МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Методические указания для выполнения
лабораторных работ
по дисциплине

Объектно-ориентированный анализ и программирование

для студентов специальности
080700.62
«Бизнес-информатика»

Томск - 2012

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации

Утверждаю:
Зав. каф. АОИ
профессор
_____Ю.П. Ехлаков
«__» _____2012 г.

Методические указания для выполнения лабораторных работ по дисциплине

Объектно-ориентированный анализ и программирование

для студентов специальности

080700.62

«Бизнес-информатика»

Разработчик:
ст. преподаватель каф. АОИ
_____ Н.В. Пермякова

Содержание

1. ЛАБОРАТОРНАЯ РАБОТА №1. ОБЩИЕ НАВЫКИ РАБОТЫ СО СРЕДОЙ 4
2. ЛАБОРАТОРНАЯ РАБОТА №2. СОЗДАНИЕ ДИАГРАММЫ ПРЕЦЕДЕНТОВ 13
3. ЛАБОРАТОРНАЯ РАБОТА № 3. СОЗДАНИЕ ДИАГРАММЫ СОСТОЯНИЙ 27
4. ЛАБОРАТОРНАЯ РАБОТА № 4. СОЗДАНИЕ ДИАГРАММ ВЗАИМОДЕЙСТВИЯ 33
5. ЛАБОРАТОРНАЯ РАБОТА № 5. СОЗДАНИЕ ДИАГРАММЫ КЛАССОВ 48
6. ЛАБОРАТОРНАЯ РАБОТА № 6. «ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. РАБОТА С КЛАССОМ VECTOR» 94
7. ЛАБОРАТОРНАЯ РАБОТА № 7. «ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. СОЗДАНИЕ КЛАССА ГРАФИЧЕСКОГО ИЗОБРАЖЕНИЯ» 98
ЛАБОРАТОРНАЯ РАБОТА №8. «НАСЛЕДОВАНИЕ И ПОЛИМОРФИЗМ. СОЗДАНИЕ КЛАССА-НАСЛЕДНИКА» 102
ЛАБОРАТОРНАЯ РАБОТА № 9. «ПЕРЕГРУЗКА ОПЕРАЦИЙ» 105
10. ЛАБОРАТОРНАЯ РАБОТА № 10. «СОЗДАНИЕ МАССИВА ОБЪЕКТОВ. ОДНОРОДНЫЕ ОБЪЕКТЫ. РАЗНОРОДНЫЕ ОБЪЕКТЫ.» 107
11. ЛАБОРАТОРНАЯ РАБОТА № 11. «СОЗДАНИЕ КЛАССА ДЛЯ РАБОТЫ С ФАЙЛАМИ. СИСТЕМА МЕНЮ» 110
12.ЛАБОРАТОРНАЯ РАБОТА №12. «РАЗРАБОТКА И СОЗДАНИЕ СОБСТВЕННОГО КЛАССА» 115
СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ 116

1. Лабораторная работа №1. Общие навыки работы со средой

1.1. Цель работы

Цель лабораторной работы – приобретение общих навыков работы со средой Rational Rose 2003: мощным инструментом анализа и проектирования объектно-ориентированных информационных систем.

Модель, созданная в Rational Rose, содержит диаграммы UML и их элементы, предоставляя детальную информацию о составе системы и аспектах ее функционирования. Использование модели в качестве эскиза или чертежа, позволяет решить проблему, присущую традиционному подходу к разработке информационных систем (рис.1.1).

Рисунок 1.1. Традиционный подход к разработке информационных систем.

Невзирая на то, что требования были документированы, весь проект находится в голове ведущего разработчика, и никто, кроме него, не понимает достаточно хорошо архитектуру системы. Когда ведущий разработчик оставляет команду, информация уходит вместе с ним. Модель Rational Rose предлагает иной подход (рис. 1.2).

Рисунок 1.2. Подход к разработке информационных систем с использованием Rational Rose.

Проект изначально документирован. Разработчики могут собраться вместе и обсудить принимаемые по проекту решения до фактического написания кода. Нет необходимости беспокоиться, что каждый из них пойдет своим путем в проектировании частей одного и того же приложения.

Rational Rose это средство, которое может быть использовано всеми участниками проекта. Фактически это - хранилище информации о проекте, обращаясь к которому, каждый участник проекта извлекает то, что ему нужно. Кроме того, Rational Rose позволяет генерировать "скелетный код" на большом количестве объектно-ориентированных языков, включая C++, Java, Visual Basic и PowerBuilder. Более того, можно выполнять обратное проектирование кода, создавая таким образом модели уже существующих систем. Весьма выгодно иметь модели в Rational Rose для уже существующих приложений. Если сделано изменение в модели, Rational Rose позволяет модифицировать код для его реализации. Если был изменен код, можно автоматически обновить соот-

ветствующим образом модель. Благодаря этому удается поддерживать соответствие между моделью и кодом, уменьшая риск "устаревания" первой.

Среду Rational Rose можно расширить с помощью RoseScript, языка программирования, поставляемого вместе с ней. На RoseScript можно написать код для автоматического внесения изменений в модель, для создания отчетов и выполнения других задач.

1.2. Элементы интерфейса Rational Rose

1.2.1. Общее описание

В пособии рассматривается версия Rational Rose 2002, функционирующая под управлением ОС Windows. Среда поддерживает работу со всеми типами канонических диаграмм языка UML посредством меню, основной и контекстной панелей инструментов. Содержимое последней изменяется в зависимости от типа текущей диаграммы.

Кроме того, пользователю предоставляются контекстные всплывающие меню, выводимые при щелчке правой кнопкой мыши. Браузер среды позволяет быстро и легко получать доступ к диаграммам и другим элементам модели. Для вывода расширенной справки используется клавиша F1.

Список основных элементов интерфейса с указанием закрепленных за ними функций приведен в таблице 1.1.

Таблица 1.1

Элемент	Назначение
1. Браузер (Browser)	Быстрая навигации по модели.
2. Окно документирования (Documentation window)	Документирование элементов модели.
3. Панели инструментов (Toolbars)	Доступ к наиболее распространенным командам.
4. Окно диаграмм (Diagram window)	Просмотр и редактирование одной или нескольких диаграмм UML.
5. Журнал (Log)	Просмотр ошибок и отчетов о результатах выполнения команд.

Номера графических фрагментов на рис. 1.3 соответствуют номерам элементов интерфейса, приведенных в таблице 1.1.

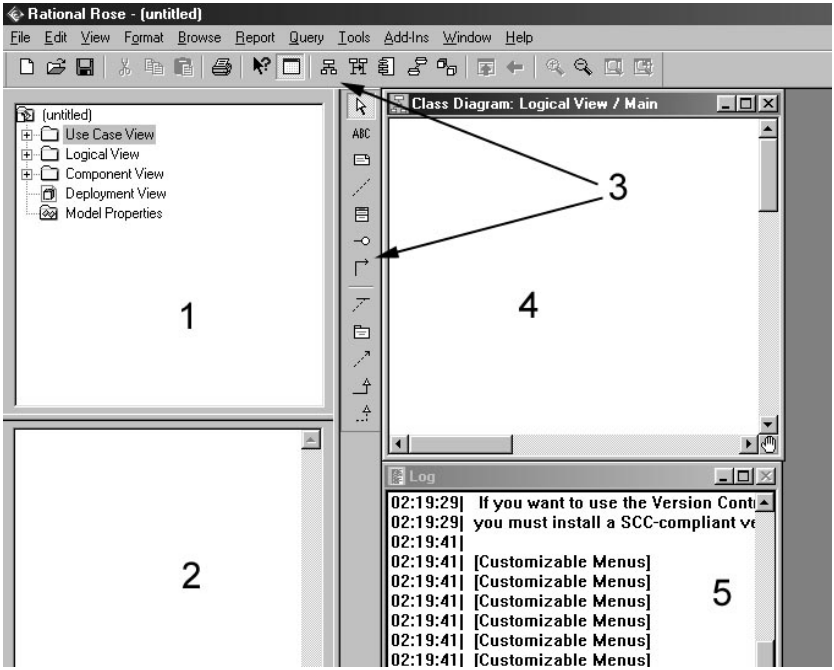

Рисунок 1.3. Основные элементы интерфейса Rational Rose 2002

1.2.2. Браузер

Окно браузера содержит дерево проекта, благодаря которому возможна понятная и простая навигация по модели. Все, что добавляется к модели, выводится в окне браузера.

С помощью браузера можно просматривать элементы модели в каждом из четырех представлений, перемещать и редактировать элементы, а также добавлять новые. Щелкнув правой кнопкой мыши на элементе в браузере, можно связать адрес URL с элементом, прочитать его спецификацию, удалить или переименовать элемент.

Информация в браузере представлена в виде дерева. Каждый элемент модели может содержать другие элементы, находящиеся ниже него в иерархии. Знак "-" около элемента означает, что его ветвь полностью раскрыта. Знак "+" - ветвь свернута.

Браузер поддерживает четыре группы представлений:

- представление прецедентов;
- логическое представление;
- представление компонентов;
- представление развертывания.

Это соответствует концепции визуального моделирования в UML, рассматривающей процесс моделирования как некий поуровневый спуск от наиболее общей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы.

Представление прецедентов предназначено для хранения концептуальной модели, логическое представление содержит логическую модель системы и, наконец, представления компонентов и развертывания - два представления физической модели.

По умолчанию браузер появляется в левой верхней части экрана. Его можно перетащить в любое другое место, закрепить там либо закрыть вовсе.

Для того, чтобы зафиксировать браузер в пределах окна необходимо щелкнуть правой кнопкой мыши на границе окна браузера. Во всплывающем меню выбрать пункт ***Allow Docking (Разрешить прикрепление)***. Рядом с этим пунктом появится отметка о том, что он выделен. Теперь браузер можно передвигать в пределах окна Rational Rose, но он будет сразу прикрепляться к одной из границ этого окна.

Для отмены прикрепления необходимо снять пометку пункта ***Allow Docking (Разрешить прикрепление)*** аналогичным образом.

Для скрытия или визуализации окна браузера необходимо щелкнуть правой кнопкой мыши на границе окна браузера. В появившемся меню выделить пункт ***Hide (Скрыть)***. Или выбрать пункт меню ***View*** > ***Browser*** **(*Вид*** > ***Браузер*)**. Окно браузера будет показано или скрыто.

1.2.3. Окно документирования

Это окно предназначено для документирования элементов модели Rational Rose. С его помощью, например, можно сделать описание каждого актера на диаграмме прецедентов. При документировании класса все, что указано в этом окне, появится затем как комментарий в сгенерированном коде, что избавляет разработчика от необходимости впоследствии вносить комментарии вручную. Эта же документация включается в отчеты, формируемые средой Rational Rose.

При выборе в браузере или на диаграмме элемента, окно документирования автоматически обновляется, показывая текст, соответствующий выбранному элементу.

Как и окно браузера, окно документирования можно перемещать. По умолчанию оно появляется в левом нижнем углу окна Rational Rose, но может быть передвинуто или скрыто.

1.2.4. Панели инструментов

Панели инструментов Rational Rose обеспечивают быстрый доступ к наиболее распространенным командам. Стандартная панель видна всегда, ее кнопки соответствуют командам, которые могут использоваться для работы с различными диаграммами.

Специфичные пиктограммы стандартной панели перечислены в таблице 1.2.

Таблица 1.2

Пиктограмма	Наименование	Назначение
	Selects or deselects an item	Предоставляет возможность выделять объект
	Browse Class Diagram	Находит и открывает диаграмму классов
	Browse Interaction Diagram	Находит и открывает диаграмму последовательности или диаграмму кооперации
	Browse Component Diagram	Находит и открывает диаграмму компонентов
	Browse State Machine Diagram	Находит и открывает диаграмму деятельности или состояний
	Browse Deployment Diagram	Находит и открывает диаграмму размещения
	Browse Parent	Находит и открывает родительскую диаграмму
	Browse Previous Diagram	Находит и открывает предыдущую диаграмму

Пиктограммы, расположенные в правой части панели инструментов (при ее настройке по умолчанию), отвечают за масштабирование диаграммы. Другие пиктограммы панели (такие как создание нового файла модели, его сохранение, печать диаграммы и проч.) присутствуют в любом развитом пакете для ОС Windows. Их назначение интуитивно понятно.

Среда Rational Rose, предоставляет возможность изменения и настройки панели инструментов. Для этого следует выбрать пункт меню ***Tools > Options (Инструменты > Параметры)***, затем вкладку ***Toolbars (Панели инструментов).***

Показать или скрыть стандартную панель инструментов можно следующим образом. Выберите пункт ***Tools > Options (Инструменты > Параметры).*** Выберите вкладку ***Toolbars (Панели инструментов).*** Установите или сбросьте флажок ***Show Standard Toolbar (Показать стандартную панель инструментов).***

Контекстная панель диаграммы изменяет свое содержимое в зависимости от типа диаграммы UML. Описание контекстных панелей дается по мере рассмотрения диаграмм соответствующего типа. Если нужно показать или скрыть контекстную панель инструментов диаграммы выберите пункт ***Tools > Options (Инструменты > Параметры).*** Выберите вкладку ***Toolbars (Панели инструментов).***Установите или сбросьте флажок ***Show Diagram Toolbar (Показать панель инструментов диаграммы)***.

Для увеличения размера кнопок панели инструментов щелкните правой кнопкой мыши на требуемой панели. Во всплывающем меню выберите пункт ***Use Large Buttons (Использовать большие кнопки).***

Для настройки панели инструментов щелкните правой кнопкой мыши на требуемой панели. Выберите пункт ***Customize (Настроить).*** Чтобы добавить или удалить кнопки, выберите соответствующую кнопку и затем щелкните мышью на кнопке ***Add (Добавить)*** или ***Remove (Удалить).***

Внешний вид окна настоек панели инструментов приведен на рисунке 1.4.

Рисунок 1.4. Внешний вид окна настроек панели инструментов.

1.2.5. Окно диаграммы

В окне диаграммы выводится одна или несколько диаграмм UML создаваемой модели. Окно диаграммы и браузер связаны между собой – изменение информации об элементе на диаграмме автоматически приводит к изменению информации в браузере и наоборот. Это позволяет поддерживать модель в непротиворечивом состоянии.

1.2.6. Журнал

Журнал содержит информацию, генерируемую средой в процессе создания и модификации модели системы. В журнал помещаются сообщения об ошибках, возникающих при генерации кода, отражаются результаты выполнения ряда операций над средой Rational Rose и моделью. Окно журнала невозможно закрыть, но можно минимизировать.

1.3. Создание модели

Первым шагом при работе с Rational Rose является создание моделей. Их можно строить либо "с нуля", либо взяв за основу существующую каркасную модель. Среда Rational Rose позволяет хранить модель в одном файле, имеющем расширение .mdl (model).

Для создания модели необходимо выбрать в меню пункт ***File > New (Файл > Создать).*** Если установлен ***Мастер каркаса (Framework Wizard)***, то

на экране появится список доступных каркасов (рис. 1.5). Следует выбрать каркас и щелкнуть на кнопке OK. Если работа с каркасами не планируется, щелкните на кнопке Cancel.

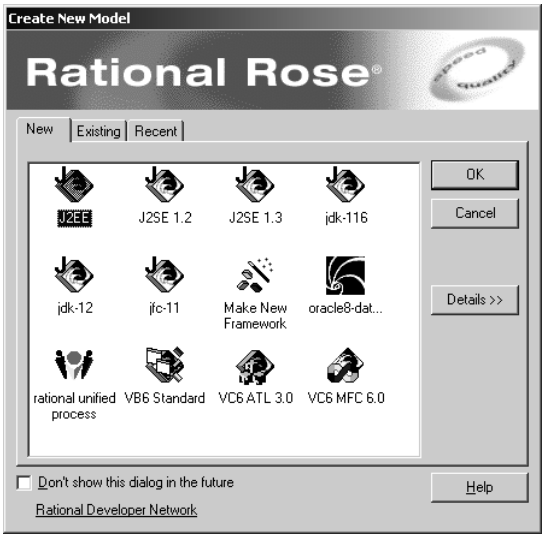

Рисунок 1.5. Мастер каркаса в Rational Rose 2002.

1.4. Сохранение модели

При работе с Rational Rose рекомендуется периодически сохранять файлы во время работы с ними. Вся модель сохраняется в одном файле. Кроме того, в отдельном файле можно сохранить журнал.

Для сохранения модели выберите в меню пункт ***File > Save (Файл > Сохранить)*** или щелкните мышью на кнопке ***Save (Сохранить)*** стандартной панели инструментов.

Для сохранения журнала выделите его окно, выберите в меню пункт ***File > Save log As (Файл > Сохранить журнал как)***. Введите название журнала. Или выделите окно журнала. Щелкните мышью на кнопке ***Save (Сохранить)*** стандартной панели инструментов. Введите имя файла журнала.

1.5. Экспорт и импорт моделей

Одним из главных преимуществ объектно-ориентированной парадигмы является возможность повторного использования, применимая не только к коду, но и к самой модели. Для максимально полной ее реализации Rational Rose поддерживает экспорт и импорт моделей и их элементов. Вы можете

экспортировать модель или ее фрагменты и затем импортировать их в другие модели.

Для экспорта модели выберите в меню пункт *File > Export Model* (*Файл > Экспортировать модель*). Введите имя экспортируемого файла.

Для экспорта пакета классов на диаграмме классов выберите пакет, который нужно экспортировать. Выберите в меню пункт *File > Export <Package> (Файл > Экспортировать <пакет>*). Введите имя экспортируемого файла.

Для экспорта класса на диаграмме классов выберите класс, который нужно экспортировать. Выберите в меню пункт *File > Export <Class> (Файл > Экспортировать <класс>).* Введите имя экспортируемого файла.

Для импорта модели, пакета или класса выберите в меню пункт *File > Import Model (Файл > Импортировать модель*). Укажите файл, который требуется импортировать. Можно импортировать файлы моделей (.MDL), petal (.PTL), категорий (.CAT) и подсистем (.SUB).

1.6. Публикация модели в Web

С помощью Rational Rose можно публиковать модели на Web-страницах — в корпоративной сети предприятия (Intranet), в Internet или на сайте файловой системы (file system site). Таким образом, все желающие смогут изучить ваши модели, даже не будучи пользователями Rational Rose и не распечатывая большое количество соответствующей документации.

Для публикации модели в сети выберите в меню пункт ***Tools > Web Publisher (Инструменты > Мастер публикации в сети).*** В окне ***Мастера публикации*** (см. рис. 1.6) укажите представления модели и пакеты. Выберите требуемый уровень детализации. Уровень ***Documentation Only (Только документация)*** соответствует наиболее общей информации, при этом не будут показаны никакие свойства элементов модели. При выборе уровня ***Intermediate (Промежуточный)*** будут показаны те свойства элементов модели, которые описаны на вкладке ***General (Общие)*** их спецификаций. Уровень ***Full (Полный)*** означает публикацию всех свойств, включая те, что содержатся на вкладке ***Detail*** спецификации элементов модели.

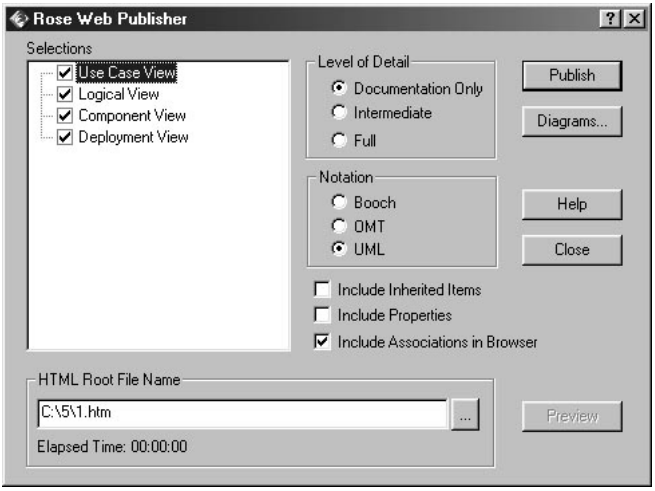

Рисунок 1.6. Окно мастера публикаций.

Выберите требуемую нотацию. По умолчанию будет использоваться та нотация, что принята по умолчанию в среде Rose. Укажите, хотите ли вы опубликовать наследуемые элементы ***(inherited items)***. Укажите, надо ли публиковать свойства. Введите название корневого файла HTML.

Если вы хотите использовать для диаграмм графические файлы, нажмите на кнопку ***Diagrams (Диаграммы)***. Появится окно ***Diagram Options (Параметры диаграммы)***, как показано на рис. 1.7.

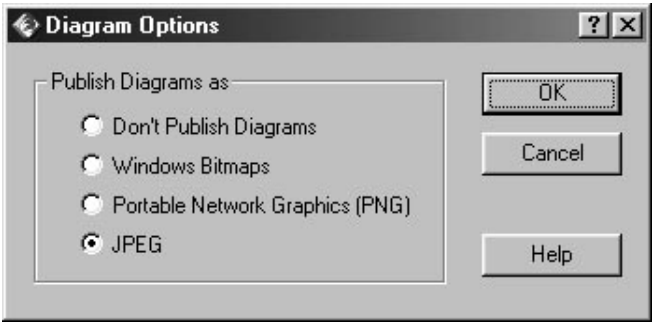

Рисунок 1.7. Окно выбора параметров публикации диаграмм.

Укажите формат, в котором будут опубликованы ваши диаграммы. Можно выбрать Windows Bitmaps (Растровые изображения в среде Windows), Portable Network Graphics (PNG) (Переносимая по сети графика) или JPEG либо отказаться от публикации диаграмм.

Закончив, щелкните на кнопке ***Finish (Готово)***. В результате будут созданы все Web-страницы, требуемые для публикации модели.

Нажатие мышью на кнопке ***Preview (Предварительный просмотр)*** позволяет увидеть результат.

1.7. Задание и отчетность

Лабораторная работа носит ознакомительный характер. Полученные в результате ее выполнения навыки являются базовыми для выполнения последующих заданий. Предлагается на практике опробовать работу описанных пунктов меню, настроить основную панель среды и опубликовать отчет в Intranet. Защита выполняется путем собеседования.

2. Лабораторная работа №2. Создание диаграммы прецедентов

2.1. Цель работы

Лабораторная работа направлена на формирование навыков разработки диаграммы прецедентов с использованием инструментальной среды. В пособии описаны основные приемы создания, модификации и специфицирования диаграммы прецедентов и ее элементов в Rational Rose 2002. Организация материала соответствует последовательности этапов создания модели. В пособии описаны процессы: создания диаграммы прецедентов и задания ее параметров, размещения актеров и прецедентов и задания их спецификаций, установление отношений между элементами диаграммы, добавления и специфицирования прочих элементов.

Пособие не содержит описания семантики основных элементов языка UML и приемов объектно-ориентированного анализа и проектирования (ООАП). Эти сведения образуют материал лекционных и практических занятий.

2.2. Основные операции, выполняемые над диаграммой прецедентов

2.2.1. Главная диаграмма прецедентов

В среде Rational Rose диаграммы прецедентов создаются в ***представлении прецедентов (Use Case View)***. Главная диаграмма (Main) предлагается по умолчанию. При моделировании системы существует возможность разрабатывать столько дополнительных диаграмм прецедентов, сколько этого требует проект.

Для получения доступа к главной диаграмме прецедентов щелкните мышью в браузере на значке "+" рядом с представлением прецедентов, что приведет к его раскрытию. Появится главная диаграмма прецедентов, представленная в древовидном списке элементом с именем Main. Дважды щелкнув на основной диаграмме, откройте ее.

2.2.2. Создание новой диаграммы прецедентов

Если в рамках проекта создается несколько диаграмм прецедентов, логично располагать их в рамках представления прецедентов, хотя среда позволяет расположить их в любой ветке дерева проекта.

Для создания новой диаграммы прецедентов щелкните правой кнопкой мыши на пакете представления прецедентов в браузере. Во всплывающем меню выберите пункт ***New > Use Case Diagram (Создать > Диаграмма прецедентов)***. Выделив новую диаграмму, введите ее имя. Дважды щелкнув на названии этой диаграммы в браузере, откройте ее.

2.2.3. Открытие диаграммы прецедентов

Чтобы открыть имеющуюся диаграмму прецедентов найдите ее в дереве, отображаемом в браузере. Дважды щелкнув на имени диаграммы, откройте ее. Или выберите в меню команду ***Browse > Use Case Diagram (Обзор > Диаграмма прецедентов)***.В списке пакетов выделите тот, который содержит требуемую диаграмму. В списке диаграмм прецедентов выберите диаграмму, которую нужно открыть. Нажмите ОК.

2.2.4. Удаление диаграммы прецедентов

Иногда требуется удалить созданные диаграммы прецедентов. На начальном этапе разработки проекта, при проведении "мозгового штурма" сферы применения системы, обычно создается большое количество таких диаграмм. Следует удалять из проекта избыточные диаграммы. Это делается непосредственно в браузере. Будьте внимательны: уничтожив диаграмму, вы не сможете отменить эту операцию.

Для удаления диаграммы прецедентов щелкните правой кнопкой мыши на диаграмме в браузере. В появившемся меню выберите пункт ***Delete (Удалить)***.В среде Rational Rose невозможно удалить главную (Main) диаграмму.

2.2.5. Связывание файлов и ссылок

В среде Rational Rose вы можете связать с диаграммой прецедентов файл или объект, расположенный по ссылке в сети Internet, используя URL этого объекта. Таким образом, можно связать с диаграммой любой вспомогательный документ, например спецификации требований высокого уровня или документы концептуального характера. Все связанные файлы и ссылки будут показаны в браузере под соответствующей диаграммой прецедентов. По двойному щелчку мыши на файле или ссылке в браузере откроется соответствующее приложение и загрузится документ, находящийся в файле или по указанному адресу в сети Internet.

Для связывания файла с диаграммой прецедентов щелкните правой кнопкой мыши на соответствующей диаграмме в браузере. В открывшемся меню выберите пункт ***New > File (Создать > Файл).*** В диалоговом окне ***Open***

(Открыть) укажите, какой файл нужно прикрепить к диаграмме. Нажмите кнопку ***Open (Открыть)***, чтобы выполнить прикрепление.

Для связывания ссылки с диаграммой прецедентов щелкните правой кнопкой мыши на соответствующей диаграмме в браузере. В открывшемся меню выберите пункт ***New > URL (Создать > Ссылка)***. Введите адрес.

Чтобы открыть прикрепленный файл найдите его в браузере. Дважды щелкните на имени файла.

Если нужно открыть прикрепленную ссылку найдите ее в браузере и дважды щелкните мышью на ссылке. Запустится ваш Web-браузер и будет открыта нужная ссылка. Или щелкните правой кнопкой мыши на ссылке в браузере. В появившемся меню выберите пункт ***Open (Открыть).***

Для удаления связанного файла или адреса щелкните правой кнопкой мыши на файле или адресе в браузере. В появившемся меню выберите пункт ***Delete (Удалить)***.

2.3. Панель инструментов

При открытии диаграммы прецедентов, на контекстной панели инструментов появляются соответствующие пиктограммы. Если содержимое панели не переопределяется пользователем, набор пиктограмм соответствуют приведенным в таблице 2.1.

Таблица 2.1

Пиктограмма	Наименование	Назначение
	Selection Tool	предоставляет возможность выделять объект
	Text Box	добавляет текст к диаграмме
	Note	добавляет к диаграмме примечание
	Anchor Note to Item	связывает примечание с объектом на диаграмме
	Package	помещает на диаграмму новый пакет
	Use case	помещает на диаграмму новый прецедент
	Actor	помещает на диаграмму нового актера
	Unidirectional Association	рисует направленную ассоциацию между актером и прецедентом
	Dependency or Instantiates	рисует отношение зависимости между элементами диаграммы

	Generalization	рисует отношение обобщения

2.4. Работа с актерами

2.4.1. Добавление актеров на диаграмму

Имеются два способа добавления актеров: на открытую диаграмму прецедентов и непосредственно в браузер. Из браузера актера можно перенести на одну или несколько диаграмм прецедентов.

Для того, чтобы поместить актера на диаграмму прецедентов нажмите кнопку ***Actor (Актер)*** панели инструментов. Или выделите в меню пункт ***Tools > Create > Actor (Инструменты > Создать > Актер)***. Щелкните мышью где-нибудь на диаграмме прецедентов, чтобы поместить туда нового актера. Он получит имя NewClass. Выделив только что созданного актера, введите его имя. Обратите внимание, что актер автоматически появляется в браузере, и размещается в той же ветке, что и диаграмма прецедентов, на которой он изображен.

2.4.2. Добавление актеров в браузер

Для добавления актера в браузер щелкните правой кнопкой мыши в браузере на узле дерева, которое соответствует нужному представлению. Выберите в открывшемся меню пункт ***New > Actor (Создать > Актер)***. В браузере появится новый актер под названием NewClass. Слева от его имени будет расположена пиктограмма актера, принятая в UML. Выделив нового актера, введите его имя. Чтобы поместить актера на диаграмму, перетащите его мышью из браузера на диаграмму прецедентов.

2.4.3. Удаление актеров

Возможны два способа удаления актера: только с диаграммы прецедентов или из модели в целом. Если вы удаляете актера из модели, он будет удален из браузера и со всех диаграмм прецедентов. При удалении с диаграммы актер останется на других диаграммах прецедентов и в браузере.

Для удаления актера с диаграммы прецедентов выделите его на диаграмме. Нажмите клавишу Delete.

Для удаления актера из модели выделите актера на диаграмме. Выберите в меню модели пункт ***Edit > Delete from Model*** или нажмите сочетание клавиш ***CTRL+D.*** Или щелкните правой кнопкой мыши на актере в браузере. Выберите пункт ***Delete (Удалить)*** в открывшемся меню. Среда Rational Rose удалит актера со всех диаграмм прецедентов и из браузера.

2.4.4. Спецификация актеров

Каждый актер в Rational Rose имеет подробную спецификацию. В окне спецификации актера вы можете определить его имя, стереотип, количество

экземпляров, называемый также множественностью ***(multiplicity)*** и другие детали. Ниже рассматриваются основные атрибуты этого элемента диаграммы прецедентов.

Окна спецификаций актера и класса похожи. Это логично, поскольку в Rational Rose актер рассматривается как особая форма класса. Окно спецификации актера содержит те же поля, что и окно класса, но для актера некоторые из этих полей заблокированы (disabled).

Открыть окно спецификации актера можно следующим образом. Щелкните правой кнопкой мыши на актере в диаграмме прецедентов или в браузере. В открывшемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. Появится окно спецификации актера. Или выделите актера на диаграмме прецедентов. Выберите пункт меню ***Browse > Specification (Обзор > Спецификация)*** или нажмите сочетание клавиш CTRL+B.

Большинство вкладок этого окна применимо к классам и недоступно для задания спецификации актеров, за исключением вкладок ***General (Общие)***, ***Detail (Подробно)***, ***Relations (Отношения)*** и ***Files (Файлы)***. Некоторые из параметров на этих вкладках также используются только для классов и недоступны.

Каждому актеру должно быть дано уникальное имя. Его можно присвоить в окне спецификации актера, непосредственно на диаграмме прецедентов или в браузере. Щелкните правой кнопкой мыши на пиктограмме актера, расположенной на диаграмме прецедентов или в браузере. В открывшемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. В поле ***Name (Имя)*** введите имя актера. Или выделив актера на диаграмме прецедентов или в браузере, введите его имя.

Если нужно ввести текстовое описание для актера, выделите актера в браузере. В окне документирования, расположенном по умолчанию ниже окна браузера введите описание. Или щелкните правой кнопкой мыши на актере в браузере или на диаграмме прецедентов. В открывшемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. В области ***Documentation (Документирование)*** окна спецификации введите описание актера.

Для назначения актеру стереотипа щелкните правой кнопкой мыши на пиктограмме актера в браузере или на диаграмме прецедентов. В открывшемся меню выберите пункт ***Open Specification (Открыть спецификацию)***.В поле ***Stereotype (Стереотип)*** укажите стереотип актера.

В среде Rational Rose можно указать, какое количество экземпляров актера введено в модель. Например, существует множество людей, играющих роль актера-клиента, но только один человек, играющий роль актера-менеджера. Чтобы зафиксировать этот факт, можно использовать поле ***Multiplicity (Множественность)*** окна спецификации. Список предопределенных значений приведен в таблице 2.2.

Таблица 2.2

Множественность	**Значение**
n (по умолчанию)	много
0..0	нуль
0..1	нуль или один

0..n	нуль или больше
1..1	ровно один
1..n	один или больше

Кроме того, существует возможность ввести значение множественности согласно форматам, приведенным в таблице 2.3.

Таблица 2.3

Формат	**Значение**
<число>	ровно число
<число 1>..<число 2>	между числом 1 и числом 2
<число>..n	число или больше
<число 1>,<число 2>	число 1 или число 2
<число 1>,<число 2>..<число 3>	ровно число 1 или между числом 2 и числом 3
<число 1>..<число 2>,<число 3>..<число 4>	между числом 1 и числом 2 или между числом 3 и числом 4

Для задания множественности актера щелкните правой кнопкой мыши на пиктограмме актера в браузере или на диаграмме прецедентов. В открывшемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. Перейдите на вкладку ***Detail (Подробно)***. Выберите нужную вам множественность в раскрывающемся списке ***Multiplicity (Множественность)*** или введите ее самостоятельно, используя один из указанных выше форматов.

Для создания абстрактного актера создайте его в браузере или на диаграмме прецедентов. Щелкните правой кнопкой мыши на актере в браузере или на диаграмме. В открывшемся меню выберите пункт ***Open Specification (Открыть спецификацию)***.Выберите вкладку ***Detail***.Установите флажок ***Abstract (Абстрактный)***.

Среда Rational Rose поддерживает правило языка UML, которое предписывает выделять курсивом имя абстрактного актера.

2.4.5. Просмотр отношений, в которых участвует актер

Вкладка ***Relations (Отношения)*** окна спецификации актера содержит список всех отношений, в которых он участвует. Перечисляются отношения актера, как с прецедентами, так и с другими актерами. Список содержит имена отношений, актеров и прецедентов, которые в нем участвуют.
С помощью этой вкладки можно просматривать, добавлять и удалять отношения.

Для просмотра отношений, в которых участвует актер, щелкните правой кнопкой мыши на пиктограмме актера в браузере или на диаграмме прецедентов. В открывшемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. Отношения актера перечислены на вкладке ***Relations (Отношения)***.

Для просмотра спецификации отношения дважды щелкните мышью на отношении в списке. Появится окно спецификаций отношения. Или щелкните правой кнопкой мыши на отношении в списке. В открывшемся меню выберите

пункт ***Specification (Спецификация)***. Появится окно спецификаций отношения. Подробное описание параметров отношений приведено в разделе 2.5. Для удаления отношения щелкните правой кнопкой мыши на отношении в списке. В открывшемся меню выберите пункт ***Delete (Удалить)***.

2.4.6. Связывание файлов и ссылок с актером

Процедура связывания реализуется посредством вкладки ***Files (Файлы)*** окна спецификации актера. Связываемые с актером файлы и Web-страницы должны содержать документы, касающиеся только этого актера, например диаграмму потока работ, иллюстрирующую задачи актера и процесс, в котором он участвует. Если актером является другая система, с ним можно связать относящуюся к ней спецификацию и документацию.

Все связываемые с актером файлы и ссылки появляются в браузере под именем этого актера. Двойной щелчок мыши на файле или ссылке в браузере приводит к запуску соответствующего приложения и загрузке в него требуемого файла или ссылки.

Связать с актером файл можно следующим образом. Щелкните правой кнопкой мыши где-нибудь в свободном белом поле вкладки ***Files***. В открывшемся меню выберите пункт ***Insert File (Вставить файл)***. В диалоговом окне найдите файл, который нужно прикрепить. Нажмите кнопку ***Open (Открыть)***, чтобы связать файл с актером. Имя и путь этого файла появятся во вкладке ***Files*** и в браузере ниже имени актера.

Для связывания с актером ссылки на Web-страницу щелкните правой кнопкой мыши в свободном белом поле вкладки ***Files***. В открывшемся меню выберите пункт ***Insert URL (Вставить ссылку)***. Введите адрес страницы. Он появится в браузере под именем актера.

Чтобы открыть файл, связанный с актером, вызовите спецификацию актера. Дважды щелкните мышью на имени файла во вкладке ***Files***. Rational Rose автоматически запустит соответствующее приложение и загрузит в него файл. Или найдите файл в браузере. Файл находится под именем соответствующего актера. Дважды щелкните мышью на имени файла. Rational Rose автоматически запустит соответствующее приложение и загрузит в него файл. Или откройте спецификацию актера. Щелкните правой кнопкой мыши на файле во вкладке ***Files.*** В открывшемся меню выберите пункт ***Open File / URL (Открыть файл / ссылку)***. Rational Rose автоматически запустит соответствующее приложение и загрузит в него файл. Или найдите файл в браузере. Файл расположен под именем соответствующего актера. Щелкните правой кнопкой мыши на файле в браузере. В открывшемся меню выберите пункт ***Open (Открыть)***. Rational Rose автоматически запустит соответствующее приложение и загрузит в него файл.

Для перехода по связанной с актером ссылке на Web-страницу откройте окно спецификации актера. Дважды щелкните мышью на ссылке во вкладке ***Files***. Rational Rose автоматически запустит ваш Web-браузер и загрузит в него ссылку. Или найдите ссылку в браузере. Она находится под именем соответствующего актера. Дважды щелкните мышью на ссылке. Rational Rose автома-

тически запустит ваш Web-браузер и загрузит в него ссылку. Или откройте спецификацию актера. Щелкните правой кнопкой мыши на ссылке во вкладке ***Files***. В открывшемся меню выберите пункт ***Open File / URL (Открыть файл / ссылку)***. Rational Rose автоматически запустит ваш Web-браузер и загрузит в него ссылку. Или найдите ссылку в браузере. Она расположена под именем соответствующего актера. Щелкните правой кнопкой мыши на ссылке в браузере. В открывшемся меню выберите пункт ***Open (Открыть)***. Rational Rose автоматически запустит ваш Web-браузер и загрузит в него ссылку.

Для удаления связанного с актером файла или ссылки щелкните правой кнопкой мыши на файле или ссылке в браузере. В открывшемся меню выберите пункт ***Delete (Удалить)***.

2.4.7. Просмотр экземпляров актера

При моделировании системы может потребоваться узнать, в каких диаграммах последовательности и диаграммах кооперации участвует данный актер. Для этого в Rational Rose предусмотрено меню ***Report (Отчет)***.

Для просмотра списка диаграмм последовательности и диаграмм кооперации, содержащих того или иного актера, выделите его на диаграмме прецедентов. Выберите в меню пункт ***Report > Show Instances (Отчет > Показать экземпляры)***. Rational Rose выведет список всех диаграмм последовательности и диаграмм кооперации, содержащих данного актера. Чтобы открыть диаграмму, дважды щелкните на ней в списке или нажмите кнопку ***Browse (Обзор)***.

2.5. Работа с прецедентами

2.5.1. Добавление прецедентов

Существуют два способа добавления прецедентов в модель: помещение его на активную диаграмму прецедентов или непосредственно в браузер. Из браузера его можно перетащить на требуемую диаграмму.

Поместить новый прецедент на диаграмму прецедентов можно следующим образом. Нажмите пиктограмму ***Use Case (прецедент)*** панели инструментов. Или выберите в меню пункт ***Tools > Create > Use Case*** (***Инструменты > Создать > Прецедент).*** Щелкните мышью где-нибудь внутри диаграммы прецедентов, чтобы поместить туда новый прецедент. Он будет назван NewUseCase. Выделив этот прецедент, введите его имя.

Обратите внимание, что новый прецедент был автоматически добавлен в браузер и помещен ниже представления, в котором хранится содержащая прецедент диаграмма.

Поместить на диаграмму прецедентов имеющийся прецедент можно следующим образом. Перетащите прецедент из браузера на открытую диаграмму. Или выберите в меню пункт ***Query > Add Use Cases (Запрос > Добавить прецеденты)***. Появится окно диалога, с помощью которого можно выбирать и добавлять существующие прецеденты. В раскрывающемся списке пакетов

выберите тот из них, который содержит нужные прецеденты. Перетащите требуемые прецеденты из списка прецедентов ***(Use Cases)*** в список выделенных прецедентов ***(Selected Use Cases)***. Нажмите на OK. Прецеденты будут добавлены к диаграмме.

Если нужно поместить прецедент в браузер, щелкните правой кнопкой мыши на пакете представления прецедентов в браузере. В открывшемся меню выберите пункт ***New > Use Case (Создать > Прецедент)***. Новый прецедент под названием NewUseCase появится в браузере. Слева от него будет выведена пиктограмма прецедента UML. Выделив новый прецедент, введите его имя. Чтобы поместить прецедент на диаграмму, перетащите его туда из браузера.

2.5.2. Удаление прецедентов

Прецедент можно удалить с одной диаграммы или со всех диаграмм модели. В начале работы над проектом часто создается много прецедентов, полезных при изучении области применения проекта. Но после утверждения окончательного набора прецедентов, избыточные следует удалить.

Для удаления прецедента с диаграммы проделайте следующие шаги. Выделите прецедент на диаграмме. Нажмите на клавишу Delete.

Обратите внимание, что, хотя прецедент удален с диаграммы прецедентов, он остался в браузере и на других диаграммах системы.

Для удаления прецедента из модели выделите прецедент на диаграмме. Выберите в меню пункт ***Edit > Delete from Model*** (***Правка > Удалить из модели***) или нажмите сочетание клавиш CTRL+D. При этом прецедент будет удален со всех диаграмм и из браузера. Или щелкните правой кнопкой мыши на прецеденте в браузере. В появившемся меню выберите пункт ***Delete*** (***Удалить***). При этом прецедент будет удален со всех диаграмм и из браузера.

2.5.3. Спецификация прецедентов

В Rational Rose можно создать подробную спецификацию для каждого прецедента. Спецификации помогают документировать такие атрибуты прецедентов, как имена, приоритеты и стереотипы.

Открыть спецификацию прецедента можно следующим образом. Щелкните правой кнопкой мыши на прецеденте в браузере или на диаграмме прецедентов. В появившемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. Или выделите прецедент на диаграмме прецедентов. Выберите в меню пункт ***Browse > Specification*** (***Обзор > Спецификация***) или нажмите сочетание клавиш CTRL+B.

Присвоить имя прецеденту можно непосредственно на диаграмме или с помощью окна спецификации.

Для присвоения имени прецеденту выделите его в браузере или на диаграмме прецедентов. Введите имя. Или щелкните правой кнопкой мыши на прецеденте в браузере или на диаграмме. В появившемся меню выберите пункт ***Open Specification*** (***Открыть спецификацию***). В поле ***Name*** (***Имя***) введите имя прецедента.

Для добавления к прецеденту текстового описания выделите прецедент в браузере. В окне документирования наберите описание прецедента. Щелкните правой кнопкой мыши на прецеденте в браузере или на диаграмме. В появившемся меню выберите пункт ***Open Specification*** (***Открыть спецификацию***). В области ***Documentation*** (***Документирование***) окна спецификации введите описание прецедента.

Назначить стереотип прецедента можно следующим образом. Щелкните правой кнопкой мыши на прецеденте в браузере или на диаграмме прецедентов. В открывшемся меню выберите пункт ***Open Specification*** (***Открыть спецификацию***). Введите стереотип в поле ***Stereotype*** (***Стереотип***).

При определении прецедентов можно назначить каждому из них приоритет. Это позволяет установить порядок, в котором прецеденты будут разрабатываться. При описании спецификаций прецедента в Rational Rose его приоритет указывается в поле Rank (Ранг). Для назначения прецеденту приоритета щелкните правой кнопкой мыши на прецеденте в браузере или на диаграмме прецедентов. В открывшемся меню выберите пункт ***Open Specification*** (***Открыть спецификацию***). На вкладке ***General*** (***Общие***) введите приоритет в поле ***Rank*** (***Ранг***).

В окне спецификации прецедента вы можете увидеть все диаграммы последовательности, диаграммы кооперации, диаграммы классов, прецедентов и состояний, указанные в браузере под этим прецедентом. Для этого используется вкладка ***Diagrams (Диаграммы)*** окна спецификации прецедента.

Здесь содержатся названия диаграмм и их типы, отображаемые соответствующими пиктограммами. Двойной щелчок мыши на любой из диаграмм приведет к ее открытию. В ходе данной лабораторной работы диаграммы других типов не создаются, поэтому список будет пуст.

Вкладка ***Files (Файлы)*** окна спецификации прецедента позволяет прикрепить файл к прецеденту. Файл может содержать документ, описывающий поток событий данного прецедента, документы с описанием сценариев для прецедента, спецификации требований или другие документы с приблизительными прототипами прецедента. Можно также связать с прецедентом ссылку на Web-страницу.

Чтобы прикрепить к прецеденту файл или сопоставить ссылку руководствуйтесь рекомендациями пункта 2.3.6.

2.6. Работа с отношениями

2.6.1. Отношение коммуникации

Для добавления на диаграмму отношения коммуникации нажмите кнопку ***Unidirectional Association (Однонаправленная ассоциация)*** панели инструментов. Проведите мышью, удерживая ее левую кнопку, от актера к прецеденту. Между прецедентом и актером будет показана стрелка, соответствующая отношению.

Создаваемая таким образом коммуникация трактуется средой как отношение направленной ассоциации (Unidirectional Association). Если есть необ-

ходимость сделать ассоциацию ненаправленной, щелкните правой кнопкой мыши на отношении коммуникации. В открывшемся меню снимите пометку с пункта ***Navigable,*** щелкнув по нему мышью.

Для удаления отношения коммуникации выделите отношение на диаграмме прецедентов. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите сочетание клавиш CTRL+D.

2.6.2 Отношения расширения

Существуют два альтернативных способа добавления отношения расширения на диаграмму. Первый способ использовался в ранних версиях Rational Rose и поддерживается Rational Rose 2002 с целью обеспечения совместимости.

Нажмите кнопку ***Unidirectional Association (Направленная ассоциация)*** панели инструментов. При нажатой левой кнопке проведите мышью от предоставляющего дополнительную функциональность прецедента к применяющему ее прецеденту. Между этими прецедентами будет нарисована ассоциация. На отношении щелкните правой кнопкой мыши и в появившемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. В раскрывающемся списке ***Stereotype (Стереотип)*** выберите пункт ***extend (расширение)***. Если данный пункт недоступен, введите слово “extend” с клавиатуры. Оно появится в списке и будет доступно в дальнейшем. Щелкнув мышью на кнопке ОК, закройте окно спецификации. Слово «extend» появится над стрелкой, соответствующей отношению обобщения. Если это слово не выводится, щелкните правой кнопкой мыши на линии отношения и в открывшемся меню пометьте пункт ***Stereotype Label (Метка стереотипа)***.

Rational Rose 2002 поддерживает UML 1.4. Решением консорциума OMG отношение расширения рассматривается как базовое. С целью поддержания нововведений в среду добавлена пиктограмма, позволяющая добавить отношения расширения и включения на диаграмму прецедентов альтернативным способом. Необходимо предварительно настроить панель инструментов диаграммы прецедентов, так как по умолчанию пиктограммы скрыты.

Для настройки панели инструментов выполните следующую последовательность шагов. Нажмите правую кнопку мыши на панели инструментов диаграммы прецедентов. В раскрывшемся меню выберите пункт ***Customize… (Настройка…).*** Внешний вид возникающего в результате окна настройки панели инструментов приведен на рис. 2.1.

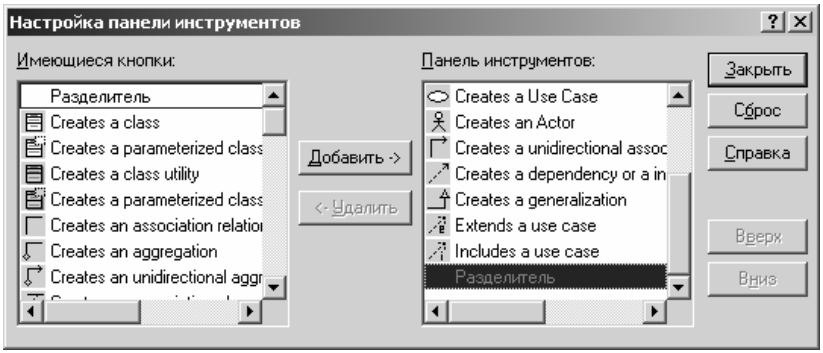

Рисунок 2.1. Внешний вид окна настройки панели инструментов

Левый список содержит перечень всех пиктограмм, которые могут быть использованы при построении диаграммы прецедентов. Позиционируйтесь на элементе с именем ***"Extend a Use case"***. Нажмите кнопку "Добавить", чтобы поместить его на панель инструментов диаграммы прецедентов. Рекомендуется аналогичным образом добавить на панель инструментов пиктограмму для создания отношения включения. Ей соответствует элемент списка с именем ***"Include a Use case"***.

Для добавления отношения расширения, созданного описанным способом, нажмите кнопку ***"Extend a Use case"*** панели инструментов. При нажатой левой кнопке проведите мышью от предоставляющего дополнительную функциональность прецедента к применяющему ее прецеденту.

Для удаления отношения расширения независимо от способа его создания, выделите отношение на диаграмме прецедентов. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите сочетание клавиш CTRL+D.

2.6.3. Отношение включения

Так же, как и отношение расширения, отношение включения отнесено к базовым типам отношений согласно спецификации языка UML 1.4. Для него определено два способа добавления на диаграмму: первый - для совместимости с диаграммами, созданными в ранних версиях Rational Rose, второй поддерживается Rational Rose 2002.

Нажмите кнопку ***Unidirectional Association (Направленная ассоциация)*** панели инструментов. При нажатой левой кнопке проведите мышью от базового прецедента к прецеденту, функциональность которого следует включить в базовый прецедент. Между этими прецедентами будет нарисована ассоциация. На отношении щелкните правой кнопкой мыши и в появившемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. В раскрывающемся списке ***Stereotype (Стереотип)*** выберите пункт ***include (включение)***. Если данный пункт недоступен, введите слово "include" с клавиатуры.

Оно появится в списке и будет доступно в дальнейшем. Щелкнув мышью на кнопке OK, закройте окно спецификации. Слово «include» появится над стрелкой, соответствующей отношению включения. Если это слово не выводится, щелкните правой кнопкой мыши на линии отношения и в открывшемся меню пометьте пункт ***Stereotype Label (Метка стереотипа)***.

Альтернативным решением является использование специальной пиктограммы ***"Include a Use case"***, вынесение которой на панель инструментов диаграммы прецедентов описано в разделе 2.5.3.

Нажмите кнопку ***"Include a Use case"*** панели инструментов. При нажатой левой кнопке проведите мышью от базового прецедента к прецеденту, функциональность которого следует включить в базовый прецедент.

Для удаления отношения включения независимо от способа его создания, выделите отношение на диаграмме прецедентов. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите сочетание клавиш CTRL+D.

2.6.4. Отношение обобщения

Добавить отношение обобщения, существующее между актерами можно следующим образом. Поместите актеров на диаграмму прецедентов. Выберите кнопку ***Generalization (Обобщение)*** панели инструментов. При нажатой левой кнопке проведите мышью от актера-потомка к актеру-предку.

Для прецедентов отношение обобщения добавляется аналогичным образом.

Для удаления отношения обобщения между актерами выделите отношение на диаграмме прецедентов. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш CTRL+D.

2.7. Работа с примечаниями

Добавление примечаний в Rational Rose выполняется с помощью кнопки ***Note (Примечание)*** панели инструментов.

Поместить на диаграмму примечание можно следующим образом. Нажмите на панели инструментов кнопку ***Note (Примечание)***. Щелкните мышью где-нибудь внутри диаграммы, чтобы поместить туда примечание. Выделив новое примечание, введите текст.

Для прикрепления примечания к элементу диаграммы, нажмите кнопку ***Anchor Note to Item (Прикрепить примечание к элементу)*** панели инструментов. Нажав левую кнопку мыши, проведите ею от примечания к прецеденту или актеру, с которым оно будет связано. Между примечанием и прецедентом или актером будет начерчена штриховая линия.

При работе над диаграммой может понадобиться удалить некоторые примечания. Выделите примечание на диаграмме. Нажмите клавишу Delete.

2.8. Работа с пакетами

В языке UML такие элементы, как актеры, прецеденты, классы и компоненты, можно сгруппировать в пакеты (packages). В частности, в представлении прецедентов можно сгруппировать в пакеты прецеденты и актеров. Внешний вид пиктограммы пакета приведен на рис 2.2.

Рисунок 2.2. Изображение пиктограммы пакета с Rational Rose

Для упорядочения элементов модели в среде Rational Rose можно создавать столько пакетов, сколько нужно. При необходимости, для лучшей организации разрешается помещать один пакет внутрь другого. При создании нового пакета Rational Rose автоматически создает диаграмму ***Package Overview (Обзор пакета)*** и ***список ассоциаций (Associations List)***.

Для добавления пакета на диаграмму проделайте следующие операции. Щелкните правой кнопкой мыши на представлении прецедентов в браузере. Если необходимо вложить вновь создаваемый пакет в существующий, щелкните правой кнопкой мыши на существующем пакете в браузере. В открывшемся меню выберите пункт ***New > Package (Создать > Пакет)***. Введите имя нового пакета.

Чтобы поместить в пакет элемент диаграммы, перетащите в браузере его пиктограмму в этот пакет.

Пакет в среде Rational Rose можно удалить с одной диаграммы прецедентов или из модели в целом. В последнем случае пакет и все его содержимое будут полностью уничтожены.

Для удаления пакета с диаграммы прецедентов выделите пакет на диаграмме прецедентов. Нажмите клавишу Delete. Обратите внимание, что, хотя пакет исчез с данной диаграммы, он остался в браузере и на других диаграммах.

Для удаления пакета из модели щелкните правой кнопкой мыши на пакете в браузере. В открывшемся меню выберите пункт Delete (Удалить).

2.9. Задание и отчетность

Необходимо создать диаграмму прецедентов в инструментальной среде Rational Rose 2002, следуя описанным принципам работы с системой. Объектом автоматизации является фирма, описанная в практическом пособии.

Документы отчетности сдаются на проверку в электронной форме и включают в себя:

- файл модели (*.mdl);

- документ Microsoft Word, содержащий описание потоков событий прецедентов моделируемой информационной системы;
- документ Microsoft Word, содержащий описание основных проектных решений;
- файл с указанием группы и фамилии студентов, выполнивших работу.

Допускается документирование потоков событий и основных проектных решений в рамках среды Rational Rose с использованием окна документирования. В этом случае помимо файла модели необходимо сдать на проверку набор HTML-страниц, представляющих собой отчет, генерируемый средствами Rational Rose. Указания по его генерации можно найти в разделе 1.6.

3. Лабораторная работа № 3. Создание диаграммы состояний

3.1. Цель работы.

Лабораторная работа направлена на формирование навыков разработки диаграммы состояний с использованием инструментальной среды. В пособии описаны основные приемы создания, модификации и специфицирования диаграммы состояний и ее элементов в Rational Rose 2002.

В среде Rational Rose на основании диаграмм состояний не генерируется программного кода. Они применяются для документирования динамики поведения объекта, которым может являться сущность предметной области, класс, или система в целом. Как и другие диаграммы языка UML, диаграммы состояний дают команде разработчиков возможность обсудить и документировать логику приложения до начала этапа кодирования.

3.2. Создание диаграммы состояний

В среде Rational Rose диаграммы состояний целесообразно создавать либо в представлении прецедентов (Use Case View), либо в логическом представлении (Logical View) модели. Положения в дереве модели определяется природой моделируемого объекта. Если моделируются состояния сущности предметной области или системы в целом, целесообразно создавать диаграмму состояний для элемента, представленного на диаграмме прецедентов. Если речь идет о моделировании состояний объектов информационной системы, создавайте диаграмму состояний класса, описывающую его состояния и переходы между ними. В этом случае диаграмма будет расположена в дереве ниже класса и помечена пиктограммой  .

Для создания диаграммы состояний щелкните правой кнопкой мыши в браузере на нужном классе. В открывшем меню выберите пункты ***New > Statechart Diagram (Создать > Диаграмму состояний)***.

3.3. Добавление состояний

Состоянием (State) является одно из возможных условий, в котором может существовать объект. Для выяснения возможных состояний объекта необходимо исследовать атрибутный состав объекта и его взаимосвязи с другими объектами. Как и другие элементы моделей Rational Rose, состояние может иметь текстовое описание, которое служит для пояснения его семантики и не включается в качестве комментария в генерируемый программный код.

Для добавления состояния нажмите кнопку ***State*** панели инструментов и щелкните мышью в том месте диаграммы состояний, где его необходимо разместить.

Для добавления к состоянию текстового описания используйте окно документирования среды или дважды щелкните мышью на состоянии, открыв окно его спецификации. Перейдите на вкладку ***General (Общие свойства)***. Введите описание в поле документирования.

3.4. Спецификация состояний

3.4.1. Понятие действия и деятельности

Пребывая в определенном состоянии, объект может выполнять некоторые действия. Например, он может генерировать отчет, осуществлять некоторые вычисления или генерировать событие, адресованное другому объекту. В среде Rational Rose информация такого рода добавляется к состоянию путем его спецификации.

Деятельностью (activity) называется поведение, реализуемое объектом в рамках состояния, в котором он пребывает. Деятельность – прерываемое поведение. Оно может выполняться до тех пор, пока объект находится в данном состоянии и естественным образом завершиться, либо может быть прервано переходом объекта в другое состояние.

Как правило, деятельность располагается в состоянии после метки ***do (делать)*** и отделяется от метки наклонной чертой (см. рис.3.1). В ранних версиях языка UML и Rational Rose 98 метка отделялась двоеточием.

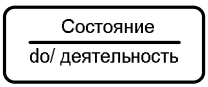

Рисунок 3.1. Состояние, содержащее деятельность.

Входным действием (entry action) называется поведение, выполняемое объектом при переходе в состояние. Данное действие выполняется не после того, как объект перешел в состояние, а на входе в него. В отличие от деятель-

ности, входное действие следует рассматривать как непрерывную операцию. Входное действие располагают внутри состояния. Ему предшествует метка ***entry (вход)***, отделяемая наклонной чертой (рис. 3.2).

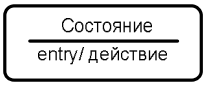

Рисунок 3.2. Состояние, содержащее входное действие.

Выходное действие ***(exit action)*** подобно входному. Однако, оно выполняется при выходе объекта из состояния (рис. 3.3). Как и входное действие, выходное действие является непрерываемым.

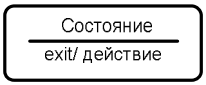

Рисунок 3.3. Состояние, содержащее действие, выполняемое при выходе.

3.4.2. Инструментальная поддержка в Rational Rose

Rational Rose 2002 несколько сужает семантику понятий "деятельность" и "действие", применительно к диаграмме состояний. Среда оперирует только понятием "action" (действие) независимо от сути поведения объекта. В ранних версиях продукта можно было оперировать как понятием действия, так и понятием деятельности, что достигалось несколько иной трактовкой меток.

Кроме того, как говорилось ранее, поведение объекта в процессе его пребывания в состоянии, может включать в себя генерацию события и посылку информации о нем другому объекту. В этом случае описанию деятельности или действия предшествует символ ***"^",*** а строка, содержащая в рамках состояния отправку информации о генерируемом событии, имеет вид:

Метка / Цель . Событие (Аргументы), где:

- ***Цель*** – объект, принимающий информацию о наступлении события;
- ***Событие*** – посылаемая информация о событии;
- ***Аргументы*** – параметры посылаемого сообщения (рис 3.4).

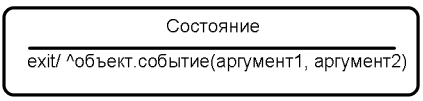

Рисунок 3.4. Состояние, включающее в себя генерацию события.

Для добавления действия, выполняемого в рамках состояния, откройте окно спецификации требуемого состояния. Перейдите на вкладку ***Action (Дей-***

ствия). Щелкните правой кнопкой мыши в области списка. В открывшемся меню выберите пункт ***Insert (Вставить)***. Дважды щелкните мышью на новом действии, появившемся в списке. В результате появится окно задания подробной спецификации действия. В выпадающем списке ***When (Когда)*** выберите нужное значение. В поле ***типа (Type)*** оставьте значение по умолчанию, которым является Action.В поле имени укажите имя действия.

Если необходимо смоделировать пересылку информации о генерации события другому объекту, повторите описанную ранее последовательность шагов. В поле типа выберите значение ***Send Event.***В поле ***Name*** укажите имя события, информацию о котором посылаете. В поле ***Argument*** через запятую перечислите аргументы. В поле ***Send Target*** укажите имя объекта, которому адресовано сообщение.

3.4.3. Задание начальных и конечных состояний

Корректная диаграмма состояний обязана содержать два специальных состояния – начальное и конечное.

Начальным (Start State) называется состояние, в котором объект находится сразу после своего создания. Наличие начального состояния является обязательным условием: лицо, читающее диаграмму, должно знать, с чего начинается жизненный цикл объекта. Допускается наличие на диаграмме только одного начального состояния.

Конечным (End State) состоянием называется состояние, в котором объект находится непосредственно перед уничтожением. Из соображений простоты визуального восприятия допускается наличие на диаграмме нескольких конечных состояний.

Для создания начального состояния нажмите кнопку ***Start State*** контекстной панели и щелкните мышью на диаграмме состояний в том месте, где его необходимо разместить. Аналогичным образом задается конечное состояние с использованием кнопки ***End State***.

3.5. Добавление переходов

3.5.1. Понятие перехода

Переходом (Transition) называется перемещение моделируемого объекта из одного состояния в другое. Совокупность переходов показывает, как объект может переходить из одного состояния в другое. На диаграмме состояний все переходы изображаются в виде стрелки, ведущей из исходного состояния в целевое.

Переходы могут быть рефлексивными, если исходное и целевое состояние совпадают. Рефлексивные переходы изображают в виде стрелки, ведущей в состояние, из которого она исходит (рис. 3.5).

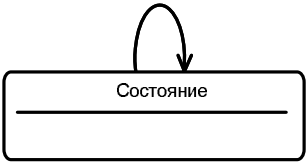

Рисунок 3.5. Рефлексивный переход.

3.5.2. Добавление перехода

Для добавления перехода нажмите кнопку ***State Transitions (Переход между состояниями)*** панели инструментов. Щелкните мышью по пиктограмме того состояния, откуда осуществляется переход. Проведите линию к тому состоянию, в которое переход должен приводить.

Для добавления рефлексивного перехода нажмите кнопку ***Transition to Self (Рефлексивный переход)*** панели инструментов.

Для добавления к переходу текстового описания используйте окно документирования среды, предварительно выделив документируемый переход на диаграмме или в браузере.

3.5.3. Спецификация перехода

Переход может характеризоваться следующим набором спецификаций: событие, аргументы, сторожевые условия, действия и посылаемой информацией о событии.

Событие (Event) –это то, что вызывает переход объекта из одного состояния в другое. События играют роль стимулов. Про них говорят, что они происходят. События размещают на диаграмме вдоль линии перехода. В зависимости от природы моделируемого объекта в качестве события можно использовать имя операции класса или обычную фразу на естественном языке.

У событий могут быть аргументы, которые Rational Rose позволяет к ним добавлять.

Большинство переходов должно иметь события, так как именно они инициируют срабатывание переходов. Тем не менее среда допускает наличие автоматических переходов, не имеющих событий. При этом объект сам перемещается из состояния в состояние со скоростью, предусматривающей выполнение входных и выходных действий.

Сторожевые условия (Guard Condition) определяют, когда переход может быть выполнен, а когда нет. На диаграмме состояний сторожевые условия располагаются вдоль линии перехода и заключаются в квадратные скобки.

Сторожевые условия задавать необязательно, однако если существует нескольких автоматических переходов из состояния или если два и более перехода срабатывают при наступлении одного и того же события, необходимо определить для них взаимно исключающие сторожевые условия.

Действием является поведение объекта, выполняемое как часть перехода. Спецификация действия, располагаемого вдоль перехода идентична спецификации действия, выполняемого во время пребывания состоянии. Действие, выполняемой при переходе размещается вдоль линии перехода после имени события, и ему предшествует косая черта (/).

Для добавления на диаграмму события и задания его аргументов откройте окно его спецификации, дважды щелкнув мышью на переходе. Перейдите на вкладку ***General (Общие)***. Введите событие в поле ***Event (Событие)***. Введите аргументы в поле ***Arguments (Аргументы)***.

Если необходимо задать сторожевое условие, откройте окно спецификации перехода, дважды щелкнув на нем мышью. Перейдите на вкладку ***Detail (Подробно)***. Введите сторожевое условие в поле ***Guard Condition (Сторожевое условие)***.

Для добавления действия, выполняемого при переходе, введите данные в поле ***Action (Действие).***

Отправить информацию о событии можно следующим образом. Дважды щелкнув мышью на переходе, откройте его спецификацию. Перейдите на вкладку ***Detail (Подробно)***. Введите событие в поле ***Send Event (Отправляемая информация о событии)***. Введите аргументы в поле ***Send Argument (Отправляемые аргументы)***. Укажите цель в поле ***Send Target (Цель события).***

3.6. Использование вложенных состояний

Для улучшения визуального восприятия диаграмм состояний спецификация языка UML предусматривает вложение одних состояний в другие.

Вложенные состояния называются ***подсостояниями (substates)***, а состояния, в которые они вложены – ***суперсостояниями (superstates)***.

Вложить состояния друг в друга можно следующим образом. Нажмите кнопку ***State (Состояние)*** контекстной панели инструментов и щелкните мышью на состоянии, в которое нужно вложить новое состояние.

Среда позволяет перетаскивание существующие на диаграмме состояний с использованием технологии drug-and-drop. В этом случае нужно перетащить пиктограмму вкладываемого состояния на пиктограмму суперсостояния.

3.7. Задание и отчетность

Необходимо дополнить модель информационной системы, описанной в практическом пособии, создав для нее диаграмму состояний с использованием Rational Rose.

Документы отчетности сдаются на проверку в электронной форме и включают в себя файл модели (*.mdl), документы Microsoft Word, содержащие дополнительную информацию по проектным решениям, файл с указанием группы и фамилии студентов, выполнивших работу.

4. Лабораторная работа № 4. Создание диаграмм взаимодействия

4.1. Цель работы

Лабораторная работа направлена на формирование навыков разработки диаграмм взаимодействия с использованием инструментальной среды. В пособии описаны основные приемы создания, модификации и специфицирования диаграмм последовательности и кооперации в Rational Rose 2002.

4.2. Создание диаграммы последовательности

Диаграммы последовательности можно создавать в группе представлений прецедентов или в группе логических представлений браузера. Как показано на рис. 4.1, диаграммы последовательности должны быть соотнесены непосредственно с прецедентом или располагаться внутри пакета.

Для создания новой диаграммы последовательности щелкните правой кнопкой мыши на соответствующем пакете или прецеденте в браузере. В открывшемся меню выберите пункт ***New > Sequence Diagram (Создать > Диаграмма последовательности)***. Дайте новой диаграмме последовательности имя. Дважды щелкнув на диаграмме в браузере, откройте ее.

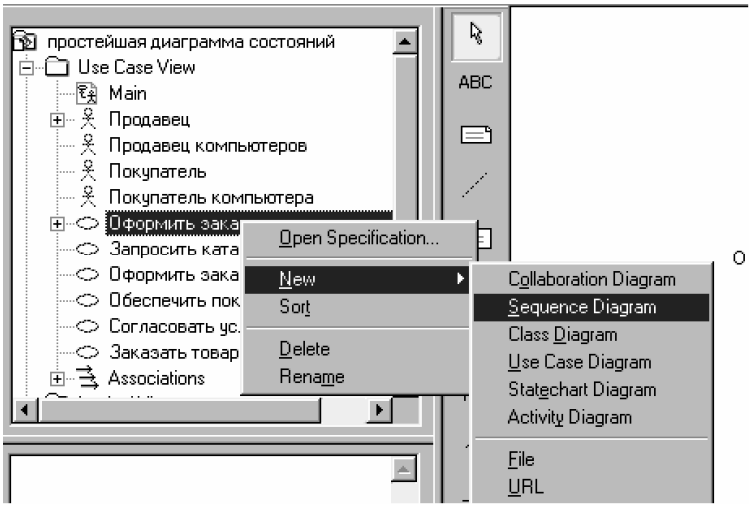

Рисунок 4.1. Создание диаграммы последовательности

Чтобы открыть диаграмму последовательности, найдите ее в представлении прецедентов браузера и дважды щелкните на ней мышью.

Для добавления на диаграмму последовательности нового элемента воспользуйтесь кнопками панели инструментов. Допускается перетаскивание актеров и классов из браузера на диаграмму.

Для удаления элемента с диаграммы последовательности выделите его на диаграмме. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш CTRL+D.

4.3. Удаление диаграммы последовательности

В процессе работы над проектом может оказаться, что некоторые диаграммы последовательности устарели или стали избыточными. Диаграммы, которые больше не нужны или не отражают проект системы, следует удалить. В среде Rational Rose удалить из модели диаграмму последовательности можно, используя браузер. Щелкните правой кнопкой мыши в браузере на диаграмме последовательности. В открывшемся меню выберите пункт Delete (Удалить).

4.4. Панель инструментов диаграммы последовательности

При открытии диаграммы последовательности контекстная панель инструментов содержит элементы, перечисленные в таблице 4.1.

Таблица 4.1.

Пиктограмма	Наименование	Назначение
	Selects or deselects an item	Превращает курсор в стрелку указателя, предоставляя возможность выделять объект
	Text Box	Добавляет к диаграмме текст
	Note	Добавляет к диаграмме примечание
	Anchor Note to Item	Связывает примечание с элементом на диаграмме
	Object (Объект)	Помещает на диаграмму новый объект
	Object Message (Сообщение для объекта)	Рисует сообщение между двумя объектами
	Message to Self (Сообщение самому себе)	Рисует рефлексивное сообщение

4.5. Диаграммы кооперации

В Rational Rose диаграмму последовательности можно преобразовать в диаграмму кооперации и наоборот, нажав клавишу F5 или выбрав в меню пункт ***Browse > Create (Sequence/Collaboration) Diagram (Обзор > Создать диаграмму (Последовательности/ Кооперации))***.

4.6. Создание диаграммы кооперации

Как и диаграммы последовательности, диаграммы кооперации обычно создаются в браузере и размещаются под прецедентом или пакетом. На рис. 4.2 показано, как добавить к модели новую диаграмму кооперации. Можно сначала создать диаграмму последовательности, а затем преобразовать ее в диаграмму кооперации. Среда автоматически сгенерирует диаграмму кооперации из указанной диаграммы последовательности.

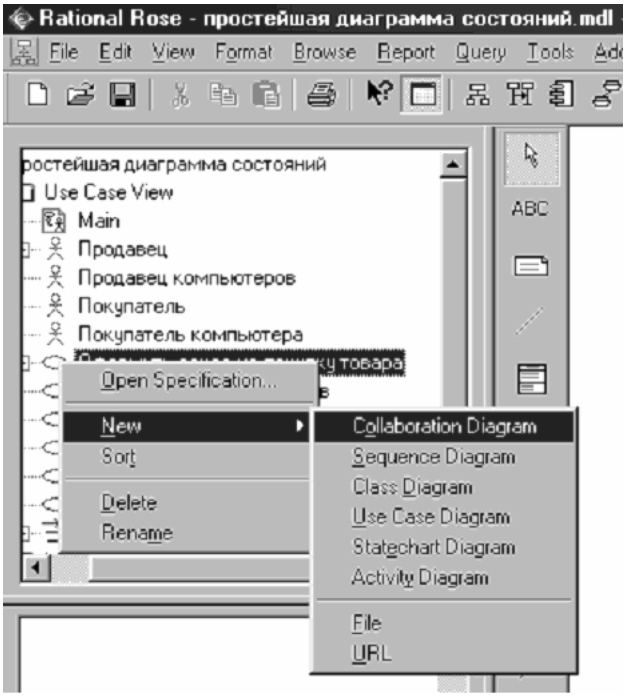

Рисунок 4.2 Создание новой диаграммы кооперации.

Для создания новой диаграммы кооперации щелкните правой кнопкой мыши на нужном варианте использования в браузере. В открывшемся меню выберите пункт ***New > Collaboration Diagram (Создать > Диаграмма кооперации)***. Дайте этой диаграмме имя. Дважды щелкнув мышью на диаграмме в браузере, откройте ее.

4.7. Панель инструментов диаграммы кооперации

Панель инструментов диаграммы кооперации напоминает панель инструментов диаграммы последовательности. Однако имеется несколько пиктограмм, недоступных на диаграмме последовательности. Они приведены в таблице 4.2.

Таблица 4.2.

Пиктограмма	Наименование	Назначение
	Object (Объект)	Помещает на диаграмму новый объект
	Class Instance	Помещает на диаграмму новый экземпляр класса
	Object Link	Создает путь коммуникации между объектами
	Link to Self	Показывает, что объект может обращаться к своим операциям
	Link Message	Показывает сообщение, передаваемое между двумя объектами или передаваемое объектом самому себе
	Reverse Link Message	Показывает сообщение, передаваемое в противоположном направлении по связи между объектами или объектом самому себе
	Data Flow	Показывает поток информации между объектами
	Reverse Data Flow	Показывает лоток информации между объектами в противоположном направлении

4.8. Работа с актерами на диаграмме взаимодействия

Поместить на диаграмму взаимодействия объект-актер можно следующим образом. Откройте диаграмму взаимодействия. Выберите актера в браузере. Перетащите его из браузера на открытую диаграмму.

Для удаления объекта-актера с диаграммы взаимодействия выделите актера на диаграмме взаимодействия. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите сочетание клавиш CTRL+D.

4.9. Добавление объектов к диаграммам взаимодействия

Одним из первых этапов в создании диаграмм последовательности и диаграмм кооперации является добавление к ним объектов. Чтобы найти их, изучите имена существительные в вашем потоке событий и сценарии.

Поместить объект на диаграмму последовательности можно следующим образом. Нажмите кнопку Object (Объект) панели инструментов. Щелкните мышью в том месте диаграммы, куда будет помещен объект. На диаграмме последовательности их располагают в ряд в верхней части. Введите имя нового объекта. Создав объект, всегда можно изменить его положение на диаграмме, перетащив его мышью.

Чтобы расположить объект между двумя существующими объектами, достаточно щелкнуть мышью между ними на втором шаге описанной последовательности.

Для добавления объекта на диаграмму кооперации нажмите кнопку Object (Объект) панели инструментов. Щелкните мышью в том месте диаграммы, куда будет помещен объект. На диаграмме кооперации его можно расположить в любом месте. Введите имя нового объекта.

4.10. Удаление объектов с диаграмм взаимодействия

В процессе работы с диаграммами взаимодействия вам может понадобиться удалить некоторые объекты. При этом Rational Rose автоматически удалит также все сообщения, которые начинаются или заканчиваются на объекте, и переименует оставшиеся сообщения.

При удалении объекта с диаграммы последовательности Rose автоматически удалит его и с соответствующей диаграммы кооперации, но оставит его класс. Аналогично, при удалении объекта с диаграммы кооперации он будет автоматически удален и с диаграммы последовательности. Если потребуется восстановить объект, можно воспользоваться командой Edit > Undo (Правка > Отменить команду) меню модели. Для удаления объекта с диаграммы последовательности или диаграммы кооперации выделите его на одной из этих диаграмм. В меню модели выберите пункт Edit > Delete from Model (Правка > Удалить из модели) или нажмите комбинацию клавиш CTRL+D.

При удалении объекта с диаграммы соответствующий класс модели сохраняется.

4.11. Спецификация объекта

Rational Rose предоставляет возможность детально специфицировать объекты диаграммы. Можно ввести имя и класс объекта, указать устойчивость его существования (persistence) и возможность наличия у него нескольких экземпляров (рис.4.3).

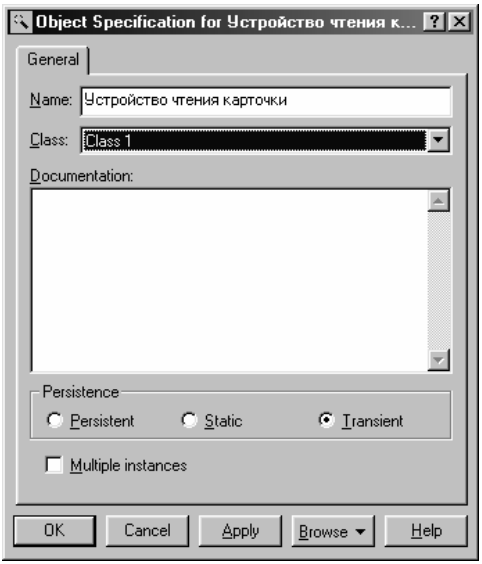

Рисунок 4.3. Окно спецификации объекта.

Открыть окно спецификации объекта можно следующим образом. Щелкните правой кнопкой мыши на объекте на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (Открыть спецификацию). Или выделите объект на диаграмме последовательности или диаграмме кооперации. В меню модели выберите команду ***Browse > Specification (Обзор > Спецификация)*** или нажмите комбинацию клавиш **CTRL+B.**

4.11.1. Именование объекта

Имя каждого присутствующего на диаграмме объекта можно задать непосредственно на диаграмме или в окне его спецификации. Дать название объекту можно следующим образом. Щелкните правой кнопкой мыши на объекте на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (Открыть спецификацию). Введите имя объекта в поле Name (Имя). Каждый объект на диаграмме должен иметь уникальное имя. В дальнейшем с помощью этого поля можно изменить имя объекта. Или выделите объект на диаграмме последовательности или диаграмме кооперации. Нажмите правую кнопку мыши так, чтобы курсор был виден на объекте. Введите имя объекта.

Если объект уже был соотнесен с классом, описание, введенное в окне документирования, будет добавлено и к классу. В противном случае оно будет связано только с объектом. Описание объекта не повлияет на генерацию кода,

описание класса будет присутствовать и в коде. Описание объекта можно просмотреть в отчете, генерируемом командой ***File > Print Specifications (Файл > Печать спецификаций)*** меню модели.

4.11.2. Соотнесение объекта с классом

По умолчанию объекту назначается класс Unspecified (Не определен). При назначении объекту класса можно либо указать уже существующий класс модели, либо создать новый класс. К моменту генерации кода все объекты должны быть соотнесены с классами.

Для соотнесения объекта с существующим классом щелкните правой кнопкой мыши на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (Открыть спецификацию). В раскрывающемся списке классов выберите имя класса или введите его с клавиатуры. После соотнесения класса с объектом название класса появится на диаграмме за именем объекта и двоеточием. Или выделите класс в логическом представлении браузера. Перетащите класс из браузера на объект диаграммы. После соотнесения класса с объектом название класса появится на диаграмме за именем объекта и двоеточием.

Если удалить класс, с которым соотнесен объект, имя класса сохранится на диаграмме рядом с объектом, но будет заключено в скобки.

Разорвать соотнесение объекта с классом можно следующим образом. Щелкните правой кнопкой мыши на объекте на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (Открыть спецификацию). В раскрывающемся списке классов выберите пункт (Unspecified) (Не определен).

Если нужно создать для объекта новый класс, щелкните правой кнопкой мыши на объекте на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (Открыть спецификацию). В раскрывающемся списке классов выберите пункт New (Создать). Перед вами появится окно спецификации для нового класса.

Если необходимо убедиться, что все объекты соотнесены с классами, выберите в меню модели пункт ***Report > Show Unresolved Objects (Отчет > Показать свободные объекты).*** Появится список всех объектов, которые еще не были соотнесены с классами.

Для того чтобы на диаграмме выводилось только имя объекта, щелкните правой кнопкой мыши на объекте на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (Открыть спецификацию). Введите имя объекта в поле Name. В раскрывающемся списке классов выберите пункт (Unspecified) (Не определен).

Если нужно показать на диаграмме имя объекта и имя его класса, щелкните правой кнопкой мыши на объекте на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (Открыть спецификацию). В раскрывающемся списке классов выберите имя класса или введите его с клавиатуры.

Если необходимо работать только с именем класса, щелкните правой кнопкой мыши на объекте на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (От-

крыть спецификацию). Удалите имя объекта из поля его имени. Теперь на диаграмме для объекта будет показано только имя соответствующего класса. Как и прежде, перед ним будет двоеточие.

4.11.3. Определение устойчивости объекта

Для определения устойчивости объекта щелкните правой кнопкой мыши на объекте на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (Открыть спецификацию). Установите переключатель Persistence в значение Persistent, Static или Transient.

Для работы с множественными экземплярами объекта щелкните правой кнопкой мыши на объекте на диаграмме последовательности или диаграмме кооперации. В появившемся меню выберите пункт Open Specification (Открыть спецификацию). Установите или сбросьте флажок Multiple Instances (Множественные экземпляры). На диаграмме кооперации для этого объекта будет показана соответствующая пиктограмма (значок множественного или одиночного объекта). На диаграмме последовательности всегда выводится пиктограмма одиночного объекта.

4.12. Работа с сообщениями

4.12.1. Добавление сообщений

Сообщение (message) — это связь между объектами, в которой один из них (клиент) требует от другого (сервера) выполнения каких-то действий. При генерации кода сообщения транслируются в вызовы функций.

На диаграмме последовательности сообщение изображают в виде стрелки, которая проводится между линиями жизни двух объектов или от линии жизни объекта к самой себе. Сообщения располагают в хронологическом порядке сверху вниз.

Поместить сообщение на диаграмму последовательности можно следующим образом. Нажмите кнопку Object Message (Сообщение объекта) панели инструментов. Нажав левую кнопку мыши, проведите ею от линии жизни объекта или актера, посылающего сообщение, к объекту или актеру, получающему сообщение Впечатайте текст сообщения.

Если нужно поместить на диаграмму последовательности рефлексивное сообщение, нажмите кнопку Message to Self (Сообщение самому себе) панели инструментов. Щелкните мышью на линии жизни объекта, посылающего и получающего сообщение Пока новое сообщение выделено, введите его текст.

4.12.2. Удаление сообщений

В процессе работы с диаграммами последовательности может потребоваться удалить ранее нарисованные сообщения. При этом оставшиеся сообщения будут автоматически перенумерованы.

Для удаления сообщения с диаграммы последовательности выделите удаляемое сообщение. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш CTRL+D.

4.12.3. Изменение порядка сообщений

Иногда требуется изменить порядок сообщений на диаграмме последовательности. В среде Rational Rose для этого достаточно перетащить сообщение на новое место. При изменении порядка следования сообщений они автоматически перенумеровываются.

Изменить порядок следования сообщений на диаграмме последовательности можно следующим образом. Выделите перемещаемое сообщение. Перетащите его вверх или вниз. Все сообщения на диаграмме будут автоматически перенумерованы .

4.12.4. Нумерация сообщений

Хотя диаграмму всегда читают сверху вниз, вы можете еще и пронумеровать сообщения, чтобы показать порядок их следования. Номера сообщений на диаграммах взаимодействия показывать необязательно. По умолчанию нумерация сообщений на диаграммах последовательности отключена.

Если нужно включить или выключить нумерацию сообщений в меню выберите пункт Tools > Options (Инструменты > Параметры). Перейдите на вкладку Diagram (Диаграмма). Установите или сбросьте флажок Sequence numbering (Нумерация сообщений), как показано на рис. 4.4.

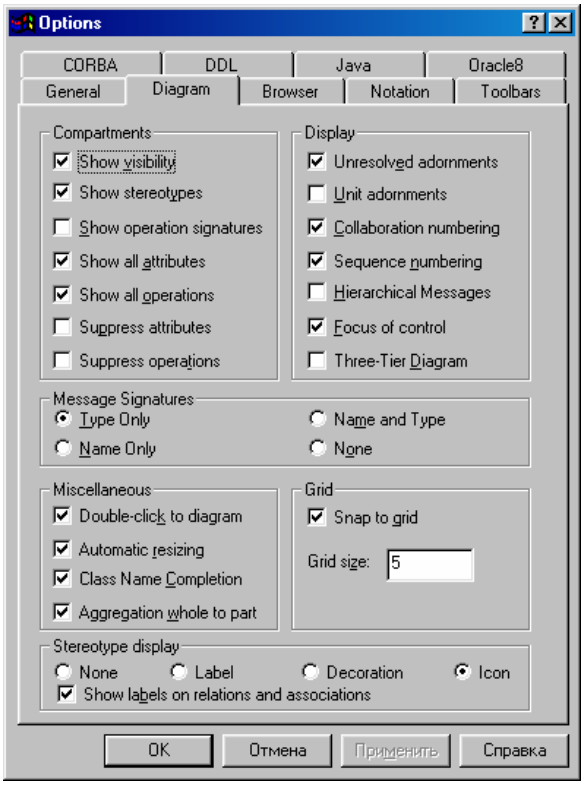

Рисунок 4.4. Установка нумерации сообщений.

4.12.5. Просмотр перехода фокуса

На диаграммах последовательности можно показать фокус (focus of control). Фокус — это маленький прямоугольник, поясняющий, какой из объектов активен и получает управление в конкретный промежуток времени. В этом заключается одно из различий между диаграммами последовательности и диаграммами кооперации, поскольку активизация может быть показана только на диаграмме последовательности.

Включить или отключить показ фокуса можно следующим образом. В меню выберите пункт Tools > Options (Инструменты > Параметры). Перейдите на вкладку Diagram (Диаграмма). Установите или сбросьте флажок Focus of control.

4.12.6. Добавление сообщений на диаграмму кооперации

Перед тем как поместить сообщение на диаграмму кооперации, необходимо установить путь коммуникации между объектами. Этот путь называется связью (link) и создается с помощью кнопки Object Link (Связь объекта) панели инструментов. После создания связи можно поместить сообщение между объектами.

Поместить сообщение на диаграмму кооперации можно следующим образом. Нажмите на панели инструментов кнопку Object Link (Связь объекта). Нажав левую кнопку, проведите мышью от одного объекта к другому, чтобы создать связь. На панели инструментов нажмите кнопку Link Message (Сообщение связи) или Reverse Link Message (Сообщение обратной, связи). Щелкните на связи между объектами. При этом будет нарисована стрелка сообщения. Выделив новое сообщение, введите текст.

Если нужно поместить на диаграмму кооперации рефлексивное сообщение нажмите кнопку Link to Self (Связь с собой) панели инструментов. Щелкните мышью на объекте, посылающем и принимающем сообщение. У этого объекта будет нарисована рефлексивная связь. На диаграмме она изображается в виде полукруга над объектом. На панели инструментов нажмите кнопку Link Message (Сообщение связи). Щелкните на рефлексивной связи объекта. При этом будет нарисована стрелка сообщения. Выделив новое сообщение, введите текст.

4.12.7. Удаление сообщений с диаграммы кооперации

Как и в случае диаграмм последовательности, с диаграмм кооперации можно удалять сообщения. При этом оставшиеся сообщения будут автоматически перенумерованы. Для удаления сообщения с диаграммы кооперации выделите удаляемое сообщение. В меню модели выберите пункт Edit > Delete from Model (Правка > Удалить из модели) или нажмите комбинацию клавиш CTRL+D.

4.12.8. Нумерация сообщений на диаграмме кооперации

Поскольку диаграмму последовательности читают сверху вниз, нумерация сообщений необязательна. Однако на диаграмме кооперации информация о порядке следования сообщений будет потеряна, если они не будут пронумерованы.

Хотя это и не рекомендуется, имеется возможность отключить нумерацию сообщений на диаграмме кооперации.

Включить или отключить нумерацию сообщений можно следующим образом. Выберите пункт меню Tools > Options (Инструменты > Параметры). Перейдите на вкладку Diagram (Диаграмма). Установите или сбросьте флажки Sequence Numbering и Collaboration numbering.

4.13. Добавление потоков данных к диаграмме кооперации

Для добавления потока данных к диаграмме кооперации на панели инструментов нажмите кнопку Data Flow (Поток данных) или Reverse Data Flow (Обратный поток данных). Щелкните на сообщении, с которым необходимо связать данные. На диаграмму будут добавлены соответствующие потоку данных стрелки. Выделив этот поток данных, введите в него возвращаемые данные.

4.14. Спецификации сообщений

4.14.1. Именование сообщений

Для детализации сообщений в среде Rational Rose можно задать большое количество различных параметров. Как и в случае прецедентов или актеров, можно вводить имена и текстовое описание сообщений.

Кроме того, разрешается определять параметры синхронизации и частоты. Рассмотрим все параметры, доступные для сообщений.

Открыть спецификации сообщений можно следующим образом. Дважды щелкните мышью на сообщении диаграммы. Появится окно спецификации сообщения.

В окне документации сообщения можно дать сообщению имя или изменить его, а также добавить текстовое описание. У каждого сообщения должно быть имя, соответствующее его цели. В дальнейшем, при соотнесении сообщений с операциями, имя сообщения будет заменено именем операции.

Дать сообщению имя можно следующим образом. Дважды щелкните на сообщении на диаграмме последовательности или диаграмме кооперации. Если получающий сообщение объект уже соотнесен с классом, операции этого класса появятся в раскрывающемся списке Name (Имя). Выберите в этом списке нужную операцию или введите имя сообщения вручную. Или выделите сообщение на диаграмме последовательности или диаграмме кооперации. Введите его имя. Если получающий сообщение объект соотнесен с классом, имя этого класса появится рядом с именем сообщения в поле класса. Это поле изменить нельзя. Если необходимо изменить получающий сообщение класс, соотнесите объект с другим классом в окне спецификации объекта.

Если сообщение уже было соотнесено с операцией, текстовое описание, создаваемое в оке документирования, будет добавлено и к соответствующей операции, в противном случае оно будет связано только с сообщением. Текстовое описание сообщения не влияет на генерацию кода. После соотнесения сообщения с операцией класса связанное с операцией текстовое описание появится в генерируемом коде в виде комментария.

4.14.2. Соотнесение сообщения с операцией

Прежде чем приступить к генерации кода, следует соотнести сообщения диаграмм последовательности и диаграмм кооперации с операциями классов. Для соотнесения сообщения с существующей операцией проследите, чтобы получающий сообщение объект (сервер) был соотнесен с классом. Щелкните правой кнопкой мыши на сообщении на диаграмме последовательности или диаграмме кооперации. Появится список операций сервера. Выберите операцию из списка.

Можно уничтожить соотнесение сообщения с операцией. Дважды щелкните на сообщении на диаграмме последовательности или диаграмме кооперации. В поле Name удалите имя операции и введите новое имя сообщения.

Если нужно создать новую операцию для сообщения, проследите, чтобы получающий сообщение объект (сервер) был соотнесен с классом. Щелкните правой кнопкой мыши на сообщении на диаграмме последовательности или диаграмме кооперации. Выберите пункт <new operation> (новая операция). Введите имя и детали новой операции. Нажмите кнопку OK**,** чтобы закрыть окно спецификации операции и завершить создание новой операции. Щелкните на сообщении правой кнопкой мыши. В появившемся списке выберите новую операцию.

Если необходимо убедиться в том, что каждое сообщение было соотнесено с операцией, выберите в меню модели пункт Report > Show Unresolved Messages (Отчет > Показать свободные сообщения). Появится список всех сообщений, которые еще не были соотнесены с классами.

4.14.3. Установка синхронизации сообщений

На вкладке Detail (Подробно) окна спецификации сообщений можно определить синхронизацию посылаемых сообщений. Соответствующая сообщению стрелка изменится, если этот параметр будет определен как "с отказом становиться в очередь", "с лимитированным временем ожидания" или "асинхронное". Доступны пять значений параметра синхронизации.

Simple (Простое). Используется по умолчанию. Означает, что все сообщения выполняются в одном потоке управления.

Synchronous (Синхронное). Применяется, когда клиент посылает сообщение и ждет ответа пользователя.

Balking (С отказом становиться в очередь). Клиент посылает сообщение серверу. Если сервер не может немедленно принять сообщение, оно отменяется.

Timeout (С лимитированным временем ожидания). Клиент посылает сообщение серверу, а затем ждет указанное время. Если в течение этого времени сервер не принимает сообщение, оно отменяется.

Asynchronous (Асинхронное). Клиент посылает сообщение серверу и продолжает свою работу, не ожидая подтверждения о получении.

Задать синхронизацию сообщения можно следующим образом. Дважды щелкните на сообщении на диаграмме Последовательности или диаграмме кооперации. В окне спецификации сообщения перейдите на вкладку Detail (Подробно). Установите переключатель Synchronization в нужное значение.

4.14.4. Установка частоты сообщения

Устанавливая частоту сообщения (message frequency), вы указываете, что оно должно посылаться через регулярные временные интервалы, например, каждые 30 с. Такое сообщение является периодичным. Частота сообщения задается на вкладке Detail (Подробно) окна спецификации сообщения.

Periodic (Периодическое). Означает, что сообщение регулярно посылается через определенные промежутки времени.

Aperiodic (Апериодическое). Сообщение посылается нерегулярно. Оно может быть отправлено только один раз или несколько раз, но через разные промежутки времени.

Частота сообщения не изменяет его вид на диаграммах последовательности и диаграммах кооперации.

Для задания частоты сообщения дважды щелкните мышью на сообщении на диаграмме последовательности или диаграмме кооперации. Перейдите на вкладку Detail (Подробно) окна спецификации сообщения. Укажите частоту сообщения, пометив в окне соответствующий переключатель.

4.15. Работа со скриптами

Для добавления скрипта на диаграмму последовательности нажмите кнопку Text Box (Текстовая область) панели инструментов. Щелкните на диаграмме там, где нужно разместить скрипт — обычно около левого края диаграммы. Выделив текстовую область, введите скрипт. Выделите текстовую область. При нажатой клавише SHIFT выделите нужное сообщение. В меню модели выберите пункт Edit > Attach Script (Правка > Прикрепить скрипт). Если передвинуть сообщение вверх или вниз на диаграмме, скрипт последует за ним.

Если нужно отделить скрипт от сообщения, выделите скрипт. В меню модели выберите пункт Edit > Detach Script (Правка > Отделить скрипт).

4.16. Переключение между диаграммами

Обычно для конкретного сценария создают либо диаграмму последовательности, либо диаграмму кооперации. Если бы не такие инструменты моделирования, как Rational Rose, разработка этих диаграмм занимала бы слишком много времени, особенно с учетом того, что оба типа отражают одну и ту же информацию.

В среде Rational Rose можно создать диаграмму последовательности из диаграммы кооперации и наоборот. Достаточно разработать диаграмму одного из этих типов, и в дальнейшем можно будет переключаться между ними.

Для создания диаграммы кооперации из диаграммы последовательности откройте диаграмму последовательности. В меню модели выберите пункт Browse > Create Collaboration Diagram (Обзор > Создать диаграмму кооперации) или нажмите клавишу F5. Будет создана диаграмма кооперации с тем же именем, что и у открытой диаграммы последовательности.

Для создания диаграммы последовательности из диаграммы кооперации откройте диаграмму кооперации. В меню модели выберите пункт Browse > Create Sequence Diagram (Обзор > Создать диаграмму последовательности) или нажмите клавишу F5. Будет создана диаграмма последовательности с тем же именем, что и у открытой диаграммы кооперации.

Для переключения между диаграммами этих типов откройте диаграмму последовательности или диаграмму кооперации. В меню модели выберите пункт Browse > Go To (Sequence or Collaboration) Diagram, (Обзор > Перейти к диаграмме (последовательности или кооперации)) или нажмите клавишу F5.

4.17. Задание и отчетность

Необходимо дополнить модель информационной системы, описанной в практическом пособии, создав для нее диаграммы взамодействия с использованием Rational Rose.

Документы отчетности сдаются на проверку в электронной форме и включают в себя файл модели (*.mdl), документы Microsoft Word, содержащие дополнительную информацию по проектным решениям, файл с указанием группы и фамилии студентов, выполнивших работу.

5. Лабораторная работа № 5. Создание диаграммы классов

5.1. Цель работы

Лабораторная работа направлена на формирование навыков разработки диаграммы классов с использованием инструментальной среды. В пособии описаны основные приемы создания, модификации и специфицирования диаграммы классов и ее элементов в Rational Rose 2002.

5.2. Основные операции

5.2.1. Главная диаграмма классов

В среде Rational Rose диаграммы классов создаются в логическом представлении модели. При построении новой модели главная диаграмма классов создается автоматически. Она размещается непосредственно под логическим представлением и обычно содержит пакеты классов модели. Вы можете создать дополнительные диаграммы классов в логическом представлении или внутри любого пакета.

Для получения доступа к главной диаграмме классов в браузере щелкните мышью на значке "+" рядом с логическим представлением, что приведет к его раскрытию. Появится главная диаграмма классов, представленная в древовидном списке элементом с именем Main. Дважды щелкнув на основной диаграмме, откройте ее.

При открытии модели сразу после запуска Rational Rose, главная диаграмма классов появится автоматически.

5.2.2. Создание новой диаграммы классов

Для создания новой диаграммы классов щелкните правой кнопкой мыши на логическом представлении браузера. В открывшемся меню выберите пункт ***New > Class Diagram (Создать > Диаграмма классов)***. Введите имя новой диаграммы. Дважды щелкнув на диаграмме в браузере, откройте ее.

Если нужно открыть существующую диаграмму классов, найдите ее в логическом представлении браузера. Дважды щелкнув на диаграмме, откройте ее.

Для добавления к диаграмме классов нового элемента воспользуйтесь кнопками контекстной панели инструментов.

Существуют два способа удаления элемента с диаграммы. Для удаления элемента только с текущей диаграммы выделите его на диаграмме и нажмите клавишу Delete.

Для удаления элемента из модели выделите его на диаграмме, в меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш ***Ctrl+D***. Или вызовите контекстное ме-

ню, щелкнув правой кнопкой мыши на удаляемом элементе в браузере. В открывшемся меню выберите пункт ***Delete (Удалить)***.

5.2.3. Удаление диаграммы классов

В процессе работы с моделью может потребоваться удалить некоторые диаграммы классов. В среде Rational Rose это делается с помощью браузера. При удалении диаграммы, содержащиеся в ней классы, не удаляются. Они сохранятся в браузере и других диаграммах.

Для удаления диаграммы классов щелкните правой кнопкой мыши на диаграмме в браузере. В открывшемся меню выберите пункт ***Delete (Удалить)***.

5.2.4. Связывание файлов и ссылок с диаграммой классов

Если имеется дополнительная информация по классам диаграммы, можно поместить ее в файл или на Web-страницу, а затем прикрепить к диаграмме классов. Связываемые таким образом файлы и ссылки будут относиться ко всем классам диаграммы. Если необходимо связать их только с одним конкретным классом, следует прикреплять их именно к этому классу. Последовательность шагов прикрепления файла или ссылки к диаграмме или ее элементу описана в пункте 1.6.

5.3. Панель инструментов

Если на панели инструментов показаны не все кнопки, описанные в таблице 5.1, щелкните на ней правой кнопкой мыши и выберите пункт ***Customize (Настроить)*** в появившемся меню. Откроется диалоговое окно, с помощью которого их можно добавить на контекстную панель.

Таблица 5.1.

Пиктограмма	Наименование	Назначение
	Class (Класс)	Добавляет на диаграмму новый класс
	Interface (Интерфейс)	Добавляет на диаграмму новый интерфейсный класс
	Association (Ассоциация)	Добавляет ненаправленную ассоциацию
	Aggergation (Агрегация)	Добавляет отношение агрегации
	Link Attribute (Атрибут отношения)	Связывает класс с отношением ассоциации
	Package (Пакет)	Добавляет на диаграмму новый пакет
	Dependency or instantiates(Зависмость	Добавляет отношение зависмости

	или наполнение)	
	Generalization (Обобщение)	Добавляет отношение обобщения
	Realize (Реализация)	Добавляет отношение реализации
	Unidirectional Association (Однонаправленная ассоциация)	Добавляет однонаправленную ассоциацию
	Parameterized Class (Параметризованный класс)	Добавляет на диаграмму новый параметризованный класс
	Parameterized Class Utility (Утилита параметризованного класса)	Добавляет на диаграмму новую утилиту параметризованного класса
	Instantiated Class (Класс-наполнитель)	Добавляет на диаграмму новый класс-наполнитель
	Instantiated Class Utility (Утилита класса-наполнителя)	Добавляет на диаграмму новую утилиту класса-наполнителя

5.4. Работа с классами

5.4.1. Добавление класса

После создания диаграммы классов нужно добавить новые классы в модель. Среда Rational Rose позволяет работать с несколькими типами классов: регулярные, параметризованные, классы-наполнители, утилиты классов, утилиты параметризованных классов, утилиты классов-наполнителей и метаклассы.

Rational Rose предоставляет возможность детально описывать классы. Каждому классу можно дать имя, определить его стереотип, указать видимость, и ряд других параметров.

Поместить на диаграмму стандартный класс можно несколькими способами: с помощью панели инструментов, браузера и меню.

Если поместить новый класс непосредственно в браузер, он не появится ни на одной диаграмме, однако его можно разместить на ней, к примеру, перетащив мышкой. Можно расположить новый класс непосредственно на диаграмме. В этом случае он будет автоматически добавлен и в браузер.

Поместить новый класс на диаграмму классов можно с использованием кнопки ***Class (Класс)*** контекстной панели инструментов. Или в меню выберите пункт ***Tools > Create > Class (Инструменты > Создать > Класс)***. Щелкните мышью где-нибудь внутри диаграммы классов. Ваш новый класс будет назван NewClass. Rational Rose выведет список всех существующих классов.

Чтобы создать новый класс, замените слово NewClass в списке новым именем класса (рис. 5.1). Обратите внимание, что он появился не только на диаграмме, но и в логическом представлении браузера.

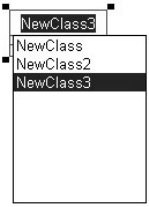

Рисунок 5.1. Добавление нового класса.

Кроме того, можно добавить новый класс с помощью диаграмм взаимодействия. Для этого откройте диаграмму последовательности или диаграмму кооперации. Щелкните правой кнопкой мыши на объекте диаграммы. В появившемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. В раскрывающемся списке классов выберите пункт ***New (Новый)***. Перед вами появится окно спецификации нового класса. В поле ***Name*** введите имя класса.

Если диаграммы взаимодействия были созданы в представлении прецедентов браузера, то создаваемые этим методом классы также будут появляться в представлении прецедентов. В логическое представление их можно перенести с помощью мыши.

Если нужно поместить на диаграмму существующий класс перетащите его из браузера на открытую диаграмму классов. Или выберите в меню пункт ***Query > Add Classes (Запрос > Добавить классы)***. Появится диалоговое окно добавления классов (рис. 5.2). В раскрывающемся списке ***Package (Пакет)*** выберите пакет, содержащий ваш класс. Перетащите нужные классы из списка ***Classes (Классы)*** в список ***Selected Classes (Выбранные классы)***. Если необходимо добавить все классы, нажмите кнопку ***All (Все)***. Нажмите кнопку OK. Выбранные классы будут добавлены к открытой диаграмме.

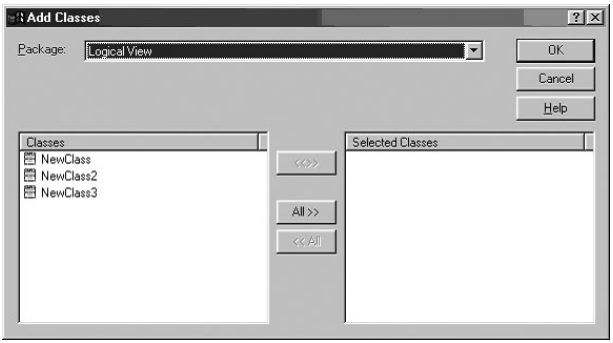

Рисунок 5.2. Добавление существующих классов на диаграмму.

Для добавления нового класса в браузер щелкните правой кнопкой мыши на логическом представлении браузера. Если вы добавляете класс к пакету, щелкните правой кнопкой на имени пакета. В открывшемся меню укажите пункт ***New > Class (Создать > Класс)***. Если необходимо создать утилиту класса или интерфейс, выберите пункт ***New > Class Utility (Создать > Утилита класса)*** или ***New > Interface (Создать > Интерфейс)***. Новый класс под названием NewClass появится в браузере. Выделите класс и введите его имя. Чтобы расположить новый класс на диаграмме классов, перетащите его на открытую диаграмму с помощью мыши.

5.4.2. Удаление класса

Как и для другие элементы модели, классы могут быть удалены . Вы можете удалить класс с диаграммы, но оставить его на других диаграммах. Можно удалить его из модели.

Для удаления класса с диаграммы классов выделите его на диаграмме. Нажмите клавишу ***Delete***.

Обратите внимание, что, хотя класс исчез с диаграммы, он остался на других диаграммах и в браузере.

Для удаления класса из модели выделите его на диаграмме. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш CTRL+D. Или щелкните правой кнопкой мыши на классе в браузере. В открывшемся меню выберите пункт ***Delete***. Класс будет удален со всех диаграмм классов и из браузера.

5.4.3. Спецификация класса

Большинство определяемых для класса параметров доступно в окне спецификации класса, показанном на рис. 5.3. В частности, это окно позволяет указать стереотип класса, а также его видимость и устойчивость (persistence).

В среде Rational Rose 2002 существует возможность описывать классы Java, CORBA и т.д., выбирая мастер каркаса. Если выбран один из этих языков, появляющееся окно спецификации будет отличаться от приведенного на рис. 5.3. Для вызова окна, приведенного ниже, используйте во всплывающем окне пункт ***Open Standard Specification (Открыть стандартную спецификацию)***.

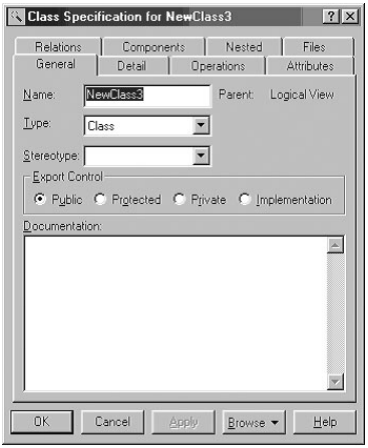

Рисунок 5.3. Окно спецификации класса.

5.5. Создание параметризованного класса

Параметризованный класс (parameterized class) — один из специальных типов классов. Он применяется для создания семейства других классов. Обычно параметризованный класс является разновидностью контейнера, его еще называют шаблоном.

Для добавления параметризованного класса нажмите кнопку ***Parameterized Class (Параметризованный класс)*** контекстной панели инструментов. Щелкните мышью где-нибудь на диаграмме, чтобы поместить туда новый класс. Введите имя класса. Или добавьте обычный класс на диаграмму или в браузер с помощью одного из описанных выше методов. Откройте окно спецификации класса. В поле ***Type (Тип)*** этого окна укажите ParameterizedClass. Нажмите ОК. Или выберите пункт ***Tools > Create > Parameterized Class (Инструменты > Создать > Параметризованный класс)*** в меню модели. Щелкните где-нибудь на диаграмме, чтобы поместить туда новый класс. Введите имя класса.

Аргументы класса указываются на вкладке Attributes. На основе аргументов создаются элементы стандартного класса.

Аргументом может быть другой класс, тип данных или выражение-константа. Вы можете задавать неограниченное количество аргументов.

Для добавления аргумента откройте окно спецификации класса. Перейдите на вкладку ***Detail (Подробно)***. Щелкните правой кнопкой мыши в поле ***Name*** области ***Formal Arguments (Формальные аргументы)***. В открывшемся меню выберите пункт ***Insert (Вставить)***. Введите имя аргумента. Щелкнув мышью, раскройте список ***Type (Тип)***. Выберите один из типов аргумента или

введите свой (рис. 5.4). Щелкните в поле ***Default Value (Значение по умолчанию)*** и введите значение аргумента по умолчанию. (Это делать не обязательно).

Для удаления аргумента откройте окно спецификации класса. Перейдите на вкладку ***Detail (Подробно)***. Щелкните правой кнопкой мыши на удаляемом аргументе. В открывшемся меню выберите пункт Delete.

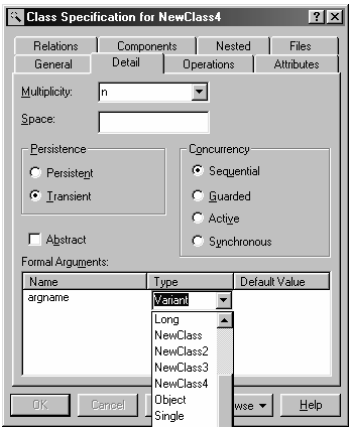

Рисунок 5.4. Задание аргументов параметризованного класса.

5.6. Создание класса-наполнителя

Класс-наполнитель (Instantiated Class) является параметризованным классом, аргументы которого имеют фактические значения. В соответствии с нотацией UML, имя аргумента класса-наполнителя заключается в угловые скобки (< >).

Для добавления класса-наполнителя нажмите кнопку ***Instantiated Class*** панели инструментов. Щелкните мышью на диаграмме там, где будет располагаться новый класс. Введите имя класса с аргументами, заключив их в угловые скобки (< >). Или добавьте класс на диаграмму классов или в браузер с помощью одного из описанных выше методов. Откройте окно спецификации класса. В поле Type (Тип) укажите InstantiatedClass. Нажмите OK. Или выберите пункт ***Tools > Create > Instantiated Class (Инструменты > Создать > Класс-наполнитель)*** в меню модели. Щелкните мышью где-нибудь на диаграмме, чтобы поместить туда новый класс и введите имя класса.

5.7. Создание утилиты, метакласса и класса-наполнителя

Утилита класса (class utility) — это совокупность операций. Например, в системе может быть совокупность математических функций (квадратный корень, кубический корень и т.д.), которые используются всей системой и не

слишком хорошо подходят для какого-либо конкретного класса. Эти функции можно собрать вместе и объединить в утилиту класса, которая будет использоваться другими классами системы.

Утилиты классов, как правило, применяют для расширения функциональных возможностей языка программирования или для хранения общих элементов функциональности многократного использования, необходимых в нескольких системах.

Утилитой параметризованного класса (parameterized class utility) является параметризованный класс, содержащий только набор операций. Это шаблон для создания утилит класса.

Утилитой класса наполнителя (instantiated class utility) называется утилита параметризованного класса, параметры которой имеют фактические значения.

Метакласс (metaclass) - это класс, экземпляры которого являются классами, а не объектами. К числу метаклассов относятся параметризованные классы и утилиты параметризованных классов.

Все элементы добавляются аналогично классам с использованием соответствующих кнопок контекстной панели.

5.8. Наименование класса

Каждому классу модели Rational Rose необходимо дать уникальное имя. Большинство организаций имеет собственные соглашения по именованию классов. В общем случае используются существительные в единственном числе. Не следует использовать существительные во множественном числе.

Обычно имена классов не содержат пробелов. Это делается по практическим причинам и из соображений удобочитаемости - языки программирования, как правило, не поддерживают пробелы в именах классов. Старайтесь, чтобы имена были относительно короткими.

Несмотря на то, что название ListOfEmployeesThatAreOnProbation (Список сотрудников, проходящих испытательный срок) хорошо описывает назначение класса, оно слишком сложное. Имя EmployeeList (Список сотрудников) будет в данном случае лучшим вариантом.

Используемый для именования классов регистр символов определяется обычно группой разработчиков. Если, например, класс соответствует списку пользователей, можно назвать его employeelist (хотя вообще-то так именуется экземпляры класса), EmployeeList (так - оптимально), Employeelist или EMPLOYEELIST. Таким образом, группа разработчиков может придерживаться своих соглашений по именованию. Важно чтобы принятый подход применялся ко всем классам модели.

Дать классу имя можно следующим образом. Выделите класс в браузере или на диаграмме классов. Введите его имя. Или откройте окно спецификации класса, введите имя в поле ***Name (Имя)***.

Для добавления к классу текстового описания выделите класс в браузере. Введите текст в окно документации. Или откройте окно спецификации класса. Введите информацию в области Documentation.

5.9. Назначение стереотипа класса

5.9.1. Общие сведения

Стереотип - это механизм, позволяющий категоризировать классы. Допустим, стоит задача найти все формы в модели. Для этого можно создать стереотип Form (Форма) и назначить его всем окнам приложения. В дальнейшем при поиске форм нужно только искать классы с этим стереотипом. На языке UML определены три основных стереотипа: ***Boundary (Граница)***, ***Entity (Объект)*** и ***Control (Управление)***. В среде Rational Rose их значительно больше.

5.9.2. Пограничные классы

Пограничными классами (Boundary Classes) называются такие классы, которые расположены на границе системы со всем остальным миром. Они включают в себя формы, отчеты, интерфейсы с аппаратурой (такой, как принтеры или сканеры) и интерфейсы с другими системами. Обозначение пограничного класса, принятое в UML приведено на рис 5.5.

Рисунок 5.5. Обозначение пограничного класса.

Для выявления пограничных классов необходимо исследовать диаграммы прецедентов. Для каждого взаимодействия между актером и прецедентом должен существовать хотя бы один пограничный класс (рис. 5.6). Именно он позволяет актеру взаимодействовать с системой.

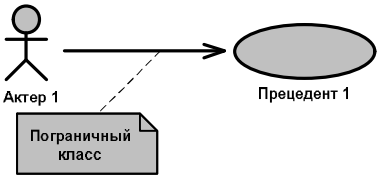

Рисунок 5.6. Актер, взаимодействующий с системой посредством пограничного класса.

Необязательно создавать уникальные пограничные классы для каждой пары "актер — прецедент". Например, если два актера инициируют один и тот же прецедент, они могут использовать общий пограничный класс для взаимодействия с системой (рис. 5.7).

Изучение диаграмм прецедентов помогает обнаружить пограничные классы.

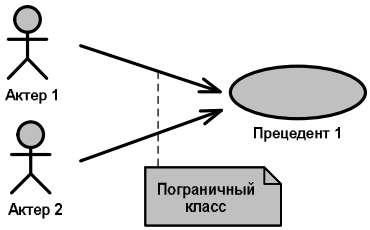

Рисунок 5.7. Два актера, взаимодействующие с системой посредством одного пограничного класса.

5.9.3. Классы-сущности

Классы-сущности (entity classes) содержат информацию, хранимую постоянно. Классы-сущности можно обнаружить в потоке событий и на диаграммах взаимодействия. Они имеют наибольшее значение для пользователя, и потому в их названиях часто применяют термины из предметной области. Пиктограмма класса-сущности приведена на рис. 5.8.

Рисунок 5.8. Обозначение класса-сущности.

Часто для каждого класса-сущности создают таблицу в базе данных. В этом заключается еще одно отличие от типичного подхода к конструированию системы. Вместо того чтобы с самого начала задавать структуру базы данных, теперь есть возможность разрабатывать ее на основе информации, получаемой из объектной модели. Это позволяет отслеживать соответствие всех полей базы данных и требований к системе. Требования определяют поток событий. Поток событий определяет объекты, классы и атрибуты классов. Каждый атрибут класса-сущности становится полем в базе данных. Применяя такой подход, проще отслеживать соответствие между полями базы данных и требованиями к системе, что уменьшает вероятность сбора ненужной информации.

5.9.4.Управляющие классы

Управляющие классы (control classes) отвечают за координацию действий других классов. Часто у прецедента имеется один управляющий класс, контролирующий последовательность событий этого прецедента.

Такой класс не несет в себе никакой функциональности - остальные классы посылают ему мало сообщений. Но сам он посылает множество сообщений. Управляющий класс делегирует ответственность другим классам. По этой причине управляющий класс часто называют классом-менеджером. Внешний вид управляющего класса приведен на рис. 5.9.

Рисунок 5.9. Обозначение управляющего класса.

В системе могут применяться управляющие классы, общие для нескольких прецедентов. Например, класс SecurityManager (Менеджер безопасности) может отвечать за контроль событий, связанных с безопасностью, а класс TransactionManager (Менеджер транзакций) - заниматься координацией сообщений, относящихся к транзакциям с базой данных. Могут быть и другие менеджеры для работы с такими элементами функционирования системы, как разделение ресурсов, распределенная обработка данных и обработка ошибок.

С помощью управляющих классов можно изолировать функциональность системы. Инкапсуляция в один класс, например, координации безопасности минимизирует последствия вносимых изменений. Любые изменения отвечающей за безопасность логики системы затронут только менеджера безопасности.

Помимо упомянутых выше стереотипов, вы можете создавать и свои собственные. Для этого нужно ввести новый стереотип в поле Stereotype, после чего он будет доступен в текущей модели Rational Rose. Разрешается добавлять стереотип для использования во всех моделях и даже создавать для него кнопки и пиктограммы панели инструментов.

Назначить стереотип классу можно следующим образом. Откройте окно спецификации класса. В раскрывающемся списке выберите нужный стереотип или введите его сами.

Для вывода имени стереотипа на диаграмму щёлкните правой кнопкой мыши на классе диаграммы классов. В открывшемся меню выберите пункт ***Options > Stereotype Display > Label (Параметры > Показ стереотипа > Метка)***. Над именем этого класса появится имя стереотипа, заключенное в двойные угловые скобки (« »).

Если нужно показать пиктограмму стереотипа на диаграмме, щелкните правой кнопкой мыши на классе диаграммы классов. В открывшемся меню выберите пункт ***Options > Stereotype Display > Icon (Параметры > Показ стереотипа > Пиктограмма)***. Представление класса на диаграмме изменится в соответствии с используемой пиктограммой.

Не у всех стереотипов есть пиктограммы. Если у стереотипа нет значка, появится только его имя.

Отключить показ стереотипов на диаграмме можно следующим образом. Щелкните правой кнопкой мыши на классе диаграммы классов. В открывшемся меню выберите пункт ***Options > Stereotype Display > None (Параметры > Показ стереотипа > Ничего)***. У класса по-прежнему будет стереотип, видимый в окне спецификации, но не отображаемый на диаграмме.

В среде Rational Rose существует возможность назначать собственные стереотипы и определять для них пиктограммы, посредством модификации *.ini файла.

5.10. Задание видимости класса

Параметр ***Visibility (Видимость)*** показывает, будет ли класс виден вне своего пакета. Можно указать для класса одно из возможных значений.

Public (Открытый). Этот класс виден всем остальным классам системы.

Protected, Private (Защищенный, закрытый). Класс может быть виден во вложенных в него классах, "друзьям" (friends) этого класса и из самого класса.

Package or Implementation (Пакет или реализация). Класс может быть виден только из классов того же пакета.

Для установки видимости класса установите параметр ***Export control (Контроль экспорта)*** в значение Public, Protected, Private или Implementation.

5.11. Задание множественности класса

Поле ***Multiplicity (Множественность)*** позволяет указать, сколько у данного класса должно быть экземпляров.

Множественность управляющего класса обычно равна 1. Например, во время работы приложения, скорее всего понадобится только один экземпляр менеджера безопасности.

Для задания множественности класса откройте окно спецификации класса. Перейдите на вкладку Detail (Подробно). Укажите множественность в раскрывающемся списке или введите ее значение.

5.12. Задание требований к хранению класса

Можно указать количество абсолютной или относительной памяти, которая, по мнению разработчика, потребуется для каждого объекта класса. Для этой цели служит поле ***Space (Пространство)*** окна спецификации класса.

Это поле не применимо для утилит классов, утилит классов-наполнителей и утилит параметризованных классов.

Для указания пространства класса откройте окно спецификации класса. Перейдите на вкладку Detail. В поле Space введите требования по хранению класса.

5.13. Задание устойчивости класса

В среде Rational Rose на основе модели можно генерировать DDL (Data Definition Language - Язык Описания Данных). DDL определяет структуру базы данных.

При генерации DDL приложение Rational Rose ищет устойчивые (persistent) классы. Поле Persistence окна спецификации класса применяется для определения этого параметра. Он может принимать одно из следующих значений:

- ***Persistent (Устойчивый).*** Класс сохраняется и после завершения работы приложения. Иначе говоря, содержащаяся в объектах класса информация будет сохраняться в базе данных или каким-то другим способом, обеспечивающим длительное хранение.
- ***Transient (Временный).*** Информация, содержащаяся в объектах класса, не будет сохраняться после завершения работы приложения.

Это поле нельзя использовать для утилит классов, утилит параметризованных классов и утилит классов-наполнителей.

Задать устойчивость класса можно следующим образом. Откройте окно спецификации класса. Перейдите на вкладку Detail (Подробно). В области Persistence выберите пункт Persistent или Transient.

5.14. Задание параллелизма класса

Параллелизм (concurrency) позволяет описать, как будет вести себя класс в присутствии нескольких потоков управления. Для класса доступны четыре значения этого параметра:

- ***Sequential (Последовательный).*** Это значение по умолчанию, оно показывает, что класс будет вести себя нормально (т.е. операции будут выполняться так, как ожидается) при наличии только одного потока управления, но в присутствии нескольких потоков управления поведение класса не гарантируется.
- ***Guarded (Ограждающий).*** При наличии нескольких потоков управления класс будет вести себя, как ожидается, но чтобы классы различных потоков не мешали друг другу, они должны взаимодействовать друг с другом.
- ***Active (Активный).*** Класс будет иметь свой собственный поток управления.
- ***Synchronous (Синхронный).*** При наличии нескольких потоков управления класс будет вести себя, как ожидается. Между ним и классами других потоков не требуется какого-то специального взаимодействия, так как класс может самостоятельно обрабатывать взаимные исключения.

Для указания параллелизма класса откройте окно его спецификации. Перейдите на вкладку Detail (Подробно). Установите переключатель Concurrency в значение, соответствующее требуемому параллелизму.

5.15. Создание абстрактного класса

Абстрактным называется класс, который не наполняется конкретным содержанием (не инстанцируется). Иными словами, если класс А - абстрактный, в памяти никогда не будет объектов типа А.

Обычно абстрактные классы применяют при работе с наследованием. В них содержатся данные и поведение, общие для нескольких других классов.

На языке UML название абстрактного класса на диаграмме пишут курсивом. Создайте обычный класс одним из описанных выше способов. Откройте окно спецификации класса. Перейдите на вкладку Detail (Подробно). Установите флажок Abstract.

5.16. Просмотр атрибутов класса

Окно спецификации класса содержит информацию о том, какие атрибуты класса уже были созданы. Если нужно просмотреть атрибуты класса откройте окно спецификации класса. Перейдите на вкладку Attributes (Атрибуты). Здесь перечисляются атрибуты класса с указанием их видимости, стереотипа, имени, типа данных и значения по умолчанию.

5.17. Использование вложенных классов

В Rational Rose классы можно вкладывать друг в друга. Во вложенные (nested) классы можно вкладывать другие классы, организуя столько уровней вложения, сколько необходимо.

Для создания вложенного класса откройте окно спецификации родительского класса. Перейдите на вкладку ***Nested (Вложенные)***. Щелкните правой кнопкой мыши на белом поле этой вкладки. В открывшемся меню выберите пункт Insert (Вставить). Введите имя вложенного класса.

Если нужно показать вложенный класс на диаграмме классов откройте диаграмму классов. Выберите в меню модели пункт ***Query > Add Classes (Запрос > Добавить классы)***. Перетащите вложенный класс из списка Classes (Классы) в список Selected Classes (Выбранные классы). Вложенный класс представлен в формате: Родительский класс :: Вложенный класс. Щелкните мышью на OK. Вложенный класс появится на диаграмме с именем родительского класса, заключенным в скобки.

Для удаления из модели вложенного класса откройте окно спецификации родительского класса. Перейдите на вкладку Nested (Вложенные). Щелкните правой кнопкой мыши на имени вложенного класса, который нужно удалить. В открывшемся меню выберите пункт Delete (Удалить). Вложенный класс будет удален со всех диаграмм классов.

5.18. Связывание файлов и ссылок с классом

Файлы и ссылки можно прикреплять к диаграмме классов, а также связывать непосредственно с классом. Например, к классу модели можно прикрепить файл с его исходным кодом или файл с описанием действий по тестированию его функциональных возможностей. В Rational Rose связывание файлов и ссылок с классом выполняется с помощью браузера или окна спецификации класса. Техника прикрепления файла и сопоставления ссылки подробно описана в пункте 1.6.

5.19. Просмотр диаграмм взаимодействия

При необходимости изменить класс полезно точно знать, где в системе он используется. Два типа диаграмм взаимодействия — диаграммы последовательности и диаграммы кооперации - позволяют понять, где и как применяется класс. Чтобы узнать, какие диаграммы последовательности и кооперации содержат объекты данного класса, можно воспользоваться меню ***Report (Отчет)***. Для просмотра всех диаграмм взаимодействия, содержащих определенный класс, укажите класс на диаграмме классов. В меню модели выберите пункт ***Report > Show Instances (Отчет > Показать экземпляры)***. Среда покажет список всех диаграмм последовательности и кооперации, содержащих экземпляры данного класса. Чтобы открыть диаграмму, дважды щелкните на ней в списке или нажмите кнопку ***Browse (Обзор)***.

5.20. Работа с примечаниями

На диаграмме классов можно размещать примечания, содержащие дополнительную информацию о конкретном классе, пакете, атрибуте, операции или отношении. Хотя примечания не влияют на генерируемый код, они помогают разработчикам и другим участникам проекта лучше понять модель. Примечания располагаются и удаляются на данном типе диаграмм способом, аналогичным рассмотренному при изучении диаграмм других типов.

5.21. Работа с пакетами

5.21.1. Общие сведения

Пакеты (packages) применяются для группирования классов, обладающих некоторой общностью. Объединять классы можно, как угодно, однако существует несколько наиболее распространенных подходов.

Во-первых, можно группировать классы по стереотипу. В таком случае получается один пакет с классами-сущностями, один с пограничными классами, один с управляющими классами и т.д. Этот подход может быть полезен с точки зрения размещения готовой системы, поскольку все находящиеся на клиентских машинах пограничные классы уже оказываются в одном пакете.

Второй подход заключается в объединении классов по их функциональности. Например, в пакете Security (Безопасность) будут содержаться все классы, отвечающие за безопасность приложения. Другие пакеты могут называться Employee Maintenance (Работа с сотрудниками). Reporting (Подготовка отчетов) и Error Handling (Обработка ошибок). Преимущество этого метода заключается в возможности повторного использования пакетов. Если внимательно подойти к группированию классов, можно получить практически не зависящие друг от друга пакеты. Например, пакет Security можно использовать и в других приложениях.

Наконец, применяют комбинацию двух указанных подходов. Для дальнейшей организации классов разрешается вкладывать пакеты друг в друга. На высоком уровне, например, можно сгруппировать классы по функционально-

сти, создав пакет Security. Внутри него можно создать другие пакеты, сгруппировав отвечающие за безопасность классы по функциональности или по стереотипу.

5.21.2. Добавление пакетов

Очередным этапом разработки модели является добавление пакетов. Пакеты классов создают в логическом представлении браузера.

Для добавления на диаграмму классов существующего пакета перетащите пакет на диаграмму из браузера.

Поместить на диаграмму классов новый пакет можно следующим образом. Нажмите кнопку Package (Пакет) панели инструментов. Щелкните мышью внутри диаграммы Классов, чтобы поместить туда пакет. Введите его имя.

Для добавления пакета в браузер щелкните правой кнопкой мыши на логическом представлении браузера. Если вы создаете пакет внутри существующего пакета, щелкните правой кнопкой мыши на нем в браузере. Выберите пункт меню модели ***New > Package (Создать > Пакет)***. Введите имя нового пакета. Если нужно поместить в пакет элемент, перетащите элемент в этот пакет в браузере.

5.21.3. Удаление пакетов

Пакет можно удалить только с диаграммы классов или из модели в целом. При удалении пакета из модели удаляется и все его содержимое. Для удаления пакета с диаграммы выделите его на диаграмме классов. Нажмите клавишу Delete. Обратите внимание, что, хотя пакет и исчез с диаграммы классов, он сохранился в браузере и на других диаграммах.

Для удаления пакета из модели щелкните правой кнопкой мыши на пакете в браузере. В открывшемся меню выберите пункт ***Delete (Удалить)***. Или выделите пакет на диаграмме классов. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш CTRL+D.

Будьте внимательны: при удалении пакета из модели все классы и диаграммы пакета также будут удалены.

5.22. Работа с атрибутами

5.22.1. Добавление атрибутов

Атрибут — это фрагмент информации, связанный с классом. Rational Rose предоставляет возможность добавлять атрибуты к классам модели. С атрибутами можно связать три основных фрагмента информации: имя атрибута, тип его данных и первоначальное значение. Имя и тип атрибута должны быть определены перед генерацией кода, первоначальное значение задавать необязательно.

Добавление атрибута выполняется непосредственно на диаграмме классов, в браузере или в окне спецификации класса.

С атрибутом можно связать некоторое описание. Как правило, это короткое описание или определение атрибута. В генерируемый код оно войдет в качестве комментария. Таким образом, путем документирования атрибута документируются фрагменты кода.

Для добавления атрибута к классу щелкните правой кнопкой мыши на классе диаграммы. В открывшемся меню выберите пункт ***New Attribute (Новый Атрибут)***. Введите имя атрибута в формате Имя : Тип данных = Начальное значение. Тип данных нужен для генерации кода. Начальное значение необязательно. Если необходимо еще добавить атрибутов, нажмите клавишу Enter и введите новые атрибуты непосредственно на диаграмму классов. Или щелкните правой кнопкой мыши на классе в браузере. В открывшемся меню выберите пункт ***New > Attribute (Создать > Атрибут)***. Под классом в браузере появится новый атрибут с именем ***name (имя)***. Введите имя этого атрибута. Или откройте окно спецификации класса (рис. 5.10). Перейдите на вкладку ***Attributes (Атрибуты)***. Если у класса уже имеются атрибуты, они будут перечислены на этой вкладке. Щелкните правой кнопкой мыши где-нибудь внутри области атрибутов. В открывшемся меню выберите пункт Insert (Вставить). Введите имя нового атрибута. Задайте видимость, стереотип, тип данных и значение по умолчанию в соответствующих колонках.

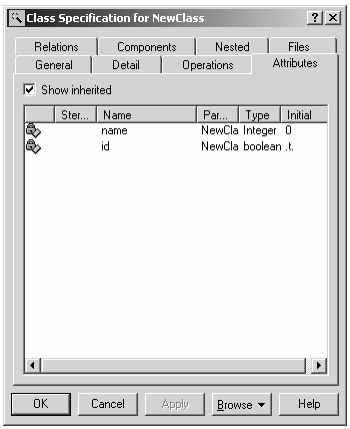

Рисунок 5.10. Добавление атрибута в окне спецификации класса.

Для добавления к атрибуту текстового описания выделите атрибут в браузере или на диаграмме классов. Введите описание атрибута в окне документации. Или щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. Введите описание атрибута в области документации окна спецификации атрибута класса.

5.22.2 Удаление атрибутов

В процессе работы может потребоваться удалить ранее созданные атрибуты. Например, часто при изменении требований к системе пропадает необходимость в конкретном атрибуте. В среде Rational Rose это легче делать в браузере. Можно также использовать диаграмму классов. При удалении атрибута с диаграммы классов он будет автоматически удален со всех остальных диаграмм классов и из модели. Для удаления атрибута класса щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт Delete (Удалить). Или выделите атрибут на диаграмме классов. С помощью клавиши Backspace удалите с диаграммы имя атрибута, тип данных и начальное значение. Щелкните мышью где-нибудь на диаграмме. Rational Rose подтвердит удаление атрибута перед тем, как завершить это действие.

5.22.3. Спецификация атрибута

Спецификация атрибута включает в себя, помимо прочего, тип данных, значение по умолчанию, стереотип и видимость атрибута. Все спецификации можно просматривать или изменять в окне спецификации атрибута, показанном на рис. 5.11.

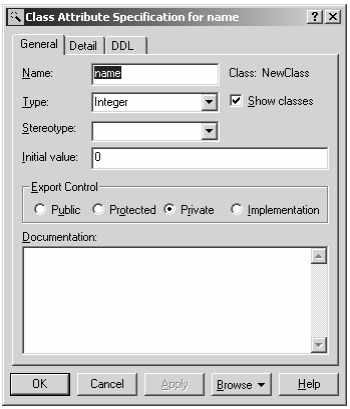

Рисунок 5.11. Окно спецификации атрибута.

Открыть окно спецификации атрибута можно следующим образом. Щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

Одной из главных характеристик атрибута является тип данных. Он специфичен для используемого языка. Это может быть, например, тип string, integer, long или boolean. Перед началом генерации кода необходимо указать тип данных каждого атрибута.

В качестве типов данных можно использовать либо встроенные типы языка программирования (string, integer, long и т.д.), либо определенные в вашей модели имена классов. Для того чтобы имена определенных в модели

классов выводились в раскрывающемся списке типов данных атрибутов, установите флажок Show Classes (Показать классы).

Для задания типа данных атрибута щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). Появится окно спецификации атрибута класса. Укажите тип данных в раскрывающемся списке типов или введите собственный. Или выделите атрибут на диаграмме классов. После имени атрибута введите двоеточие и тип его данных.

Подобно актерам, прецедентам и классам, атрибут может характеризоваться стереотипом. Стереотип атрибута является способом его классификации. Например, некоторые атрибуты могут соответствовать полям базы данных, а другие нет. Для каждого такого типа можно определить свой, стереотип.

В Rational Rose необязательно назначать стереотипы атрибутам. Стереотипы не требуются для генерации кода, но при их использовании легче читать и понимать модель.

Для назначения стереотипа атрибуту щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). Откроется окно спецификации атрибута класса. Укажите стереотип в раскрывающемся списке или введите новый стереотип. Или выделите атрибут в браузере. Для того чтобы отредактировать имя атрибута, щелкните на нем один раз. Перед именем появятся символы “<< >>”. Введите внутри угловых скобок имя стереотипа.

5.22.4. Задание начальных значений атрибута

Атрибуты могут иметь значения по умолчанию. Для генерации кода, как и в случае стереотипов, задавать начальные значения необязательно. Тем не менее, при их наличии генерируемый код будет соответствующим образом инициализировать атрибут. Для задания начального значения атрибута щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). В поле Initial Value (Начальное значение) введите значение атрибута по умолчанию. Или выделите атрибут на диаграмме классов. После типа данных атрибута введите знак равенства, а затем значение по умолчанию.

5.22.5. Задание видимости атрибута

Одной из центральных концепций объектно-ориентированного программирования является инкапсуляция. Благодаря наличию атрибутов и операций, каждый класс инкапсулирует некоторое количество данных и поведения. К преимуществам такого подхода относится возможность создания небольших самодостаточных фрагментов кода.

Так как атрибуты содержатся внутри класса, они скрыты от других классов. В связи с этим нужно указать, какие классы имеют право читать и изме-

нять атрибуты. Это свойство называется видимостью атрибута. Допустимы четыре значения этого параметра.

Public (Общий, открытый). Атрибут виден всем остальным классам. Любой класс может просмотреть или изменить значение атрибута. В соответствии с нотацией UML общему атрибуту предшествует знак "+".

Private (Закрытый). Атрибут не виден другим классам. В соответствии с нотацией UML закрытый атрибут обозначается знаком "-".

Protected (Защищенный). Атрибут доступен только самому классу и его потомкам. Нотация UML для защищенного атрибута предполагает использование знака "#".

Package or Implementation (Пакетный). Атрибут является общим, но только в пределах своего пакета. Данный тип видимости не обозначается никаким специальным значком.

В общем случае атрибуты рекомендуется делать закрытыми или защищенными. Это позволяет лучше контролировать сам атрибут и код. При использовании закрытых или защищенных атрибутов удается избежать ситуации, когда значение атрибута изменяется всеми классами системы. Вместо этого логика изменения атрибута будет заключена в том же классе, что и сам атрибут. Задаваемые параметры видимости влияют на генерируемый код.

В среде Rational Rose поддерживаются два набора нотаций видимости, приведенные в таблице 5.2. Первый включает в себя четыре значка Rational Rose. Второй - нотация UML (+,-,#), для общих, закрытых и защищенных атрибутов соответственно.

Таблица 5.2.

Нотация Rational Rose	**Нотация UML**	**Описание**
	**“+”**	Public – доступен всем
	“-“	Private – доступен только классу, в котором определен
	**“#”**	Protected – доступен только классу, в котором определен и его потомкам
	Нет обозначения	Package or Implementation - доступен для классов в одном пакете

На диаграмме классов разрешается применять любую из этих нотаций. Ниже описывается возможность переключения между ними. На рис. 5.12 приведен пример класса, видимость атрибутов которого соответствует нотации UML.

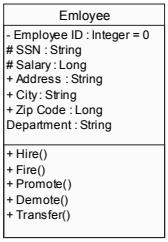

Рисунок 5.12. Пример класса, видимость атрибутов которого выполнена в нотации UML.

На рис. 5.13. показан тот же класс, но уже в соответствии с |нотацией Rational Rose.

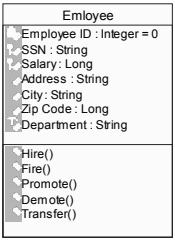

Рисунок 5.13. Пример класса, видимость атрибутов которого выполнена в нотации Rational Rose.

Для задания значения видимости атрибута щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт ***Open Specification (Открыть спецификацию)***. Появится окно спецификации атрибута класса. В поле ***Export Control (Контроль экспорта)*** выберите видимость атрибута: Public, Protected, Private или Implementation. По умолчанию видимость всех атрибутов установлена в Private. Или выделите атрибут на диаграмме классов. Если для обозначения видимости вы используете нотацию UML, щелкните мышью на значке "+", "-" или "#" рядом с атрибутом. В появившемся списке выберите значение видимости. Если для обозначения видимости используется нотация Rational Rose, щелкните мышью на значке видимости слева от имени атрибута. В появившемся списке значков выберите требуемую видимость.

Изменить нотацию для обозначения видимости можно следующим образом. В меню модели выберите пункт ***Tools > Options (Инструменты > Параметры)***. Перейдите на вкладку Notation (Нотация). Установите флажок Visibility as icons (Отображать пиктограммы) для использования нотации Rational Rose или сбросьте его для применения нотации UML.

Изменение значения этого параметра приведет к смене нотации только для новых диаграмм и не затронет уже существующие диаграммы. Чтобы сменить нотацию для существующей диаграммы, необходимо закрыть, а затем снова загрузить в инструментальную среду файл модели.

5.22.6. Задание метода локализации атрибута

Метод локализации атрибута (containment) показывает, каким образом атрибут хранится в классе. Возможны три значения этого параметра.

By value (По значению). Предполагается, что атрибут содержится внутри класса. Например, если атрибут относится к типу string, эта строка будет содержаться внутри определения класса.

By reference (По ссылке). Предполагается, что атрибут локализован вне класса, но класс содержит указатель на него. Например, у класса Timecard (Карточка табельного учета) может быть атрибут типа Employee (Сотрудник). Сам объект employee размещен вне объекта timecard. Таким образом, этот атрибут является указателем на внешний объект employee.

Unspecified (Не определен). Метод локализации атрибута еще не определен. В этом случае при генерации кода по умолчанию применяется значение By value этого параметра.

Задать метод локализации атрибута можно следующим образом. Щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию) или Open Standard Specification (Открыть стандартную спецификацию), если вы работаете с загруженным мастером каркаса. Появится окно спецификации атрибута класса. Перейдите на вкладку Detail (Подробно). Укажите значение метода локализации атрибута (containment): By value, By reference или Unspecified. Значение этого параметра по умолчанию - Unspecified.

5.22.7. Определение статичного атрибута

При добавлении атрибута к классу каждый экземпляр класса получит свою собственную копию этого атрибута. Статичный атрибут (static) - это такой атрибут, который используется всеми экземплярами класса.

На языке UML статичный атрибут помечают символом "$". Сделать атрибут статичным можно следующим образом. Щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию) или Open Standard Specification (Открыть стандартную спецификацию), если вы загружали мастер каркаса. Появится окно спецификации атрибута класса. Перейдите на вкладку Detail (Подробно). Установите флажок Static, чтобы сделать атрибут статичным. Перед именем атрибута на диаграмме классов появится символ "$".

5.22.8. Определение производного атрибута

Производным (derived) называется атрибут, созданный из одного или нескольких других атрибутов. В UML производные атрибуты помечают символом"/". Сделать атрибут производным можно следующим образом. Щелкните правой кнопкой мыши на атрибуте в браузере. В открывшемся меню выберите пункт Open Specification (Открыть спецификаций) или Open Standard Specification (Открыть стандартную спецификацию), если вы используете мастер каркаса. Появится окно спецификации атрибута класса. Перейдите на вкладку Detail (Подробно). Установите флажок Derived (Производный). Перед именем атрибута на диаграмме классов появится символ "/".

5.23. Работа с операциями

5.23.1. Общие сведения

Операцией называется связанное с классом поведение. Операция состоит из трех частей: имени, параметров и типа возвращаемого значения. Параметры - это аргументы, получаемые операцией "на входе". Тип возвращаемого значения относится к результату действия операции.

На диаграмме классов можно показывать как имена операций, так и их параметры и типы возвращаемого значения. На некоторых диаграммах полезно показывать полную сигнатуру операций. Если же нужно упростить диаграмму, лучше оставить только имена.

В языке UML операции имеют следующую нотацию:

Имя операции (аргумент1 : тип данных аргумента1, аргумент2 : тип данных аргумента2, ...) : тип возвращаемого значения.

Создав диаграммы последовательности и диаграммы кооперации, вы проделаете большую часть работы, требуемой для выявления операций. Операции классифицируются по 4 основным типам, описанным ниже.

5.23.2. Операции реализации

Операции реализации (implementor operations) реализуют некоторую бизнес-функциональность. Такие операции можно найти, исследуя диаграммы взаимодействия. Диаграммы этого типа фокусируются на бизнес-функциональности, и каждое сообщение диаграммы скорее всего можно соотнести с операцией реализации.

Необходимо, чтобы каждую операцию реализации можно было проследить до соответствующего требования. Это достигается на различных этапах моделирования. Операция выводится из сообщения на диаграмме взаимодействия, сообщения выделяются из подробного описания потока событий, который создается на основе прецедента, а последний - на основе требований. Возможность проследить всю эту цепочку гарантирует, что каждое требование будет воплощено в коде, а каждый фрагмент кода реализует какое-то требование.

5.23.3. Операции управления

Операции управления (manager operations) управляют созданием и разрушением объектов. В эту категорию попадают конструкторы и деструкторы классов. В среде Rational Rose не требуется вручную создавать конструкторы и деструкторы классов. При генерации кода среда предоставит возможность сделать это автоматически.

5.23.4. Операции доступа

Атрибуты обычно бывают закрытыми или защищенными. Тем не менее, другие классы иногда должны просматривать или изменять их значения. Для этого предназначены ***операции доступа (access operations)***. Такой подход дает возможность безопасно инкапсулировать атрибуты внутри класса, защитив их от других классов, но все же позволяет осуществлять контролируемый доступ к ним.

Создание операций Get и Set (получения и изменения значения) для каждого атрибута класса является промышленным стандартом. Как и в случае операций управления, операции доступа не нужно вводить вручную. При генерации кода Rational Rose автоматически создаст операции Get и Set для каждого атрибута класса.

5.23.5. Вспомогательные операции

Вспомогательными (helper operations) называются такие операции класса, которые необходимы ему для выполнения его ответственностей, но о которых другие классы не должны ничего знать. Это закрытые и защищенные операции класса.

Как и операции реализации, вспомогательные операции можно обнаружить на диаграммах последовательности и кооперации. Часто такие операции являются рефлексивными сообщениями.

Для идентификации операций выполните следующие действия. Изучите диаграммы последовательности и кооперации. Большая часть сообщений на этих диаграммах является операциями реализации. Рефлексивные сообщения будут вспомогательными операциями. Рассмотрите управляющие операции. Возможно, требуется добавить конструкторы и деструкторы. Это необязательно – Rational Rose может сделать это самостоятельно при генерации кода.

Рассмотрите операции доступа. Для каждого атрибута класса, с которым будут работать другие классы, необходимо создать операции Get и Set. Как и в случае управляющих операций, это необязательно делать вручную – Rational Rose может создать указанные операции автоматически.

5.23.6. Добавление операций

Как и атрибуты, операции можно добавить в модель Rational Rose на диаграмму классов или в браузер. Можно также воспользоваться окном спецификации класса.

После создания операции с ней можно связать какое-либо текстовое описание. Оно будет включено в генерируемый код в качестве комментария. В описании операции обычно указывается ее назначение, параметры и тип возвращаемого значения.

Для добавления операции к классу щелкните правой кнопкой мыши на классе диаграммы классов. В открывшемся меню выберите пункт ***New > Operation (Создать > Операция)***. Введите имя операции в формате Имя (Аргумент1 : Тип данных аргумента1) : Тип возвращаемого значения. Если необходимо еще добавить операции, нажмите клавишу Enter и введите новые операции непосредственно на диаграмму классов. Или щелкните правой кнопкой мыши на классе в браузере. В открывшемся меню выберите пункт ***New > Operation (Создать > Операция)***. Под названием этого класса в браузере появится новая операция opname. Введите ее имя. В браузере нельзя указывать аргументы операции и тип возвращаемого значения; как и в случае атрибутов, это делается на диаграмме классов. Или откройте окно спецификации класса данной операции (или стандартное окно спецификации). Перейдите на вкладку Operations (Операции). Здесь будут перечислены уже имеющиеся операции класса. Щелкните правой кнопкой мыши где-нибудь внутри области операций, как показано на рис. 5.14.

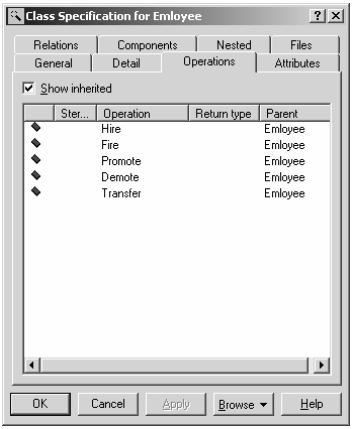

Рисунок 5.14. Добавление новой операции в окно спецификации класса.

В открывшемся меню выберите пункт Insert (Вставить). Введите имя новой операции в колонке Operation. В соответствующих колонках задайте видимость, стереотип и тип возвращаемого значения операции.

Текстовое описание для операции выполняется подобно текстовому описанию атрибутов.

5.23.7. Удаление операций

Удалить операцию можно с диаграммы классов или из браузера. При удалении с диаграммы операция автоматически удаляется из модели в целом.

При удалении операции следите за тем, чтобы модель оставалась согласованной. Возможно, вы использовали операцию на диаграммах последовательности и кооперации. При удалении она автоматически преобразуется в сообщение на этих диаграммах. Следовательно, нужно обновить соответствующим образом диаграммы последовательности и кооперации.

Если требуется определить, какие диаграммы используют операцию откройте окно спецификации класса этой операции (или стандартное окно спецификации). В нижней части окна нажмите кнопку Browse (Обзор) и выберите Show Usage (Показать использование).

Для удаления операции класса щелкните правой кнопкой мыши на операции в браузере. В открывшемся меню выберите пункт Delete (Удалить). Или выделите операцию на диаграмме классов. С помощью клавиши Backspace сотрите имя операции и ее сигнатуру. Щелкните мышью где-нибудь в другом месте диаграммы. Rational Rose подтвердит удаление перед завершением этой процедуры. Или откройте окно спецификации класса данной операции (или стандартное окно спецификации). Перейдите на вкладку Operations (Операции). Щелкните правой кнопкой на удаляемой операции. В открывшемся меню выберите пункт Delete (Удалить). Rational Rose подтвердит удаление перед завершением этой процедуры.

5.23.8. Спецификация операции

В спецификации операции можно задать ее параметры, тип возвращаемого значения и видимость. Все спецификации можно просмотреть и изменить в окне спецификации операции.

Открыть спецификацию операции можно следующим образом. Щелкните правой кнопкой мыши на операции в браузере. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию). Или откройте окно спецификации класса операции. Перейдите на вкладку Operations (Операции). Дважды щелкните мышью на соответствующей операции.

5.23.9. Задание возвращаемого класса операции

Возвращаемым классом (return class) операции называется тип данных ее результата. При определении возвращаемого класса можно использовать

либо встроенные типы языка программирования (такие, как string, char или integer) либо определенные в модели классы.

Для задания возвращаемого класса операции щелкните правой кнопкой мыши на операции в браузере. Откройте окно спецификации класса этой операции (или стандартное окно спецификации). Укажите возвращаемый класс в раскрывающемся списке или введите свой тип. Или выделите операцию на диаграмме классов. После имени операции введите двоеточие, а затем тип возвращаемого значения.

5.23.10. Назначение стереотипа для операции

Как и в случае других элементов модели, для классификации операций создаются их стереотипы. Как упоминалось ранее, существуют четыре наиболее распространенных стереотипа операций.

Implementor (Реализация). Операции, реализующие некоторую бизнес-логику.

Manager (Управляющая). Конструкторы, деструкторы и операции управления памятью.

Access (Доступ). Операции, позволяющие другим классам просматривать или редактировать атрибуты данного класса. Как правило, такие операции называют Get<имя атрибута> или 8et<имя атрибута>

Helper (Вспомогательная). Закрытые или защищенные операции, которые используются классом, но не видны другим классам.

Назначение операциям стереотипов не требуется для генерации кода. Тем не менее, они облегчают понимание модели. Кроме того, они помогают убедиться в том, что ни одна операция не была пропущена.

Для назначения стереотипа операции щелкните правой кнопкой мыши на операции в браузере. Откройте окно спецификации класса этой операции (или стандартное окно спецификации). В соответствующем раскрывающемся списке укажите стереотип или введите новый. Или выделите операцию в браузере. Чтобы отредактировать имя операции, один раз щелкните на ней мышью. Перед именем появятся символы "<< >>". Внутри этих скобок введите стереотип.

5.23.11. Задание видимости операции

Как уже упоминалось выше, видимость показывает, каким образом данные и поведение инкапсулируются в класс. Для операций допустимы четыре значения этого параметра (их визуальное представление аналогично представлению видимости атрибутов).

Public (Общая). Операция доступна всем остальным классам. Любой класс может запросить ее выполнение. ***Private (Закрытая).*** Операция не доступна ни одному другому классу. ***Protected (Защищенная).*** Доступ к операции разрешен только для самого класса и его потомков. ***Package or Implementation (Пакетная).*** Операция доступна только классам данного пакета.

В то время как атрибуты обычно бывают закрытыми или защищенными, операции могут быть общими, закрытыми, защищенными или пакетными.

Принимая решение по поводу видимости операции, подумайте, какие классы должны знать о ней. При генерации кода Rational Rose учтет установленную видимость.

Как говорилось ранее, на диаграммах классов разрешается использовать стандартную нотацию UML или собственную нотацию Rational Rose. Допускается переключение между ними.

Для операции область видимости назначается способом аналогичным назначению видимости атрибутов.

5.23.12. Добавление аргументов к операции

Аргументы, или параметры, операции - это получаемые ею входные данные. Для каждого аргумента должны быть заданы имя и тип данных. На диаграмме классов аргументы и их типы указываются в скобках после имени операции. При желании для аргументов можно задавать также их значения по умолчанию. В таком случае нотация UML будет иметь вид, приведенный на рис. 5.15.

Имя операции (аргумент1 : тип данных аргумента1 = значение по умолчанию аргумента1) : тип возвращаемого значения операции.

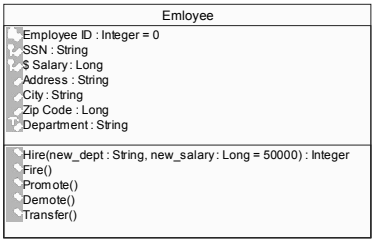

Рисунок 5.15. Добавление аргументов к операции.

При генерации кода Rational Rose будет генерировать имя операции, ее аргументы, их типы данных и значения по умолчанию, а также тип возвращаемого значения. Если к операции было добавлено текстовое описание, Rational Rose создает комментарии. Для добавления аргумента к операции откройте окно спецификации операции (или стандартное окно спецификации). Перейдите на вкладку Detail (Подробно). Щелкните правой кнопкой мыши в области аргументов. В открывшемся меню выберите Insert (Вставить). Введите имя аргумента. Щелкните мышью на колонке Type (Тип) и введите тип данных аргумента. При необходимости щелкните на колонке Default (По умолчанию) и введите значение аргумента по умолчанию.

Для удаления аргумента операции откройте окно спецификации операции (или стандартное окно спецификации). Перейдите на вкладку Detail (Подробно). В списке аргументов щелкните правой кнопкой мыши на удаляемом

аргументе. В открывшемся меню выберите пункт Delete (Удалить). Подтвердите удаление.

5.23.13. Определение протокола операции

Протокол операции описывает, какие операции и в каком порядке может выполнять клиент над объектом. Например, если операцию А нельзя запускать до завершения операции Б, это можно отметить в поле протокола операции А.

Вводимая таким образом информация будет включена в генерируемый код в качестве комментария, но не повлияет на сам код. Экран протокола операции представлен на рис. 5. 16.

Для задания протокола операции откройте окно спецификации операции (или стандартное окно спецификации). Перейдите на вкладку Detail (Подробно). Введите протокол в поле Protocol.

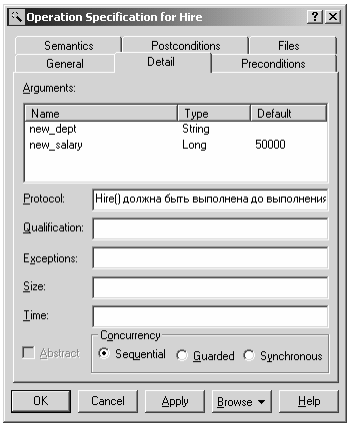

Рисунок 5.15. Добавление новой операции в окно спецификации.

5.23.14. Определение уточнений операции

Поле Qualification (Уточнение) позволяет идентифицировать любые уточнения операции, связанные с конкретным языком программирования. Все, что вы здесь введете, войдет в генерируемый код в качестве комментария, но на сам код не повлияет. Для ввода уточнения операции откройте окно спецификации операции (или стандартное окно спецификации). Перейдите на вкладку Detail (Подробно). Введите уточнения в поле Qualification.

5.23.15. Задание исключительных ситуаций операции

В поле Exceptions (Исключительные ситуации) можно перечислить исключительные ситуации для данной операции. Эта информация войдет в качестве комментария в генерируемый код, но на сам код не повлияет.

5.23.16. Определение размера операции

В поле Size можно указать предполагаемый объем памяти, требуемой операции во время ее выполнения. Эта информация войдет в качестве комментария в генерируемый код.

5.23.17. Задание времени выполнения операции

Время операции - это предполагаемое время, требуемое для ее выполнения. Указываемая информация войдет в сгенерированный код в качестве комментария. Для задания времени операции откройте окно спецификации операции (или стандартное окно спецификации). Перейдите на вкладку Detail (Подробно). Введите время в поле Time.

5.23.18. Задание параллелизма операции

Значение переключателя Concurrency (Параллелизм) определяет поведение операции при наличии нескольких потоков управления. Возможны три значения этого параметра:

Sequential (Последовательная). Предполагается, что операция будет выполняться правильно при наличии только одного потока управления. Выполнение операции должно завершиться перед тем, как начнется выполнение другой операции.

Guarded (Охраняемая). Предполагается, что операция будет выполняться правильно в нескольких потоках управления, но только если взаимодействие классов гарантирует взаимное исключение выполняемых операций.

Synchronous (Синхронная). Предполагается, что операция будет правильно выполняться при наличии нескольких потоков. После обращения операция будет выполняться в одном потоке управления вплоть до своего завершения.

Другие операции могут в то же самое время выполняться в других потоках. Класс должен сам позаботиться о решении проблем взаимных исключений, так что взаимодействие с другими классами не требуется.

Сведения о параллелизме операции появятся в сгенерированном коде в качестве комментариев. Для задания параллелизма операции откройте окно спецификации операции (или стандартное окно спецификации). Перейдите на вкладку Detail (Подробно). Установите переключатель Concurrency (Параллелизм) в требуемое значение.

5.23.19. Задание предусловий и постусловий операции

Предусловия - это такие условия, которые должны быть выполнены перед запуском операции. Любые предусловия операции можно ввести на вкладке Preconditions (Предусловия) окна спецификации операции. Постусловиями называются такие условия, которые должны всегда выполняться после завершения работы операции. Постусловия задаются на вкладке Postconditions (Постусловия) окна спецификации операции.

5.23.20. Определение семантики операции

В поле Semantics (Семантика) окна спецификации операции можно описать, что будет делать операция. Для передачи логики используется псевдокод или обыкновенное описание. Все, что вводится в этом поле, появится в готовом коде в виде комментария. Если какая-либо диаграмма взаимодействия иллюстрирует семантику операции, можно ввести имя диаграммы в нижней части вкладки.

Для описания семантики операции откройте окно спецификации операции (или стандартное окно спецификации). Перейдите на вкладку Semantics (Семантика). В поле Semantics введите семантику операции.

5.23.21. Связывание файлов и ссылок с операцией

Некоторая информация, касающаяся данной операции, может содержаться во внешнем файле или на Web-странице. Например, это может быть документ с требованиями, подтверждающими необходимость операции, или с описанием того, что она будет делать.

Rational Rose позволяет связать файл или ссылку с операцией. Прикрепленный таким образом файл или ссылку можно открыть непосредственно в браузере. Информация о способах связывания внешнего файла с элементом диаграммы приводилась неоднократно. Для операции класса эта операция выполняется по аналогичной схеме.

5.24. Изображение атрибутов и операций диаграмме классов

5.24.1. Общие сведения.

Язык UML позволяет изображать на диаграммах классов как все детали, так и только те, что необходимы. В Rational Rose можно настроить диаграммы классов так, чтобы показать все атрибуты и операции, скрыть операции, скрыть атрибуты, показать некоторые атрибуты или операции, показать операции вместе с их полными сигнатурами или только их имена, Показать или скрыть видимость атрибутов и операций, показать или скрыть стереотипы атрибутов и операций.

В типичном проекте создается большое количество диаграмм классов. На одних основное внимание уделяется отношениям, а детали атрибутов и операций практически не отображаются. Другие показывают сами классы, но ничего не сообщают об операциях и атрибутах. На третьих могут быть представле-

ны все атрибуты и операции с наиболее подробной информацией о них. В среде Rational Rose один и тот же класс можно располагать на любом необходимом вам количестве диаграмм классов. А с помощью параметров можно задать отображение деталей описания атрибутов и операций.

Значения параметров по умолчанию задаются в окне, открываемом при выборе пункта меню ***Tools > Options (Инструменты > Параметры)***.

5.24.2. Изображение атрибутов

Подавление вывода атрибутов приведет не только к исчезновению атрибутов с диаграммы, но и к удалению линии, показывающей место расположения атрибутов в классе.

Существует два способа изменения параметров представления атрибутов на диаграмме. Можно установить нужные вам значения для каждого класса индивидуально либо изменить значения требуемых параметров по умолчанию до начала создания диаграммы классов. Внесенные таким образом изменения влияют только на вновь создаваемые диаграммы.

Для вывода всех атрибутов класса выделите класс на диаграмме. Щелкните правой кнопкой мыши. Выберите пункт ***Options > Show All Attributes (Параметры > Показать все атрибуты)***. Или выделите класс на диаграмме. В меню модели выберите пункт ***Edit > Diagram Object Properties > Show All Attributes (Правка > Свойства объектов диаграммы > Показать все атрибуты)***.

Если нужно показать только избранные атрибуты класса выделите класс на диаграмме. Щелкните правой кнопкой мыши на классе. Выберите пункт Options > Select Compartment Item. В окне Edit Compartment укажите нужные вам атрибуты. Или выделите класс на диаграмме. В меню модели выберите пункт ***Edit > Compartment.*** В окне Edit Compartment укажите нужные вам атрибуты.

Для подавления вывода всех атрибутов класса диаграммы выделите класс на диаграмме. Щелкните правой кнопкой мыши на классе. В контекстнозависимом меню выберите пункт ***Options > Suppress Attributes (Параметры > Подавить атрибуты)***. Или выделите класс на диаграмме. В меню модели выберите пункт ***Edit > Diagram Object Properties > Suppress Attributes (Правка > Свойства объектов диаграммы > Подавить атрибуты)***.

Для изменения принятого по умолчанию вида атрибута в меню модели выберите пункт ***Tools > Options (Инструменты > Параметры)***. Перейдите на вкладку Diagram (Диаграмма). Для установки значений параметров отображения атрибутов по умолчанию воспользуйтесь флажками Suppress attributes (Подавить атрибуты) и Show all attributes (Показать все атрибуты).

Изменение значений по умолчанию повлияет только на новые диаграммы. Вид существующих диаграмм классов не изменится.

5.24.3. Изображение операций

Как и в случае атрибутов, имеется несколько вариантов представления операций на диаграммах показать все операции, показать некоторые операции, скрыть все операции, подавить вывод операций.

Кроме того, существует возможность показать только имя операции. На диаграмме будет представлено имя операции, но не аргументы или тип возвращаемого значения. Или показать полную сигнатуру операции. На диаграмме будет представлено не только имя операции, но и все ее параметры, их типы данных и тип возвращаемого значения операции.

Для отображения всех операций класса выделите класс на диаграмме. Щелкните правой кнопкой мыши на классе. В контекстно-зависимом меню выберите ***Options > Show All Operations (Параметры > Показать все операции)***. Или выделите класс на диаграмме. В меню модели выберите пункт ***Edit > Diagram Object Properties > Show All Operations (Правка > Свойства объектов диаграммы > Показать все операции)***.

Если нужно показать только избранные операции класса выделите класс на диаграмме. Щелкните правой кнопкой мыши на классе. В контекстно-зависимом меню выберите ***Options > Select Compartment Items***. В окне Edit Compartment укажите нужные вам операции. Или выделите класс на диаграмме. В меню модели выберите пункт ***Edit > Compartment***. В окне Edit Compartment укажите нужные вам операции.

Для подавления вывода всех операций класса диаграммы выделите класс на диаграмме. Щелкните правой кнопкой мыши на имени класса. В контекстно-зависимом меню выберите ***Options > Suppress Operations (Параметры > Подавить операции)***. Или выделите класс на диаграмме. В меню модели выберите пункт ***Edit > Diagram Object Properties > Suppress Operations (Правка > Свойства объектов диаграммы > Подавить операции)***.

Если требуется показать на диаграмме классов сигнатуру операции выделите класс на диаграмме. Щелкните правой кнопкой мыши на имени класса. В контекстно-зависимом меню выберите ***Options > Show Operation Signature (Параметры > Показать сигнатуру операции)***. Или выделите класс на диаграмме. В меню модели выберите пункт ***Edit > Diagram Object Properties > Show Operation Signature (Правка > Свойства объектов диаграммы > Показать сигнатуру операции)***.

Для изменения принятого по умолчанию вида операции в меню модели выберите пункт ***Tools > Options (Инструменты > Параметры)***. Перейдите на вкладку Diagram (Диаграмма). Для установки значений параметров отображения операций по умолчанию воспользуйтесь флажками Suppress operations (Подавить операции), Show all operations (Показать все операции) и Show operation signatures (Показать сигнатуры операции).

5.24.4. Изображение стереотипов

В среде Rational Rose можно показывать или не показывать стереотипы атрибутов и операций. Стереотипы выводятся в угловых скобках (<< >>) перед именем атрибута или операции. Для отображения стереотипов атрибутов и операций класса выделите класс на диаграмме. Щелкните правой кнопкой мыши на имени класса. В контекстно-зависимом меню выберите ***Options > Show Compartment Stereotypes***. Или выделите класс на диаграмме. В меню

модели выберите пункт ***Edit > Diagram Object Properties > Show Compartment Stereotypes***.

Для указания того, нужно ли отображать стереотипы по умолчанию в меню модели выберите пункт ***Tools > Options (Инструменты > Параметры)***. Перейдите на вкладку Diagram (Диаграмма). Для установки значения этого параметра воспользуйтесь флажком Show stereotypes (Показать стереотипы).

5.25. Соотнесение операций с сообщениями

Как уже упоминалось, каждое сообщение диаграммы последовательности или кооперации должно быть соотнесено с операцией. Если диаграмма последовательности имеет вид, приведенный на рис. 5.16., то операция1 будет расположена внутри класса B. При создании диаграмм последовательности и кооперации вы можете использовать для сообщений не имена операций, а фразы на естественном языке.

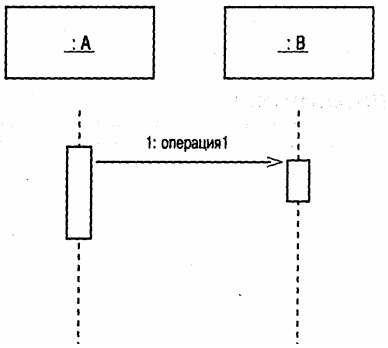

Рисунок 5.16. Фрагмент диаграммы взаимодействия.

Однако при идентификации операций придется соотнести с ними соответствующие сообщения. В результате диаграмма последовательности изменится.

Идентифицируя операции, изучите все сообщения на диаграммах последовательности и кооперации. Прежде чем генерировать код, убедитесь в том, что каждое сообщение соотнесено с соответствующей операцией. Операции можно соотнести с сообщениями, только если объекты диаграммы взаимодействия уже соотнесены с классами.

Соотнести сообщение с существующей операцией можно следующим образом. Убедитесь, что получающий сообщение объект (сервер) уже соотнесен с классом. Щелкните правой кнопкой мыши на сообщении диаграммы последовательности или кооперации. Появится список операций сервера. Выберите операцию в списке.

Для удаления соответствия между сообщением и операцией дважды щелкните мышью на сообщении диаграммы последовательности или кооперации. В поле имени удалите имя операции и введите новое сообщение.

Если требуется определить для сообщения новую операцию убедитесь, что получающий сообщение объект (сервер) уже соотнесен с классом. Щелкните правой кнопкой мыши на сообщении диаграммы последовательности или кооперации. Выберите пункт New operation (новая операция). Введите имя и детальное описание новой операции. Параметры окна спецификации операции рассматривались выше. Нажмите на кнопку OK, чтобы закрыть окно спецификации операции и добавить новую операцию. Щелкните правой кнопкой мыши на сообщении. Выберите новую операцию в появившемся списке.

Если нужно проверить, все ли сообщения диаграммы соотнесены с операциями в меню модели выберите пункт ***Report > Show Unresolved Messages (Отчет > Показать свободные сообщения)***. Появится список всех сообщений, которые еще не были соотнесены с операциями.

5.25. Отношения

5.25.1. Отношение ассоциации

Ассоциация (association) — это семантическое отношение между классами. Ассоциация дает классу возможность узнавать об общих атрибутах и операциях другого класса. Ее рисуют на диаграмме классов в виде линии (см. рис. 5.17). После того как классы связаны отношением ассоциации, они могут передавать друг другу сообщения на диаграмме последовательности или кооперации. Ассоциации могут быть ненаправленными (иногда говорят - двунаправленными) (рис. 5.17) или однонаправленными. На языке UML двунаправленные ассоциации изображают в виде простой линии без стрелок или со стрелками с обеих сторон. На однонаправленной ассоциации ставят только одну стрелку, показывающую направление отношения. После определения ассоциации Rational Rose помещает в классы соответствующие дополнительные атрибуты.

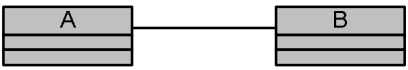

Рисунок 5.17. Ненаправленная ассоциация.

Однонаправленные отношения ассоциации позволяют выявить классы, являющиеся кандидатами на повторное использование. Если ассоциация двунаправленная, каждый класс-участник должен знать о другом. При генерации кода для двунаправленной ассоциации создаются дополнительные атрибуты у каждого класса.

Ассоциации могут быть рефлексивными. Рефлексивная ассоциация предполагает, что один экземпляр класса взаимодействует с другими экземплярами этого же класса.

5.25.2. Создание ассоциации

В среде Rational Rose ассоциации создают непосредственно на диаграмме классов. Контекстная панель инструментов диаграммы классов содержит, кнопки для построения как одно, так и двунаправленных ассоциаций.

Если создана двунаправленная ассоциация, ее впоследствии можно преобразовать в однонаправленную, изменив возможность навигации.

Для создания двунаправленной ассоциации на диаграмме классов нажмите кнопку Association (Ассоциация) панели инструментов. Или в меню модели выберите пункт ***Tools > Create > Association (Инструменты > Создать > Ассоциация)***. Проведите мышью линию ассоциации от одного класса к другому.

Для создания на диаграмме классов однонаправленной ассоциации нажмите кнопку Unidirectional Association (Однонаправленная ассоциация) панели инструментов. Или в меню модели выберите пункт ***Tools > Create > Unidirectional Association (Инструменты > Создать > Однонаправленная ассоциация)***. Проведите мышью линию ассоциации от одного класса к другому.

Для задания возможности навигации по ассоциации щелкните правой кнопкой мыши на том конце отношения, на котором нужно показать стрелку. В открывшемся меню выберите пункт Navigable (Навигация). Или откройте окно спецификации требуемого отношения. Перейдите на вкладку Role Detail (Роль, детали) для того конца ассоциации, для которого нужно подключить возможность навигации. Измените возможность навигации с помощью флажка Navigable.

Для создания на диаграмме классов рефлексивной ассоциации нажмите кнопку Association (Ассоциация) панели инструментов диаграммы. Или в меню модели выберите пункт ***Tools > Create > Association (Инструменты > Создать > Ассоциация)***. Проведите линию ассоциации от класса до какого-нибудь места вне класса. Отпустите кнопку мыши. Проведите линию ассоциации обратно к классу.

Для добавления к ассоциации текстового описания дважды щелкните мышью на ассоциации. Перейдите на вкладку General (Общие). Введите описание в поле документации. Или выделите ассоциацию. В меню модели выберите пункт ***Browse > Specification (Обзор > Спецификация)***. Перейдите на вкладку General (Общие). Введите описание в поле документации.

Для преобразования отношения в ассоциацию выделите требуемое отношение. В меню модели выберите пункт ***Edit > Change Into > Association (Правка > Преобразовать в > Ассоциация).***

5.25.3. Удаление ассоциации

Возможны два способа удаления ассоциации. При удалении с одной диаграммы ассоциация по-прежнему будет существовать на других диаграммах классов. В этом случае Rational Rose будет знать о наличии ассоциации и отслеживать ее "за сценой". При удалении ассоциации из модели отношение будет удалено со всех диаграмм классов и перестанет отслеживаться Rational Rose.

Для удаления ассоциации только с одной диаграммы выделите ассоциацию. Выберите пункт ***Edit > Delete (Правка > Удалить)*** или нажмите клавишу Delete. При таком способе удаления ассоциация не удаляется из модели.

Для удаления ассоциации из модели выделите ассоциацию. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш CTRL+D. Или откройте окно спецификации для одного из участвующих в ассоциации классов. Перейдите на вкладку Relations. Щелкните на отношении правой кнопкой мыши. В открывшемся меню выберите пункт Delete.

5.25.4. Отношение зависимости

Отношение зависимости (dependency) также отражает связь между классами, но делает это несколько иначе. Зависимости всегда однонаправленные, они показывают, что один класс зависит от определений, сделанных в другом. Специальные атрибуты для классов, связанных зависимостью, не создаются. Зависимости изображают в виде стрелки, проведенной пунктирной линией (рис. 5.18).

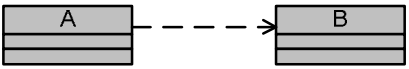

Рисунок 5.18. Отношение зависимости.

5.25.5. Создание зависимостей

Создать новую зависимость можно с помощью кнопки Dependency панели инструментов диаграммы классов.

Для создания зависимости на диаграмме классов нажмите кнопку Dependency (Зависимость) панели инструментов. Или выберите пункт ***Tools > Create > Dependency (Инструменты > Создать > Зависимость)***. Щелкните мышью на классе, который будет зависеть от другого. Проведите линию зависимости к другому классу.

Для добавления к зависимости текстового описания дважды щелкните мышью на зависимости. Перейдите на вкладку General (Общие). Введите описание в поле документации. Или выделите зависимость. Выберите пункт ***Browse > Specification (Обзор > Спецификация)***. Перейдите на вкладку General (Общие). Введите описание в поле документации.

Для преобразования в зависимость отношения другого типа выделите его на диаграмме или в браузере. Выберите пункт ***Edit > Change Into > Uses Dependency (Правка > Преобразовать в > Зависимость)***.

5.25.6. Удаление зависимостей

Допустимы два способа удаления зависимости: только с одной диаграммы или из модели в целом. Для удаления зависимости с диаграммы выделите зависимость. Выберите в меню пункт ***Edit > Delete (Правка > Удалить)*** или нажмите клавишу Delete. Удаление зависимости с диаграммы не приводит к удалению ее из модели.

Для удаления зависимости из модели выделите зависимость. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш CTRL+D. Или откройте окно спецификации для одного из участвующих в отношении классов. Перейдите на вкладку Relations (Отношения). Щелкните на отношении правой кнопкой мыши. В открывшемся меню выберите пункт Delete (Удалить).

5.25.7. Зависимости между пакетами

Зависимости можно устанавливать не только между классами, но и между пакетами. Фактически, это единственный тип отношений, существующий между пакетами. Как и в случае классов, зависимость между пакетами изображают пунктирной линией. Отношение зависимости между пакетами A и B означает, что некоторый класс пакета A связан однонаправленным отношением с некоторым классом пакета B. Иначе говоря, класс A должен знать что-либо о классе B. Зависимости определяют возможность повторного использования пакетов. На приведенном выше рисунке пакет A зависит от B. Это означает, что при создании других приложений пакет A можно использовать только совместно с пакетом B. Однако сам пакет B можно использовать повторно, так как он не зависит ни от чего больше.

Зависимости между пакетами можно обнаружить, исследуя отношения на диаграмме классов. Если два класса из различных пакетов связаны, эти пакеты также связаны.

Создавая зависимости между пакетами, старайтесь по мере возможности избегать циклических зависимостей. Такая зависимость предполагает, что класс из пакета A должен знать о классе из пакета B, а еще какой-то класс пакета B должен знать о классе из пакета A. Следовательно, ни один пакет нельзя самостоятельно использовать повторно, и изменения в одном из них неизбежно повлияют на другой. Таким образом, теряется одно из преимуществ пакетов - они становятся взаимозависимы.

Чтобы разбить циклическую зависимость, разделите пакет на два. В нашем примере можно взять классы пакета B, от которых зависит A, и переместить их в третий пакет C. Тогда зависимости пакетов будут выглядеть так, как показано на рис. 5.19.

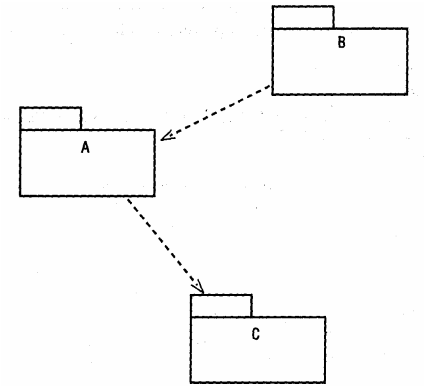

Рисунок 5.19. Зависимость между пакетами.

Определив зависимости между пакетами, можно добавить их к модели на диаграмме классов. Как правило, на одной из диаграмм классов изображают все пакеты и отношения между ними. Как и в случае классов, зависимости между пакетами создаются с помощью кнопки Dependency панели инструментов диаграммы классов.

Допустимы два способа удаления зависимости между пакетами: только с одной диаграммы классов и модели в целом. Если удалить зависимость между пакетами, классы которых все еще связаны, возникнут проблемы при генерации кода. С помощью пункта меню ***Report > Show Access Violations (Отчет > Показать нарушения доступа)*** можно проверить, имеются ли в вашей модели проблемы такого рода.

Для удаления отношения зависимости между пакетами с диаграммы классов выделите зависимость между пакетами. Выберите пункт ***Edit > Delete (Правка > Удалить)*** или нажмите клавишу Delete.

5.25.8. Отношение агрегации

Агрегация (aggregations) представляет собой более тесную форму ассоциации. Агрегация - это отношение между целым и его частями. Агрегацию изображают в виде линии с ромбиком у класса, являющегося целым (рис. 5.20).

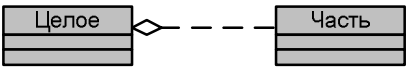

Рис. 5.20. Отношение агрегации.

Один класс может участвовать в нескольких отношениях агрегации с другими классами. Как и ассоциации, агрегации могут быть рефлексивными. Рефлексивные агрегации предполагают, что один экземпляр класса состоит из одного или нескольких других экземпляров того же класса. При генерации кода для агрегации автоматически создаются поддерживающие ее дополнительные атрибуты.

Агрегации указывают на диаграммах классов. Для создания агрегации можно воспользоваться кнопкой Aggregation (Агрегация) панели инструментов диаграммы.

Для добавления агрегации на диаграмму классов уажмите кнопку Aggregation (Агрегация) панели инструментов. Или выберите пункт ***Tools > Create > Aggregation (Инструменты > Создать > Агрегация)***. Проведите линию агрегации от класса-части к целому.

Для создания на диаграмме классов рефлексивной агрегации нажмите кнопку Aggregation (Агрегация) панели инструментов диаграммы. Или в меню модели выберите пункт ***Tools > Create > Aggregation (Инструменты > Создать > Агрегация)***. Проведите линию агрегации от класса до какого-нибудь места вне класса. Отпустите кнопку мыши. Проведите линию агрегации обратно к классу.

Для добавления текстового описания к агрегации дважды щелкните мышью на агрегации. Перейдите на вкладку General (Общие). Введите описание в поле документации.

Если нужно преобразовать отношение на диаграмме классов в агрегацию выделите требуемое отношение. В меню модели выберите пункт ***Edit > Change Into > Aggregation (Правка > Преобразовать в > Агрегация)***. Или откройте окно спецификации преобразуемого отношения. Перейдите на вкладку Role Detail (Роль, детали). Установите флажок Aggregate (Агрегация).

Агрегации можно удалять с одной только диаграммы или из модели в целом. Для удаления агрегации с диаграммы выделите агрегацию. Выберите пункт ***Edit > Delete (Правка > Удалить)*** или нажмите клавишу Delete. Удаление агрегации с диаграммы не означает удаления ее из модели. Для удаления агрегации из модели выделите агрегацию. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш CTRL+D.

5.25.9. Отношение обобщения

С помощью ***обобщений (generalization)*** показывают отношения наследования между двумя классами. Большинство объектноориентированных языков непосредственно поддерживает концепцию наследования. Она позволяет одному классу наследовать все атрибуты, операции и отношения другого. На языке UML отношение наследования называют обобщением и изображают в виде стрелки от класса потомка к классу-предку (рис. 5.21).

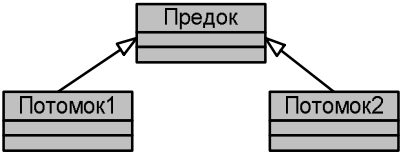

Рисунок 5.21. Отношение обобщения.

Отношения обобщения позволяют сэкономить время и усилия как при разработке, так и при дальнейшей поддержке программы. Для выявления обобщений на диаграмме можно развертывать структуру наследования сверху вниз или снизу вверх. В первом случае необходимо искать классы, у которых имеется несколько разновидностей.

Создаваемая структура должна быть управляемой. Иерархия со слишком большим количеством уровней может стать плохо управляемой - каждое изменение в верхней части структуры немедленно отразится во всех производных классах. Являясь, с одной стороны, преимуществом, это может затруднить анализ и управление системой. В некоторых языках большое количество уровней усложняет работу приложения.

При создании обобщений может потребоваться перенести некоторые атрибуты или операции из одного класса в другой. В браузере достаточно перетащить атрибут из любого порожденного класса в класс Employee. Не забудьте удалить копию атрибута из второго класса-потомка.

Отношения обобщения можно добавлять к модели Rational Rose с помощью диаграмм классов. Часто даже создают специальные диаграммы классов, посвященные структуре наследования. Они содержат минимальное количество данных и атрибутов.

Для создания обобщения на диаграмме классов нажмите кнопку Generalization (Обобщение) панели инструментов. Или выберите пункт ***Tools > Create > Inherits (Инструменты > Создать > Наследование)***. Проведите линию обобщения от подкласса к суперклассу.

Если нужно преобразовать отношение на диаграмме классов в обобщение выделите требуемое отношение. В меню модели выберите пункт ***Edit > Change Into > Inherits (Правка > Преобразовать в > Наследование)***

Обобщение можно удалить с одной диаграммы или из модели. При удалении из модели помните, что атрибуты, операции и отношения суперкласса больше не будут наследоваться подклассами. Если они нужны в подклассе, их необходимо туда добавить.

Для удаления обобщения с диаграммы выделите обобщение. Выберите пункт ***Edit > Delete (Правка > Удалить)*** или нажмите клавишу Delete. Удаление обобщения с диаграммы не означает удаления его из модели. Для удаления обобщения из модели выделите обобщение. В меню модели выберите пункт ***Edit > Delete from Model (Правка > Удалить из модели)*** или нажмите комбинацию клавиш CTRL+D.

5.26. Выявление отношений

Для выявления отношений следует изучить созданные к этому моменту элементы модели. Почти вся информация об отношениях уже описана на диаграммах последовательности и кооперации. Просмотрите их еще раз, и найдете ассоциации и зависимости. Агрегации и обобщения можно обнаружить, изучив имеющиеся классы. Для обнаружения отношений сделайте следующее.

1. Начните с изучения диаграмм последовательности и кооперации. Если класс A посылает сообщение классу B, между этими классами должна быть установлено отношение. Как правило, обнаруживаемые таким способом отношения - это ассоциации или зависимости.
2. Исследуйте классы на предмет наличия отношений целое-часть. Любой класс, состоящий из других классов, может принимать участие в отношениях агрегации.
3. Исследуйте классы на предмет отношений обобщения. Постарайтесь найти все классы, у которых есть несколько типов или вариантов. Например, у вас может быть класс Employee (Сотрудник). В вашей компании имеются два типа сотрудников - получающих почасовую оплату и оклад. Это значит, что вам нужно создать классы HourlyEmp и SalariedEmp, наследующие от класса Employee общие для всех сотрудников атрибуты, операции и отношения. Атрибуты, операции и отношения, уникальные для сотрудников какого-то типа, необходимо поместить в классы HourlyEmp и SalariedEmp.
4. Изучите классы еще раз и найдите остальные отношения обобщения. Постарайтесь обнаружить такие классы, которые имеют много общего. В частности, у вас могут быть классы CheckingAccount (Счет до востребования) и SavingsAccount (Сберегательный счет). Их данные и поведение сходны. Для общих элементов двух классов можно создать третий класс Account (Счет).

Будьте осторожны, работая с классами, у которых слишком много отношений. Одной из особенностей хорошо спроектированного приложения является сравнительно небольшое количество отношений в системе. Класс, у которого много отношений, должен знать о большом числе других классов системы. В результате его трудно будет использовать во второй раз. Кроме того, сложно будет вносить изменения в готовое приложение. Если изменится любой из классов, это может повлиять на данный.

5.27. Работа с отношениями

Подробно рассмотрим спецификации отношений в Rational Rose. Разрешается добавлять в модель такие элементы отношений, как имена ассоциаций, ролевые имена и квалификаторы, объясняющие причину появления отношения.

Для генерации кода необходимо указать также множественность отношения, иначе будут заданы значения по умолчанию.

Многие спецификации являются необязательными. Если не хватает какой-то необходимой спецификации. Rational Rose сообщит об этом при генерации кода.

5.27.1. Задание множественности

Множественность (multiplicity) показывает, сколько экземпляров одного класса взаимодействуют с помощью отношения с одним экземпляром другого класса в данный момент времени.

Помните, что значение множественности позволяет понять, является ли данное отношение обязательным.

Обычно для форм, экранов и окон множественность равна 0..1 с каждой стороны отношения. Это означает, что формы могут существовать независимо одна от другой, хотя это и не всегда правильно.

Для задания множественности отношения щелкните правой кнопкой мыши на одном конце отношения. В открывшемся меню выберите пункт Multiplicity (Множественность). Укажите нужную множественность. Повторите шаги для другого конца отношения.

5.27.2. Использование имен отношений

Отношения можно уточнить с помощью имен отношений или ролевых имен. Имя отношения — как правило глагол или глагольная фраза, описывающая, зачем нужно отношение. Имена отношений определять необязательно. Обычно это делают, если причина создания отношения неочевидна. Имя показывают около линии соответствующего отношения.

В Rational Rose можно указать также направление отношения. Например, можно сказать, что компания нанимает сотрудника, но не наоборот. Направление действия имени устанавливается в окне спецификации отношения.

Для задания имени отношения выделите его. Введите имя. Или откройте окно спецификации отношения. Перейдите на вкладку General (Общие). Введите имя отношения в поле имени.

Для указания направления имени откройте окно спецификации отношения. Перейдите на вкладку Detail (Подробно). В поле Name direction (Направление имени) укажите направление имени отношения.

5.27.3. Использование стереотипов

Как и другим элементам модели, отношениям разрешается назначать стереотипы. Они применяются для классификации отношений. Например, используется два типа ассоциаций. Для этих типов можно создать стереотипы. Стереотипы пишут вдоль линии ассоциации в двойных угловых скобках (« »).

Стереотип отношения задается на вкладке General (Общие) окна спецификации отношения. Для задания стереотипа отношения откройте окно спецификации отношения. Перейдите на вкладку General (Общие).Введите стереотип в поле Stereotype.

5.27.4. Использование ролей

Ролевые имена применяют в отношениях ассоциации или агрегации для описания назначения отношения. Ролевые имена - это обычно имена существительные или фразы. Их показывают на диаграмме рядом с классом, играющим ответственную роль. Как правило, пользуются или ролевым именем, или именем отношения, но не обеими сразу. Как и имена отношений, ролевые имена необязательны, их указывают, только если цель отношения неочевидна.

В окне спецификации отношения можно добавить к роли текстовое описание. При генерации кода оно войдет в комментарий. Чтобы увидеть роль на диаграмме, щелкните правой кнопкой мыши на отношении и в открывшемся меню выберите пункт Role name.

Для задания ролевого имени щелкните правой кнопкой мыши на нужном конце ассоциации. В открывшемся меню выберите пункт Role name (мя роли). Введите ролевое имя. Или откройте окно спецификации ассоциации. Перейдите на вкладку Role General (Роль, общие) для той роли, которой нужно дать имя. Введите имя роли в поле Role (Роль).

Для добавления текстового описания к роли откройте окно спецификации требуемой ассоциации. Перейдите на вкладку Role General (Роль, общие) роли. Введите описание в поле Documentation (Документация).

5.27.5. Задание управления экспортом

При генерации кода для классов, связанных ассоциацией, создаются атрибуты. Видимость этих атрибутов определяется значением переключателя Export Control (Управление экспортом). Как и для любых других атрибутов, возможны четыре значения видимости:

Public (Общий) - доступ к атрибуту осуществляется из всех остальных классов. ***Private (Закрытый)*** - доступ к атрибуту невозможен ни из какого другого класса. ***Protected (Защищенный).*** Доступ к атрибуту возможен только из самого класса и его потомков. ***Package or Implementation (Пакетный).*** Доступ осуществляется только из классов того же пакета. В двунаправленных ассоциациях управление экспортом можно установить для атрибутов обоих концов отношения. В однонаправленных ассоциациях это делается только для одного конца. Управление экспортом можно задать на вкладке Role A General (Роль A, общие) или Role B General (Роль B, общие) окна спецификации отношения.

Назначить управление экспортом для роли можно следующим образом щелкните правой кнопкой мыши на имени роли. В открывшемся меню выберите пункт Export Control (Управление экспортом). Или откройте окно спецификации отношения. Перейдите на вкладку Role General (Роль, общие) роли. Установите переключатель Export Control (управление экспортом) в нужное значение: Public, Protected, Private или Implementation.

5.27.6. Использование статичных отношений

Как уже упоминалось, при создании кода для отношений ассоциации и агрегации генерируются атрибуты. Поле Static (Статичная) определяет будут ли эти атрибуты статичными. Статичным называется атрибут, который используется всеми экземплярами класса.

Если роль статичная, то создаваемый при генерации кода соответствующий ей атрибут также будет статичным. На диаграмме Классов перед статичной ролью ставится знак $.

Для определения статичной ассоциации щелкните правой кнопкой мыши на нужном конце ассоциации. В открывшемся меню выберите пункт Static (Статичная). Или откройте окно спецификации требуемой ассоциации. Перейдите на вкладку Detail (Подробно) роли, которую нужно сделать статичной. Установите флажок Static (Статичная).

5.27.7. Использование дружественных отношений

Дружественное (friend) отношение предполагает, что класс-клиент имеет право доступа к атрибутам и операциям класса-сервера, не являющимся общими. Это свойство можно задать для ассоциаций, агрегаций, зависимостей и обобщений. В исходный код класса-сервера войдет логика, поддерживающая дружественную видимость для клиента.

Допустим, что у нас имеется двунаправленная ассоциация, связывающая классы Person и Company и мы установили флажок Friend для этого отношения. Тогда при генерации кода на языке C++ в файл Company.H будет добавлена строка "friend class Person". Это означает, что класс Person получит право доступа к частям класса Company, не являющимся общими.

Для создания дружественного отношения щелкните правой кнопкой мыши на соответствующем конце отношения. В открывшемся меню выберите пункт Friend. Или откройте окно спецификации требуемого отношения. Перейдите на вкладку Detail (Подробно) роли, которую нужно сделать дружественной. Установите флажок Friend (Друг).

5.27.8. Задание метода включения

Параметр Containment of Button определяет, каким образом будут включаться созданные атрибуты агрегации: по значению или по ссылке. Если два класса связаны отношением агрегации, то в класс-целое войдут атрибуты для каждого классовости. В поле Containment of Button устанавливается, будут ли эти данные атрибутами по значению или по ссылке.

Значение By Value (По значению) этого параметра предполагает, что целое и часть создаются и разрушаются одновременно. Например, если между классами Window (Окно) и Button (Кнопка) установлена агрегация по значению, соответствующие объекты создаются и разрушаются в одно и то же время. На языке UML агрегацию по значению помечают закрашенным ромбиком.

Агрегация по ссылке (By Reference) предполагает, что целое и часть создаются и разрушаются в разное время. Если класс EmployeeList состоит из классов Employee, то агрегация по ссылке означает, что как класс EmployeeList, так и классы Employee могут находиться или не находиться в памяти компьютера в любой момент времени независимо друг от друга. Если они имеются в памяти, то взаимодействуют друг с другом посредством агрегации. Классы Employee и EmployeeList создаются и разрушаются в разное время. Агрегация по ссылке отображается в виде пустого ромбика.

Для установки метода включения щелкните правой кнопкой мыши на том конце ассоциации, где требуется задать включение. В открывшемся меню выберите пункт Containment. Укажите метод включения: By Reference, By Value или Unspecified (Не определен). Или откройте окно спецификации требуемого отношения. Перейдите на вкладку Role Detail (Роль, детали) нужной роли. Укажите метод включения: By Reference, By Value или Unspecified (Не определен).

5.27.9. Работа с квалификаторами

Квалификаторы (qualifiers) применяются для того, чтобы уменьшить область действия ассоциации. Допустим, что между классами Person и Company установлено отношение ассоциации и что для данного значения атрибута Person ID (Идентификационный номер) существуют две взаимодействующие с классом компании. Это можно показать на диаграмме с помощью квалификаторов.

Для добавления на диаграмму квалификатора щелкните правой кнопкой мыши на том конце ассоциации, куда требуется добавить квалификатор. В открывшемся меню выберите пункт New Key / Qualifier (Создать ключ / квалификатор). Введите имя и тип нового квалификатора. Или откройте окно спецификации требуемой ассоциации. Перейдите на вкладку Role Detail (Роль, детали) нужной роли. Щелкните правой кнопкой мыши в окне Keys / Qualifiers (Ключи / Квалификаторы). В открывшемся меню выберите пункт Insert (Вставить). Введите имя и тип нового квалификатора.

Для удаления квалификатора откройте окно спецификации требуемой ассоциации. Перейдите на вкладку Role Detail (Роль, детали) нужной роли. Щелкните правой кнопкой мыши на удаляемом квалификаторе. В открывшемся меню выберите пункт Delete (Удалить).

5.27.10. Работа с элементами отношения

Элементом отношения (link element), известным также как класс ассоциаций (Association class), называется место, где хранятся относящиеся к ассоциации атрибуты. Допустим, что у есть два класса: Student и Course, и необходимо добавить на диаграмму атрибут Grade (Год обучения). В этом случае возникает вопрос: в какой класс добавить атрибут? Если мы поместим его в класс Student, то придется вводить атрибут для каждого посещаемого студентом курса, что значительно увеличит этот класс. Если же мы поместим его в

класс Course, то придется задавать атрибут для каждого посещающего этот курс студента. Для решения подобной проблемы можно создать класс ассоциаций. В него следует поместить атрибут Grade, относящийся в большей степени к отношению между курсом и студентом, чем к какому-то классу конкретно.

Для задания элемента отношения откройте окно спецификации требуемого отношения. Перейдите на вкладку Detail (Подробно). Укажите элемент отношения в поле Link Element (Элемент отношения).

5.27.11. Использование ограничений

Ограничением (constraint) называется некоторое условие, которое должно выполняться. В среде Rational Rose можно установить ограничения для отношения в целом или для одной роли. При генерации кода все ограничения войдут в комментарий. Для установки ограничения отношения откройте окно спецификации отношения. Перейдите на вкладку Detail (Подробно). Введите ограничение в поле Constraints (Ограничения).

Если нужно установить ограничения для одной роли откройте окно спецификации отношения. Перейдите на вкладку Role Detail (Роль, детали) нужной роли. Введите ограничения в поле Constraints (Ограничения).

5.28. Задание и отчетность

Необходимо дополнить модель информационной системы, описанной в практическом пособии, создав для нее диаграмму классов с использованием Rational Rose.

Документы отчетности сдаются на проверку в электронной форме и включают в себя файл модели (*.mdl), документы Microsoft Word, содержащие дополнительную информацию по проектным решениям, файл с указанием группы и фамилии студентов, выполнивших работу.

6. Лабораторная работа № 6. «Введение в объектно-ориентированное программирование. Работа с классом Vector»

6.1. Цель работы

Ознакомиться с основными принципами объектно-ориентированного программирования и синтаксисом C++ для создания объектно-ориентированных приложений на примере класса Vector.

6.2. Порядок выполнения работы

6.2.1. Синтаксис C++ для работы с классами

В языке программирования C++ основным элементом объектно-ориентированных приложений является ***класс***. Синтаксис описания класса:

class <имя класса>
{
список членов (описание полей и методов)
};

Члены класса можно разделить на две группы. *Поля* класса – данные, каким –то образом идентифицирующие объект класса, либо хранящиеся в объекте класса. *Методы* класса – функции, которые могут использовать поля класса и внешние данные.

Каждый член класса имеет атрибут доступа, с их помощью определяется “зона видимости” для члена класса. Всего таких атрибутов 3:

- *public* - член класса может использоваться любой функцией. Как правило, этот атрибут используется для функций-членов классов (методов); т.е. такой член видим и вне класса.
- *private* - член класса может использоваться только функциями-членами класса и функциями-друзьями класса. Как правило используется для данных класса (полей), т.е. скрывает данные внутри класса.
- *protected* - член класса может использоваться функциями-членами класса и функциями-друзьями класса, а так же членами классов, для которых этот класс является базовым (предком).

Существуют специальные функции-члены, которые отвечают за создание объектов класса, их копирование и удаление. Это так называемые *конструкторы* и *деструкторы*. *Конструкторы* – это функции-члены класса, основная цель которых инициализация объекта или распределение памяти для хранения объекта. Конструктор имеет то же имя, что и класс. Так же, как и любая функция, конструктор может иметь и не иметь параметров. Конструктор без параметров называется конструктором по умолчанию. Конструктора может не быть, тогда C++ генерирует собственный конструктор по умолчанию.

Деструктор разрушает созданный объект. У деструктора нет параметров. Деструктор вызывается, если закончена часть программы, в которой видим объект. Деструктора носит имя класса с лидирующим знаком «~» - тильда. Если деструктор не описан в классе явно, то используется деструктор по умолчанию. Деструктор не может перегружаться.

6.2.2. Описание данных и методов класса Vector

Класс VECTOR содержит набор методов для работы с массивом.

```
class VECTOR
  {
  protected :
          int n; // количество элементов массива
          int *vector; // массив целочисленных элементов
  public :
```

```
VECTOR(int size, const flag = 0); /* конструктор, создает объект массив, размерностью size, константный параметр flag задает способ заполнения массива: значение по умолчанию, равное 0 - элементы массива задаются случайным образом, при других значениях элементы массива задаются с клавиатуры */
void ENTER(int number); /* метод, осуществляющий ввод нового элемента в позицию number*/
void PRINT_ALL(); /* метод печати элементов массива */
int DELETE(int number); /* метод удаления элемента с номером number*/
int GETN(); /* метод, возвращающий количество элементов массива */
int GETELEM(int number); /*метод, возвращающий значение элемента с номером number*/
void SAVEFILE(const char* filename); /*метод, сохраняющий элементы массива в текстовом файле с именем filename*/
~VECTOR(); /* деструктор класса*/
};
```

6.2.3. Задание на выполнение и пример выполнения.

- Самостоятельно рассмотрите реализацию методов класса VECTOR. Обратите внимание на синтаксис описания методов.
- Получите вариант индивидуального задания у преподавателя.

Рассмотрим пример выполнения практического задания:

Используя методы класса VECTOR, решить следующую задачу: *Создать массив размерности n. Размерность массива задавать с клавиатуры, элементы массива задать случайным образом. Найти количество ненулевых элементов массива.*

Для решения этой задачи напишем свой собственный класс MyVector с одним методом Run, решающим поставленную задачу. Опишем класс в файле с расширением h, например myvec.h:

```
class MyVector
 {
   public:
   void Run();
 };
```

Метод Run опишем в файле с расширением cpp, например myvec.cpp:

```
#include "vec.h" /* подключение заголовочного файла, содержащего описание класса VECTOR*/
#include "myvec.h" /* подключение заголовочного файла, содержащего описание класса MYVECTOR*/
```

```
#include <iostream.h> /* подключение стандартных заголовочных файлов
*/
#include <conio.h>
#include <stdio.h>

 void MyVector :: Run()
  {
    clrscr();
    cout << "Введите количество элементов массива: ";
    int n;
    cin >> n;
    VECTOR a(n); /* описывается и создается объект a – масссив размер-
    ности n. */
    a.PRINT_ALL(); /* для объекта выполнить метод печати */
    int k = 0;
    /* в цикле просматривать все элементы массива, используя метод по-
    лучения значения массива с номером i */
    for(int i=0;i<n;i++)
            if(a.GETELEM(i) == 0) k++;
    if (k==0) cout << "В массиве нет нулевых элементов";
    else cout << "В массиве %d нулевых элементов" << endl;
    getch();
  }
```

Для запуска написанного метода в функции main() должен быть создан объект класса MyVector. К этому объекту нужно применить метод Run. Напишем функцию main() в отдельном файле:

```
#include "myvec.h" /* подключить заголовочный файл, содержащий опи-
сание класса MyVector*/

void main()
 {
  MyVector Obj; /* Создать объект класса MyVector*/
  Obj.Run(); /* Применить к объекту метод Run*/
 }
```

Для компиляции программы создадим проект. В проект должны быть включены 3 файла с расширением cpp:

- Файл с функцией main()
- Файл с описанием метода Run класса MyVector
- Файл с описанием методов класса VECTOR.

После успешной компиляции защитите выполненное задание.

7. Лабораторная работа № 7. «Введение в объектно-ориентированное программирование. Создание класса графического изображения»

7.1. Цель работы

Научиться практически применять теоретический материал, изученный на практическом занятии № 1 для разработки собственного класса на примере класса графического изображения.

7.2.Общая структура класса в С++

Синтаксис описания класса и основные определения были рассмотрены на предыдущем занятии. Рассмотрим синтаксис конструкторов, деструкторов, обыкновенных методов класса.

Пусть в программе описан класс:

class <имя класса>
{
<имя класса> (тип арг1, тип арг2,…);
~<имя класса>();
тип <имя метода 1>(тип арг1, тип арг2,…);
тип <имя метода 2>(тип арг1, тип арг2,…);
…
}

Синтаксис описания конструктора и деструктора:

<имя класса-хозяина> :: <имя класса> (тип арг1, тип арг2,…)
{ …}
<имя класса-хозяина> :: ~<имя класса> ()
{…}

Синтаксис описания обыкновенных методов:

тип возвращаемого значения <имя класса-хозяина> :: <имя метода> (тип арг1, тип арг2,…)
{…}

Синтаксис описания объекта класса:

<имя класса> <имя объекта>(арг1, арг2,…);

Имя объекта формируется по правилам формирования имен переменных. В круглых скобках пишутся фактические значения для аргументов конструктора.

Если поля и(или) методы класса описаны атрибутом public, эти элементы класса можно применять к объектам класса, используя следующий синтаксис:

<имя объекта>.<имя метода>(фактические аргументы метода);

Если поля и (или) методы класса описаны другими атрибутами доступ к ним можно получить только через другие методы этого класса.

При использовании объектно-ориентированного подхода нужно помнить, что:

- одним из принципов называется инкапсуляция – скрытие данных внутри класса, поэтому поля класса как правило описываются атрибутами private или protected;
- конструкторы используются для инициализации объектов класса, распределения памяти для полей класса, конструкторов может быть несколько;
- конструктора в классе может не быть;
- деструкторы в классе необходимы только в том случае, если при создании объекта класса выделяется память, в остальных случаях они не обязательны.

7.3. Порядок выполнения работы и пример выполнения

- получите вариант задания у преподавателя;
- разработайте структуру класса по выбранному варианту
- напишите программу, позволяющую передвигать объект класса с помощью клавиш управления курсором.

При разработке класса можно воспользоваться следующим примером:

Разработать класс или классы для работы с графической точкой.

Опишем класс-сущность – класс, задающий параметры(свойства) графической точки. За свойства точки можно принять ее координаты на экране и ее цвет. Поля класса опишем атрибутом private, для того, чтобы ограничить доступ к данным извне. Конструктор класса определяет или инициализирует свойства объекта. Метод change изменяет координаты точки на соответствующие приращения. Методы getX() и getY() возвращают координаты точки. Метод draw(int c = -1) при вызове с параметром по умолчанию рисует точку цветом объекта, в противном случае рисует точку цветом, заданным в параметре.

```
class pixel
 { private:
   int x,y,color;
   public:
   pixel(int a,int b,short c);
   void change(int dx, int dy);
   void draw(const int = -1);
   int getX();
   int getY();
};
```

Опишем второй класс, который будет определять способы работы с точкой. Параметрами методов такого класса должны быть объекты или указатели на объекты класса pixel. С такими параметрами можно выполнить следующие действия – сделать точку видимой, нарисовать ее. Это действие выполняет метод put(pixel *a). Стереть точку – метод clear(pixel *a), использует метод рисования точки цветом фона. Изменить положение точки – метод move(pixel

*a, int dx, int dy), где переменные dx и dy – приращения координат объекта-точки.

```
class action
 {
  public :
    void put(pixel *a);
    void clear(pixel *a);
    void move(pixel *a,int dX, int dY);
 };
```

Опишем методы классов:

```
pixel :: pixel (int a, int b, short c)
    {
  x = a;
  y = b;
 color = c;
    }
 void pixel :: change(int dx, int dy)
  {
 x+=dx;
 y+=dy;
  }
 void pixel :: draw(const int c)
  {
  if (c==-1)
  putpixel(x,y,color);
              else putpixel(x,y,c); }
```

```
int pixel :: getX()
{
 return x;
}
int pixel :: getY()
{
  return y;
}
int pixel :: getColor()
{
  return color;
}
```

```
void action::put(pixel *a)
  {
    a->draw();
  }
 void action::clear(pixel *a)
  {
    int color = getbkcolor();
    a->draw(color);
  }
```

```
void action::move(pixel *a,int dx,int dy)
  {
  this -> clear(a);
  a->change(dx,dy);
  this -> put(a);
  }
```

Обратите внимание на синтаксис вызова метода, или обращения к полю класса: *<имя объекта>. <имя метода или поля>* либо *<ссылка на объект>-> <имя метода или объекта>*.

Напишем класс, демонстрирующий работу этих классов. Создадим объект-точку с заданными координатами, проинициализируем графику. Будем управлять объектом-точкой с помощью объекта класса action.

```
class MyGraph
  {
```

```
  public:
  void Run();
 };
void MyGraph :: Run ()
 {
 int gdriver = DETECT, gmode, errorcode;
 int x=100,i;
 int dX=20;
 initgraph(&gdriver, &gmode, " ");
 pixel p1(x,200,4);
 action b;

 for (i=0;i<10;i++)
 {
 delay(300);
 b.move(&p1,dX,0);
 }
 getch();
closegraph();
  }


 void main()
 {
   MyGraph obj;
   obj.Run();
 }
```

Примечание: для рисования равносторонних фигур можно воспользоваться следующим алгоритмом:

1. x,y – координаты центра фигуры;
2. n – количество сторон;
3. r – радиус описанной окружности;
4. установить указатель графического экрана в точку с координатами (x,y-r);
5. цикл ($i := \overline{1;n}$)

   5.1. провести линию в точку с координатами

$$x + \sin(\frac{i \cdot 2 \cdot \pi}{n}) \cdot r, y - \cos(\frac{i \cdot 2 \cdot \pi}{n}) \cdot r$$

Лабораторная работа №8. «Наследование и полиморфизм. Создание класса-наследника»

8.1. Цель работы

Практическое применение принципов наследования и полиморфизма – построение иерархической схемы классов на примере создания класса-наследника для ранее написанного класса.

8.2. Принцип наследования

Предположим, что в одной программе используются класс pixel и класс circle, объектами которого являются окружности. Объекты класса circle отличаются от объектов класса pixel, но у них много общего. Определим класс circle как наследник класса pixel. Класс circle в этом случае будет называться *производным классом*, класс pixel – базовым классом. Класс circle наследует все свойства базового класса, описанные атрибутами доступа protected и public. Для этого класса нет необходимости описывать поля x, y, color.

Для того, чтобы в C++ описать класс, как производный используется конструкция:

```
class a {
        private: int c;
        protected: float к;
        public :
                a();
                metod1();
                metod2();
}
```

```
class b : public a {
        public:
                b();
                metod4(); }
```

В данном примере класс b наследует поле K, методы metod1();metod2(), класс b имеет свой собственный метод – metod4().

Таким образом, производные классы могут «пользоваться» всеми методами, описанными в базовом классе соответствующими атрибутами, а так же могут иметь и свои собственные поля и методы.

Ссылки на объекты производного класса могут быть описаны как ссылки на объекты базового класса, т.е. допустим следующий фрагмент программы:

```
…
a *obj1,obj2; // ссылки на объект класса a
obj1 = new a(); // выделение памяти под объект класса a
obj2 = new b(); // выделение памяти под объект класса b
```

Конструкторы классов не наследуются, деструкторы могут наследоваться.

8.3. Принцип полиморфизма

Вернемся к классам, описывающим точки и окружности. В классе circle нужно переписать (переопределить) метод рисования – классы - потомки не всегда могут использовать неизмененные методы классов-предков. Но у метода рисования может быть оставлено тоже самое имя, в этом случае демонстрируется принцип полиморфизма – возможность называть методы, одинаковые по назначению одинаковыми именами. При работе с объектами существует два вида полиморфизма – *статический и динамический.*

Статический полиморфизм имеет тот же самый механизм, что и обыкновенная перегрузка функций в С, без требований к списку параметров. Статический полиморфизм носит название *механизма раннего связывания.* Еще на этапе компиляции известно, к какому классу относится объект, этот объект «жестко» связан со своими методами и полями.

Рассмотрим пример:

```
class a {
        private: int c;
        protected: float к;
        public :
                a();
                metod1();
                metod2();
                metod3();
}
```

```
Class b : public a {
        public:
                b();
                metod4();
                metod3();}
```

В данном случае полагается что для производного класс будет переопределяться metod3(). При компиляции программы все объекты описанные как a, будут связываться с методом класса a, все объекты описанные как b, будут связываться с методом класса b.

Динамический полиморфизм имеет место при работе со ссылками на объекты. Предположим, что в программе описана ссылка на объект базового класса, в процессе работы под объект может быть выделена память и как под объект базового класса и как под объект производного класса. Тогда при использовании статического полиморфизма произойдет связь с методами базового класса и при любом выделении памяти будет вызываться метод из базового класса.

Для решения этой проблемы используют *механизм позднего связывания:* переопределяемые функции в описании класса объявляют виртуальными (ключевое слово virtual). При компиляции таких классов в памяти компьютера создается таблица виртуальных методов vtbl. В этой таблице хранятся адреса всех виртуальных методов. При компиляции программы каждый объект дополняется ссылкой на vtbl – vptr, значение этой ссылки сам компилятор и заполняет. На этапе компиляции все вызовы виртуальных методов заменяются на значение vptr, а на этапе выполнения выбирается адрес из vtbl.

Рекомендуется делать виртуальными деструкторы, для того, чтобы гарантировать правильное освобождение памяти из под объекта, т.к. при этом будет вызван именно тот деструктор, который нужен.

```
class a {
        private: int c;
        protected: float к;
        public :
                a();
                metod1();
                metod2();
            virtual metod3();
}
```

```
Class b : public a {
        public:
                b();
                metod4();
        virtual metod3();}
```

8.4. Порядок выполнения работы

Напишите класс-наследник для класса графического изображения. Подумайте, нужно ли будет изменять класс, выполняющий действия над объектом. Продемонстрируйте механизмы раннего и позднего связываний. При работе над собственным классом можно воспользоваться следующим примером.

Опишем класс circle_ как производный от класса pixel. Поля x,y, color наследуются, поэтому в базовом классе их необходимо описать атрибутом protected. В производном классе появляется новое поле r – радиус окружности, новый метод getR – возвращение радиуса. Переписывается метод рисования и описывается как виртуальный.

```
class circle_ : public pixel
 {
  protected:
  int r;
  public:
    circle_(int a,int b, short c, int rad);
    int getR();
    virtual void draw(const c = -1);
 };
```

Обратите внимание на описание конструктора производного класса, такая запись обозначает, что вызывается конструктор базового класса, фактические параметры которого заменяют формальные параметры конструктора производного класса. Вызов конструктора базового класса обязателен.

```
circle_ :: circle_(int a, int b, short c, int rad) : pixel(a,b,c)
  {
   r = rad;
  }

 int circle_ :: getR()
  {
   return r;
```

```
 }
void circle_ :: draw(const int c)
{
if (c==-1)
 {
setcolor(color);
circle(x,y,r);
 }
else {setcolor(c);
       circle(x,y,r);}
}
```

Выполните изменения в выполняемом классе таким образом, чтобы выбор базового или производного объекта осуществлял пользователь. Попробуйте сделать функцию рисования не виртуальной. Объясните результат.

Лабораторная работа № 9. «Перегрузка операций»

9.1. Цель работы

Целью данной работы является изучение перегрузки операций в языке C++ и применение перегрузки в собственных классах.

9.2. Перегрузка операций в C++

9.2.1. Понятие перегрузки

Для наглядности при использовании классов можно переопределить операции (как бинарные, так и унарные). Например, написан класс работы с матрицами – необходимо сложить две матрицы одинаковой размерности, наглядно будет написать в теле программе c = a + b. Для того, чтобы такая запись была возможна, необходимо перегрузить, переопределить операции присвоения и сложения.

Синтаксис:

<тип результата операции> operator <оператор> (тип операнда);

Подразумевается, что одним из операндов обязательно будет объект класса-хозяина.

9.2.2. Перегрузка арифметических операций

Рассмотрим пример перегрузки операции сложения. Вернемся к описанию класса окружности. Перегрузим операцию сложения таким образом, чтобы запись a+<целое число>, где a – объект-окружность, обозначала увеличение радиуса объекта на заданное число.

Перегрузка должна быть добавлена в класс circle_ и может выглядеть следующим образом:

```
void circle_ :: operator + (int k)
  {
    r+=k;
  }
```

Перегрузка может не возвращать никакого результата, потому что по заданию изменяется объект, к которому применяется оператор сложения.

Если требуется получить другой объект класса с измененными параметрами необходимо перегрузить оператор присваиваивания.

9.2.3. Перегрузка операций << и >>.

Отдельно рассмотрим перегрузку операций вставки в поток и операций взятия из потока.

Операцию << перегружают для облегчения вывода объекта на экран. Стандартный поток вывода cout работает с текстовым экраном, поэтому целесообразнее пример перегрузки показывать на объектах, которые должны быть выведены на текстовый экран. Рассмотрим перегрузку на примере класса VECTOR, используемого на первом занятии.

В описание класса добавим:

friend ostream& operator << (**ostream& os**, VECTOR& a);

Перегрузка обязательно должна быть объявлена дружественной функцией, ostream& os – ссылка на класс-хозяин, который формируется внутри перегрузки, второй параметр – ссылка на объект, к которому применяется перегрузка.

Определим дружественную функцию, например следующим образом:

```
ostream& operator << (ostream& os,VECTOR& a)
 {
 os << "-----------------------------" << endl;
 os << "Size vector : " << a.n <<endl;
 os << "Elements vector : " << endl;
for (int i = 0; i< a.n; i++)
              {
               os << "vector[" << i << "] = " << a.vector[i] << endl;}
 os << "-----------------------------" << endl;
 return os;
}
```

После этого в программе могут быть записаны следующие строки:

```
…
VECTOR a(10);
Cout << a;
…
```

Аналогичные правила требуется применять и для перегрузки операции взятия из потока. Классом-хозяином в этом случае будет класс istream, типом

функции будет ссылка на объект, получаемый при выполнении перегрузки. Внутри перегрузки будет выполняться ввод элементов вектора с клавиатуры.

friend VECTOR& operator >> (istream &is, VECTOR& a);

9.3. Порядок выполнения работы

Получите задание на выполнение у преподавателя. Напишите предложенные перегруженные операторы для ранее написанных классов графических изображений. Продемонстрируйте работу перегруженных операторов.

Примечание: при перегрузке логических операторов выберите параметр или параметры объекта, по которым Вы будете проводить сравнение. Вспомните работу конструкции if и понятия лжи и истины в С.

10. Лабораторная работа № 10. «Создание массива объектов. Однородные объекты. Разнородные объекты.»

10.1. Цель работы

Научиться создавать динамические массивы объектов, используя для этого конструкторы копирования и перегруженную операцию присвоения.

10.2. Инструментарий C++ для создания копий объектов

10.2.1. Конструктор копирования

Для того, чтобы облегчить создание одинаковых объектов в С++ существует специальный вид конструкторов – конструктор копирования. Синтаксис конструктора копирования:

<Имя класса> (Ссылка на объект класса);

В теле конструктора копирования переопределяются все поля класса, которые необходимы для корректного создания объекта. Например, для объекта-точки переопределяются поля–координаты и поле цвета. Всегда нужно помнить, что если какие-либо поля остались не определенными, их значения по всем правилам С заменяются значениями, которые хранятся на данный момент в памяти компьютера, используемой для хранения этих полей. Пример описаний конструкторов копирования для классов pixel и circle_, эти описания добавляются в соответствующие классы:

```
pixel(pixel &a);
circle_( circle_ &a);
```

Сами конструкторы копирования могут выглядеть следующим образом:

```
pixel :: pixel(pixel &a)
    {
     x = a.x;
     y = a.y;
     color = a.color;
```

```
}
```

Значения полей нового объекта определяются как значения полей уже существующего объекта a.

```
Circle_ :: circle_(circle_ &a) : pixel(a)
 {
  r = a.r;
 }
```

В конструкторе копирования для класса circle_ остается переопределить только поле, задающее значение радиуса, потому что остальные поля определятся при вызове конструктора копирования базового класса.

Вызвать конструктор копирования можно следующим образом:

```
circle_ p1(x,200,4,50);
circle_ p2(p1);
```

После этого объект p2 – копия объекта p1. Т.е. окружность красного цвета, с координатами центра (x, 200) и радиусом 50.

10.2.2. Конструктор без параметров

Задача конструктора без параметров – создать экземпляр объекта с определенными значениями полей. Использоваться такой конструктор может для создания массива объектов заданного класса. Если конструктор имеет параметры, то он не может быть использован в операторе new для создания массива объектов. Например, при компиляции такой программы возникает синтаксическая ошибка:

```
…
pixel *a; // конструктор при таком описании не вызывается.
A = new pixel[20]; // синтаксическая ошибка, при выделении памяти под
```

20 объектов вызывается конструктор, у которого должны быть записаны при вызове три параметра.

Если бы в классе pixel был описан конструктор без параметров, такая запись была бы возможна. Но все 20 объектов, созданных при выделении памяти имели бы одинаковые значения полей координат и цвета.

Пример конструктора без параметров:

```
pixel:: pixel()
{ x = 20;
  y = 20;
  color = 0;
}
```

10.2.3.Перегрузка операции присваивания

Перегрузка операции присваивания решает ту же задачу, что и конструктор копирования и может использоваться при имеющихся перегруженных арифметических операциях.

Синтаксис перегрузки

<Ссылка на объект класса> operator = (Ссылка на объект класса)

Внутри перегруженного оператора присваивания выполняются те же действия, что и внутри тела конструктора копирования.

Например, для класса pixel конструктор копирования может быть записан следующим образом:

pixel& operator = (pixel &a); - описание

```
pixel& pixel :: operator = (pixel & a)
  {
   x = a.x;
   y = a.y;
   color = a.color;
   return *this;
  }
```

Переопределяются все поля класса, возвращается указатель на созданный объект.

Вызов перегруженного оператора присваивания может быть выполнен следующим образом:

```
pixel p1(20,20,3);
pixel p2 = p1;
```

10.3. Способы создания массива объектов

Рассмотрим несколько способов создания массивов объектов:

- Добавить в описание классов конструктор без параметров и перегруженный оператор присваивания. Описать указатель на объект класса, выделить память под нужное количество объектов. Для изменения объектов использовать перегруженный оператор присваивания.
- Описать в программе массив указателей на базовый класс, для каждого элемента массива выделять память под объект с заданными параметрами. Обратите внимание на то, что размер массива в этом случае должен задаваться константной величиной, но элементами такого массива могут быть как объекты базового класса, так и объекты производного класса.
- Использовать для создания массива списковую структуру, элементами которой будут указатель на следующий элемент списка и указатель на объект. При добавлении объекта в список выделяется память не только под элемент списка, но и память под объект. В таком случае длина массива может быть переменной и в массиве могут быть как объекты базового, так и объекты производного классов.

10.4. Порядок выполнения работы

По заданному варианту напишите программу, создающую массив объектов и продемонстрируйте на объектах массива передвижение объектов и (или) ранее сделанную перегрузку операторов.

11. Лабораторная работа № 11. «Создание класса для работы с файлами. Система меню»

11.1. Цель работы

Изучение и практическое применение классов ifstream, ofstream, fstream. Формирование навыков работы с разными типами файлов. Самостоятельная разработка класса на примере класса «Меню».

11.2. Потоки в C++

11.2.1. Стандартные потоки

Поток – это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику. Чтение данных из потока называется извлечением, вывод в поток – помещением или включением. Сам поток определяется как последовательность байтов и не зависит от конкретного устройства, с которым происходит обмен. По направлению обмена потоки разделяются на входные, выходные и двунаправленные. C++ поддерживает три вида потоков, на основе которых пользователь сам может строить потоки – стандартные, файловые, строковые.

Класс iostream помимо методов и функций при описании создает четыре своих стандартных объекта:

- cout – консольный поток вывода,
- cin, - консольный поток ввода,
- cerr – консольный поток сообщений об ошибках,
- clog – консольный поток сообщений об ошибках.

Операции << (вставка в поток) и >> (взятие из потока) являются перегруженными в классах istream (входные потоки) и ostream (выходные оттоки), наследуются всеми производными от них классами.

11.2.2.Файловые потоки

Файловые потоки могут быть входными(только для записи), выходными(только для чтения), потоки допускающие ввод и вывод (двунаправленные). В классе ios определены следующие режимы открытия файлов:

- ios :: app – открывает поток для записи в конец файла, пишет в конец;
- ios::ate – указатель потока установить в конец файла, пишет в текущую позицую;
- ios::binary – открыть поток в двоичном режиме;
- ios::in – открыть поток для ввода;
- ios::nocreate – генерирует ошибку, если файл не существует;
- ios::noreplace – устанавливает ошибку, если файл уже существует;
- ios::out – открыть для вывода;
- ios::trunc – очистить файл, если он уже существует

Общие методы для ввода-вывода в поток:

Метод ***open(имя файла, режим).***

Приведем примеры открытия файлов:

…

fstream f; // опишем объект класса fstream

f.open(“My.txt”,ios::in); // открыть файл My.txt для чтения

f.open(“MyFile”,ios::in|ios::binary); // открыть файл MyFile для чтения в двоичном (бинарном) режиме.

Если при открытии файла не указан флаг ios::binary файл по умолчанию считается текстовым.

Метод ***fail()*** – принимает значение 0, если не произошла ошибка в потоковых операциях.

Метод ***close()*** – закрыть файл .

Метод ***good()*** **–** принимает значение 0, если произошла какая-нибудь ошибка в потоке, в противном случае принимает любое ненулевое значение.

Метод ***eof()*** **–** принимает значение 0, если поток не достиг конца файла.

Перегруженный оператор ***!*** – определяет состояние ошибки (может быть записано**!f –** если f – объект потокового класса)

Последовательный ввод-вывод текста в поток.

Перегруженные операторы << и >>.

Метод ***getline(строка, максимально возможное количество символов строки, разделитель для строк)*** - разделитель для строк определен по умолчанию и равен ‘\n’, т.е. при вызове функции третий параметр может быть опущен.

Следующий фрагмент программы выводит в текстовый файл с именем a.out матрицу размером из n строк и m столбцов:

```
clrscr();
int n,m;
randomize();
cout << "Введите количество строк: ";
cin >> n;
cout << "Введите количество столбцов: ";
cin >> m;
ofstream f;
f.open("M1.txt",ios::out);
f<<n<<' ';
f<<m<<endl;
for(int i=0;i<n;i++)
  {
    for(int j=0;j<m;j++)
     { int x = random(100);
       f<< setw(4)<<x;
     }
     f<<endl;
}
```

Следующий фрагмент программы считывает матрицу из текстового файла:

```
clrscr();
int n,m;
randomize();
ifstream f;
f.open("M1.txt",ios::in);
f>>n;
f>>m;
int **x;
x = new int* [n];
for(int i=0;i<n;i++)
   x[i] = new int [m];
for(i=0;i<n;i++)
  {
    for(int j=0;j<m;j++)
     {
       f>>x[i][j];
    cout << setw(4) << x[i][j];
    }
      cout << endl;
  }
 getch();
for(i=0;i<n;i++)
   delete [] x[i];
delete [] x;
```

Последовательный двоичный режим потока ввода-вывода файлов.

Метод ***write****(указатель на массив символов, количество байт, которое нужно записать в поток).*

Метод ***read****(указатель на массив символов, количество байт, которое нужно прочитать из потока).*

Произвольный доступ в двоичных файлах.

Класс istream: метод ***seekg****(смещение в байтах, указание места, от которого происходит смещение).*

Класс ostream: метод ***seekp****(смещение в байтах, указание места, от которого происходит смещение).*

Второй аргумент в этих методах может принимать следующие значения:

- ios::beg – начало файла,
- ios::cur – текущая позиция файла,
- ios::end – конец файла.

Следующий пример сохраняет nxn чисел в двоичном файле:

```
clrscr();
int n;
randomize();
cout << "Введите количество строк: ";
cin >> n;
ofstream f;
f.open("M1.dat",ios::out|ios::binary);
for(int i=0;i<n;i++)
 {
 for(int j=0;j<n;j++)
    { int x = random(100);
      f.write((const char*)&x,sizeof(x));
    }
 }
 f.close();
```

Чтение матрицы из двоичного файла:

```
 clrscr();
 int n,m;
 randomize();
 ifstream f;

f.open("M1.dat",ios::in|ios::binary);
 f.read((char*)&n,sizeof(n));
 f.read((char*)&m,sizeof(m));
 int **x;
 x = new int* [n];
 for(int i=0;i<n;i++)
     x[i] = new int [m];
```

```
for(i=0;i<n;i++)
  {
    for(int j=0;j<m;j++)
     {
f.read((char*)&x[i][j],sizeof(x[i][j]));
cout << setw(4) << x[i][j];
     }
    cout << endl;
  }
  getch();
 for(i=0;i<n;i++)
   delete [] x[i];
delete [] x;
```

Обмен элементов первого столбца с элементами последнего столбца может быть запрограммирован следующим образом:

```
clrscr();
 int n,m;
 randomize();
 ifstream f;
 ofstream f1;
 f.open("M1.dat",ios::in|ios::binary);
 f1.open("M1.dat",ios::ate|ios::binary);
 f.seekg(0,ios::beg);
 f.read((char*)&n,sizeof(n));
 f.read((char*)&m,sizeof(m));
 int x,y;
 for(int i=0;i<n;i++)
  {
   for(int j=0;j<m;j++)
{
  f.read((char*)&x,sizeof(x));
  cout << setw(4) << x;
}
  cout << endl;
  }
  cout << endl;
```

```
long fs = 0+2*sizeof(n);
   long ls = fs+sizeof(x)*(m-1);
   int z;
   for (i=0;i<n;i++)
  {f.seekg(fs,ios::beg);
   f.read((char*)&x,sizeof(x));
   f.seekg(ls,ios::beg);
   f.read((char*)&y,sizeof(y));
   f1.seekp(fs,ios::beg);
   f1.write((char*)&y,sizeof(y));
   f1.seekp(ls,ios::beg);
   f1.write((char*)&x,sizeof(x));
   fs+=m*sizeof(x);
   ls+=m*sizeof(y); }
  f1.close();
  f.seekg(0+2*sizeof(n),ios::beg);
   for(i=0;i<n;i++)
    {for(int j=0;j<m;j++)
  {
   f.read((char*)&x,sizeof(x));
   cout << setw(4) << x;
  }
   cout << endl;
   }
   f.close();
```

Обратите внимание: один и тот же файл открыт как поток для ввода и редактирования.

11.3. Принципы проектирования меню

При выполнении данного практического задания предлагается выполнить работу с помощью текстового меню. Меню должно отвечать следующим требованиям:

- пользователь не должен вводить несколько раз одни и те же данные (например, имя файла);
- на протяжении всей работы программы на экране должна быть информация о выбранном на данный момент пункте меню;
- если в процессе работы меню программы по каким-то причинам удаляется с экрана, на экране должна быть информация о способе возврата в меню;
- по возможности меню должно предусматривать все некорректные действия со стороны пользователя и правильно обрабатывать такие ситуации (например, невозможно обрабатывать данные, которые еще не введены).

Для программирования меню в объектах Вы можете воспользоваться следующими схемами.

- Меню - текстовые строки, расположенные на экране вертикально. Текущий пункт меню выделен курсором в виде определенного символа или цветом. Для программирования такого меню можно предложить следующую иерархию классов. Класс для работы со строками содержит конструктор, задающий значение строки, защищенное поле для строки, поле для длины строки, метод вывода строки заданным цветом в заданном месте экрана. Класс меню содержит конструктор, задающий массив из объектов-строк, размерность этого объекта, цвета для активных и неактивных пунктов меню, координаты первого пункта меню, счетчик пунктов меню. Полями класса в таком случае могут быть массив из объектов строк, размерность массива, указатель текущего пункта меню, цвета меню, координаты меню. Методами такого меню могут быть – рисование всего меню, рисование отдельного пункта меню, управление меню.
- Меню – окна экрана, расположенные как вертикально, так и горизонтально. В этом случае могут быть разработаны два класса – класс для работы с окнами и класс, объединяющий окна в меню.

11.4. Порядок выполнения работы

Получите задание на практическое занятие. В соответствии с заданием разработайте класс для работы с файлом. Разработайте два или один класс для

меню и стартовый класс для выполнения программы. Класс для работы с файлом может содержать следующие поля и методы:

- поля, задающие размерность данных (матрицы или массива), хранящихся в файле;
- поле для хранения имени файла;
- конструктор, создающий бинарный файл;
- метод, выводящий на экран или в текстовый файл содержимое бинарного файла (или перегрузка оператора <<);
- метод, обрабатывающий файл в соответствии с заданием.
- деструктор, освобождающий выделенную память.

12.Лабораторная работа №12. «Разработка и создание собственного класса»

12.1. Цель работы

Комплексное применение теоретического материала, изученного в курсе «Объектно-ориентированный анализ и программирование» для создания пользовательского класса.

12.2. Основные моменты проектирования классов

Разработку любого класса необходимо начинать с изучения задачи, для решения которой разрабатывается класс.

Например, предлагается решить следующую задачу: *Неорграф задан матрицей смежности. Найти список ребер неорграфа.*

Объектом в этом примере будет граф. Граф характеризуется количеством вершин и количеством ребер, таким образом, выделены уже 2 поля класса. Но, для полного и однозначного описания графа двух этих данных недостаточно. Поэтому еще одним полем класса должна быть структура, описывающая граф. В предложенном примере это матрица смежности, которая будет третьим полем класса.

В задании не оговариваются способы задания графа. Предусмотрим следующие случаи:

- вся информация считывается из файла
- вся информация задается с клавиатуры
- вся информация передается с помощью двух констант и динамической матрицы языка С.

Так как за инициализацию объекта отвечают конструкторы то в классе необходимо предусмотреть 3 конструктора и перегруженный оператор взятия из потока для способа задания графа с клавиатуры.

Для вывода на экран информации о графе предусмотрим перегруженный оператор вставки в поток.

По условию задачи необходимо построить список ребер. Для этого предусмотрим метод, строящий список ребер и сохраняющий его в матрице $2 \times 2 \cdot m$, в первой строке которой записаны начала ребер, во второй – соответствующие им концы ребер.

Замечание: необязательно решение задачи будет методом разрабатываемого класса. Возможно, для решения задачи Вы будете только использовать написанные методы.

Так как в конструкторе выделяется память под матрицу смежности, в классе обязательно должен быть написан деструктор.

Описание класса в таком случае выглядит следующим образом:

```
class Graph
 {
  private:
      int n,m;
      int **a;
  public:
   Graph(int n1,int m1);
   Graph(int n1,int m1,int **a1);
Graph(char *name);
 int** List_R();
friend ostream& operator <<
(ostream & out, Graph &a);
friend istream& operator >>
(istream & input, Graph &a);
~Graph()
 };
```

Замечание: программа была бы более объектно-ориентированной, если бы в дополнение к классу, описывающему граф, был написан класс, описывающий матрицу и способы работы с ней.

Тексты методов класса и пример, работающий с графом из 4 вершин и 4 ребер, записаны в папке GRAPH. Для запуска программы создайте проект.

12.3. Порядок выполнения работы

Получите вариант для выполнения. Выделите объект или объекты, с которыми Вы будете работать. Разработайте структуру класса, исходя из поставленной задачи. Решите поставленную задачу с помощью методов класса.

Список рекомендуемой литературы

1. Павловская Т.А. С/С++: Программирование на языке высокого уровня: Учебник для вузов – Спб.: Питер, 2002. – 460 с.
2. Павловская Т.А. С++. Объектно-ориентированное программирование: практикум: учебное пособие для вузов – СПб.: Питер, 2005. – 464 с.
3. Леоненков А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM RATIONAL ROSE : Учебное пособие для вузов / А. В. Леоненков. - М. : Интернет-Университет Ин-

формационных Технологий, 2006 ; М. : БИНОМ. Лаборатория знаний, 2006. - 318[2] с.: ил.