ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
«Казанский национальный исследовательский технический университет им. А.Н.Туполева»

Кафедра автоматизированных систем обработки информации и управления

С.А. ЗАРАЙСКИЙ

Методические указания к лабораторным работам по дисциплине «Инструментальные средства информационных систем» (Б1.В.16**)**

Лабораторная работа № 3
Диаграммы классов

Казань – 2023

Лабораторная работа № 3
Диаграммы классов

Цели лабораторной работы.

1. Изучить основные сведения о языке UML в части построения диаграммы классов (class diagrams).
2. Повторить пример построения диаграммы классов из лабораторной работы.
3. Выполнить построение диаграммы классов (class diagrams) для своего варианта задания учебного проекта.

Теоретические положения. Что такое объект.

Объект (object) - это некая сущность реального мира или концептуальная сущность. Объект может быть чем-то конкретным, например, грузовик или мой компьютер, или концептуальным, как, например, химический процесс, банковская операция, торговый заказ, кредитная история или ставка прибыли. **Объектом** называется концепция, абстракция или вещь с четко определенными границами и значением для системы. Каждый объект в системе имеет три характеристики: **состояние, поведение и индивидуальность.**

Состояние, поведение и индивидуальность объекта.

Состоянием (state) объекта называется одно из условий, в которых он может находиться. Состояние системы обычно меняется во времени и определяется набором свойств, называемых *атрибутами* (attribute), значений свойств и отношений между объектами. Например, объект учебный курс (CourseOffering) в системе регистрации учебных курсов может находиться в одном из двух состояний: открыт для записи или закрыт для записи. Если количество студентов, зарегистрировавшихся на курс, меньше десяти, запись на курс продолжается. После регистрации десятого студента она прекращается.

Поведение (behavior) определяет, как объект реагирует на запросы других объектов и что может делать сам объект. Поведение реализуется с помощью набора *операций* (operation) для объекта. В системе регистрации курсов объект учебный курс может иметь операции **добавить студента** и **удалить студента**.

Индивидуальность (identity) означает, что каждый объект уникален, даже если его состояние идентично состоянию другого объекта. Например: Алгебра 101, секция 1 и Алгебра 101, секция 2 - два объекта в системе регистрации курсов. Хотя они оба являются учебными курсами, каждый из них уникален.

В языке UML объект изображается в виде прямоугольников, а его имя пишется с подчеркиванием - см. рис. 1.

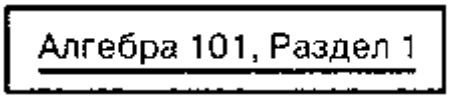


Рис. 1. Нотация языка UML для объекта.

Что такое класс

Класс (class) - это описание группы объектов с общими свойствами (атрибутами), поведением (операциями), отношениями с другими объектами и семантикой. Таким образом, класс представляет собой шаблон для создания объекта. Каждый объект является экземпляром конкретного класса и не может быть экземпляром нескольких классов. Например, класс учебный курс (CourseOffering) может определяться следующими характеристиками:

- атрибуты - место занятий, время занятий;
- операции - получить место занятий, получить время занятий, добавить студента на курс.

Математика 1, Математика 2, Информатика - это объекты, принадлежащие классу учебный курс. Каждый объект имеет значения атрибутов и доступ к операциям, определенным классом учебный курс.

«Хороший» класс представляет одну и только одну абстракцию, то есть должен отражать одну основную сущность. Например, класс, способный хранить информацию о студентах и данные о курсах, которые студент посещает в течение нескольких лет, не является «хорошим» классом, потому что не представляет одну сущность. Такой класс необходимо разделить на два связанных класса: студент и история студента.

Названия классов выбираются в соответствии с понятиями предметной области. Имя должно быть существительным в единственном числе, наиболее точно характеризующим предмет.

Иногда трудно отличить объект от класса. Почему, например, Математика 1, Математика 2 - объект, а не класс? Что отличает его от объекта Информатика? Ответы на эти вопросы субъективны. Изучив эти объекты, можно заключить, что у них одинаковая структура и поведение. Они лишь являются разными учебными предметами семестра. Кроме того, в системе регистрации курсов можно обнаружить множество схожих сущностей с одинаковой структурой и поведением: Музыка 101, секция 1; История 101, секция 1; История 101, секция 2 и т.п. Значит, допустимо создание единого **класса** учебный курс (CourseOffering).

В языке UML классы изображаются в виде разделенных прямоугольников. В верхней секции указывается имя класса, средняя секция содержит его структуру - атрибуты, а нижняя описывает его поведение - операции. Класс показан на рис. 2.

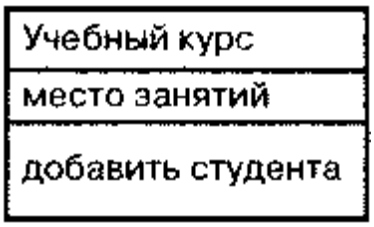


Рис. 2. Нотация языка UML для класса

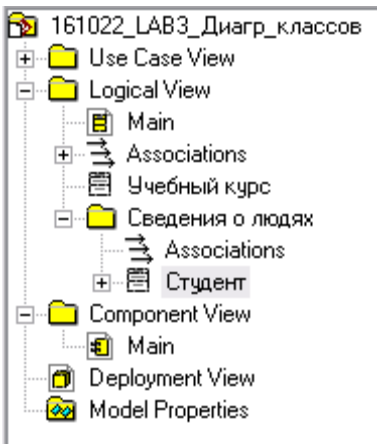


Рис.3. Класс, созданный в окне Rational Rose

Рецептов для поиска классов не существует. Rational Unified Process содержит средства, помогающие обнаружить в системе классы типа **управляющий элемент**, **граничный элемент** и **сущность**. Эти три стереотипа соответствуют концепции «модель - представление - управление» и позволяют аналитику отделить друг от друга представление, предметную область и управление в системе.

По причине того, что процесс анализа и проектирования является итеративным, список классов со временем изменится. Начальный набор классов, скорее всего, будет отличаться от итогового. Поэтому для описания начального набора классов, обнаруженных в системе, часто используется **термин «класс- кандидат».**

Стереотипы и классы

Некоторые основные **стереотипы** класса - это **сущность, граничный элемент, элемент управления, сервисный элемент** и **исключение.**

Стереотип класса указывается под его именем и заключается в двойные треугольные скобки. Если требуется, стереотип можно отобразить графическим значком или выделить цветом. В программе Rational Rose есть изображения для стереотипов Rational Unified Process - управляющего элемента, сущности и граничного элемента. Эти стереотипы показаны на рис. 4.

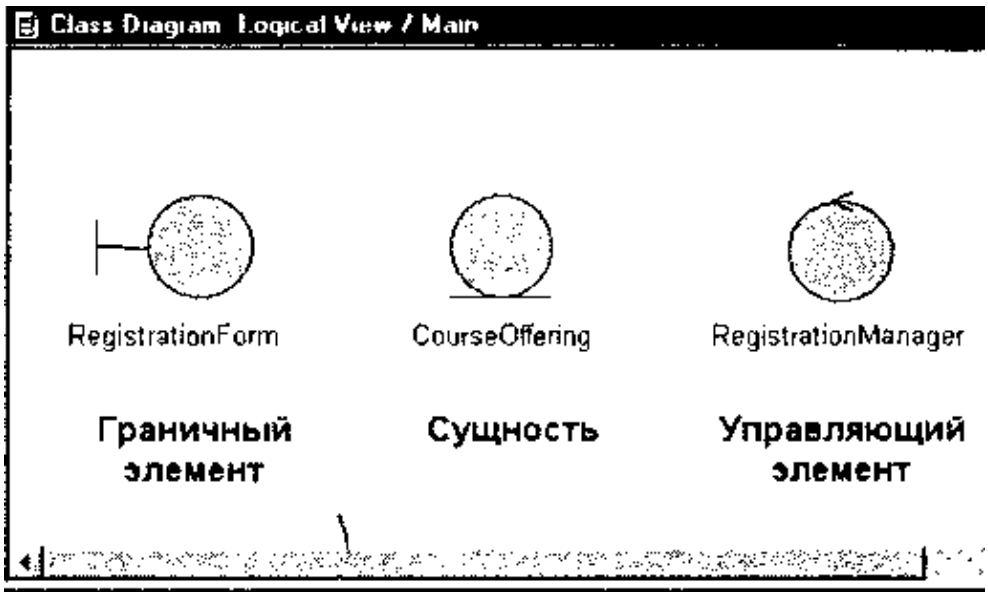


Рис. 4. Классы со стереотипами

Классы-сущности

Класс-сущность (entity class) используется для моделирования данных и поведения с длинным жизненным циклом. Этот тип классов может представлять сущности реального мира или внутренние элементы системы. Такие классы обычно не зависят от окружения, то есть они нечувствительны к взаимодействию окружающей среды с системой. Следовательно, **они не зависят от приложения и могут использоваться в различных приложениях**.

Первый шаг - изучить обязанности, описанные в **потоке событий** для выявления прецедентов (что система должна делать). **Классы-сущности** - это обычно те классы, которые требуются системе для выполнения определенных обязанностей. Использование существительных для описания обязанностей может стать хорошим началом. Исходный список нужно профильтровать, так как он будет содержать слова, не относящиеся к предметной области, языковые выражения, избыточные слова и существительные, описывающие структуру класса.

Классы-сущности обычно определяются на стадии проработки. Их часто называют классами предметной области, потому что они представляют собой абстракции предметов реального мира.

Граничные классы

Граничные классы (boundary class) обеспечивают взаимодействие между окружающей средой и внутренними элементами системы. Такие классы предоставляют **интерфейс** для пользователя или другой системы (то есть для актера). Они составляют внешне зависимую часть системы и используются для моделирования интерфейсов системы.

Для обнаружения **граничных классов** изучают пары актер/сценарий. Такие классы, определенные на фазе проработки, обычно, являются классами верхнего уровня. Например, вы можете смоделировать окно, но не моделировать его диалоговые элементы и кнопки. В этом случае вы опишете требования пользовательского **интерфейса**, но не реализуете его.

Требования к пользовательскому **интерфейсу** порой недостаточно ясны. Обычно используются термины «дружественный» и «гибкий». Но дружественный интерфейс разными людьми трактуется по-разному. Здесь могут пригодиться **прототипы**. Пользователь должен посмотреть и почувствовать систему, чтобы реально оценить, что значит «дружественный» и то, что это значит, затем представляется как структура и поведение граничного класса. На этапе проектирования такие классы совершенствуются и выносятся на обсуждение вопросов реализации пользовательского **интерфейса**. Граничные классы также используются для обеспечения связи с другими системами. На этапе проектирования эти классы совершенствуются и выносятся на обсуждение вопросов реализации протоколов взаимодействия.

Управляющие классы

Управляющие классы (control class) служат для моделирования последовательного поведения одного или нескольких прецедентов и координации событий, реализующих заложенное в них поведение. **Управляющие классы** можно представить как **классы, «исполняющие» прецедент** и определяющие его динамику. Они обычно зависят от приложения.

На ранней стадии проработки управляющие классы добавляются для каждой пары актер/прецедент. Такие классы определяют поток событий в прецедентах.

Вопрос использования управляющих классов очень субъективный. Многие авторы утверждают, что их применение приводит к отделению данных от поведения. Это может случиться, если **управляющие классы** выбраны неаккуратно. Если **управляющий класс** реализует что-то большее, чем последовательное действие, то он делает слишком много. Например: в системе регистрации учебных курсов студент выбирает курс, и если курс доступен, студента на него записывают.

Кто должен знать, как прикрепить студента, - управляющий класс или класс, представляющий курс занятий? Правильный ответ - класс, представляющий курс занятий. Управляющий класс знает лишь, когда студент должен быть прикреплен. «Плохой» управляющий класс знает не только когда, но и как прикрепить студента.

Управляющий класс для каждой пары актер/прецедент создается на начальном этапе. При дальнейшем анализе и проектировании управляющие классы могут исключаться, разделяться или объединяться.

Документирование классов

После того как класс создан, информацию о нем необходимо отразить в документации. Документация предназначена для описания **назначения класса,** а не его структуры. Например, **класс студент** может быть описан следующим образом:

Информация, необходимая для регистрации студента и оплаты обучения. Студент - это человек, обучающийся в университете.

А вот пример неправильного описания:

Имя, адрес и телефон студента.

Оно раскрывает только структуру класса, которую можно увидеть, посмотрев на список атрибутов, и не поясняет, для чего нужен данный класс.

Трудности при выборе имени и описании класса могут свидетельствовать о том, что это недостаточно хорошая абстракция. В следующем списке перечислены возможнее варианты:

- можно определить имя и дать краткое, четкое описание - хороший класс- кандидат;

- можно определить имя и выбрать описание, похожее на описание другого класса, - объединить классы;

- можно определить имя, но потребуется целая книга, чтобы описать назначение класса, - разделить класс;

- нельзя определить имя и дать описание - требуется дополнительный анализ для выделения правильных абстракций.

Чтобы описать классы в программе Rational Rose:

1. Выберите класс в списке браузера.
2. Установите курсор в окне описания и введите описание класса.

Описание класса представлено на рис. 5.

Пакеты

Если в системе существует немного классов, управлять ими достаточно легко. Многие системы состоят из большого количества классов, поэтому необходим механизм, позволяющий разбить их на группы и облегчающий управление и повторное использование. Здесь оказывается полезной концепция пакетов. ***Пакет*** (package) в логическом представлении модели - это набор классов и других связанных пакетов. Путем объединения классов в пакеты мы можем получить представление модели на более высоком уровне. Изучая содержимое пакета, мы, наоборот, получаем более детальное представление.

Каждый **пакет** содержит **интерфейс**, реализуемый набором его общедоступных **классов** (public classes), то есть тех, с которыми могут общаться **классы** из других пакетов.

Рис. 7. Нотация языка UML для пакетов

Остальные классы пакета - это классы реализации (implementation classes), которые не взаимодействуют с классами в других пакетах.

В сложной системе для облегчения восприятия пакеты могут быть созданы на этапе проработки. В более простой системе классы, выделенные на этапе анализа, могут быть сгруппированы в один **пакет**, представляющий саму систему.

В ходе дальнейшего анализа и проектирования пакеты нужны для группировки **классов**, используемых в системной архитектуре.

В языке UML пакеты изображаются в виде папок (см. рис.7).

Объекты и классы в системе регистрации курсов

Рассмотрим сценарий добавление учебного курса (Add a Course Offering to Teach), который является внутренним потоком для прецедента выбор предметов для преподавания (Select Courses to Teach). Данный **сценарий** позволяет преподавателю выбрать учебный курс для конкретного семестра.

Хотя мы рассматриваем этот процесс пошагово, на практике большинство шагов могут быть выполнены одновременно.

Выбор граничных классов

Рассматриваемый прецедент взаимодействует только с актером преподаватель. Действие, выполняемое указанным сценарием, - это только одна из возможностей, обеспечиваемых прецедентом (он также определяет, что преподаватель может изменять, удалять, просматривать и печатать курсы). Это означает, что в системе должен быть механизм, позволяющий преподавателю выбирать желаемое действие. Для обеспечения потребностей преподавателя создается специальный класс - параметры курса преподавателя (ProfessorCourseOptions). Дополнительно мы можем указать класс, который служит для добавления новых курсов, доступных преподавателю, - добавление учебного курса (AddACourseOffering).

Выбор классов-сущностей

Данный сценарий состоит из **предметов, учебных курсов** и **назначения преподавателей**. Мы можем выделить три **класса-сущности**: предмет (Course), учебный курс (CourseOffering) и преподаватель (Professor).

Выбор управляющих классов

Добавим один управляющий класс с целью обработки потока событий для прецедента - менеджер курсов преподавателя (ProfessorCourseManager).

Выбранные классы (с установленными стереотипами сущность, управляющий элемент или граничный элемент) могут быть добавлены к модели (см. рис. 10).

Так как актер преподаватель уже существует, при создании класса преподаватель программа Rational Rose предупредит, что одно и то же имя используется в разных разделах.

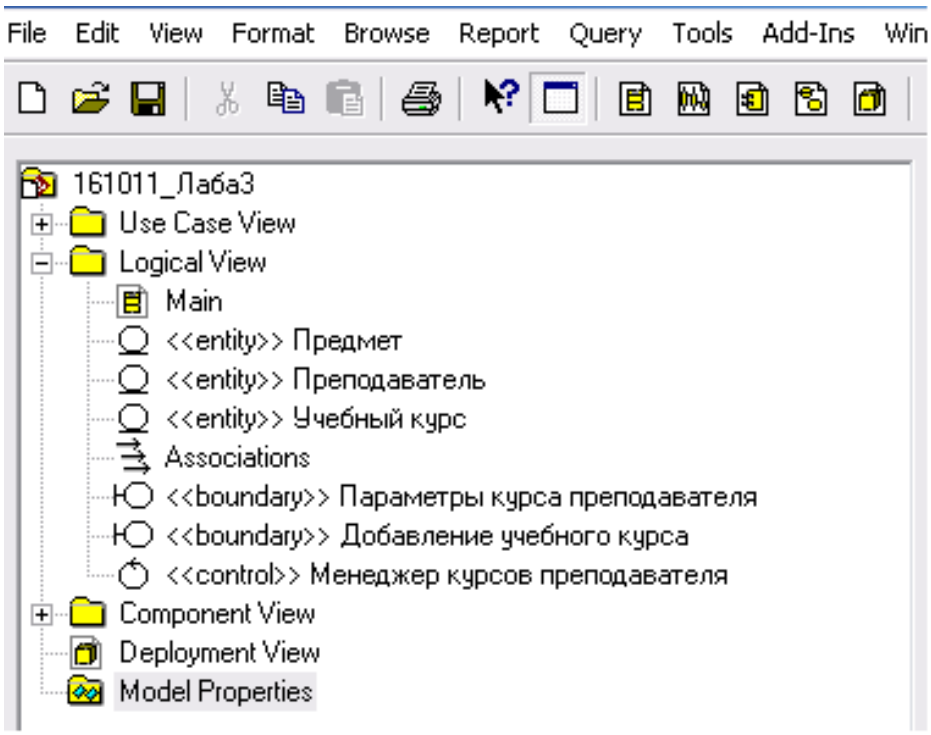


Рис. 10. Классы для сценария
«Добавление учебного курса»

Создание пакетов

Следующий шаг - объединить классы в пакеты. На данном этапе выделим шесть классов: предмет, учебный курс, преподаватель, параметры курса преподавателя, добавление учебного курса и менеджер курсов преподавателя. Их можно разделить на три логические группы: объекты, специфичные для университета; объекты, содержащие информацию о людях; интерфейсы для актеров. Таким образом, мы можем создать следующие пакеты: Интерфейсы (Interfaces), Объекты университета (UniversityArtifacts) и Сведения о людях (Peoplelnfo). Затем классы помещаются в соответствующие пакеты (см. рис. 11).

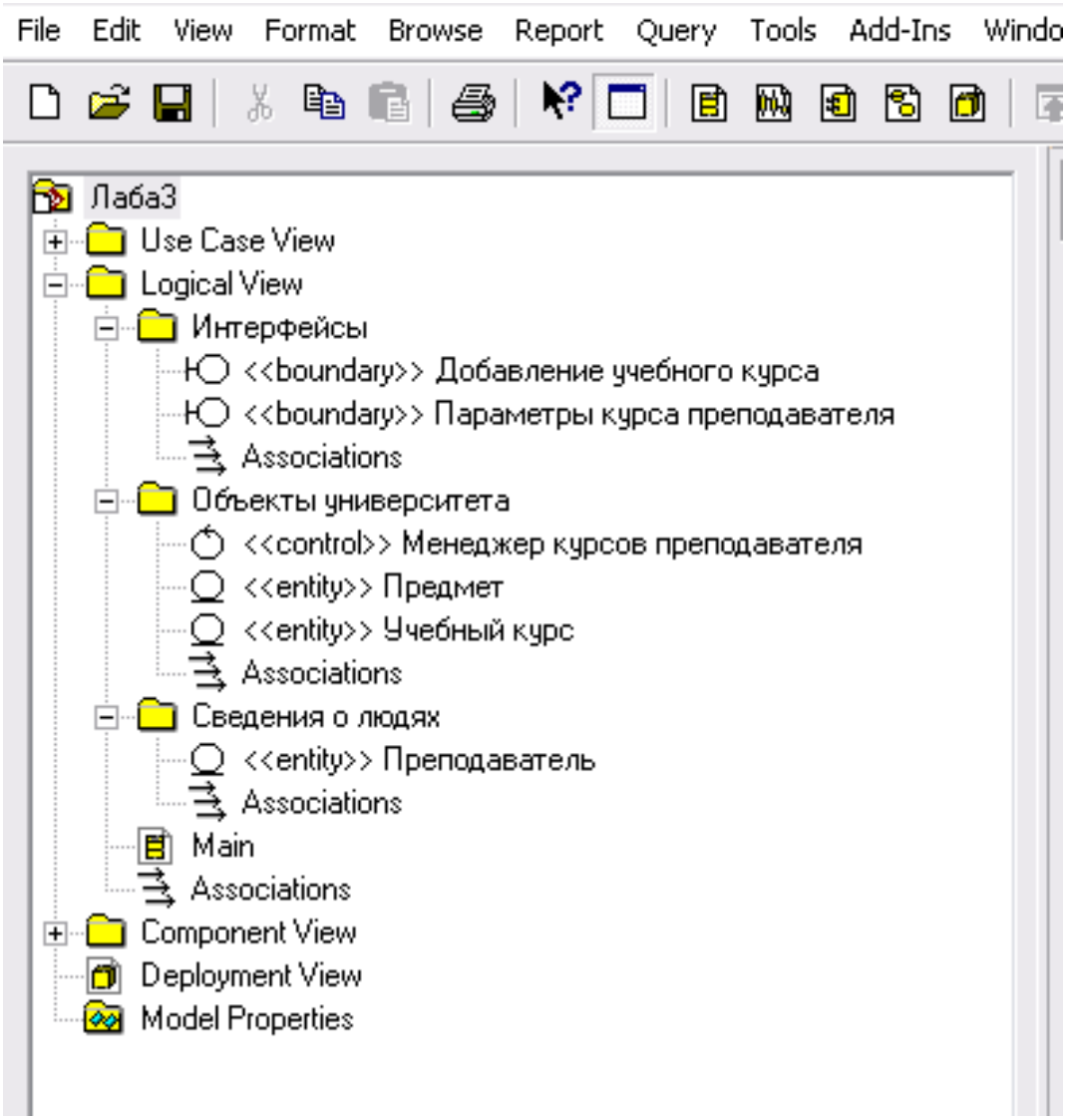


Рис. 11. Пакеты в браузере

Диаграммы классов

По мере того как новые классы добавляются в систему, их текстовое представление становится неудобным. ***Диаграммы классов*** (class diagrams) помогают графически представить некоторые или все классы в модели.

Главная диаграмма классов в логическом представлении модели обычно отображает пакеты системы. **Каждый пакет также имеет свою главную диаграмму классов, которая обычно содержит общедоступные классы пакета.** Другие диаграммы создаются по необходимости. Типичные примеры использования диаграмм классов:

- просмотр всех классов реализации в пакете;
- просмотр структуры и поведения одного или нескольких классов;
- просмотр иерархии наследования классов.

Программа Rational Rose автоматически создает главную диаграмму классов (**Main**) в логическом представлении модели.

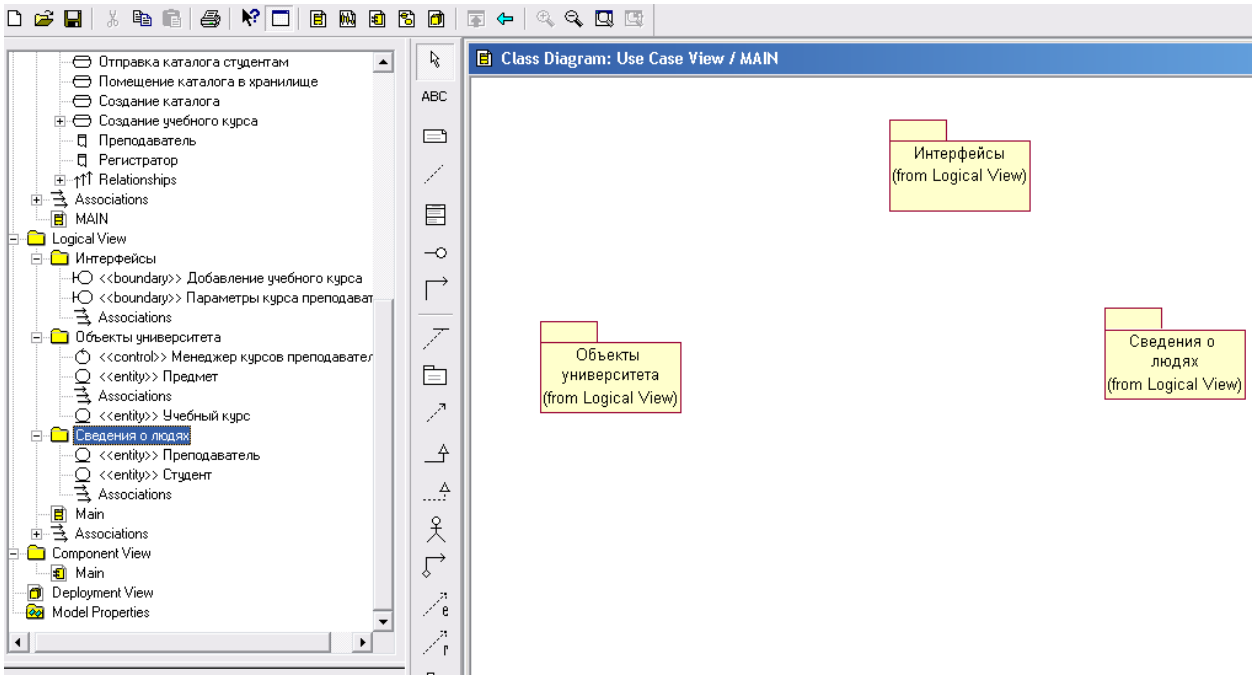


Рис.12. Главная диаграмма классов

Главная диаграмма классов для **пакета Объекты университета** изображена на рис. 13. Заметьте, что класс учебный курс (CourseOffering) на ней отсутствует. Это класс реализации в пакете, и мы решили не показывать его на главной диаграмме. По мере добавления пакетов и классов в модель могут быть созданы дополнительные диаграммы.

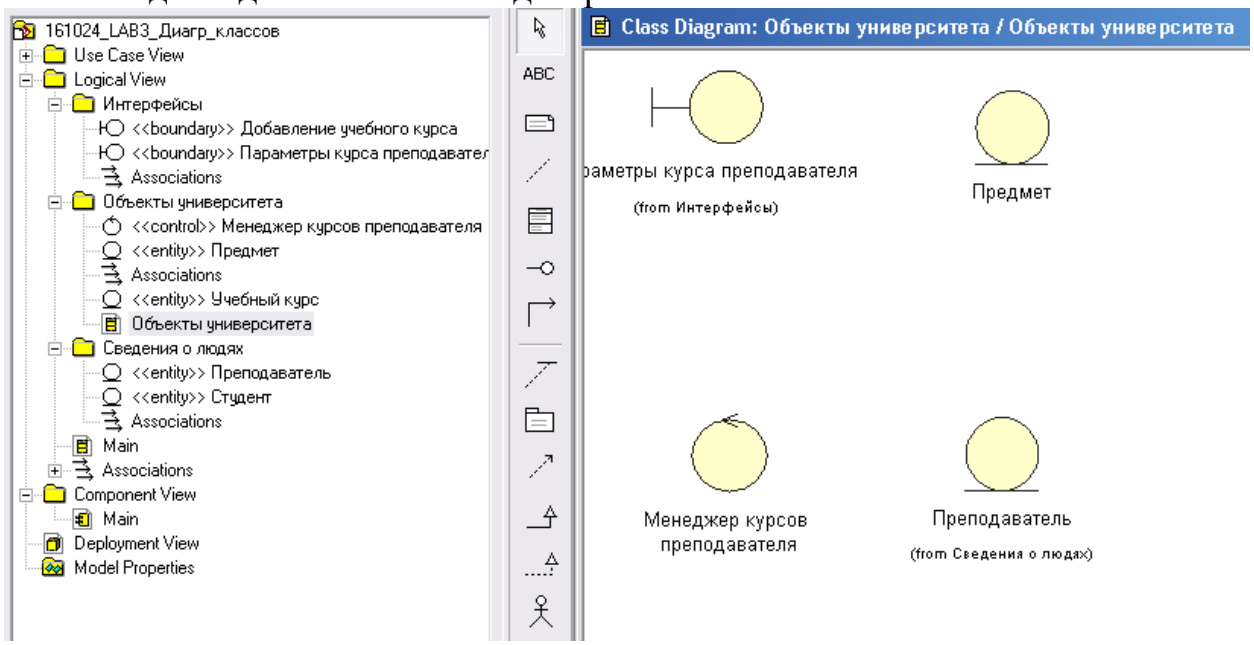


Рис. 13. Главная диаграмма классов пакета Объекты университета

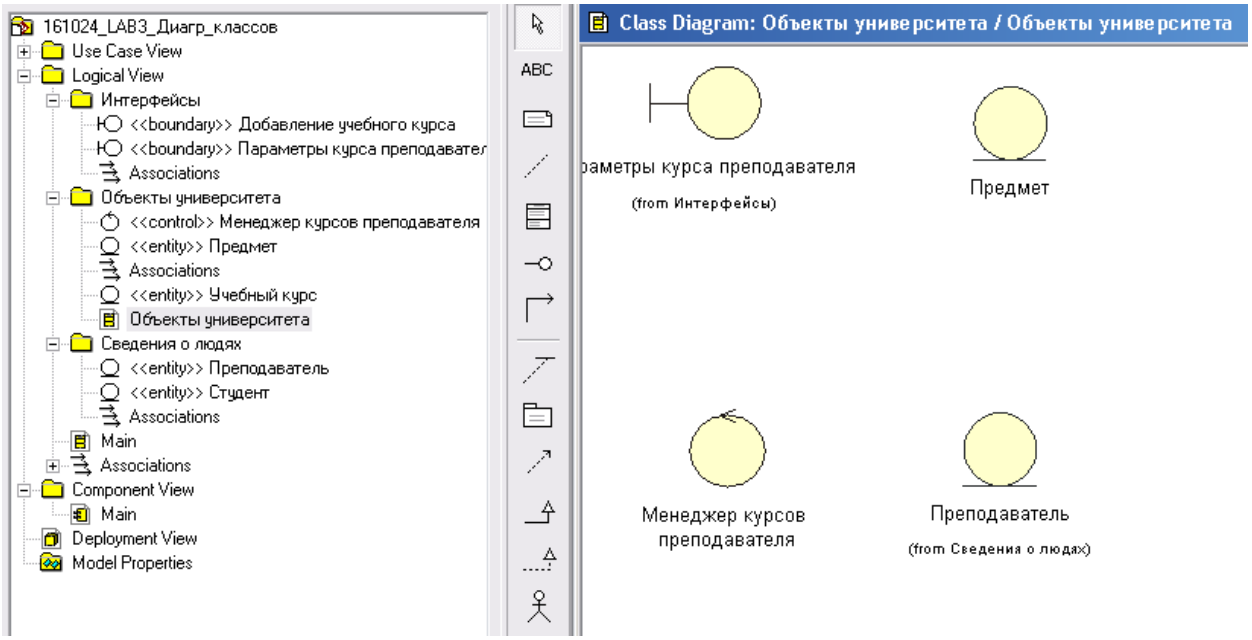


Рис. 14. Диаграмма классов

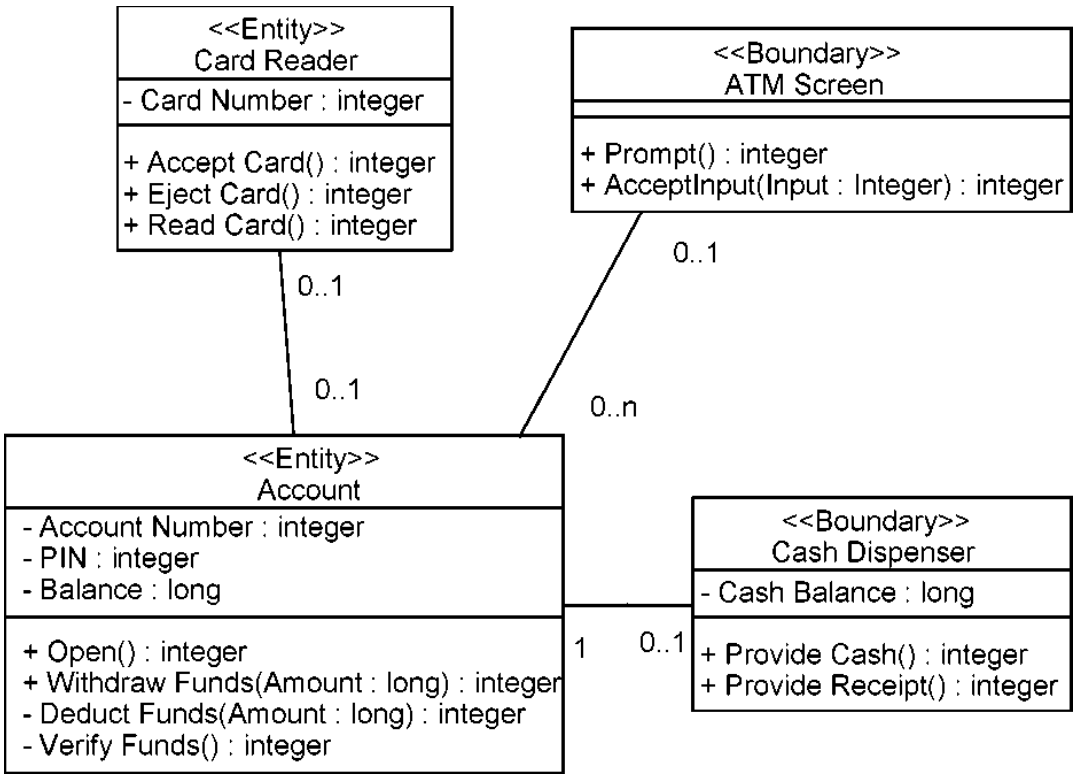


Рис. 15. Диаграмма классов для варианта использования
«Снять деньги»

ЗАКЛЮЧЕНИЕ

Объекты - это компьютерное представление сущностей (предметов реального мира или понятий, придуманных человеком). Объект - это концепция, абстракция или вещь с четко определенными границами и значением для системы. Каждый объект в системе имеет три характеристики:

состояние, поведение и индивидуальность. Состояние объекта - одно из условий, в которых он может находиться. Поведение характеризует объект и показывает, как он реагирует на запросы других объектов. Индивидуальность означает, что каждый объект уникален, даже если его состояние идентично состоянию другого объекта.

Класс - это описание группы объектов с общими свойствами (атрибутами), поведением (операциями), отношениями с другими объектами (ассоциативными или агрегационными) и семантикой. В языке UML классы изображаются в виде разделенных прямоугольников. В секциях прямоугольника указываются имя, структура и поведение класса. После того как класс создан, его необходимо описать в документации. Документация предназначена для описания назначения класса, а не его структуры.

Стереотипы обеспечивают возможность создания новых типов элементов моделирования и должны основываться на элементах, входящих в метамодель языка UML. На этапе анализа выделяют три основных стереотипа для классов: класс- сущность, граничный класс и управляющий класс. Эти стереотипы используются для определения классов в разрабатываемой системе.

Пакет в логическом представлении модели - это набор классов и других связанных пакетов. Путем объединения классов в пакеты мы можем получить представление модели на более высоком уровне. Изучая содержимое пакета, мы получаем более детальное представление.

Диаграммы классов помогают графически изобразить некоторые или все классы системы. Диаграммы классов можно создать и в представлении модели прецедента. Они обычно прикрепляются к прецеденту и содержат представления классов, участвующих в их выполнении.

2. Порядок выполнения лабораторной работы

1. Изучить методические материалы по лабораторной работе №3.
2. Повторить все диаграммы лабораторной работы.
3. Составить диаграмму классов для своего варианта задания.

4.. Оформить лабораторную работу. Отчёт составляется в печатной (на бумаге) форме, каждому студенту. Отчет включает описание всей проделанной работы (интерпретацию постановки задачи, глоссарий проекта, дополнительные спецификации, описание элементов диаграммы вариантов использования) и скриншот полученной диаграммы. Отправить лабораторную работу в виде: ФамилияСтудента_лр3 ИСИС_44ХХ (44ХХ – код группы).

5. Защитить лабораторную работу. Защита работы сопровождается необходимыми пояснениями и подробными и точными ответами на контрольные вопросы из нижеприведённого списка.

Контрольные вопросы.

1. Что такое объект в UML?
2. Что такое состояние, поведение и индивидуальность объекта
3. Что такое класс в UML?
4. Что такое «хороший класс»?
5. Стереотипы и классов?
6. Что значит документировать класс? Примеры.
7. Пакеты. Назначение.

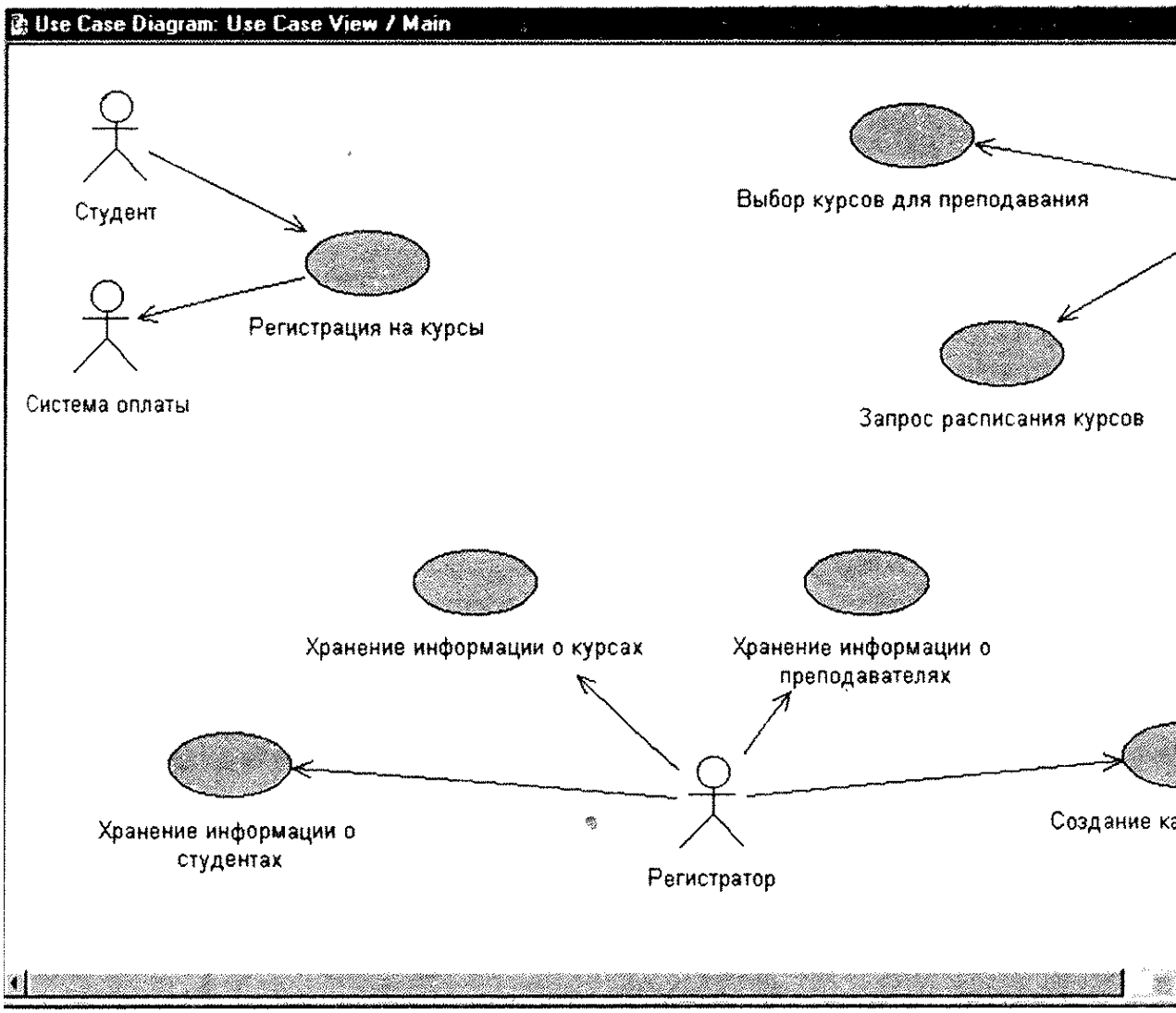
Литература:

1. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование: Пер. с англ. - М.: ДМК Пресс, 2001. - 176 с.: ил. (Серия «Объектно-ориентированные технологии в программировании»).

2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. Учебник. М.: Финансы и статистика, 2003, 352 с.

3. Вендров А.М. Практикум по проектированию программного обеспечения экономических информационных систем. М.: Финансы и статистика, 2002,192 с.

Приложение А. Главная диаграмма претендентов



Приложение В. Дополнительная диаграмма претендентов