Министерство образования и науки РФ

Санкт-Петербургский государственный
Электротехнический университет «ЛЭТИ»

Объектно-ориентированное программирование

Методические указания
к курсовому проектированию по дисциплине
«Объектно-ориентированное программирование»

Санкт-Петербург
Издательство СПбГЭТУ «ЛЭТИ»
2006

УДК 681.3.06

Объектно-ориентированное программирование: Методические указания к курсовому проектированию по дисциплине «Объектно-ориентированное программирование» / Сост. Г. В. Разумовский, А. Ф. Казак, С. А. Романенко, Д. В. Тихонов. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2006, 32 с.

Содержат требования по оформлению пояснительной записки к курсовому проекту по дисциплине «Объектно-ориентированное программирование», общие рекомендации по объектно-ориентированному анализу и проектированию программ и задания на курсовой проект.

Предназначено для студентов специальностей 220100.

Утверждено
редакционно-издательским советом университета
в качестве методических указаний

1. Цель и задачи курсового проектирования

Целью курсового проектирования является закрепление и углубление теоретических знаний, приобретение практических навыков по проектированию и разработке программного обеспечения на объектно-ориентированном языке.

В задачи курсового проектирования входят:

- изучение особенностей конкретной предметной области, относящейся к заданию на курсовой проект, и разработка технического задания на программный комплекс (ПК);
- объектно-ориентированное проектирование ПК с использованием языка UML;
- разработка ПК на объектно-ориентированном языке;
- написание программной документации.

2. Содержание пояснительной записки

Курсовой проект должен включать оттестированный ПК и пояснительную записку. Пояснительная записка проекта должна иметь следующую структуру:

- техническое задание;
- описание процесса проектирования ПК;
- руководство оператора;
- исходные тексты ПК.

3. Общие рекомендации по объектно-ориентированному анализу и проектированию ПК

Основная концепция объектно-ориентированного анализа и проектирования состоит в рассмотрении предметной области и логического решения задачи с точки зрения объектов. Процесс объектно-ориентированного проектирования – это построение моделей и их использование для описания различных аспектов ПК. Средством описания таких моделей является универсальный язык моделирования UML (Unified Modeling Language). Это язык для визуализации, описания, проектирования и документирования основных понятий программных систем. Он

стандартизует эти понятия и систему обозначений, а также является языком общения между программистами, аналитиками и заказчиками систем.

Процесс проектирования ПК можно разбить на следующие этапы:

1. Разработка технического задания.
2. Описание вариантов использования ПК.
3. Создание прототипа интерфейса пользователя.
4. Разработка объектной модели.
5. Построение диаграммы программных классов.
6. Описание поведения ПК.
7. Построение диаграмм действий.

3.1 Разработка технического задания

Процесс разработки ПК – это сумма видов деятельности, необходимых для преобразования требований пользователей в программную систему. Процесс определения требований к программному комплексу включает следующие шаги:

- перечисление общих требований (цель и назначение комплекса, стоимость, время разработки и другие);
- определение функциональных требований;
- определение нефункциональных требований (производительность, зависимость от платформы, надежность, расширяемость и другие).

Общие требования, функциональные требования высокого уровня и нефункциональные требования описываются в техническом задании. Такой тип описания используется на начальном этапе формулирования требований к программному комплексу.

Порядок построения и оформления технического задания на разработку программы или программного изделия для вычислительных машин, комплексов и систем независимо от их назначения и области применения описан в ГОСТ 19.201-78 «Техническое задание. Требование к содержанию и оформлению». Техническое задание в пояснительной записке должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;

- требования к программной документации;
- стадии и этапы разработки;
- порядок контроля и приемки.

В разделе «Введение» указывают наименование, краткую характеристику области применения ПК и объекта, в котором используют ПК.

В разделе «Основания для разработки» должно быть указано наименование и (или) условное обозначение темы разработки.

В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение ПК.

Раздел «Требования к программе или программному изделию» должен содержать следующие подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости.

В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т. п.

В подразделе «Требования к надежности» должны быть указаны требования к обеспечению надежного функционирования (обеспечения устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т.п.).

В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации, при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав технических средств с указанием их основных технических характеристик.

В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования и программным средствам, используемым программой.

В разделе «Требования к программной документации» должен быть указан предварительный состав программной документации и, при необходимости, специальные требования к ней.

В разделе «Стадии и этапы разработки» устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также сроки разработки и исполнители.

В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

3.2 Описание вариантов использования ПК

Развернутое описание функциональных требований осуществляется на этапе проектирования комплекса. Для того чтобы детализировать требования, необходимо выделить процессы, происходящие в заданной предметной области. Описание таких процессов на UML выполняется в виде прецедентов (use case). Прецеденты являются сценарием или вариантом использования ПК при взаимодействии с внешней средой. Они являются продолжением описаний требований и функциональных спецификаций, указанных в техническом задании. Прецедент изображается в виде эллипса, в котором содержится имя прецедента. Название прецедента обязательно включает в себя глагол, выражающий суть выполняемой функции. С помощью прецедентов описывается функционирование ПК с точки зрения внешнего пользователя, который называется в UML актором (actor). Актор представляет собой любую внешнюю по отношению к моделируемой системе сущность (человек, программная система, устройство), которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. Актор на диаграмме изображается пиктограммой в виде человечка, под которым указано его имя. Совокупность функций, реализуемых ПК, изображается в виде диаграммы (use case diagram). Для построения диаграммы необходимо определить акторы, прецеденты (функции) и взаимоотношение между акторами и прецедентами, и между прецедентами, если один прецедент расширяет или использует другой. В языке UML для вариантов использования и действующих лиц поддерживается несколько типов связей. Это связи коммуникации (communication), использования (uses) и расширения (extends).

Связь коммуникации — это связь между прецедентом и актором. На языке UML связь коммуникации изображают в виде стрелки. Направление стрелки показывает, кто инициирует коммуникацию. При задании коммуникации необходимо указать данные, которые вводит или получает пользователь. Кроме данных на концах стрелки можно указать кратности отношения, которые характеризуют количество взаимодействующих между собой акторов и прецедентов. На диаграммах прецедентов наиболее распространенными являются две формы записи кратности 1 и 1 .. *. Первая форма записи означает, что один актор (прецедент) участвует во взаимодействии, а вторая форма записи, что один или несколько акторов (прецедентов) участвуют во взаимодействии.

Связь использования предполагает, что один прецедент всегда применяет функциональные возможности другого. С помощью таких связей структурируют прецеденты, показывая тем самым, какой прецедент является составной частью другого прецедента. Такой включаемый прецедент является абстрактным прецедентом в том смысле, что он не может исполняться независимо от других прецедентов, а лишь в их составе. Связь использования изображается с помощью стрелок и слова «uses» (использование). Направление стрелки указывает, какой прецедент используется для реализации функциональности другого прецедента.

Связь расширения задается в том случае, если необходимо показать родственные отношения между двумя прецедентами. Один из них является базовым, а другой его расширением. Базовый прецедент не зависит от расширяющих прецедентов и способен функционировать без них. С другой стороны, расширяющие прецеденты без базового прецедента функционировать не могут. Связи расширения изображают в виде стрелки со словом «extends» (расширение), которая имеет направление от базового прецедента к расширяемому.

Прецеденты необходимо ранжировать, чтобы в начальных циклах разработки реализовать наиболее приоритетные из них. Разбиение функциональности системы на отдельные прецеденты служит примерно той же цели, что и разбиение сложного алгоритма на подпрограммы. Основная стратегия должна заключаться в том, чтобы сначала сконцентрировать внимание на тех прецедентах, которые в значительной мере определяют базовую архитектуру ПК.

Пример диаграммы прецедентов приведен на рис. 1. На этой диаграмме показаны функции ПК, предназначенного для составления расписания занятий. Из диаграммы прецедентов видно, что пользователями данного ПК являются диспетчер, методический отдел и старосты групп. Кроме того, этот ПК должен взаимодействовать с хранилищем данных (базой данных) и с другим ПК, который занимается учетом помещений. На диаграмме основные функции ПК показаны с помощью следующих прецедентов:

- «Печатает расписание»;
- «Просматривает расписание»;
- «Составляет расписание сессии»;
- «Составляет расписание занятий»;
- «Сохраняет данные для расписания».

3.3 Создание прототипа интерфейса пользователя

Описание прецедента выражает общую сущность процесса без детализации его реализации. Проектные решения, связанные с интерфейсом пользователя, при этом опускаются. Для разработки пользовательского интерфейса необходимо описать процесс в терминах реальных проектных решений, на основе конкретных технологий ввода-вывода информации. Когда речь идет об интерфейсе пользователя, прецеденты разбиваются на экранные формы, которые определяют содержимое диалоговых окон и описывают способы взаимодействия с конкретными устройствами. Для каждой экранной формы указываются поля ввода и перечень элементов управления, действия пользователя (нажать кнопку, выбрать пункт меню, ввести данные, нажать правую/левую кнопку мыши) и отклики системы (отобразить данные, вывести подсказку, переместить курсор). Такое описание интерфейса представляется в виде таблицы экранных форм. Пример фрагмента такой таблицы для ПК составления расписания представлен в табл.1. В него включены две экранные формы «Список заявок на экзамен» и «Карточка заявки на экзамен». Первая форма должна отображать таблицу с полями номер группы, название предмета, дата и время экзамена, а также содержать кнопки «Добавить», «Удалить», «Изменить». Вторая форма должна содержать поля ввода для номера группы, названия предмета, даты и времени экзамена и кнопки: «Выбрать предмет»,

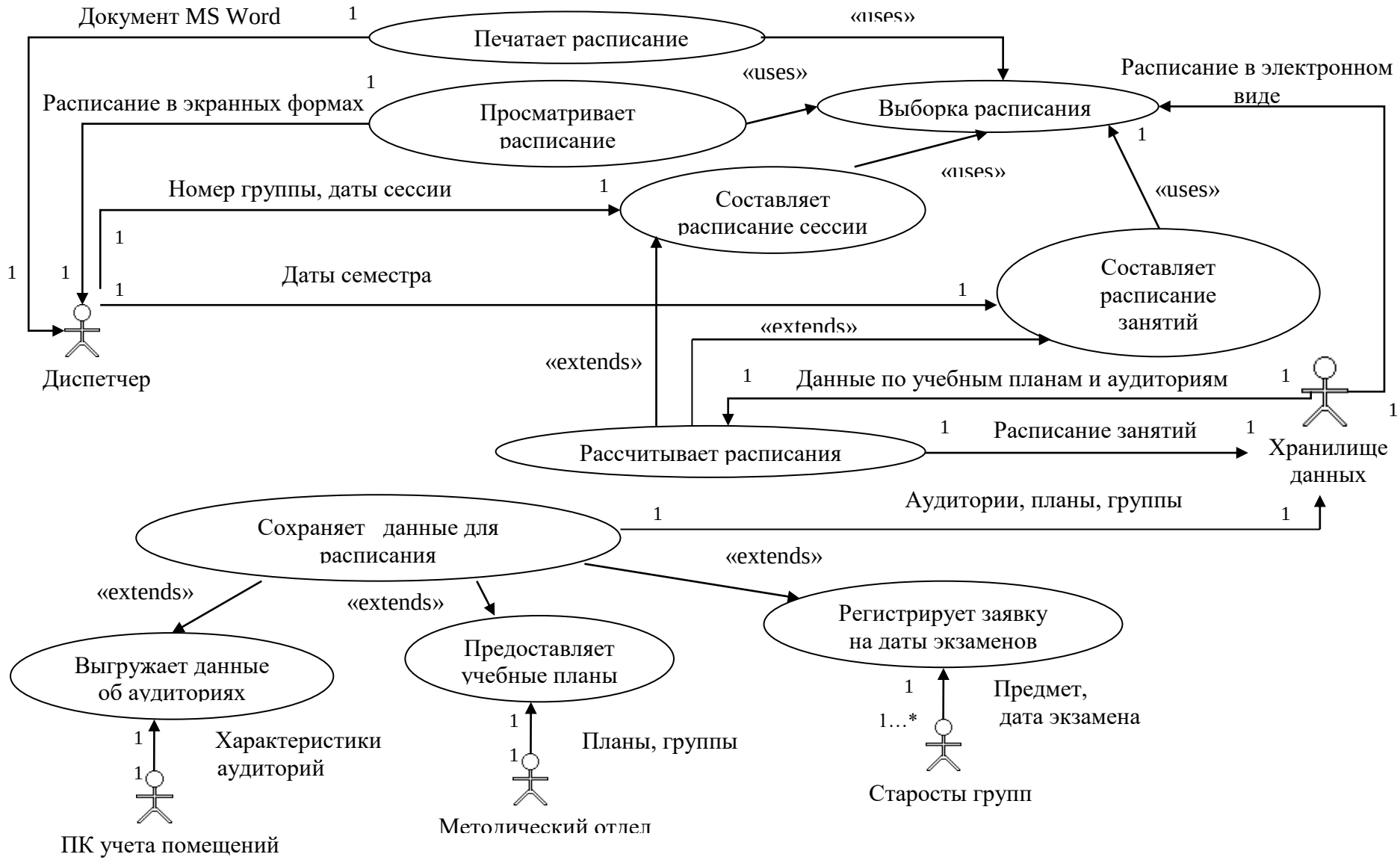
Документ MS Word
Печатает расписание
«uses»
Расписание в экранных формах
Просматривает расписание
Выборка расписания
Расписание в электронном виде
Номер группы, даты сессии
Составляет расписание сессии
Даты семестра
Составляет расписание занятий
Диспетчер
«extends»
Данные по учебным планам и аудиториям
Расписание занятий
Рассчитывает расписания
Хранилище данных
Аудитории, планы, группы
Сохраняет данные для расписания
Выгружает данные об аудиториях
Предоставляет учебные планы
Регистрирует заявку на даты экзаменов
Характеристики аудиторий
Планы, группы
Предмет, дата экзамена
ПК учета помещений
Методический отдел
Старосты групп

Рис. 1

«Сохранить» и «Отмена». Для каждой формы указывается последовательность действий пользователя и реакция ПК на эти действия.

Таблица 1

Экранная форма	**Элементы управления**	**Действия пользователя**	**Отклик системы**
Список заявок на экзамен	Таблица с полями: номер группы, название предмета, дата и время экзамена. Кнопки: «Добавить», «Удалить», «Изменить».	Создать новую заявку (нажать кнопку «Добавить»).	Открыть форму «Карточка заявки на экзамен».
Карточка заявки на экзамен	Поля ввода: номер группы, название предмета, дата и время экзамена. Кнопки: «Выбрать предмет», «Сохранить», «Отмена».	Нажать кнопку «Выбрать предмет»	Открыть форму со списком предметов. Заполнить поле «Предмет» в карточке заявки названием выбранного предмета.
		Нажать кнопку «Сохранить»	Добавить заявку в хранилище данных. Вывести сообщение с результатом сохранения заявки в хранилище данных (сохранена / не сохранена). Автоматически закрыть форму.

3.4 Разработка объектной модели ПК

Следующим шагом проектирования ПК является разработка объектной модели. Объектная модель не описывает структуру ПК, она отображает основные понятия предметной области в виде совокупности типов объектов (сущностей). В языке UML используется термин «класс», а не сущность, поэтому этот термин можно также применять при описании объектной модели. Однако следует отличать классы предметной области от программных классов, которые существуют в языках программирования. Сила языка UML заключается в том, что понятие «класс» можно использовать на разных уровнях абстракции описания ПК, при этом используется одна и та же система обозначений и терминология. На языке UML объектная модель представляется статической диаграммой классов (class diagram), в которой присутствуют сущности (классы), ее атрибуты и операции, и ассоциации между сущностями. Сущности строится путем выделения их из предметной области и анализа прецедентов. На диаграмме сущность обозначается прямоугольником, внутри которого записывается имя сущности, ее атрибуты и операции.

Атрибуты описывают свойства сущности. В объектную модель включаются те атрибуты, для которых определены соответствующие требования или для которых предполагается хранить определенную информацию. Атрибут характеризуется именем и типом. Для атрибута рекомендуется использовать простые типы данных (число, строка, дата, время и другие).

Описание операций помогает определить поведение объектов сущности. На этом этапе, прежде всего, определяется внутреннее поведение каждого объекта сущности, без учета взаимодействия с другими объектами предметной области. На диаграмме обычно указывается только имя операции, а ее подробное описание приводится в отдельной таблице. В таблице должно содержаться краткое описание назначения операции, ее имя и список входных и выходных параметров.

Ассоциация между сущностями отражает некоторое бинарное отношение между ними. Ассоциация обозначается проведенной между сущностями линией, с которой связывается определенное имя. Имя записывается в глагольной форме, и оно должно отражать семантический смысл отношения. Стрелка на линии указывает, в каком направлении нужно

читать имя. На концах линии могут содержаться выражения, определяющие количественную связь между экземплярами сущности (кратность). Кратность определяет, сколько экземпляров одной сущности может быть ассоциировано с одним экземпляром другой сущности. Примеры кратностей:

0 ..* - нуль или больше,

1 .. * - один или больше,

1 – ровно один.

Необходимо устанавливать отношения ассоциации между двумя сущностями в том случае, если объект одной сущности должен знать об объекте другой. Прежде всего, следует включать в модель те ассоциации, которые отражают структурные отношения («содержит», «включает», «хранит» и т.д.), или те, которые должны сохраняться в течение некоторого времени.

В качестве примера рассмотрим описание объектной модели ПК составления расписания. На диаграмме сущностей (рис.2) выделены понятия предметной области, их атрибуты и связи между ними, которые используются при составлении расписания. С точки зрения предметной области, наиболее важными являются такие сущности, как «Расписание», «Экзамен», «Преподаватель», «Дисциплина», «Аудитория», «Занятие», «Группа», «Заявка на экзамен», «План на семестр». Для каждой сущности описан состав атрибутов, раскрывающий внутреннюю структуру объекта. Так атрибутами сущности «Аудитория» являются ее тип, номер и вместимость. Между сущностями определены бинарные отношения, которые указывают, как они соотносятся друг с другом. Например, между сущностями «Преподаватель» и «Дисциплина» определено отношение «Читает лекции» с кратностью один ко многим, указывающее, что один преподаватель может читать лекции по нескольким дисциплинам, либо он не имеет лекционной нагрузки.

На диаграмме для некоторых сущностей введены операции, определяющие их поведение. Так для сущности «Заявка на экзамен» определены операции «Создать», «Установить предмет», «Установить дату» и «Установить время». Детальное описание первых двух операций приведено в табл. 2. В этой же таблице детально описываются некоторые операции сущности «Расписание».

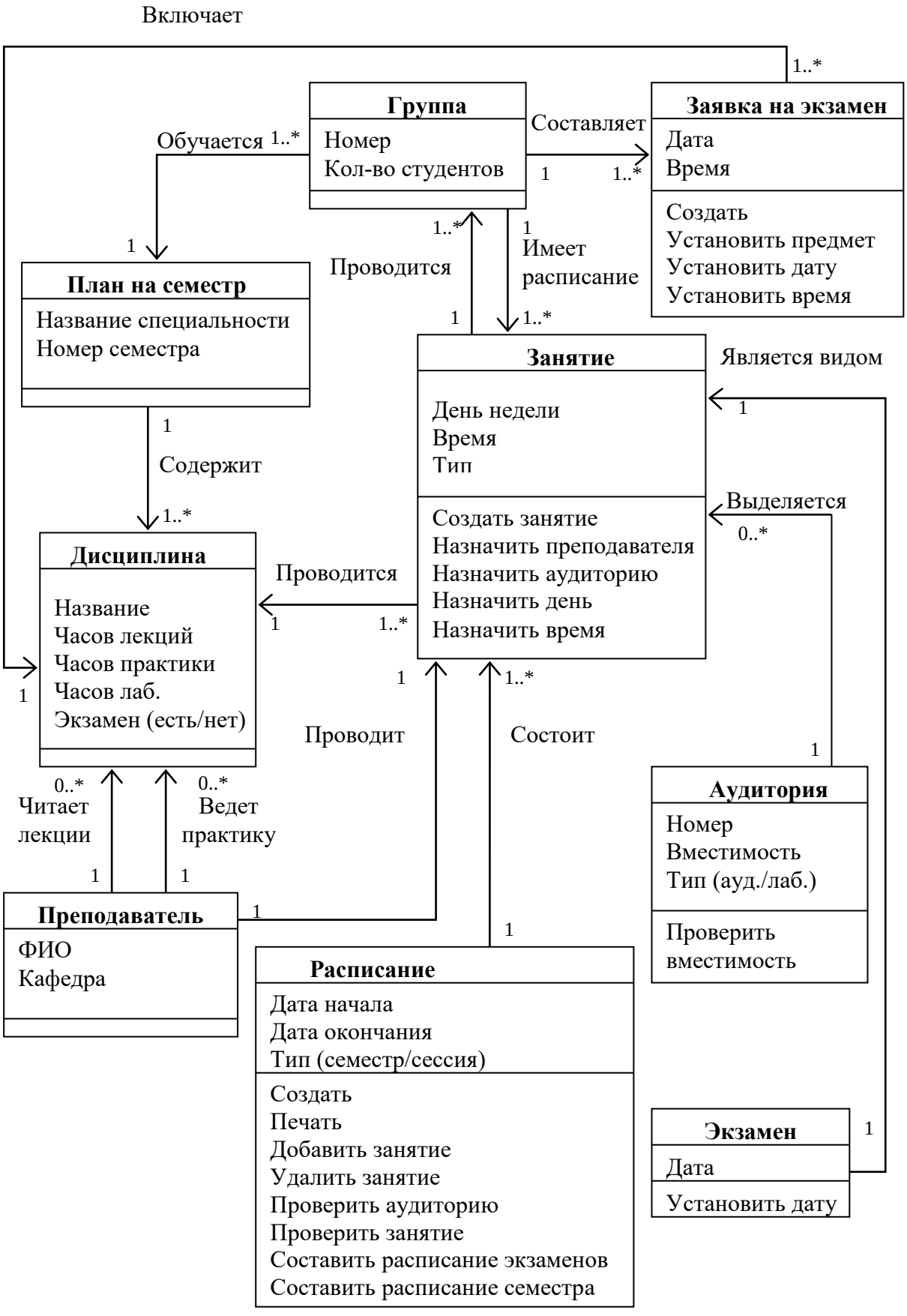
Включает
1..*
Группа
Номер
Кол-во студентов
Составляет
1
1..*
Заявка на экзамен
Дата
Время
Создать
Установить предмет
Установить дату
Установить время
Обучается
1..*
1
План на семестр
Название специальности
Номер семестра
1..*
Проводится
1
Имеет расписание
1..*
Занятие
День недели
Время
Тип
Создать занятие
Назначить преподавателя
Назначить аудиторию
Назначить день
Назначить время
Является видом
1
Выделяется
0..*
1
Содержит
1..*
Дисциплина
Название
Часов лекций
Часов практики
Часов лаб.
Экзамен (есть/нет)
Проводится
1
1..*
1
1..*
Проводит
Состоит
0..*
0..*
Читает лекции
Ведет практику
1
1
Преподаватель
ФИО
Кафедра
1
1
Аудитория
Номер
Вместимость
Тип (ауд./лаб.)
Проверить вместимость
Расписание
Дата начала
Дата окончания
Тип (семестр/сессия)
Создать
Печать
Добавить занятие
Удалить занятие
Проверить аудиторию
Проверить занятие
Составить расписание экзаменов
Составить расписание семестра
Экзамен
Дата
Установить дату
1

Рис. 2

Таблица 2

Сущность	**Имя операции**	**Параметры операции**			**Тип возвраща-емого значения**	**Назначение операции**
		Вид	**Название**	**Тип**		
Заявка на экзамен	Создать	Вх.	Номер группы	Строка	Заявка	Создает пустую заявку на экзамен и назначает номер группы
	Установить предмет	Вх.	Предмет	Предмет	Пусто	Устанавлив-ливает учебный предмет в заявке на экзамен
Расписание	Создать	Вх.	Дата начала	Дата	Расписание	Создает пустое расписание с заданными датами начала и окончания
		Вх.	Дата окончания	Дата		
	Печать	Вх.	Номер группы	Строка	Логический	Печатает фрагмент расписания по заданным параметрам
		Вх.	Дата начала	Дата		
		Вх.	Дата окончания	Дата		
	Составить расписание экзаменов	Вх.	Список аудиторий	Список аудиторий	Логический	Составляет расписание экзаменов по заданным заявкам на экзамены и списку доступных аудиторий. Через выходной параметр возвращает список созданных объектов типа «Экзамен»
		Вх.	Список заявок на экзамен	Список Заявок на экзамены		
		Вых.	Список сформированных занятий типа «Экзамен»	Список Экзаменов		

3.5 Построение диаграммы программных классов

Диаграмма классов (class diagram) иллюстрирует спецификации будущих программных классов и интерфейсов. Она строится на основе объектной модели. В описание класса указываются три раздела: имя класса, состав компонентов класса и методы класса. Графически класс изображается в виде прямоугольника. Имя программного класса может совпадать с именем сущности или быть другим. Но поскольку для записи идентификаторов переменных в языках программирования используют латинские буквы, то и имена программных классов и имена их атрибутов, как правило, записываются латинскими буквами. Атрибуты и операции класса перечисляются в горизонтальных отделениях этого прямоугольника. Атрибутам и методам классов должны быть присвоены права доступа. Права доступа помечаются специальными знаками:

\+ - означает открытый (public) доступ;

— - означает скрытый (private) доступ;

- означает наследуемый (protected) доступ.

При описании атрибутов после двоеточия указывается их тип, а при описании методов класса возвращаемое значение (для конструкторов возвращаемое значение не указывается).

В диаграмме классов могут вводиться дополнительно новые атрибуты, операции и связи или осуществляться конкретизация ассоциаций, указанных в объектной модели. На диаграмме классов могут быть три вида отношений: ассоциация, агрегация и наследование.

На диаграмме классов ассоциация имеет такое же обозначение, как и в объектной модели. На линиях ассоциации может присутствовать стрелка. Это стрелка видимости, которая показывает направление посылки запросов в ассоциации. Стрелка видимости также показывает, какой из классов содержит компоненты для реализации отношения ассоциации, иными словами, кто является инициатором посылки запроса к другому объекту. Ассоциация без стрелки является двунаправленной.

Агрегирование - это отношение между классами типа целое/часть. Агрегируемый класс в той или иной форме является частью агрегата. На практике это может быть реализовано по-разному. Например, объект класса-агрегата может хранить объект агрегируемого класса, или хранить ссылку на него. Агрегирование изображается на диаграмме полым ромбом на конце линии со стороны агрегирующего класса (агрегата). Если агрегируемый

объект может быть создан только тогда, когда создан агрегат, а с уничтожением агрегата уничтожаются и все агрегируемые объекты, то такое агрегирование называется сильным и отображается в виде закрашенного ромба.

Наследование - это отношение типа общее-частное между классами. Его следует вводить в том случае, когда поведение и состояние различных классов имеют общие черты. Наследование связывает конкретные классы с общими или в терминологии языков программирования производные классы (подклассы) с базовыми классами (суперклассами). На диаграммах наследование изображается в виде стрелки с полым треугольником, идущей от производного класса к базовому. Если один производный класс наследует несколько базовых, то такое наследование называется множественным.

Графические изображения перечисленных выше отношений показаны на рис. 3.

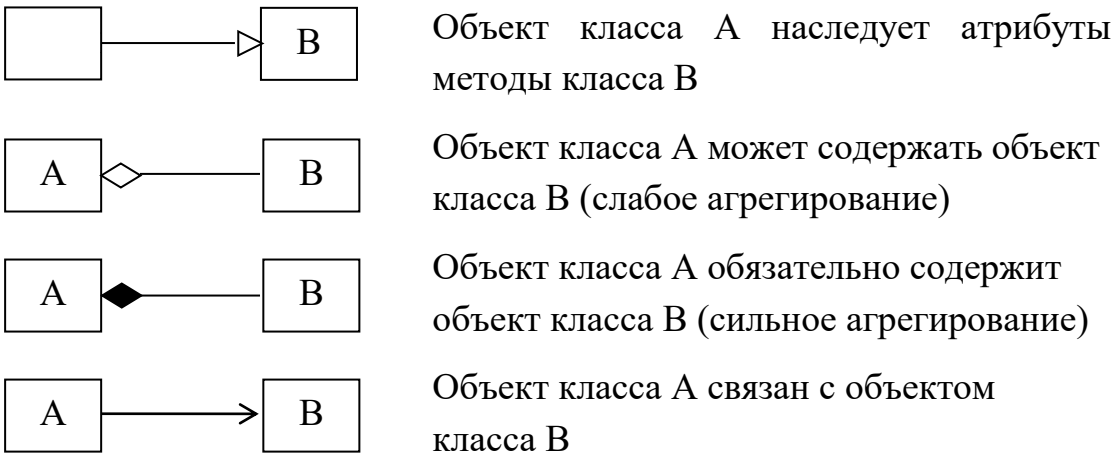

Рис. 3

На рис. 4 приведена диаграмма программных классов для ПК составления расписания. На этой диаграмме, по сравнению с диаграммой сущностей, появились два новых класса DBDescriptor и addExamOrderForm. Класс DBDescriptor предназначен для взаимодействия ПК с базой данных, а класс addExamOrderForm – для реализации экранной формы, с помощью которой осуществляется добавление заявки. В диаграмме присутствуют все виды связей. Например, объекты класса Exam (соответствует сущности «Экзамен») наследуют атрибуты и методы класса Lesson (соответствует сущности «Занятие»), объект класса Plan (соответствует сущности «План на

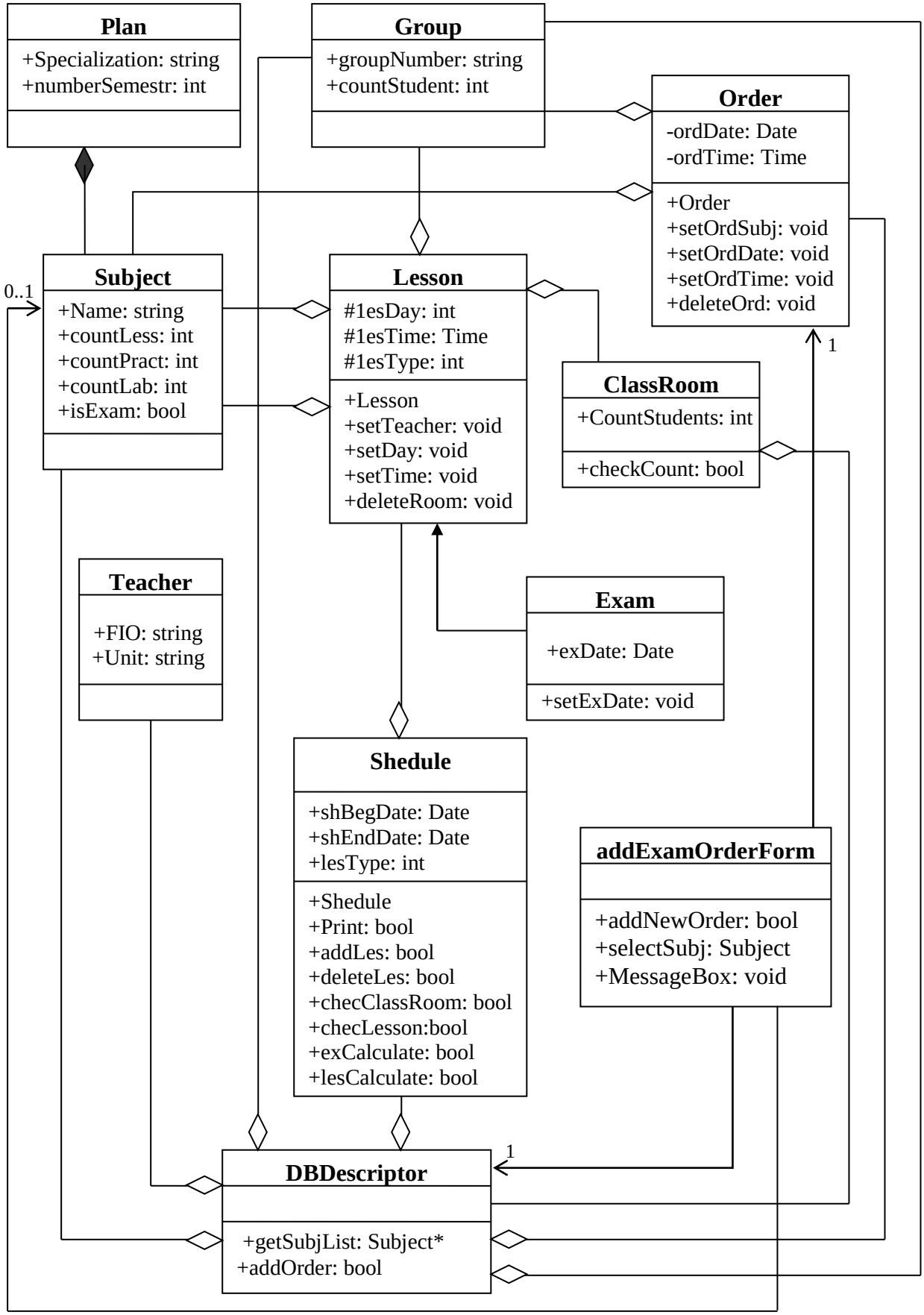
Plan
+Specialization: string
+numberSemestr: int
Group
+groupNumber: string
+countStudent: int
Order
-ordDate: Date
-ordTime: Time
+Order
+setOrdSubj: void
+setOrdDate: void
+setOrdTime: void
+deleteOrd: void
Subject
+Name: string
+countLess: int
+countPract: int
+countLab: int
+isExam: bool
0..1
Lesson
#1esDay: int
#1esTime: Time
#1esType: int
+Lesson
+setTeacher: void
+setDay: void
+setTime: void
+deleteRoom: void
ClassRoom
+CountStudents: int
+checkCount: bool
1
Teacher
+FIO: string
+Unit: string
Exam
+exDate: Date
+setExDate: void
Shedule
+shBegDate: Date
+shEndDate: Date
+lesType: int
+Shedule
+Print: bool
+addLes: bool
+deleteLes: bool
+checClassRoom: bool
+checLesson:bool
+exCalculate: bool
+lesCalculate: bool
addExamOrderForm
+addNewOrder: bool
+selectSubj: Subject
+MessageBox: void
DBDescriptor
+getSubjList: Subject*
+addOrder: bool
1

Рис. 4

семестр») обязательно агрегирует объекты класса Subject (соответствует сущности «Дисциплина») и каждый объект класса addExamOrderForm связан с одним объектом класса Order (соответствует сущности «Заявка на экзамен»).

3.6 Описание поведения ПК

Поведение ПК представляет собой описание того, какие действия выполняет ПК, без определения механизма их реализации. Одной из составляющей такого описания является диаграмма последовательностей (seguence diagram). Диаграмма последовательностей является схемой, которая для определенного сценария прецедента показывает генерируемые пользователями и объектами события (запросы) на выполнение некоторой операции и их порядок. Диаграммы последовательности имеют две размерности: вертикальная представляет время, горизонтальная - различные объекты. Чтобы построить диаграмму последовательностей необходимо выполнить следующие действия:

1. Идентифицировать пользователей и объекты программных классов, участвующие в начальной стадии реализации сценария прецедента, и их изображения в виде прямоугольников расположить наверху в одну линию. Для каждого пользователя и объекта нарисовать вертикальную пунктирную линию, которая является линией их жизни. Внутри прямоугольника указываются подчеркнутое имя объекта и имя класса, к которому принадлежит объект.
2. Из объектной модели выбрать те операции, которые участвуют в реализации сценария. Если такие операции не были определены при построении диаграммы программных классов, то необходимо их описать и внести в модель.
3. На диаграмме последовательностей каждому запросу на выполнение операции должна соответствовать горизонтальная линия со стрелкой, начинающаяся от вертикальной линии того пользователя или объекта, который вызывает операцию, и заканчивающаяся на линии жизни того пользователя или объекта, который будет ее выполнять. Над стрелкой указывается номер операции, число итераций, имя операции и в скобках ее параметры. После описания операции может следовать комментарий, поясняющий смысл операции и начинающийся со знака "//".

Операция, которая реализует запрос, на линии жизни объекта обозначается прямоугольником. Порядок выполнения операций определяется ее номером, который указывается перед именем, и положением горизонтальной линии на диаграмме. Чем ниже горизонтальная линия, тем позже выполняется операция. В диаграммах последовательности принято применять вложенную систему нумерации, так как это позволяет отобразить их вложенность. Нумерация операций каждого уровня вложенности должна начинаться с 1.

Чтобы указать, что операция повторяется, следует задать после ее порядкового номера в квадратных скобках итеративное выражение. В выражении можно использовать звездочку (*) для указания того, что операция вызывается более чем один раз, или явно задать количество раз повторений (например, 1.. 10 или 1.. количество экземпляров объекта).

На диаграмме последовательностей можно описать вызов операции по условию (конструкция if-else) и показать моменты создания и уничтожения объектов. Если объект создается или уничтожается на отрезке времени, представленном на диаграмме, то его линия жизни начинается и заканчивается в соответствующих точках, в противном случае линия жизни объекта проводится от начала до конца диаграммы. Символ объекта рисуется в начале его линии жизни; если объект создается не в начале диаграммы, то сообщение о создании объекта рисуется со стрелкой, проведенной к символу объекта. Если объект уничтожается не в конце диаграммы, то момент его уничтожения помечается большим крестиком "X".

Пример диаграммы последовательности приведен на рис. 5. На этой диаграмме раскрыта последовательность вызовов операций, реализующих метод addNewOrder (добавления заявки на экзамен) класса addExamOrderForm. В реализации этого метода участвуют три объекта: orderRegistrate, examOrd и DBDescr, причем объект examOrd создается и уничтожается в процессе выполнения метода addNewOrder. Этот метод реализуется с помощью вызовов 9 операций: из них 5 операций с номерами 1.1, 1.4, 1.5, 1.6 и 1.9 принадлежат объекту examOrd, 2 операции с номерами 1.3 и 1.8.1 (или 1.8.2) принадлежат объекту orderRegistrate и 2 операции с номерами 1.2 и 1.7 принадлежат объекту DBDescr. Обе операции с номерами 1.8.1 и 1.8.2 являются альтернативными, но вызывают один и тот же метод MessageBox с разными параметрами в зависимости от условия, указанного в начале описания операции с номером 1.8.1.

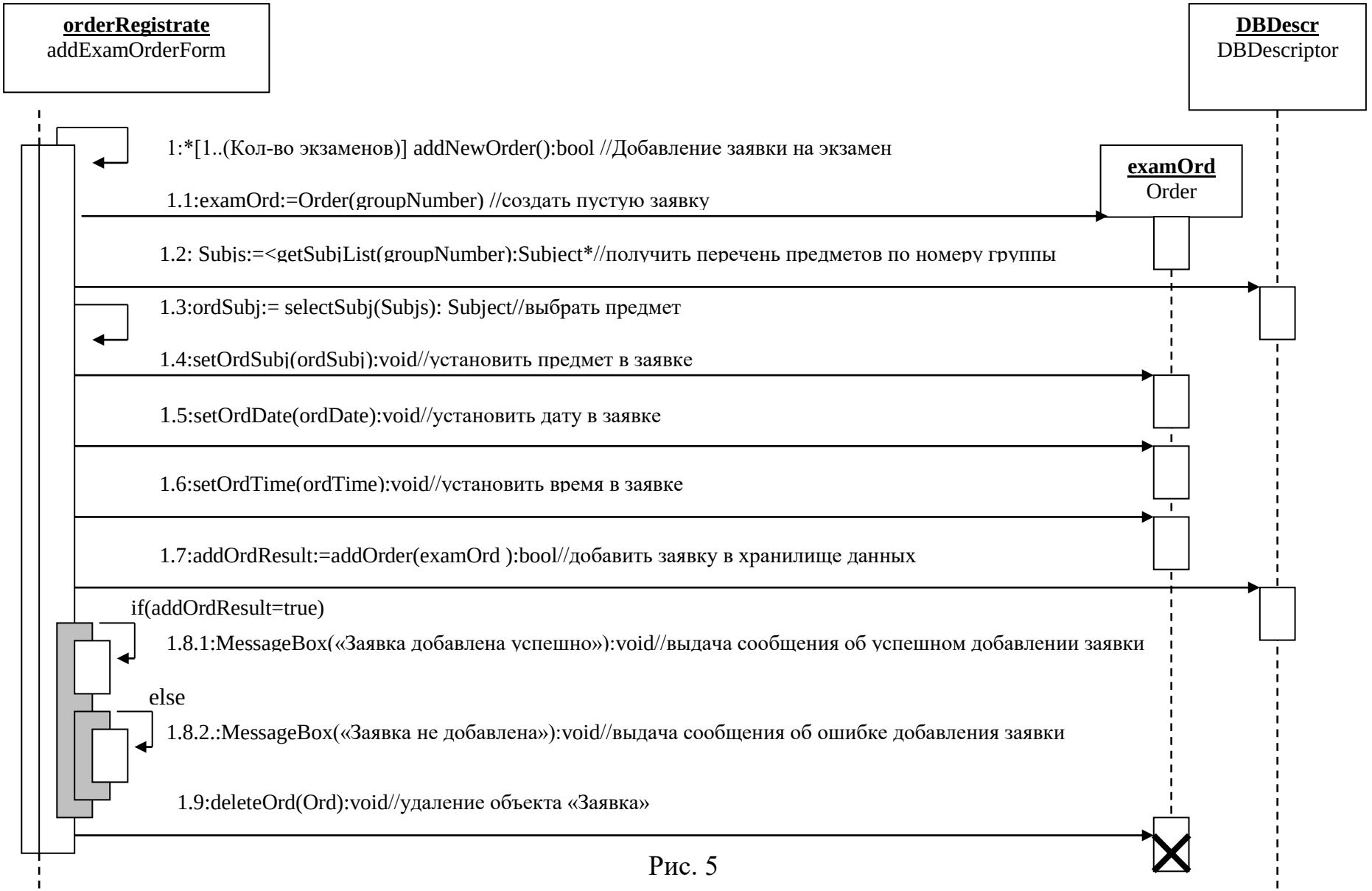
orderRegistrate
addExamOrderForm
DBDescr
DBDescriptor
examOrd
Order
1:*[1..(Кол-во экзаменов)] addNewOrder():bool //Добавление заявки на экзамен
1.1:examOrd:=Order(groupNumber) //создать пустую заявку
1.2: Subjs:=<getSubjList(groupNumber):Subject*//получить перечень предметов по номеру группы
1.3:ordSubj:= selectSubj(Subjs): Subject//выбрать предмет
1.4:setOrdSubj(ordSubj):void//установить предмет в заявке
1.5:setOrdDate(ordDate):void//установить дату в заявке
1.6:setOrdTime(ordTime):void//установить время в заявке
1.7:addOrdResult:=addOrder(examOrd):bool//добавить заявку в хранилище данных
if(addOrdResult=true)
1.8.1:MessageBox(«Заявка добавлена успешно»):void//выдача сообщения об успешном добавлении заявки
else
1.8.2.:MessageBox(«Заявка не добавлена»):void//выдача сообщения об ошибке добавления заявки
1.9:deleteOrd(Ord):void//удаление объекта «Заявка»

Рис. 5

3.7 Построение диаграммы действий

Диаграмма действий (activity diagram) строится для сложных операций. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются действия, а дугами — переходы от одного действия к другому. Она очень похожа на блок-схемы алгоритмов. Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояние. Диаграмму деятельности принято строить таким образом, чтобы действия следовали сверху вниз. Отличительной чертой диаграммы действий является то, что в ней можно отобразить параллельные процессы. Для этой цели используется специальный символ (линия синхронизации), который позволяет задать разделение и слияние потоков управления. При этом разделение имеет один входящий переход и несколько выходящих, а слияние, наоборот, имеет несколько входящих переходов и один выходящий.

Для построения диаграммы используются пиктограммы «действие», «переход», «выбор» и «линии синхронизации» и другие. Эти пиктограммы представлены на рис. 6.

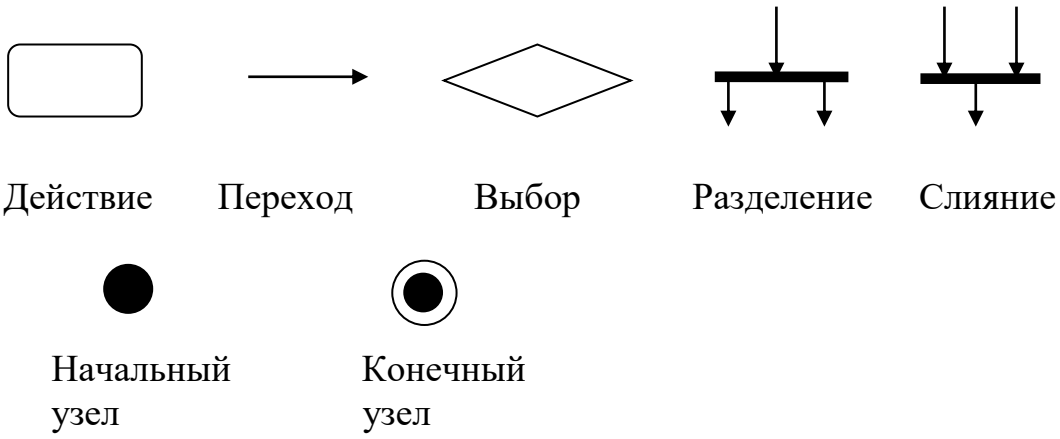

Рис. 6

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий. При этом действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Чтобы связать объекты с действиями,

необходимо явно указать их на диаграмме деятельности. Для графического представления объектов, используются прямоугольник, в котором указывается подчеркнутое имя класса. Подчеркнутое имя означает, что на диаграмме задается объект, а не его класс. Далее после имени можно указать в прямых скобках значения атрибутов объекта после выполнения предшествующего действия. Такие прямоугольники объектов присоединяются к переходам отношением зависимости с помощью пунктирной линией со стрелкой.

На рис. 7 представлена диаграмма действий, описывающая алгоритм выполнения операции составления расписания экзаменов. Алгоритм реализует цикл обработки заявок на экзамен с учетом даты, количества студентов в группе и вместимости аудитории. В этой диаграмме задействованы следующие объекты: «Заявка», «Аудитория», «Расписание», «Экзамен».

3.8 Средства построения UML диаграмм

Средства построения UML-диаграмм можно разделить на две группы:

1. Графические редакторы.
2. CASE-средства для проектирования и разработки программного обеспечения.

Графические редакторы позволяют построить UML-диаграммы как отдельные картинки. Для построения диаграмм можно использовать любой графический редактор, позволяющий изображать геометрические фигуры, линии и надписи. Примерами таких редакторов являются Microsoft Paint и Microsoft Word.

Существуют графические редакторы, которые поддерживают спецификацию UML и предоставляют возможность размещать на картинке готовые пиктограммы различных видов UML-диаграмм. Примером такого графического редактора является программное средство Microsoft Visual Studio, входящее в состав Microsoft Office. Графические редакторы просты в использовании, но не проверяют корректность диаграмм и не связаны и не согласованы с исходным кодом программы.

В качестве основных CASE-средств объектно-ориентированного проектирования программного обеспечения можно назвать продукты Rational

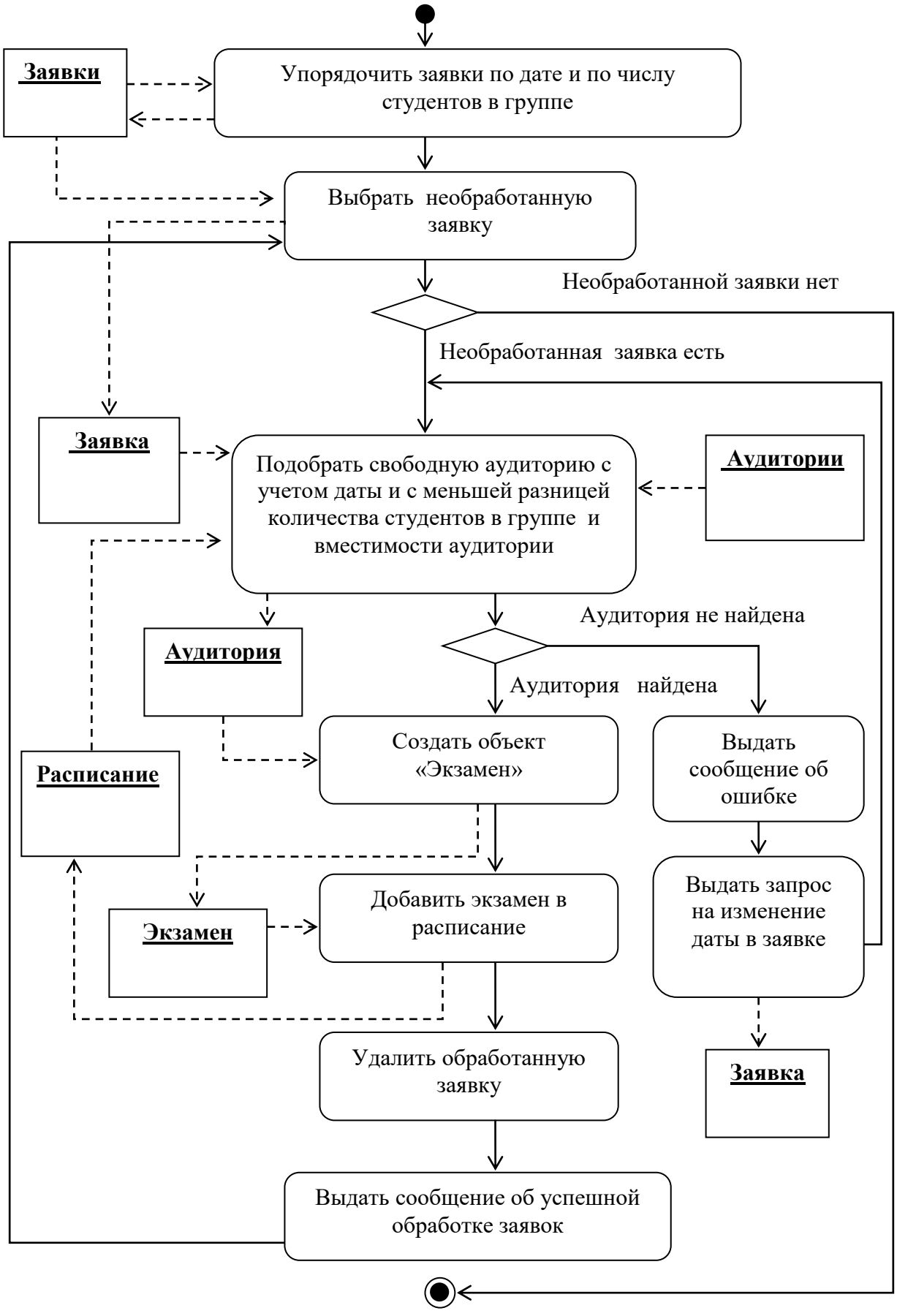
Заявки
Упорядочить заявки по дате и по числу студентов в группе
Выбрать необработанную заявку
Необработанной заявки нет
Необработанная заявка есть
Заявка
Подобрать свободную аудиторию с учетом даты и с меньшей разницей количества студентов в группе и вместимости аудитории
Аудитории
Аудитория не найдена
Аудитория
Аудитория найдена
Создать объект «Экзамен»
Выдать сообщение об ошибке
Расписание
Добавить экзамен в расписание
Выдать запрос на изменение даты в заявке
Экзамен
Удалить обработанную заявку
Заявка
Выдать сообщение об успешной обработке заявок

Рис. 7

Rose компании Rational Software Corporation и Together компании Object International. Они предоставляют удобную среду для разработки и графического представления UML-диаграмм, поддержку нескольких языков программирования, проверку целостности и взаимной непротиворечивости UML-диаграмм, автоматическую генерацию исходного кода по диаграммам и обратный инжиниринг – построение UML-диаграмм по исходному коду программы. Их различия заключаются, в основном, в принципах организации базы данных проекта и в работе с исходным кодом на языке программирования. CASE-средства очень мощные инструменты, но они не русифицированы и для их освоения и использования требуется большая квалификация программиста.

При использовании автоматизированных средств построения UML-диаграмм необходимо помнить, что они могут не полностью поддерживать стандартную графическую нотацию языка UML и в каждом из них существуют свои особенности. В курсовом проектировании можно использовать любое из перечисленных выше средств построения UML-диаграмм, но наиболее простыми и доступными являются Microsoft Word и Microsoft Visual Studio.

4. Программная документация

Документация на ПК может быть представлена в двух видах: техническая и эксплутационная. К технической документации относятся:

- техническое задание (ГОСТ 19.201-78), определяющее требования, предъявляемые к ПК, необходимые стадии и сроки разработки, виды испытаний;
- спецификация (ГОСТ 19.202-78), содержащая состав ПК и документации на него;
- программа и методика испытаний (ГОСТ 19.301-79), содержащая требования, подлежащие проверке при испытании ПО, а также порядок и методы их контроля;
- тексты программ (ГОСТ 19.401-78) - записи программ с необходимыми комментариями;
- описание программы (ГОСТ 19.402-78), в котором содержатся сведения о логической структуре и функционировании ПК;

- пояснительная записка (ГОСТ 19.201-78), включающая общее описание алгоритмов и функционирования ПК.

Эксплутационная документация может включать в себя следующий комплект документов:

- ведомость эксплуатационных документов (ГОСТ 19.507-79) - перечень эксплуатационных документов на ПК;
- формуляр (ГОСТ 19.501-78), который определяет основные характеристики ПК, комплектность и сведения об эксплуатации;
- описание применения (ГОСТ 19.502-78), в котором содержаться сведения о назначении, области применения ПК, методах и классе решаемых задач;
- руководство системного программиста (ГОСТ 19.503-79), включающее сведения для проверки, обеспечения функционирования и настройки программ на условия конкретного приложения;
- руководство программиста (ГОСТ 19.504-79), содержащее сведения, необходимые для эксплуатации ПК;
- руководство оператора (ГОСТ 19.505-79) - сведения, необходимые для обеспечения процедуры общения оператора с ЭВМ в процессе выполнения (работы) ПК;
- руководство по техническому обслуживанию (ГОСТ 19.508-79) - содержит описание применяемых тестовых и диагностических программ при обслуживании технических средств.

Структура и оформление руководство оператора устанавливается в соответствии с ГОСТ 19.105-78. Оно должно содержать следующие разделы:

- назначение программы;
- условия выполнения программы;
- описание задачи;
- входные и выходные данные.
- выполнение программы;
- проверка программы;
- сообщения оператору.

В разделе «Назначение программы» должны быть указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

В разделе «Условия выполнения программы» должны быть указаны условия, необходимые для выполнения программы (минимальный и (или) максимальный состав аппаратурных и программных средств и т.п.).

В разделе «Описание задачи» должны быть указаны определения задачи и методы ее решения.

В разделе «Входные и выходные данные» должны быть указаны сведения о входных и выходных данных.

В разделе «Выполнение программы» должна быть указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузки и управляет выполнением программы, а также ответы программы на эти команды.

В разделе «Проверка программы» должны быть приведено описание способов проверки, позволяющих дать общее заключение о работоспособности программы (контрольные примеры, методы прогона, результаты).

В разделе «Сообщения оператору» должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска программы и т.п.).

Допускается содержание разделов иллюстрировать поясняющими примерами, таблицами, схемами, графиками. Приложение к руководству оператора должно содержать тексты программ.

5. Задания на курсовое проектирование

В курсовом проекте должен быть спроектирован и разработан на объектно-ориентированном языке ПК с учетом требований, указанных в задании. ПК должен быть разработан, как приложение ОС Windows с оконным интерфейсом, и обеспечивать хранение данных в файле или базе данных (например, Access), используя драйвер ODBC. По согласованию с преподавателем студент может выполнить другое задание или изменить заданное. Обязательными требованиями при разработке кода ПК являются использование следующих конструкций языка C++:

- закрытые и открытые члены классов;

- наследование;
- конструкторы с параметрами и копирования;
- деструкторы;
- абстрактные базовые классы;
- виртуальные функции;
- обработка исключительных ситуаций;
- динамическое создание объектов;

Рекомендуется также включить в программу дружественные функции, переопределение функций или операторов, шаблонные классы.

Задание 1. Разработать ПК для работников библиотеки. В ПК должны храниться сведения об имеющихся в библиотеке книгах и о читателях библиотеки. Библиотекарю могут потребоваться следующие сведения:

- какие книги закреплены за читателем;
- кто автор и как называется книга с заданным шифром.

Библиотекарь может вносить следующие изменения:

- запись нового читателя в библиотеку;
- пополнение библиотеки;
- списывание старой книги;
- изменение шифра книги.

Необходимо предусмотреть возможность выдачи справки о количестве читателей библиотеки и работе библиотеки за месяц (количество выданных книг, число записавшихся читателей).

Задание 2. Разработать ПК для завуча школы. В ПК должны храниться сведения об учениках школы, учителях, предметах, которые они преподают. Завуч может добавлять, изменять и удалять информацию о учителе или ученике. Завучу могут потребоваться следующие сведения:

- какие преподаватели ведут тот или иной предмет;
- в каких классах преподает учитель;
- количество учеников в школе и в указанном классе;
- отчет об успеваемости в школе (количество отличников, двоечников с указанием их фамилий).

Задание 3. Разработать ПК для работников приемной комиссии. В ПК должны храниться сведения об абитуриентах, датах экзаменов и номерах аудиторий. Работник приемной комиссии может вносить и удалять

информацию об абитуриенте, даты экзаменов и номера аудиторий. Ему могут потребоваться следующие сведения:

- список абитуриентов по факультетам;
- полученные абитуриентом оценки за экзамены;
- дата экзамена и номер аудитории, где будет проводиться экзамен по данному предмету;
- количество абитуриентов, сдавших экзамены на 2, 3, 4, 5 по предметам;
- отчет о работе приемной комиссии (количество абитуриентов на каждом факультете и в какие дни и где проводятся экзамены).

Задание 4. Разработать ПК для работников почты. В ПК должна храниться информация о клиентах почты, о почтальонах и о газетах (журналах). Работник почты может добавлять, изменять и удалять эту информацию. Ему могут потребоваться следующие сведения:

- какие клиенты выписывает данную газету;
- какая номенклатура газет и в каком количестве они поступают на почту;
- справка о почтальонах и отчет об их работе (адреса клиентов, по которым они разносят почту).

Задание 5. Разработать ПК для администратора магазина. В ПК должна храниться информация о магазине, товарах и продавцах. Администратор магазина может добавлять, изменять и удалять эту информацию. Ему могут потребоваться следующие сведения:

- прейскурант товаров;
- список продавцов;
- справка о магазине (адрес, специализация, ФИО директора) и отчет о состоянии магазина (какие товары и в каком количестве есть в магазине).

Задание 6. Разработать ПК для диспетчера автобусного парка. В ПК должна храниться информация о водителях, маршрутах и графике движения автобусов. Диспетчер автобусного парка может добавлять, изменять и удалять эту информацию. Ему могут потребоваться следующие сведения:

- список водителей, их стаж работы и класс;
- когда начинается или заканчивается движение автобуса на всех или отдельных маршрутах;
- справка о графике движения автобусов и отчет о его нарушениях.

Задание 7. Разработать ПК для администратора футбольной команды. В ПК должна храниться информация об игроках команды, календарь и результаты игр. Администратор футбольной команды может добавлять, изменять и удалять эту информацию. Ему могут потребоваться следующие сведения:

- список футболистов и их специализация;
- календарь игр;
- количество проведенных игр;
- отчета о результативности каждого игрока команды.

Задание 8. Разработать ПК для администратора регистратуры поликлиники. В ПК должна храниться информация об больных, врачах и их расписании работы. Администратор регистратуры может добавлять, изменять и удалять эту информацию. Ему могут потребоваться следующие сведения:

- список врачей и их специализация;
- график работы врачей (номер кабинета, дни и часы приема);
- справка о болезни некоторого больного;
- количество заболеваний по каждому виду болезни.

Задание 9. Разработать ПК для диспетчера станции технического обслуживания. В ПК должна храниться информация о владельце автомобиля, характеристики и неисправности его автомобиля, о работниках станции. Администратор станции технического обслуживания может добавлять, изменять и удалять эту информацию. Ему могут потребоваться следующие сведения:

- список работников станции и их специализация;
- марка, год выпуска и пробег автомобиля данного владельца;
- количестве отремонтированных автомобилей за месяц;
- отчет о работе станции в виде перечня и дат устранения неисправностей у автомобилей и ФИО работников станции, устранявших неисправности.

Задание 10. Разработать ПК для администратора гостиницы. В ПК должна храниться информация о проживающих клиентах, номерах гостиницы и служащих гостиницы. Администратор гостиницы может добавлять, изменять и удалять эту информацию. Ему могут потребоваться следующие сведения:

- список работников гостиницы;
- список свободных номеров с указанием его вместимости;
- прейскурант цен.
- справка о жильцах гостиницы (ФИО, срок проживания, номер);
- отчет о работе гостиницы за месяц (число клиентов, сколько дней был занят и свободен номер).

Задание 11. Разработать ПК для менеджера музыкальных групп. В ПК должны храниться сведения о музыкальных группах, о репертуаре каждой группы, график гастролей. Менеджер может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- название, состав и год образования группы;
- репертуар группы и ее положение в хит-параде;
- лучшие три группы в хит-параде;
- отчет о гастролях группы.

Задание 12. Разработать ПК для коллекционера марок. В ПК должны храниться сведения о марках, имеющихся в коллекции, и сведения об их положении в коллекции. Коллекционер может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- марки каких стран содержатся в коллекции;
- в каких местах коллекции находятся марки указанной серии;
- названия коллекций и количество страниц в каждой коллекции;
- марка какой страны и какой серии находится в указанном месте коллекции.

Задание 13. Разработать ПК для администратора выставки собак. В ПК должны храниться сведения о собаках, их владельцах и судьях. Администратор выставки может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- список судей и какие породы они обслуживают;
- какая кличка и порода собаки у данного владельца;
- какие собаки получили награды на выставке;
- список владельцев собак указанной породы.

Задание 14. Разработать ПК для работников ГАИ. В ПК должны храниться сведения о водителях, их машинах и нарушениях. Работник ГАИ может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- кто владелец машины с указанным номером;
- какие нарушения совершил владелец машины;
- когда проходила машина технический осмотр;
- отчет о нарушениях правил за заданный период.

Задание 15. Разработать ПК для администратора кинотеатра. В ПК должны храниться сведения о фильмах, репертуаре, сеансах и проданных билетах. Администратор кинотеатра может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- репертуар кинотеатра на месяц;
- дни и сеансы демонстрации указанного фильма;
- сколько билетов продано на сеанс;
- справка о фильме (режиссер, год выпуска, жанр).

Задание 16. Разработать ПК для администратора аптеки. В ПК должны храниться сведения о болезнях и лекарствах. Администратор аптеки может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- какие лекарства применяются для лечения указанной болезни;
- имеется ли лекарство в аптеке и в каком количестве;
- какие лекарства и в каком количестве проданы за указанный период времени;
- на какую сумму проданы лекарства за месяц.

Задание 17. Разработать ПК для менеджера соревнований автогонщиков. В ПК должны храниться сведения о командах, гонщиках и трассах. Менеджер соревнований может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- состав гонщиков команды;
- призеры гонок по каждой трассе;
- график проведения соревнований;
- количество очков, набранных гонщиками.

Содержание

Редактор

Подписано в печать . .06. Формат 60 × 84 /16. Бумага офсетная.
Печать офсетная. Гарнитура «Times New Roman». Печ. л.
Тираж экз. Заказ .

Издательство СПбГЭТУ «ЛЭТИ»
197376, С-Петербург, ул. Проф. Попова, 5