МИНОБРНАУКИ РОССИИ
Глазовский инженерно-экономический институт (филиал)
федерального государственного бюджетного образовательного
учреждения высшего образования
«Ижевский государственный технический университет имени М.Т.Калашникова»
(ГИЭИ (филиал) ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

УТВЕРЖДАЮ
Директор
_____М.А. Бабушкин
15 июня 2024 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине

МДК.02.02 «Инструментальные средства разработки программного обеспечения»

09.02.07 Информационные системы и программирование

Глазов 2024

Фонд оценочных средств разработан на основе Федерального государственного образовательного стандарта по специальности среднего профессионального образования 09.02.07 "Информационные системы и программирование", утвержденного приказом Министерства образования и науки Российской Федерации 09 декабря 2016 г. № 1547 с изменениями и дополнениями (приказ Министерства просвещения Российской Федерации от 17.12.2020 № 747 «О внесении изменений в федеральные государственные образовательные стандарты среднего профессионального образования» (зарегистрирован 22.01.2021 № 62178), приказ Министерства просвещения Российской Федерации от 01.09.2022 № 796 «О внесении изменений в федеральные государственные образовательные стандарты среднего профессионального образования» (зарегистрирован 11.10.2022 № 70461))

Организация разработчик: ГИЭИ (филиал) ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»

Разработчик: Горбушин Денис Шарибзянович,
преподаватель СПО

Утверждено: Протокол Ученого совета филиала № 9, от 14 июня 2024 г.

Руководитель образовательной программы

_____ Т.А. Савельева
15 июня 2024 г.

Согласовано: Начальник отдела по учебно-методической работе

_____ И.Ф. Яковлева

15 июня 2024 г.

Оглавление

Паспорт фонда оценочных средств
по дисциплине **МДК.02.02 «Инструментальные средства разработки программного обеспечения»**
(наименование дисциплины)

№ п/п	Контролируемые разделы (темы) дисциплины*	Код контролируемой компетенции (или ее части)	Наименование оценочного средства
1	**Тема 1.1. Основные понятия**	ПК 2.1, ПК 2.2	Контрольная работа, тестовые задания, вопросы к зачету
2	**Тема 1.2. Общая характеристика инструментальных средств разработки программных продуктов**	ПК 2.1, ПК 2.2 ПК 2.3	Контрольная работа, тестовые задания, вопросы к зачету
3	**Тема 1.3. Современные технологии и инструменты интеграции**	ПК 2.1, ПК 2.2 ПК 2.3	Контрольная работа, тестовые задания, вопросы к зачету
4	**Тема 1.4. Инструментарий тестирования и анализа качества программных средств**	ПК 2.1, ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5	Контрольная работа, тестовые задания, вопросы к зачету

Код	**Наименование результата обучения**
ПК 2.1	Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент
ПК 2.2	Выполнять интеграцию модулей в программное обеспечение
ПК 2.3	Выполнять отладку программного модуля с использованием специализированных программных средств
ПК 2.4	Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения
ПК 2.5	Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования

Уровни сформированности профессиональных компетенций

ПК 2.1 Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент	
Категории	Описание показателей
Уметь	- Анализировать проектную и техническую документацию.;
	- Использовать специализированные графические средства построения и анализа архитектуры программных продуктов;

	- Определять источники и приемники данных;
	- Графические средства проектирования архитектуры программных продуктов;
Знать	- модели процесса разработки программного обеспечения
	- основные принципы процесса разработки программного обеспечения
	- методы организации работы в команде
ПК 2.2. Выполнять интеграцию модулей в программное обеспечение	
Уметь	Организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов
	Выполнять тестирование интеграции
	Организовывать постобработку данных
Знать	Основные подходы к интегрированию программных модулей.
	Современные технологии и инструменты интеграции
	Основные протоколы доступа к данным
	Методы и способы идентификации сбоев и ошибок при интеграции приложений.
ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств	
Уметь	Использовать инструментальные средства отладки программных продуктов
	Выполнять отладку, используя методы и инструменты условной компиляции
	Выявлять ошибки в системных компонентах на основе спецификаций
Знать	Основные методы отладки
	Приемы работы с инструментальными средствами тестирования и отладки
ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения	
Уметь	Оценивать размер минимального набора тестов
	Разрабатывать тестовые пакеты и тестовые сценарии
	Выполнять ручное и автоматизированное тестирование программного модуля
Знать	- Стандарты качества программной документации;
	- Встроенные и основные специализированные инструменты анализа качества программных продуктов;
ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования	
Уметь	Выявлять ошибки в системных компонентах на основе спецификаций
Знать	Основы организации инспектирования и верификации
	Встроенные и основные специализированные инструменты анализа качества программных продуктов
	Методы организации работы в команде разработчиков.

Зачетно-экзаменационные материалы

Контрольные вопросы для оценки усвоения знаний

1. Необходимые инструментальные средства разработки программ
2. Часто используемые инструментальные средства разработки программ
3. Специализированные инструментальные средства разработки программ
4. Интегрированные среды разработки
5. Средства разработки программного обеспечения
6. Определение «разработка программ». Три этапа разработки программ
7. Средства проектирования приложений
8. Средства реализации программного кода
9. Средства тестирования программ
10. Классы инструментальных средств разработки программ
11. Четыре категории инструментальных программ, применяемые при проектировании экспертных систем
12. Оболочки экспертных систем
13. Языки программирования высокого уровня Среда программирования, поддерживающая несколько парадигм
14. Средства автоматизации разработки экспертных систем
15. Общее программное обеспечение
16. Специальное программное обеспечение
17. Инструментальная система технологии программирования
18. Четыре класса компьютерной поддержки инструментальных систем технологий программирования: комплексность, ориентированность на коллективную разработку, технологическая определенность, интегрированность
19. Компоненты инструментальных систем технологий программирования
20. База данных разработки. Инструментарий. Интерфейсы.
21. Общая архитектура инструментальных систем технологий программирования
22. Инструментальная система поддержки проекта
23. Языково-зависимая инструментальная система
24. Пользовательский интерфейс. Схема организации взаимодействия компьютера и пользователя.
25. Процедурно-ориентированный подход к разработке интерфейсов
26. Объектно-ориентированный подход к разработке интерфейсов
27. Типы интерфейсов. Интерфейс-меню. Интерфейсы со свободной навигацией. Критерии оценки интерфейса пользователем
28. Факторы появления Case-технологий. Что такое Case-технология?
29. Компоненты интегрированного Case-средства.
30. Классификация по категориям Case-средств. Классификация по типам Case-средств. Вспомогательные типы Case-средств.
31. Структурный подход к разработке ИС
32. Объектно-ориентированный подход к разработке ИС

33. Факторы, усложняющие определение возможного эффекта от использования Case-средств
34. Качества организации для успешного внедрения Case-средств
35. Проблемы использования Case-средств
36. Структурный системный анализ
37. Диаграммы «сущность-связь»
38. Диаграммы классов
39. Язык графического описания UML
40. Диаграмма компонентов
41. Диаграмма композитной структуры
42. Диаграмма развёртывания
43. Диаграмма объектов
44. Диаграмма пакетов
45. Диаграмма деятельности
46. Преимущества UML
47. IDEF
48. Диаграммы переходов состояний
49. Методология функционального моделирования ИС
50. Состав функциональной модели
51. Иерархия диаграмм
52. Типы связей между функциями
53. Характеристика современных Case-средств
54. Методология ARIS. Программный продукт ARIS Express.
55. Основные элементы, используемые в нотации ARIS. Архитектура ARIS
56. Имитационное моделирование. Применение имитационного моделирования. Виды имитационного моделирования
57. Дискретно-событийное моделирование
58. Системная динамика
59. Области применения имитационного моделирования
60. Основные этапы компьютерного моделирования
61. Построение концептуальной модели системы
62. Постановка задачи машинного моделирования. Анализ задачи моделирования.
63. Определение требований к исходной информации
64. Выдвижение гипотез и принятие предположений
65. Определение параметров и переменных
66. Установление основного содержания модели
67. Обоснование критериев оценки эффективности системы
68. Определение процедур аппроксимации
69. Описание концептуальной модели
70. Проверка достоверности модели
71. Составление технической документации

В качестве зачета по дисциплине МДК. 02.02. «Инструментальные средства разработки программного обеспечения» студенту предлагается устно ответить на два вопроса из выше предложенного перечня.

Критерии оценки:

Оценка **«отлично»** выставляется студенту, если грамотно изложены и обоснован выбор документов, учащийся решает поставленную задачу. Выполняет качественно весь объем работ. Ответил на контрольный вопрос для проведения зачета подробно и глубоко.

Оценка «**хорошо**» выставляется студенту, если имеются незначительные погрешности в вышеперечисленных критериях оценки; студент ответил на контрольный вопрос для проведения зачета подробно и глубоко.

Оценка «**удовлетворительно**» выставляется студенту, если имеются значительные погрешности в вышеперечисленных критериях оценки; студент ответил на контрольный вопрос для проведения зачета не полно.

Оценка «**неудовлетворительно**» выставляется студенту, если имеются грубые погрешности в вышеперечисленных критериях оценки и не ответил на контрольный вопрос для проведения зачета.

Типовые задания для оценки освоенных умений, компетенций:

Обучающийся демонстрирует знания, практические умения и сформированность профессиональных компетенций, развитие общих компетенций при выполнении заданий. Образец задачи, предлагаемой для решения в рамках экзамена по предмету «Инструментальные средства разработки программного обеспечения»:

1. Создайте на основании темы в соответствии с вариантом смешанную диаграмму: Разработайте систему для описания порядка подготовки к экзамену, предполагающий получение отличной оценки (IDEF0 -> DFD).

2. Создайте на основании темы в соответствии с вариантом смешанную диаграмму: Разработайте систему, которая должна описывать порядка выполнения практической работы по дисциплине «Проектирование ИС» (IDEF0 -> IDEF3).

3.Создайте на основании темы в соответствии с вариантом смешанную диаграмму: Разработайте систему описания порядка получения водительских прав (DFD -> IDEF3).

4. Разработайте модель IDEF0 в соответствии с вариантом: Разработайте систему описания порядка организации городского спортивного соревнования.

5.Разработайте модель IDEF3 в соответствии с вариантом: Разработайте систему, которая должна описывать порядок организации общеинститутского студенческого мероприятия.

6. Разработайте модель DFD в соответствии с вариантом: Разработайте систему составления учебного графика дисциплин, изучаемых на факультете (IDEF0 -> IDEF3; кооперации)

7. На основании темы в соответствии с вариантом разработать диаграмму: Разработайте систему, которая должна описывать порядок поставок

товара в систему розничных киосков. Дайте обозначения для элементов системы IDEF0, разделяя их на основные и вспомогательные

8. Спроектируйте основные и вспомогательные процессы модели IDEF3 по теме: Разработайте систему описания порядка обработки заказов в службе быта.

9. Создайте элементы системы DFD и выделите основные и вспомогательные процессы: Разработайте систему описания работы одного из участков автосалона.

10. Выполните создание диаграммы классов UML по индивидуальному заданию: Разработайте систему описания работы приемного покоя в больнице.

11. Выполните создание диаграммы классов UML по индивидуальному заданию: Разработайте систему для описания порядка приема заявки на поставку продукции на хлебокомбинате.

12. Создает диаграмму вариантов использования для конкретной системы: Разработайте систему, описывающую процесс поставки сезонных товаров в оптовой фирме.

13. Создает диаграмму вариантов использования для конкретной системы: Разработайте систему описывающую процесс работы торгового отдела.

14. Проектирует деятельность системы, используя диаграммы деятельности по теме: Разработайте систему, которая должна описывать порядок поставок товара в систему розничных киосков

15. Проектирует деятельность системы, используя диаграммы деятельности по теме: Разработайте систему учета в видеопрокате.

16. Выполняет создание диаграммы развертывания UML по индивидуальному заданию: Разработайте систему учета проката на лыжной базе

17. Выполняет создание диаграммы развертывания UML по индивидуальному заданию: Разработайте систему описания процесса обслуживания клиента в банке

18. Выполняет моделирование системы с помощью диаграммы состояний: Разработайте систему описания процесса покупки товаров в Интернет-магазине

19. Выполняет моделирование системы с помощью диаграммы состояний: Разработайте систему, описывающей процесс ремонта компьютеров

20. Создайте в соответствии с вариантом смешанную диаграмму: Разработайте систему описания процесса выполнения курсовой работы (IDEF0 -> DFD).

Критерии оценки:

Оценка **«отлично»** выставляется студенту, если спроектирована диаграмма прецедентов, реализована модель базы данных.

Оценка **«хорошо»** выставляется студенту, если диаграмма прецедентов спроектирована с недочетами в части владения инструментарием программного

средства или недочетами в части проектирования диаграммы, реализованная модель базы данных содержит недочеты.

Оценка **«удовлетворительно»** выставляется студенту, если диаграмма прецедентов не спроектирована в программном средстве, но верно указаны элементы диаграммы, реализованная модель базы данных содержит недочеты.

Оценка **«неудовлетворительно»** выставляется студенту, если диаграмма прецедентов не спроектирована в программном средстве, но верно указаны элементы диаграммы, реализованная модель базы данных содержит ошибки или не реализована.

Контрольно-измерительные материалы:

1. Контрольные работы

Текущий контроль результатов усвоения УД в соответствии с рабочей программой происходит при использовании такой формы контроля, как текущая контрольная работа.

Контрольная работа № 1.
Вариант 1.

1. Классификация по сфере (области) использования программных продуктов
2. Сколько периодов обычно выделяют в истории развития Инструментальные средства разработки программного обеспечения и чем они отличаются?
3. Определение метода, нотации и средства
4. Какие основные компоненты входят в состав CASE – средств?
5. Основные функции репозитория
6. На что подразделяют все CASE – средства?

Контрольная работа № 1.
Вариант 2.

1. Перечислите три класса программных продуктов
2. Перечислите названия периодов
3. Что может быть отнесено к CASE – средствам?
4. Какими функциональными возможностями обладают компоненты CASE – средств?
5. Какими свойствами описывается каждый информационный объект, хранящийся в репозитории?
6. Классификация CASE – средств по типам.

Контрольная работа № 1.
Вариант 3.

1. С чем связано развитие инструментария технологии программирования?
2. Подробно опишите 5 и 6 этапы
3. Перечислите базовые принципы CASE – средств
4. С помощью чего осуществляется разработка диаграмм?
5. Основные типы отчетов
6. Классификация CASE – средств по категориям

Контрольная работа № 1.
Вариант 4.

1. Какие группы программных продуктов сформировались в рамках этого направления? Перечислите из и запишите определения.
2. На какой модели основано большинство CASE – средств?

3. Перечислите базовые положения CASE – средств
4. Какие типы контроля реализуются в CASE – средствах?
5. Какие свойства являются основой поддержки процесса разработки современных CASE – средств?
6. Классификация CASE – средств по уровням

Контрольная работа № 2.
Вариант 1.

1. Какие факторы имеют сильное воздействие на эффективность используемых методов синхронизации изменений?
2. Для каких целей используются системы контроля версий?
3. Виды моделей (диаграмм)
4. В чем заключается построение SADT – модели?
5. Что позволяет разрабатывать UML?
6. На какие виды подразделяются модели языка UML и что описывают?

Контрольная работа № 2.
Вариант 2.

1. Какую модель разработки предпочитают различные коллективы программистов?
2. Какая необходимость возникает при одновременной разработке нескольких программ?
3. Для чего используются данные модели?
4. Этапы построения диаграмм?
5. Для чего предназначен язык UML?
6. На какие уровни подразделяются модели языка UML и что они собой представляют?

Контрольная работа № 2.
Вариант 3.

1. Почему возникает потребность в ПО контроля и объединения версий?
2. Какой вариант является наиболее приемлемым для разработчика ПО?
3. Где используется метод SADT и на чем он реализован?
4. Какими правилами определяется синтаксис диаграмм?
5. На каких принципах основан язык UML?
6. Что называется рациональным унифицированным процессом RUP и кто его автор?

Контрольная работа № 2.
Вариант 4.

1. Для каких целей используют мощные архиваторы?
2. Перечислите основные варианты структуры и организации коллективной разработки программ?
3. Что представляет собой метод SADT?

4. Перечислите и опишите типы связей между функциями
5. Виды диаграмм в языке UML
6. Что описывается в диаграммах языка UML?

Критерии оценки:

Контрольная работа оценивается по пятибалльной шкале, стоимость одного вопроса – 1 балл. За правильный ответ можно получить 1 балл, за неверный ответ или его отсутствие баллы не начисляются.

Оценка «5» соответствует 86% – 100%

Оценка «4» соответствует 73% – 85%

Оценка «3» соответствует 53% – 72%

Оценка «2» соответствует 0% – 52%

2 Тестовый контроль

Тема бщая характеристика инструментальных средств разработки программных продуктов

1. Программное средство, предназначенное для поддержки разработки других программ, называется -…

a) аппаратным инструментом

b) программным инструментом

c) программной средой

d) инструментарий технологии программирования

2. Анализаторы обеспечивают…

a. конструирование тех или иных программных продуктов и документов на различных этапах жизненного цикла

b. автоматически приводить документы к другой форме представления или переводить документ одного вида к документу другого вида

c. статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных, либо динамический анализ программ

d. выполнять на компьютере описание процессов или отдельных их частей, представленных в виде, отличном от машинного кода

3. Преобразователи обеспечивают…

a. конструирование тех или иных программных продуктов и документов на различных этапах жизненного цикла

b. автоматически приводить документы к другой форме представления или переводить документ одного вида к документу другого вида

c. статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных, либо динамический анализ программ

d. выполнять на компьютере описание процессов или отдельных их частей, представленных в виде, отличном от машинного кода

4. Сколько классов инструментальных средств выделяют в инструментальной среде разработки и сопровождения программ?

a. 2

b. 4

c. 3

d. 5

5. Среда программирования предназначена для…

a. конструирования тех или иных программных продуктов и документов на различных этапах жизненного цикла

b. автоматического перевода документов к другой форме представления или перевода документа одного вида к документу другого вида

c. поддержки ранних этапов разработки программ и автоматической генерации программ по спецификациям

d. поддержки процессов программирования (кодирования), тестирования и отладки программ

6. Инструментальные среды программирования бывают

a. языково-ориентированные среды и среды общего назначения

b. объектно-ориентированные и языково-ориентированные среды

c. среды общего назначения и прикладные среды

d. среды общего назначения, прикладные среды, логические и математические среды

7. Для поддержки разработки программного продукта на каком-либо одном языке программирования используют…

a. среду программирования общего назначения

b. языково-ориентированную среду программирования

c. интерпретирующую среду программирования

d. прикладную среду программирования

8. Синтаксически-управляемая инструментальная среда программирования базируется на знании

a. семантики языка программирования

b. синтаксиса языка программирования

c. синтаксиса и семантики языка программирования

d. основных управляющих структур языка программирования

9. Инструментальная система технологии программирования – это…

a. программное средство, предназначенное для поддержки разработки других программ

b. устройство компьютера, специально предназначенное для поддержки разработки программного средства

c. интегрированная совокупность программных и аппаратных инструментов, поддерживающая все процессы разработки и сопровождения больших программных продуктов

d. логически связанная совокупность программных и аппаратных инструментов, поддерживающих разработку ПП

10. Устройство компьютера, специально предназначенное для поддержки разработки программного средства, называется -…

a. аппаратным инструментом

b. программным инструментом

c. программной средой

d. инструментарий технологии программирования

11. Редакторы обеспечивают…

a. конструирование тех или иных программных продуктов и документов на различных этапах жизненного цикла

b. автоматически приводить документы к другой форме представления или переводить документ одного вида к документу другого вида

c. статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных, либо динамический анализ программ

d. выполнять на компьютере описание процессов или отдельных их частей, представленных в виде, отличном от машинного кода

12. Инструменты, поддерживающие процесс выполнения программ, обеспечивают…

a. конструирование тех или иных программных продуктов и документов на различных этапах жизненного цикла

b. автоматический привод документов к другой форме представления или перевод документа одного вида к документу другого вида

c. возможность выполнять на компьютере описание процессов или отдельных их частей, представленных в виде, отличном от машинного кода

d. статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных, либо динамический анализ программ

13. Инструментальная система технологии программирования предназначена для…

a. поддержки всех процессов разработки и сопровождения в течение всего жизненного цикла ПС и ориентирована на коллективную разработку больших программных систем с длительным жизненным циклом

b. автоматического перевода документов к другой форме представления или перевода документа одного вида к документу другого вида

c. поддержки ранних этапов разработки программ и автоматической генерации программ по спецификациям

d. поддержки процессов программирования (кодирования), тестирования и отладки программ

14. Рабочее место компьютерной технологии предназначено для…

a. конструирования тех или иных программных продуктов и документов на различных этапах жизненного цикла

b. автоматического перевода документов к другой форме представления или перевода документа одного вида к документу другого вида

c. поддержки ранних этапов разработки программ и автоматической генерации программ по спецификациям

d. поддержки процессов программирования (кодирования), тестирования и отладки программ

15. Инструментальные среды программирования содержат

a. редактор, анализатор и компилятор
b. редактор, интерпретатор и компилятор
c. интерпретатор, компилятор, преобразователь
d. редактор и интерпретатор

16. Для поддержки разработки программного продукта на разных языках программирования (например, текстовый редактор, редактор связей или интерпретатор языка целевого компьютера) используют…
a. среду программирования общего назначения
b. языково-ориентированную среду программирования
c. интерпретирующую среду программирования
d. прикладную среду программирования

17. При использовании компьютерных технологий для разработки ПП жизненный цикл ПП представлен следующей цепочкой
a. прототипирование – кодогенерация – комплексная отладка и тестирование – аттестация, применение, сопровождение
b. прототипирование – разработка спецификаций – автоматизированный контроль спецификаций – кодогенерация – комплексная отладка и тестирование – аттестация, применение, сопровождение
c. разработка спецификаций – автоматизированный контроль спецификаций – кодогенерация – комплексная отладка и тестирование – аттестация, применение, сопровождение
d. прототипирование – разработка спецификаций – кодогенерация – аттестация, применение, сопровождение

18. Основными чертами инструментальной системы технологии программирования являются…
a. массовость, дискретность, результативность, определенность, понятность
b. комплексность, ориентированность на коллективную разработку, технологическая определенность, интегрированность
c. актуальность, непротиворечивость, полнота
d. комплексность, актуальность, интегрированность, массовость, понятность

Применение CASE-средств

1. Современные крупные проекты информационных систем характеризуются следующими особенностями:
a. сложность описания, требующая тщательного моделирования и анализа данных и процессов
b. наличие совокупности тесно взаимодействующих компонентов c. наличие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений

d. невозможность интеграции существующих и вновь разрабатываемых приложений;

2. Под CASE-средства понимаются программные средства, поддерживающие…

a. процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО

b. процессы тиражирования программного продукта

c. процессы создания и эксплуатации программного продукта d. процессы компилирования и интерпретации программных продуктов

3. Репозитарий Case – средства – это…

a. специализированная база данных проекта, предназначенная для отображения состояния проектируемой системы в каждый момент времени

b. компонент, обеспечивающий создание и редактирование в интерактивном режиме элементов диаграмм и связей между ними

c. компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования

d. компонент, позволяющий получать информацию о проектах в виде отчетов

e. компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта

f. набор системных утилит по обслуживанию репозитария

4. Графический редактор Case – средства – это…

a. компонент, обеспечивающий создание и редактирование в интерактивном режиме элементов диаграмм и связей между ними

b. компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования

c. компонент, позволяющий получать информацию о проектах в виде отчетов

d. компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта

5. Верификатор Case – средства – это…

a. компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования

b. компонент, позволяющий получать информацию о проектах в виде отчетов

c. компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта

d. набор системных утилит по обслуживанию репозитария

6. Документатор проекта Case – средства – это…

a. компонент, позволяющий получать информацию о проектах в виде отчетов
b. компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта
c. набор системных утилит по обслуживанию репозитария
d. компонент, обеспечивающий создание и редактирование в интерактивном режиме элементов диаграмм и связей между ними

7. Сервис Case – средства – это…
a. компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования
b. компонент, позволяющий получать информацию о проектах в виде отчетов
c. компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта
d. набор системных утилит по обслуживанию репозитария

8. Администратор проекта Case – средства – это…
a. компонент, служащий для контроля правильности построения диаграмм в заданной методологии проектирования
b. компонент, позволяющий получать информацию о проектах в виде отчетов
c. компонент, выполняющий запуск проекта, задание начальных параметров и назначение и изменение прав доступа к элементам проекта
d. набор системных утилит по обслуживанию репозитария

9. Какие методологии проектирования используют Case – средства?
a. структурного и модульного проектирования
b. структурного и объектно-ориентированного проектирования
c. объектно-ориентированного и нисходящего проектирования
d. нисходящего и восходящего проектирования

10. Структурное проектирование системы основано на…
a. объектно-ориентированной декомпозиции
b. алгоритмической декомпозиции
c. модульной декомпозиции
d. функциональной декомпозиции

11. Объектно-ориентированное проектирование системы основано на…
a. объектно-ориентированной декомпозиции
b. алгоритмической декомпозиции
c. модульной декомпозиции
d. функциональной декомпозиции

12. Case – средства представляют собой…

a. набор инструментальных средств для проектирования программного продукта
b. набор программных средств для сопровождения программного продукта
c. набор программных и инструментальных средств, поддерживающие все процессы жизненного цикла программного продукта
d. набор аппаратных средств, поддерживающих все процессы жизненного цикла программного продукта

13. Компания-разработчик приобрела новое Case – средство. Сразу ли компания получит ожидаемый результат от применения новой технологии?
a. да
b. нет

14. Сколько классов Case – средств выделяют?
a. 5
b. 3
c. 7
d. 2

15. Case – средства анализа и проектирования, предназначенные для
a. моделирования данных и генерации схем баз данных
b. построения и анализа моделей деятельности организаций (предметной области) или моделей проектируемой системы
c. обеспечения комплексной поддержки требований к создаваемой системе
d. поддержки всего жизненного цикла программного продукта

16. Case – средства управления требованиями предназначены для
a. моделирования данных и генерации схем баз данных
b. построения и анализа моделей деятельности организаций (предметной области) или моделей проектируемой системы
c. обеспечения комплексной поддержки требований к создаваемой системе
d. поддержки всего жизненного цикла программного продукта

17. Case – средства проектирования баз данных предназначены для
a. моделирования данных и генерации схем баз данных
b. построения и анализа моделей деятельности организаций (предметной области) или моделей проектируемой системы
c. обеспечения комплексной поддержки требований к создаваемой системе
d. поддержки всего жизненного цикла программного продукта

18. На каких стандартах базируется технология освоения и внедрения Case – средств?
a. ГОСТ 2107-99
b. IEEE (IEEE Std 1348-1995 и IEEE Std 1209-1992)
c. AES
d. ISO

19. Из каких этапов состоит процесс освоения и внедрения Case – средств?
a. определение потребностей в CASE-средствах, оценка и выбор CASE-средств, практическое внедрение CASE-средств
b. определение потребностей в CASE-средствах, оценка и выбор CASE-средств, выполнение пилотного проекта, практическое внедрение CASE-средств
c. определение потребностей в CASE-средствах, проектирования CASE-средств, практическое применение CASE-средств
d. проектирование CASE-средств, оценка и внедрение CASE-средств, практическое применение CASE-средств

20. Критериями для выбора CASE-средств могут являться
a. открытая архитектура, поддержка полного жизненного цикла ИС с обеспечением эволюционности ее развития, обеспечение целостности проекта, независимость от программно-аппаратной платформы и СУБД
b. модифицируемость, простота, эффективность, учет человеческого фактора, многоплатформенность c. закрытая архитектура, поддержка полного жизненного цикла ИС с обеспечением эволюционности ее развития, простота, эффективность
d. максимальная зависимость от программных и аппаратных средств системы и характеристик самой системы, жесткая привязка к конкретным информационным процессам, прочность внутренней связи отдельных компонентов системы

21. Комплексность компьютерной поддержки разработки ПП с использованием инструментальной системы технологии программирования означает
a. что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана
b. что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов
c. что все инструменты объединены единым пользовательским интерфейсом

d. что инструменты действуют в соответствии с фиксированной информационной схемой системы, определяющей зависимость различных используемых в системе фрагментов данных друг от друга

22. Ориентированность инструментальной системы технологии программирования на коллективную разработку означает
a. что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана
b. что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов
c. что все инструменты объединены единым пользовательским интерфейсом
d. что инструменты действуют в соответствии с фиксированной информационной схемой системы, определяющей зависимость различных используемых в системе фрагментов данных друг от друга

23. Технологическая определенность инструментальной системы технологии программирования означает
a. что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана
b. что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов
c. что ее комплексность ограничивается рамками какой-либо конкретной технологии программирования
d. что инструменты действуют в соответствии с фиксированной информационной схемой системы, определяющей зависимость различных используемых в системе фрагментов данных друг от друга

24. Интегрированность инструментальной системы технологии программирования по данным означает
a. что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана
b. что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов
c. что ее комплексность ограничивается рамками какой-либо конкретной технологии программирования

d. что инструменты действуют в соответствии с фиксированной информационной схемой системы, определяющей зависимость различных используемых в системе фрагментов данных друг от друга

25. Интегрированность инструментальной системы технологии программирования по пользовательскому интерфейсу означает
a. что система технологии программирования охватывает все процессы разработки и сопровождения ПС и что продукция этих процессов согласована и взаимоувязана
b. что система технологии программирования должна поддерживать управление работой коллектива и для разных членов этого коллектива обеспечивать разные права доступа к различным фрагментам продукции технологических процессов
c. что ее комплексность ограничивается рамками какой-либо конкретной технологии программирования
d. что все инструменты объединены единым пользовательским интерфейсом

Критерии оценки тестовых заданий.

Оценка	Число правильных ответов
5(отлично)	90% - 100%
4(хорошо)	70% - 89%
3(удовлетворительно)	55% - 69%
2(неудовлетворительно)	55% и менее

3. Практические работы

Применение языка UML для проектирования программного обеспечения средствами IBM Rational Rose

Теоретическая часть

Rational Rose – CASE-средство, предназначенное для автоматизации этапов анализа и проектирования ПО, а также для генерации кодов на различных языках и выпуска проектной документации.

CASE-средство (Computer Aided Software Engineering) – это инструмент, который позволяет автоматизировать процесс разработки информационной системы и программного обеспечения.

UML (Unified Modeling Language) – это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных систем. В рамках языка UML все представления о модели сложной системы фиксируются в виде специальных графических конструкций, получивших название диаграмм.

Rational Rose позволяет генерировать программный код на различных языках программирования (Java, C++, VisualBasic, PowerBuilder), а так же осуществлять обратную генерацию диаграмм (реинжиниринг) на основе программного кода и выпускать проектную документацию.

В распоряжение проектировщика системы Rational Rose предоставляет следующие типы диаграмм, последовательное создание которых позволяет получить полное представление о всей проектируемой системе и об отдельных ее компонентах:

- Use case diagram (диаграммы прецедентов);
- Deployment diagram (диаграммы топологии);
- Statechart diagram (диаграммы состояний);
- Activity diagram (диаграммы активности);
- Interaction diagram (диаграммы взаимодействия);
- Sequence diagram (диаграммы последовательностей действий);
- Collaboration diagram (диаграммы сотрудничества);
- Class diagram (диаграммы классов);
- Component diagram (диаграммы компонент).

Диаграммы прецедентов (Use case diagram)

Этот вид диаграмм позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций.

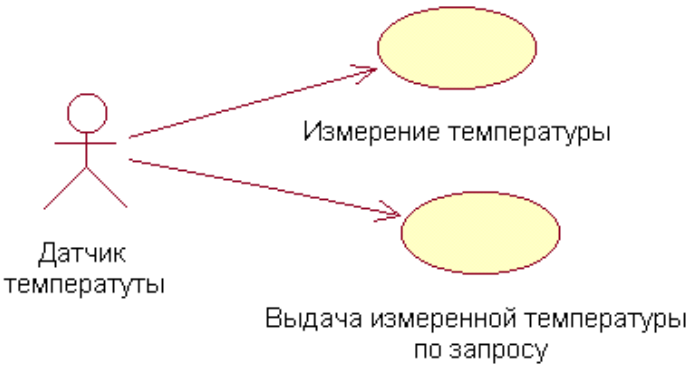


Рис.1. Диаграмма прецедентов

Каждая такая диаграмма или, как ее обычно называют, каждый Use case - это описание сценария поведения, которому следуют действующие лица (Actors).

Данный тип диаграмм используется при описании бизнес процессов автоматизируемой предметной области, определении требований к будущей программной системе. Отражает объекты как системы, так и предметной области и задачи, ими выполняемые.

Диаграммы топологии (Deployment diagram)

Этот вид диаграмм предназначен для анализа аппаратной части системы, то есть «железа», а не программ. В прямом переводе с английского Deployment означает «развертывание», но термин «топология» точнее отражает сущность этого типа диаграмм.

Для каждой модели создается только одна такая диаграмма, отображающая процессоры (Processor), устройства (Device) и их соединения.

Обычно этот тип диаграмм используется в самом начале проектирования системы для анализа аппаратных средств, на которых она будет эксплуатироваться.

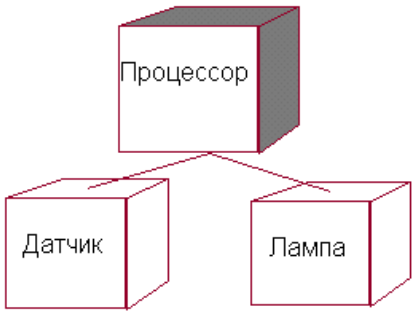


Рис.2. Диаграмма топологии

Диаграммы состояний (State Machine diagram)

Каждый объект системы, обладающий определенным поведением, может находится в определенных состояниях, переходить из состояния в состояние, совершая определенные действия в процессе реализации сценария поведения объекта. Поведение большинства объектов реальных систем можно представить с точки зрения теории конечных автоматов, то есть поведение объекта отражается в его состояниях, и данный тип диаграмм позволяет отразить это графически. Для этого используется два вида диаграмм: Statechart diagram (дмаграмма состояний) и Activity diagram (диаграмма активности).

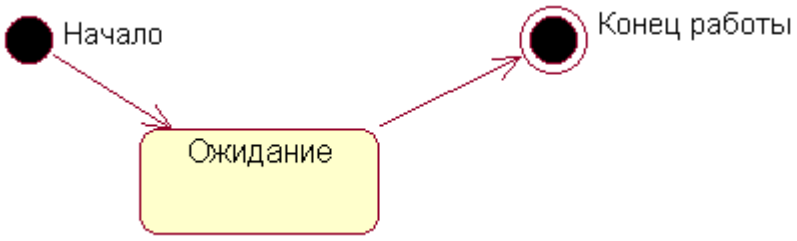


Рис.3. Диаграмма состояний

Диаграмма состояний (Statechart) предназначена для отображения состояний объектов системы, имеющих сложную модель поведения. Это одна из двух диаграмм State Machine, доступ к которой осуществляется из одного пункта меню.

Диаграммы активности (Activity diagram)

Это дальнейшее развитие диаграммы состояний. Фактически данный тип диаграмм может использоваться и для отражения состояний моделируемого объекта, однако, основное назначение Activity diagram в том, чтобы отражать бизнес-процессы объекта. Этот тип диаграмм позволяет показать не только последовательность процессов, но и ветвление и даже синхронизацию процессов.

Этот тип диаграмм позволяет проектировать алгоритмы поведения объектов любой сложности, в том числе может использоваться для составления блок-схем.

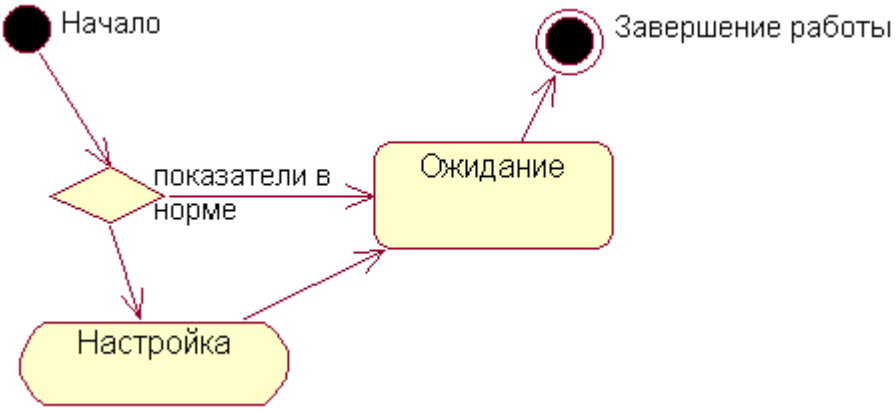


Рис.4. Диаграмма активности

Диаграммы взаимодействия (Interaction diagram)

Этот тип диаграмм включает в себя диаграммы Sequence diagram (диаграммы последовательностей действий) и Collaboration diagram (диаграммы сотрудничества). Эти диаграммы позволяют с разных точек зрения рассмотреть взаимодействие объектов в создаваемой системе.

Диаграммы последовательностей действий (Sequence diagram)

Взаимодействие объектов в системе происходит посредством приема и передачи сообщений объектами-клиентами и обработки этих сообщений объектами-серверами. При этом в разных ситуациях одни и те же объекты могут выступать и в качестве клиентов, и в качестве серверов.

Данный тип диаграмм позволяет отразить последовательность передачи сообщений между объектами.
Этот тип диаграммы не акцентирует внимание на конкретном взаимодействии, главный акцент уделяется последовательности приема/передачи сообщений. Для того чтобы окинуть взглядом все взаимосвязи объектов, служит Collaboration diagram.

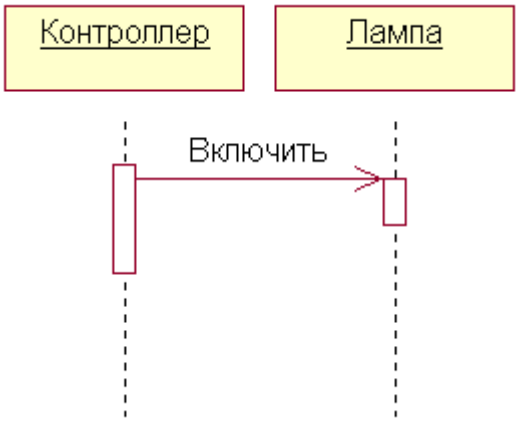


Рис.5. Диаграмма последовательностей действий

Диаграммы сотрудничества (Collaboration diagram)

Этот тип диаграмм позволяет описать взаимодействия объектов, абстрагируясь от последовательности передачи сообщений. На этом типе диаграмм в компактном виде отражаются все принимаемые и передаваемые сообщения конкретного объекта и типы этих сообщений.

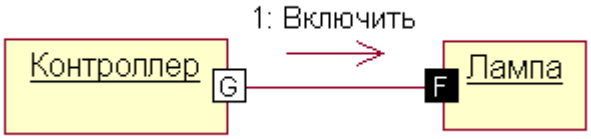


Рис.6. Диаграмма сотрудничества

По причине того, что диаграммы Sequence и Collaboration являются разными взглядами на одни и те же процессы, Rational Rose позволяет создавать из Sequence диаграммы диаграмму Collaboration и наоборот, а также производит автоматическую синхронизацию этих диаграмм.

Диаграммы классов (Class diagram)

Этот тип диаграмм позволяет создавать логическое представление системы, на основе которого создается исходный код описанных классов.
Значки диаграммы позволяют отображать сложную иерархию систем, взаимосвязи классов (Classes) и интерфейсов (Interfaces). Данный тип диаграмм противоположен по содержанию диаграмме Collaboration, на котором отображаются объекты системы. Rational Rose позволяет создавать классы при помощи данного типа диаграмм в различных нотациях. похожего на облако. Таким образом Г.Буч пытается показать, что класс - это лишь шаблон, по которому в дальнейшем будет создан конкретный объект.

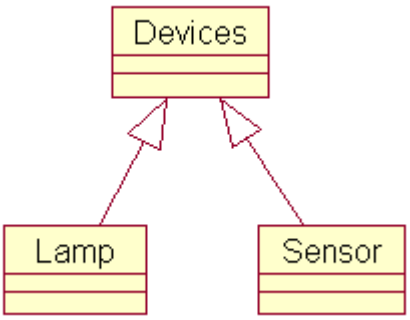


Рис.7. Диаграмма классов

Диаграммы компонентов (Component diagram)

Этот тип диаграмм предназначен для распределения классов и объектов по компонентам при физическом проектировании системы. Часто данный тип диаграмм называют диаграммами модулей.

При проектировании больших систем может оказаться, что система должна быть разложена на несколько сотен или даже тысяч компонентов, и этот тип диаграмм позволяет не потеряться в обилии модулей и их связей.

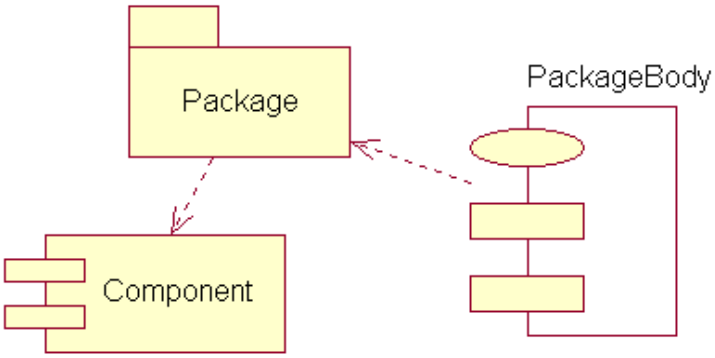


Рис.8. Диаграмма классов

1. Порядок выполнения лабораторной работы

Для выполнения всех лабораторных работ предлагается единый порядок, предусматривающий следующие шаги:

1. Ознакомиться с постановкой задачи и исходными данными.
2. Разработать предлагаемую в работе диаграмму.
3. Реализовать разработанную диаграмму в среде Rational Rose.
4. Сохранить файл модели.
5. Составить отчет по проделанной работе.
6. Представить отчет по лабораторной работе для защиты.

2. Защита отчета по лабораторной работе

Отчет по лабораторной работе должен состоять из следующих структурных элементов:

1. Титульный лист.
2. Текстовая часть.
3. Приложение: разработанная модель вариантов использования.

Текстовая часть отчета должна включать пункты:

– условие задачи;
– порядок выполнения

– краткие сведения о составе и компонентах построенной модели.

Зашита отчета по лабораторной работе заключается в предъявлении преподавателю полученных результатов и демонстрации полученных навыков при ответах на вопросы преподавателя.

ЛАБОРАТОРНАЯ РАБОТА № 1
Построение диаграммы вариантов разрабатываемого программного обеспечения средствами IBM Rational Rose

Цель работы: разработать диаграмму вариантов использования в среде Rational Rose для своего индивидуального задания.

Теоретическая часть. Моделирование бизнес-процессов предметной области

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать.

Действующее лицо (actor) – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ. Несмотря на то, что на диаграммах вариантов использования они изображаются в виде стилизованных человеческих фигурок, действующее лицо может также быть внешней системой, которой необходима некоторая информация от данной системы. Показывать на диаграмме действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования.

Действующие лица делятся на три основных типа – пользователи системы, другие системы, взаимодействующие с данной, и время. Время становится действующим лицом, если от него зависит запуск каких-либо событий в системе.

В языке UML на диаграммах вариантов использования поддерживается несколько типов связей между элементами диаграммы.

Это связи коммуникации (communication), включения (include), расширения (extend) и обобщения (generalization).

Связь коммуникации – это связь между вариантом использования и действующим лицом. На языке UML связи коммуникации показывают с помощью однонаправленной ассоциации (сплошной линии со стрелкой).

Направление стрелки позволяет понять, кто инициирует коммуникацию.

Связь включения применяется в тех ситуациях, когда имеется какой-либо фрагмент поведения системы, который повторяется более чем в одном варианте использования. С помощью таких связей обычно моделируют многократно используемую функциональность.

Связь расширения применяется при описании изменений в нормальном поведении системы. Она позволяет варианту использования только при необходимости использовать функциональные возможности другого.

На языке UML связи включения и расширения показывают в виде зависимостей с соответствующими стереотипами, как показано на рисунке 1.

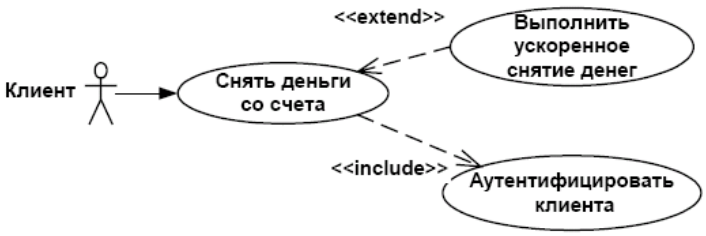
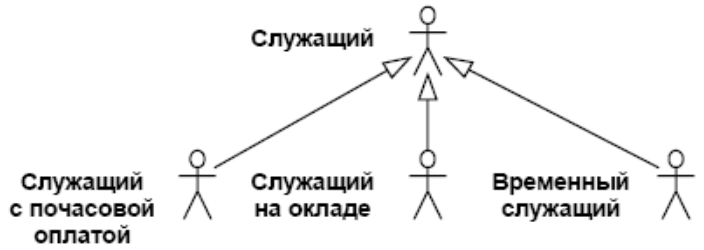


Рисунок 1 - Связи использования и расширения

С помощью связи обобщения показывают, что у нескольких действующих лиц имеются общие черты. Например, клиенты могут быть двух типов: корпоративные и индивидуальные. Эту связь можно моделировать с помощью



нотации, показанной на рисунке 3.

Рис.3. Обобщение действующего лица

Нет необходимости всегда создавать связи этого типа. В общем случае, они нужны, если поведение действующего лица одного типа отличается от поведения другого постольку, поскольку это затрагивает систему. Если оба подтипа используют одни и те же варианты использования, показывать обобщение действующего лица не требуется.

Порядок выполнения работы

1. Построить модель предметной области, согласно варианту (Приложение А) с помощью диаграммы вариантов использования UML.
2. Оформить отчет по лабораторной работе.
3. Представить отчет по лабораторной работе для защиты.

Порядок построения модели

Создание бизнес-схемы компании

1. Щелкните правой кнопкой мыши на представлении Use Case View в браузере.
2. Выберем пункт New далее Package
3. Назовем новый пакет «Общая схема»

Чтобы поместить действующее лицо в браузер:

1. Щелкните правой кнопкой мыши на пакете «Общая схема» представления Use Case View в браузере.
2. Выберите в открывшемся меню пункт New далее Actor
3. В браузере появится новое действующее лицо под названием NewClass. Слева от его имени вы увидите пиктограмму действующего лица UML.
4. Выделив новое действующее лицо, введите его имя.
5. Щелкните правой кнопкой мыши на действующем лице.

6. В открывшемся меню выберите пункт Open Specification.
7. В поле стереотипа выберите Business Actor и нажмите на кнопку ОК.
8. После создания действующих лиц сохраните модель с помощью пункта меню File затем Save.

Чтобы поместить вариант использования в браузер:

1. Щелкните правой кнопкой мыши на пакете «Общая схема» представления Use Case View в браузере.
2. Выберите в появившемся меню пункт New > Use Case
3. Новый вариант использования под названием NewUseCase появится в браузере. Слева от него будет видна пиктограмма варианта использования UML.
4. Выделив новый вариант использования, введите его название.
5. Щелкните правой кнопкой мыши на варианте использования.
6. В открывшемся меню выберите пункт Open Specification.
7. В поле стереотипа выберите Business Use Case и нажмите на кнопку ОК.

Для создания новой диаграммы вариантов использования:

1. Щелкните правой кнопкой мыши на пакете «Общая схема» представления Use Case View в браузере.
2. Из всплывающего меню выберите пункт New далее Use Case Diagram.
3. Выделив новую диаграмму, введите ее имя («Общая схема действий»).
4. Дважды щелкните на названии этой диаграммы в браузере, чтобы открыть ее.
5. Чтобы поместить действующее лицо или вариант использования на диаграмму, перетащите его мышью из браузера на диаграмму вариантов использования.
6. С помощью кнопки Unidirectional Association (Однонаправленная ассоциация) панели инструментов нарисуйте ассоциации между действующими лицами и вариантами использования.

Пример выполнения задания представлен в Приложении Б.

ЛАБОРАТОРНАЯ РАБОТА № 2
Построение UML диаграмм, изображающих логические схемы баз данных, средствами IBM Rational Rose

Цель работы: смоделировать базу данных на примере построения диаграмм классов средствами IBM Rational Rose.

Теоретическая часть. Моделирование требований к программному обеспечению

Разработка логической *структуры.* После завершения формирования принципов использования системы, наступает этап разработки ее логической структуры. В Rational Rose он именуется "Logical View". Логическое представление, концентрируется на том, как система будет реализовывать поведение, описанное в вариантах использования. Оно дает подробную картину составных частей системы и описывает взаимодействие этих частей. Логическое представление включает, помимо прочего, конкретные требуемые классы, диаграммы классов и диаграммы состояний. С их помощью конструируется детальный проект создаваемой системы.

Логическое представление содержит:

- классы,
- диаграммы классов.

  Как правило, для описания системы используется несколько диаграмм классов, каждая из которых отображает некоторое подмножество всех классов системы,
- диаграммы взаимодействия, применяемые для отображения объектов, участвующих в одном потоке событий варианта использования,
- диаграммы состояний,
- пакеты, являющиеся группами взаимосвязанных классов.

Для каждого блока использования строится диаграмма взаимодействия, на которой отображается взаимодействие во времени объектов, выполняющих поставленную задачу. На подобных диаграммах идентифицируются объекты системы и определяются сообщения, с помощью которых эти объекты взаимодействуют

Диаграммы взаимодействия (interaction diagrams) описывают поведение взаимодействующих групп объектов. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой.

Сообщение (message) – это средство, с помощью которого объект-отправитель запрашивает у объекта получателя выполнение одной из его операций.

Информационное (informative) сообщение – это сообщение, снабжающее объект-получатель некоторой информацией для обновления его состояния.

Сообщение-запрос (interrogative) – это сообщение, запрашивающее выдачу некоторой информации об объекте-получателе.

Императивное (imperative) сообщение – это сообщение, запрашивающее у объекта-получателя выполнение некоторых действий.

Существует два вида диаграмм взаимодействия: диаграммы последовательности (sequence diagrams) и кооперативные диаграммы (collaboration diagrams).

Каждый объект на диаграмме последовательностей сопровождается именем класса, к которому он принадлежит. Конкретный объект является экземпляром некоторого класса. Классы образуют логическую структуру системы.

Результатом данного этапа должна стать главная диаграмма, и детализирующие диаграммы для ее элементов.

На этом этапе следует определить классы, которые необходимы в системе. Экземпляры этих классов уже указаны на диаграммах последовательностей. Классы и их связи отражается в модели в виде диаграммы классов. Группы классов на этих диаграммах могут быть объединены в пакеты.

Проектирование логической структуры следует начинать с определения основных пакетов. Пакет – универсальное средство для группировки элементов модели. Применение пакетов позволяет сделать модель более обозримой. Пакеты могут быть вложенными друг в друга. Классы, составляющие каждый пакет, детализируются на вложенной диаграмме.

Идентификация основных абстракций заключается в предварительном определении набора классов системы (классов анализа) на основе описания предметной области.

Все классы и диаграммы, описывающие системный проект, помещаются в пакет с именем Design Model.

Диаграммы классов, реализующие варианты использования и диаграммы взаимодействия, отражающие взаимодействие объектов в процессе реализации сценариев варианта использования, помещаются в кооперацию с именем данного варианта использования и стереотипом «use-case realization». Все кооперации помещаются в пакет с именем Use-Case Realization. Связь между вариантом использования и его реализацией изображается на специальной диаграмме трассировки (рис.4).

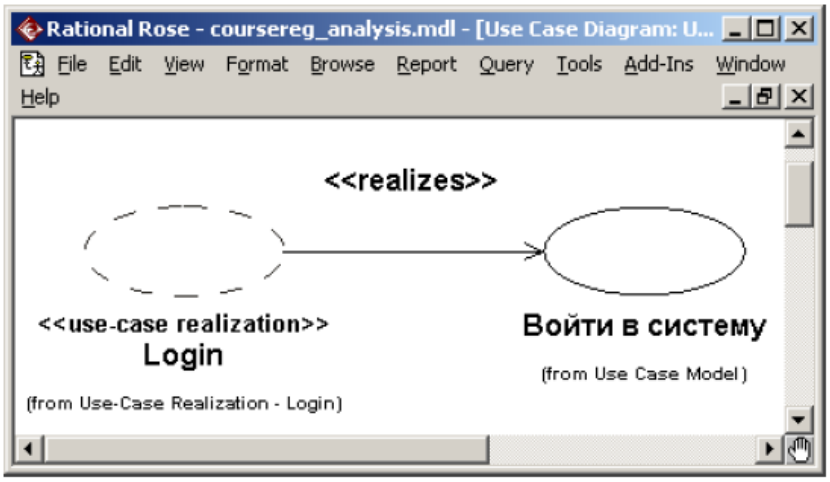


Рис.4. Диаграмма трассировки

Порядок выполнения работы

1. Для реализации сценариев варианта использования постройте
a. диаграммы классов, реализующих вариант использования,

б. диаграммы взаимодействия, отражающие взаимодействие объектов в процессе.
в. кооперативные диаграммы для построенных диаграмм взаимодействия
2. Все построенные диаграммы помещаются в кооперацию с именем данного варианта использования и стереотипом «use-case realization». Все кооперации помещаются в пакет с именем Use Case Realizations.
3. Оформить отчет по лабораторной работе.
4. Представить отчет по лабораторной работе для защиты.

Порядок построения модели

Создание классов

1. Щелкните правой кнопкой мыши на представлении Logical View.
2. Выберите в открывшемся меню пункт New\Class. Новый класс под названием NewClass появится в браузере.
3. Выделите его и введите имя класса.
4. Щелкните правой кнопкой мыши на созданном классе.
5. В открывшемся меню выберите пункт Open Specification.
6. В поле стереотипа выберите необходимый стереотип (Boundary, Control, Entity) и нажмите на кнопку ОК.
7. Откройте диаграмму Main и перетащите созданные классы.

Создание пакетов и диаграммы Traceabilities:

8. Щелкните правой кнопкой мыши на представлении Logic View.
9. В открывшемся меню выберите пункт New\Package.
10. Создайте пакет Use-Case Realizations, затем внутри него – пакеты соответствующие построенным вариантам использования.
11. В каждом из пакетов создайте соответствующие кооперации (каждая кооперация представляет собой вариант использования со стереотипом «use-case realization», который задается в спецификации варианта использования).
12. Создайте в пакете Use-Case Realizations новую диаграмму вариантов использования с названием Traceabilities, которая показывает связь между вариантом использования и его реализацией (диаграмма трассировки).

Создание диаграмм взаимодействия

Настройка

1. В меню модели выберите пункт Tools далее Options.
2. Перейдите на вкладку диаграмм.
3. Контрольные переключатели Sequence Numbering, Collaboration Numbering должны быть помечены, а Focus of Control – нет.
4. Нажмите ОК, чтобы выйти из окна параметров.

Создание диаграммы последовательности

1. Щелкните правой кнопкой мыши на кооперации
2. В открывшемся меню выберите пункт New далее Sequence Diagram.
3. Назовите новую диаграмму.
4. Дважды щелкните на ней, чтобы открыть ее.

Добавление на диаграмму действующего лица, объектов и сообщений

1. Перетащите действующее лицо из браузера на диаграмму.
2. Перетащите классы из браузера на диаграмму.
3. На панели инструментов нажмите кнопку Object Message (Сообщение объекта).

4. Проведите мышью от линии жизни действующего лица к линии жизни объекта
5. Выделив сообщение, введите его имя.
6. Повторите действия 3 – 5, чтобы поместить на диаграмму остальные сообщения (для рефлексивного сообщения используется кнопка Message to Self).

Соотнесение сообщений с операциями

1. Щелкните правой кнопкой на тексте сообщении
2. В открывшемся меню выберите пункт <new operation>. Появится окно спецификации операции.
3. В поле имени оставьте имя сообщения.
4. Нажмите на кнопку ОК, чтобы закрыть окно спецификации операции и вернуться на диаграмму.
5. Повторите действия 1 – 4, пока не соотнесете с операциями все остальные сообщения.

Создание примечаний

Чтобы поместить на диаграмму примечание:

1. Нажмите на панели инструментов кнопку Note.
2. Щелкните мышью в том месте диаграммы, куда собираетесь поместить примечание.
3. Выделив новое примечание, введите туда текст.
4. Чтобы прикрепить примечание к элементу диаграммы, на панели инструментов нажмите кнопку Anchor Notes To Item (Прикрепить примечания к элементу).
5. Нажав левую кнопку мыши, проведите указатель от примечания до элемента диаграммы, с которым оно будет связано. Между примечанием и элементом возникнет штриховая линия.
6. Чтобы создать примечание-ссылку на другую диаграмму создайте пустое примечание (без текста) и перетащите на него из браузера нужную диаграмму.

Чтобы поместить на диаграмму текстовую область:

1. На панели управления нажмите кнопку Text Box.
2. Щелкните мышью внутри диаграммы, чтобы поместить туда текстовую область.
3. Выделив эту область, введите в нее текст.

Создание кооперативной диаграммы

Для создания кооперативной диаграммы достаточно открыть диаграмму последовательности и нажать клавишу F5.

Пример выполнения задания представлен в Приложении В.

ЛАБОРАТОРНАЯ РАБОТА № 3
Разработка диаграммы компонентов и генерация программного кода в среде IBM Rational Rose

Цель работы: разработать диаграмму компонентов и сгенерировать программный код спроектированного программного обеспечения средствами IBM Rational Rose.

Диаграммы компонентов показывают, как выглядит модель на физическом уровне. На них изображены компоненты программного обеспечения и связи между ними. При этом на такой диаграмме выделяют два типа компонентов: исполняемые компоненты и библиотеки кода.

Каждый класс модели (или подсистема) преобразуется в компонент исходного кода. После создания они сразу добавляются к диаграмме компонентов. Между отдельными компонентами изображают зависимости, соответствующие зависимостям на этапе компиляции или выполнения программы. Пример диаграммы компонентов показан на рисунке 5

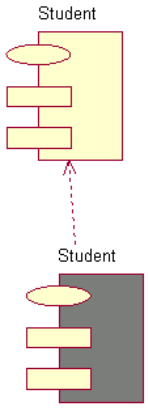


Рисунок 5 – Пример диаграммы компонентов.

Представление компонентов содержит:

– Компоненты, являющиеся физическими модулями кода.

– Диаграммы компонентов.

– Пакеты, являющиеся группами связанных компонентов.

Порядок выполнения работы

5. Для реализации построенной системы
а. постройте диаграмму компонентов
б. выполните проверку корректности модели
в. выполните генерацию кода,
6. Оформить отчет по лабораторной работе.
7. Представить отчет по лабораторной работе для защиты.

Порядок построения модели

Создание диаграммы компонентов:

1. Дважды щелкните мышью по главной диаграмме компонентов в представлении компонентов.
2. На панели инструментов нажмите кнопку Package Specification.
3. Поместите спецификацию пакета на диаграмму.
4. Введите имя спецификации пакета и укажите в окне спецификации язык программирования для генерации кода.

5. На панели инструментов нажмите кнопку Package Body.
6. Поместите тело пакета на диаграмму.
7. Введите имя тела пакета и укажите в окне спецификации язык генерации кода
8. На панели инструментов нажмите кнопку Dependency.
9. Проведите линию зависимости от тела пакета к спецификации пакета.

Соотнесение классов с компонентами:

1. В логическом представлении браузера найдите необходимый для генерации класс.
2. Перетащите этот класс на спецификацию пакета компонента в представлении компонентов браузера. В результате класс будет соотнесен со спецификацией пакета компонента.

Процесс генерации кода состоит из четырех основных шагов:

1. Проверка корректности модели.
2. Установка свойств генерации кода.
3. Выбор класса, компонента или пакета.
4. Генерация кода.

Для проверки модели:

1. Выберите в меню Tools\Check Model.
2. Проанализируйте все найденные ошибки в окне журнала, используя команду View\Log.

К наиболее распространенным ошибкам относятся такие, например, как сообщения на диаграмме последовательности или кооперативной диаграмме, не соотнесенные с операцией, либо объекты этих диаграмм, не соотнесенные с классом.

С помощью пункта меню Check Model можно выявить большую часть неточностей и ошибок в модели. Пункт меню Access Violations позволяет обнаруживать нарушения правил доступа, возникающие тогда, когда существует связь между двумя классами разных пакетов, но связи между самими пакетами нет.

Для того чтобы обнаружить нарушение правил доступа:

1. Выберите в меню Report\Show Access Violations.
2. Проанализируйте все нарушения правил доступа в окне.

Можно установить несколько параметров генерации кода для классов, атрибутов, компонентов и других элементов модели. Этими свойствами определяется способ генерации программ. Для каждого языка в Rose предусмотрен ряд определенных свойств генерации кода. Перед генерацией кода рекомендуется анализировать эти свойства и вносить необходимые изменения.

Для анализа свойств генерации кода

1. выберите Tools\Options, а затем вкладку соответствующего языка.
2. в окне списка можно выбрать класс, атрибут, операцию и другие элементы модели.

Для каждого языка в этом списке указаны свои собственные элементы модели. При выборе разных значений на экране появляются разные наборы свойств.

Для изменения свойства генерации кода для одного класса, атрибута, одной операции и т.д. нужно

1. открыть окно спецификации элемента модели. Выбрать вкладку языка (C++, Java,...) и изменить свойства. Все изменения, вносимые в окне спецификации элемента модели, оказывают влияние только на этот элемент.

   При генерации кода за один раз можно создать класс, компонент или целый пакет. Код генерируется с помощью диаграммы или браузера. При генерации кода из пакета можно выбрать или пакет логического представления на диаграмме классов, или пакет представления компонентов на диаграмме компонентов. При выборе пакета логического представления генерируются все классы этого пакета. При выборе пакета представления компонентов генерируются все компоненты этого пакета.

   После выбора класса или компонента на диаграмме выберите в меню соответствующий вариант генерации кода. Сообщения об ошибках, возникающих в процессе генерации кода, будут появляться в окне журнала.

   Во время генерации кода Rose выбирает информацию из логического и компонентного представлений модели и генерирует большой объем «скелетного» (skeletal) кода:

Генерация кода

3. Откройте диаграмму компонентов системы.
4. Выберите все объекты на диаграмме компонентов.
5. Выберите Tools\(язык для генерации кода)\Code Generation в меню.
6. Выполните генерацию кода.
7. Просмотрите результаты генерации (меню Tools\\(язык для генерации кода)\ Browse Header и Tools\\(язык для генерации кода)\Browse Body.

Приложение А

Варианты заданий

Номер варианта	Задание
1.	Опишите процесс организации работы детективного агентства с точки зрения ее работников. Разработать программный модуль «Детективное агентство», содержащий сведения о клиентах агентства и об оказанных услугах. Программный модуль предназначен для учета средств за оказанные услуги.
2.	Опишите процесс работы музея с точки зрения его служащего. Разработать программный модуль «Музей», предназначенный для использования работниками музея. В базе содержатся сведения об экспонатах музея и вносятся данные при поступлении новых экземпляров. При выполнении инвентаризации данные заносятся в базу, проводится сверка и выдаются отчеты по учету экспонатов в музее.
3.	Опишите процесс работы книжного магазина с точки зрения его служащего. Разработать программный модуль «Книжный магазин», содержащий сведения о книгах (автор, название, издательство, год издания, цена). Покупатель оформляет заявку на нужные ему книги, если таковых нет, он заносится в базу и оповещается, когда нужные книги поступают в магазин.
4.	Опишите процесс работы автостоянки с точки зрения ее служащего. Разработать программный модуль «Автостоянка». В программе содержится информация о марке автомобиля, его владельце, дате и времени въезда, стоимости стоянки, скидках, задолженности по оплате и др.
5.	Опишите процесс организации работы гостиницы с точки зрения администратора. Разработать программный модуль «Гостиница», содержащий сведения о наличии свободных мест и о проживающих в гостинице. Программный модуль предназначен для бронирования мест в гостинице и оформления проживающих.
6.	Опишите процесс работы АТС с точки зрения ее служащего. Разработать программный модуль «Картотека абонентов АТС». Картотека содержит сведения о телефонах и их владельцах. Фиксирует задолженности по оплате (абонентской и повременной). Считается, что повременная оплата местных телефонных разговоров уже введена.
7.	Опишите процесс организации работы с нарушителями правил дорожного движения с точки зрения работника полиции. Разработать программный модуль «Учет нарушений правил дорожного движения». Для каждой автомашины (и ее владельца) в базе хранится список нарушений. Для каждого нарушения

	фиксируется дата, время, вид нарушения и размер штрафа. При оплате всех штрафов машина удаляется из базы.
8.	Опишите процесс работы химчистки с точки зрения ее служащего. Разработать программный модуль «Химчистка». При записи на обслуживание заполняется заявка, в которой указываются ФИО владельца, описание изделия, вид услуги, дата приема заказа и стоимость услуги. После выполнения работ распечатывается квитанция.
9.	Опишите процесс организации рабочего дня руководителя с точки зрения его секретаря. Разработать приложение «Органайзер». Приложение предназначено для записи, хранения и поиска адресов и телефонов физических лиц и организаций, а также расписания, встреч и др. Приложение предназначено для организации рабочего дня руководителя.
10.	Создать информационную систему, содержащую сведения о деятельности турагенств продающих путевки в разные страны, курорты, отели и, включая характеристики этих отелей.
11.	Программное средство Call-центра технической поддержки сотового оператора, обеспечивающее клиента (актёр) функциями связи с менеджером (актёр), выдачи баланса, консалтинга по тарифам, подключения / отключения услуг и не менее 5 других функций. Функция со справкой по тарифам должна предполагать внутреннее голосовое меню.
12.	Программное средство банкомата, выполняющего только функцию выдачи денег. Взаимодействуют клиент, система обслуживания, купюроприёмник, кнопочная панель, определяющая снимаемую сумму и кнопки подтверждения или отмены. Функции клиента – вставить карту, ввести код, запросить баланс, набрать сумму, получить деньги. Функции система обслуживания – получать и распознавать сигналы с датчиков кнопок, выводить на кран распознанные цифры, выводить на экран необходимые приглашения и сообщения (на схеме они должны быть расшифрованы), анализировать наличие в банкомате средств, отдавать в купюроприёмник информацию о нужном количестве денег на выдачу, формировать и распечатывать выписку, функция купюроприёмника – выдача денег; функции кнопок – формирование соответствующих сигналов нажатия.
13.	Программное средство Call-центра салона красоты, обеспечивающее взаимодействие клиента, менеджера и базы учёта. Call-центр выдаёт стандартное приветствие и предоставляет клиенту голосовое меню, с помощью которого он может получить информацию об услугах салона, записаться на сеанс к стилисту, вейзажисту, массажисту, парикмахеру и т.д., связаться с менеджером, отменить, изменить время сеанса, записаться на

	повторный приём, получить справку. Менеджер может напрямую работать с базой учёта, внося туда сведения о клиентах и мастерах.
14.	Программное средство формирования командировочного удостоверения преподавателя вуза. Программное средство управляется бухгалтером. Преподаватель заполняет электронную анкету с типом командировки (по России или международная), местом назначения, временем пребывания, целью и способом проезда к месту командировки. Руководство вуза выдаёт (или не выдаёт) разрешение на командировку, в результате чего на основе анкетных данных преподавателя и реквизитов вуза, полученных из учётной базы данных, формируется проект приказа на командировку преподавателя, который получает преподаватель и предоставляет бухгалтеру. Бухгалтер оценивает по справочникам примерную стоимость проезда, проживания и величину суточных расходов и формирует величину аванса, который начисляется программным средством на зарплатную карточку преподавателя. По истечении командировки преподаватель предоставляет бухгалтеру чеки, подтверждающие использование аванса. Если потрачено меньше, чем запланировано, то осуществляется удержание с преподавателя разницы, если больше – то доначисление.
15.	Программное средство библиотечного каталога вуза, взаимодействующее со студентами, библиотекарями и преподавателями. Функции преподавателей – поиск учебников, заказ на закупку учебников; функции библиотекарей – внесение (исключение) записей в базу данных (или подтверждение по запросам студентов через интернет), распределение полномочий; функции студентов – алфавитный, тематический и не менее 2-х видов других поисков, запрос электронной версии, запрос на заказ учебника, запрос на справку, связь с библиотекарем через интернет или телефон.
16.	Разработать программное средство для регистратуры детской стоматологической клиники, имеющее агентов: секретарь, врачи и клиенты. Функции секретаря – внесение, исключение записей в базе данных (или подтверждение по запросам клиентов через интернет), запрос расписания врачей, определение времени приёма по справочнику, исключение или перенос записей по требованию врача с уведомлением клиентов, связь с врачом. Функции врача – анализ заявок на обслуживание и формирование предложений по оптимизации записей при наличии ошибок или иным обстоятельствам, заполнение карточки по результатам приёма; функции клиентов – поиск информации о расписании врачей, получение информации об опыте, образовании и достижениях врачей, запись на приём (по телефону, лично у секретаря или через

	интернет), запрос на справку у врача, связь с секретарём через интернет или телефон.
17.	Разработать программное средство регистрации клиентов в гостинице, включающее агентов: администратор, персонал, клиент. Функции администратора: внесение паспортных и анкетных данных клиента, получение сведений о свободном номерном фонде, бронирование или немедленное предоставление ключа доступа в комнату клиента, выписка клиента, выдача справок клиенту, приём заявок на вызов персонала гостиницы (горничные, плотники, слесаря и т.д.); функции персонала – регистрация времени посещения клиента и сделанных работ; функции клиента – бронирование по телефону, лично у администратора или через интернет номера в гостинице; заполнение через интернет анкеты для заселения или передача информации администратору, получение справки у администратора, связь с администратором, получение карты доступа в комнату, использование карты доступа в номер.
18.	Опишите процесс работы автомагазина с точки зрения его служащего. Разработать программный модуль «Картотека автомагазина», предназначенный для использования работниками магазина. В базе содержатся сведения об автомобилях (марка, объем двигателя, дата выпуска и др.). При поступлении заявки на покупку производится поиск подходящего варианта. Если такого нет, клиент заносится в клиентскую базу и оповещается, когда вариант появляется.
19.	Опишите процесс организации работы автостанции с точки зрения ее служащего. Разработать программный модуль «Автокасса», содержащий сведения о наличии свободных мест на автобусные маршруты. В базе должны содержаться сведения о номере рейса, маршруте, водителе, типе автобуса, дате и времени отправления, а также стоимости билетов. При поступлении заявки на билеты программа производит поиск подходящего рейса.
20.	Опишите процесс учета посещения студентов учебных занятий и успеваемости студентов с точки зрения работника деканата. Разработать программный модуль «Учет успеваемости студентов». Программный модуль предназначен для оперативного учета успеваемости студентов в сессию деканом, заместителями декана и сотрудниками деканата. Сведения об успеваемости студентов должны храниться в течение всего срока их обучения и использоваться при составлении справок о прослушанных курсах и приложений к диплому.

Приложение Б

Пример выполнения лабораторной работы № 1

Ознакомьтесь с окном Rational Rose (*рис. 1*):

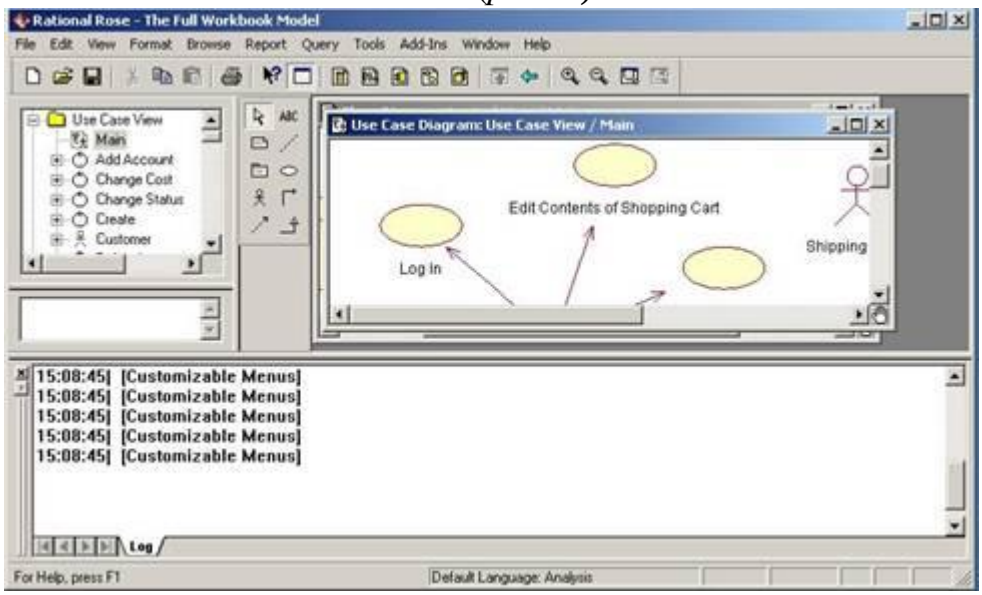


Рис. 1. Окно Rational Rose

- *браузер программы* - располагается под строкой меню и строкой инструментов в левой части окна приложения. Браузер (навигатор) является инструментом навигации по иерархической структуре модели. Знак + рядом с папкой означает, что в ней имеются дополнительные элементы;

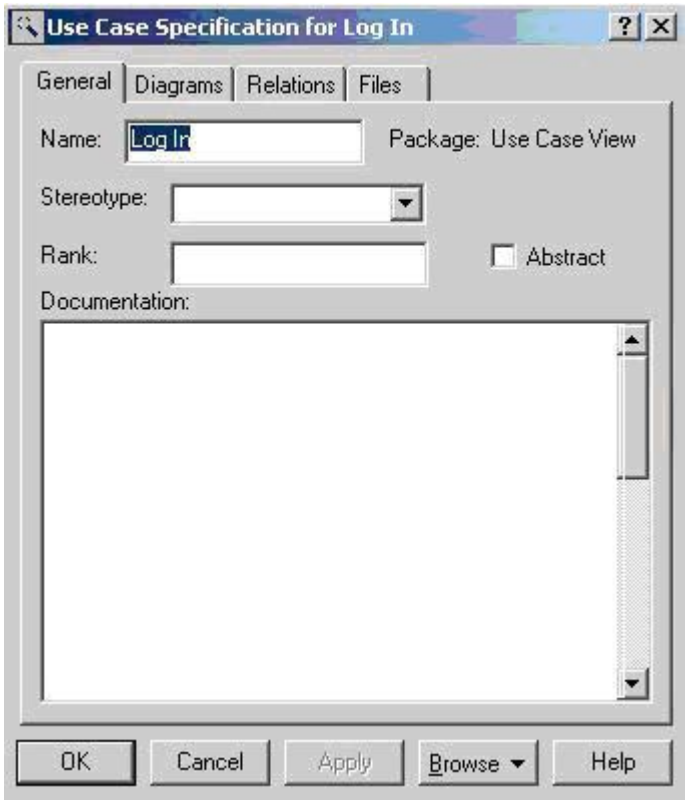


Рис. 2. Окно спецификаций

- *панель "инструменты диаграммы"* - с права от окна браузера. Она меняется в зависимости от выбранной диаграммы;

- *окно диаграммы*, используется для создания, отображения и изменения диаграмм на языке UML. В нем находятся разные диаграммы. Активная диаграмма имеет синий заголовок;
- окно спецификации – позволяет задавать характеристики элемента диаграммы (*рис.2*);
- окно докумен тации – окно для вывода словесного описания данного элемента. Описание можно вводить и в поле Documentation окна спецификации.

Как показано на *рис. 3*, кнопки строки инструментов позволяют выполнять стан дартные и специальные действия.

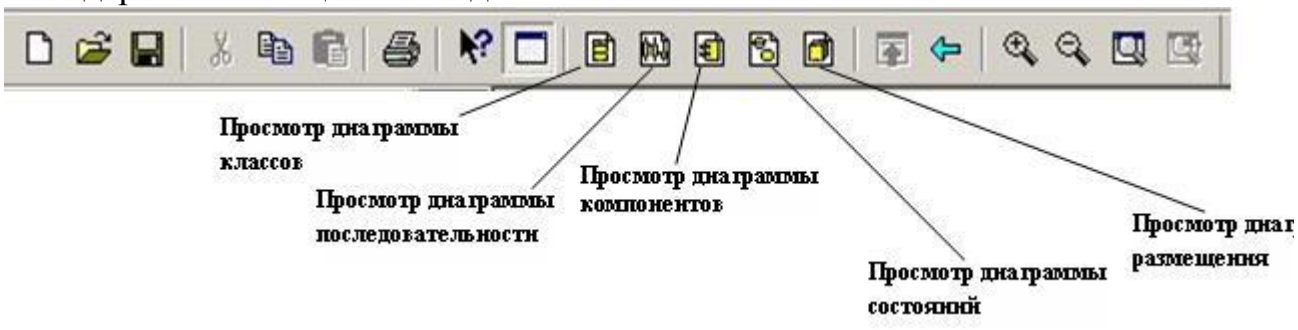


Рис. 3. Строка инструментов.

Ознакомьтесь с постановкой задачи.

Постановка задачи:

Построить модель Университетской системы для регистрации учебных курсов. Система регистрации учебных курсов используется:

- *профессором* — для задания читаемого курса;
- *студентом* — для выбора изучаемого курса;
- *регистратором* — для формирования учебного плана и расписания;
- *учетной системой* — для определения денежных затрат.

Постановка задачи:

Построить модель Университетской системы для регистрации учебных курсов. Система регистрации учебных курсов используется:

- *профессором* — для задания читаемого курса;
- *студентом* — для выбора изучаемого курса;
- *регистратором* — для формирования учебного плана и расписания;
- *учетной системой* — для определения денежных затрат.

Построение диаграммы Use Case

Диаграмма**Use Case** строится с целью определения объектов системы и их взаимодействия. Они представляются актерами, элементами Use Case и отно шениями между ними.

Первый шаг построения этой диаграммы состоит в определении *актеров* , фиксирующих роли внешних объектов, взаимодействующих с системой. В рассматриваемом примере можно выделить 4 актера: *Student (Студент)*, *Professor (Про фессор)*, *Registrar (Регистратор)* и *Billing System (Учетная система)* (*рис. 5*).

Элемент Use Case представляет определенную часть функциональности, обеспе чиваемой системой при взаимодействии с ней каждого актера.

Откройте главную диаграмму Use Case (*рис. 4*).
В окне браузера щелкните по значку + слева от пакета Use Case View.
Откройте диаграмму, выполнив двойной щелчок по значку Main.

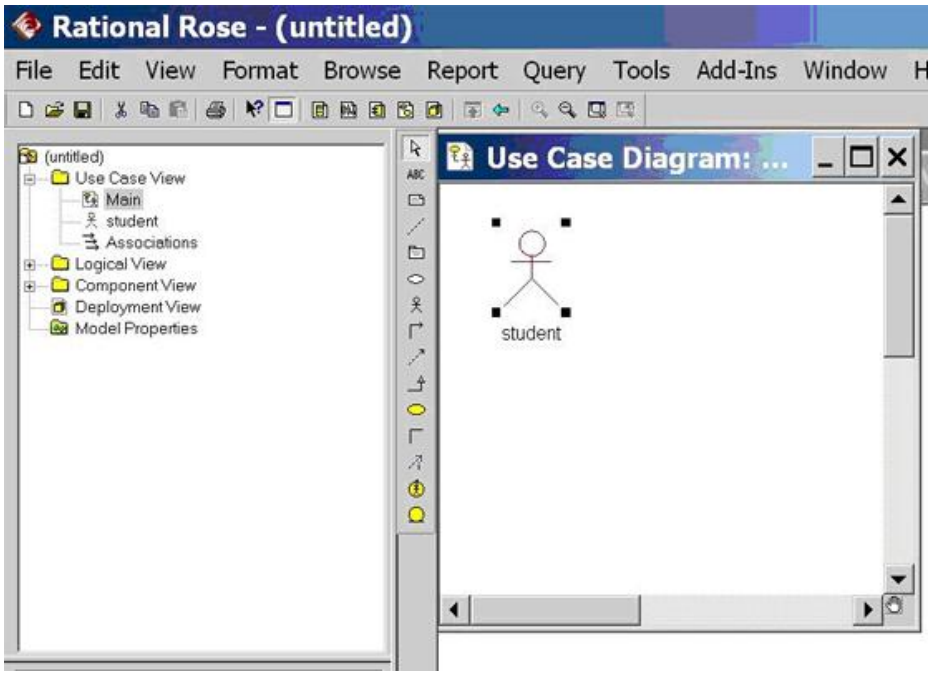


Рис. 4. Главная диаграмма Use Case

Проблемная область Университетской системы взаимодействует с 4 актерами:

1. *Student (Студент)* – регистрируется на курсы (Register for Courses);
2. *Professor (Про фессор)*. Professor запрашивает спи сок курса (Request a Course Roster);
3. *Registrar (Регистратор)*. Registrar управляет учебным планом (Manage Curriculum);
4. *Billing System (Учетная система)* - получает информацию о регистрации (*рис. 5*).

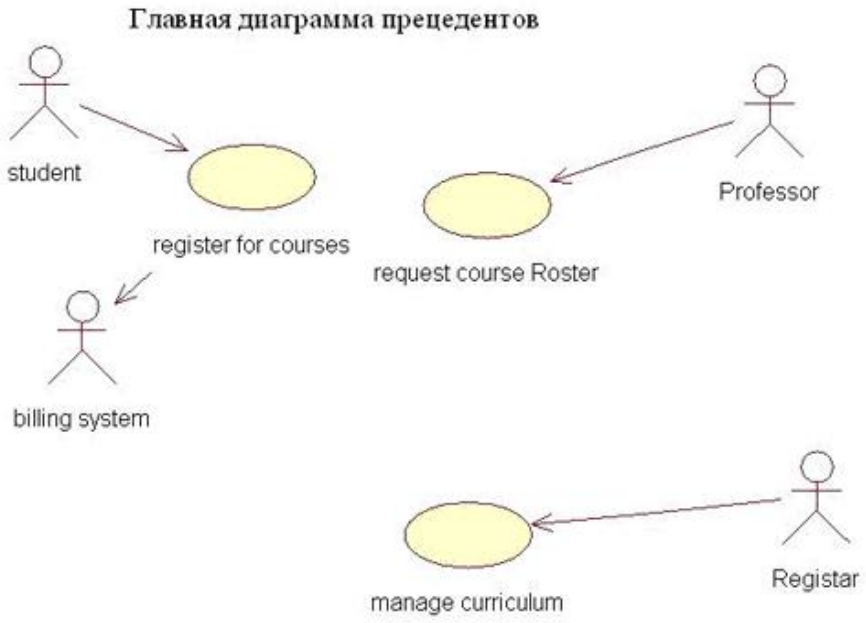


Рис. 5. Актеры и элементы Use Case

Последовательно создайте 4 актеров:

- щелкните по значку актера (человечек) на панели инструментов;
- щелкните в нужном месте диаграммы для добавления актера в диаграмму;
- введите имя Student (Студент), пока актер остается выделенным;
- повторите предыдущие шаги для ввода трех других актеров (Professor, Registrar и Billing System — Профессор, Регистратор, Учетная система).

Определите для каждого актера соответствующие элементы Use Case:

- добавьте элемент Use Case в диаграмму щелчком по значку элемента Use Case и затем щелкните в нужном месте диаграммы;
- введите имя (Register for Courses), пока элемент Use Case остается выделенным;
- повторите предыдущие шаги для ввода других элементов Use Case (Request Course Roster, Manage Curriculum).

Отобразите отношения между актерами и элементами Use Case, указав направление взаимодействия (кто инициирует взаимодействие), используя однонаправленные стрелки (unidirectional arrows):

- щелкните по значку однонаправленной ассоциации (стрелке) на панели инструментов;
- щелкните по актеру.Student и переместите линию на элемент Use Case Register for Courses.

Повторите предыдущие шаги для ввода других отношений:

- от актера Professor к элементу Use Case Request Course Roster;
- от актера Registrar к элементу Use Case Manage Curriculum

Комментарий:

В системе регистрации курсов:

- актер Student инициирует элемент Use Case Register for Courses, этот же элемент, в свою очередь, взаимодействует с актером Billing System;
- актер Professor инициирует элемент Use Case Request Course Roster;
- актер Registrar инициирует элемент Use Case Manage Curriculum (*рис. 5*).

Диаграммы последовательностей

Функциональность элемента Use Case отображается графически в диаграмме по следовательности (Sequence Diagram). Она отображает один из возмож ных путей в потоках событий элемента Use Case — например, добавление студента к курсу.

Диаграммы последовательности содержат объекты и сообщения между объек тами, которые показывают реализацию поведения.

Сценарий предполагает, что студент должен заполнить информацией (fill in info) регистрацион ную форму (registration form), а затем форма предъявляется на рассмотрение (submitted).

Очевидно, что необходим объект *registration form*, который принимает информацию от студента.

Создайте диаграмму последо вательности для элемента Use Case Register for Courses (*рис. 6*):

- Щелкните правой кнопкой по элементу Use Case Register for Courses в окне браузера;

- Выберите команду New: Sequence Diagram в появившемся контекстном меню;
- Убедитесь в добавлении в дерево окна браузера диаграммы последователь ности с именем New Diagram;

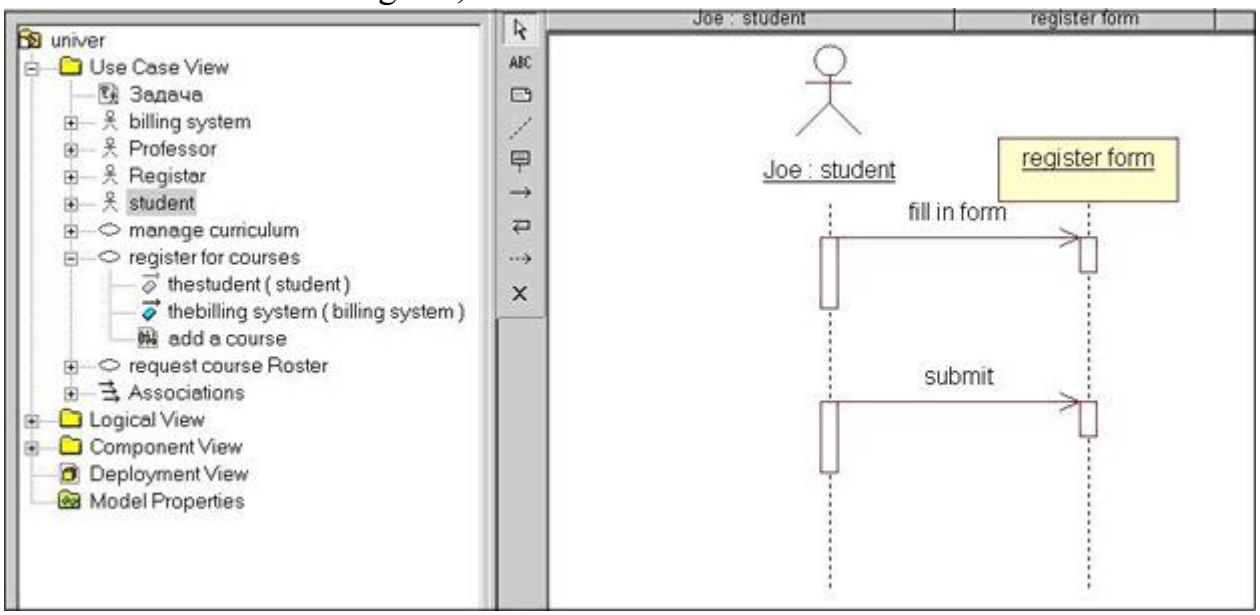


Рис. 6. Диаграмма последовательностей

- Введите имя диаграммы Add a Course, пока значок новой диаграммы остается выделенным (*рис. 6*);
- Откройте диаграмму двойным щелчком по ее значку в окне браузера;
- Переместите актера Student в окно диаграммы (он инициирует сценарий);
- Присвойте экземпляру актера конкретное имя: Joe;
- Создайте объект регистрационная форма (registration form):
  - щелкните сначала по значку объекта (прямоугольнику) на панели инструментов и затем в нужном месте диаграммы;
  - введите имя registration form, пока объект остается выделенным;
- Создайте объектные сообщения "*fill in info*" (форма должна заполняться студентом) и "*submit*":
  - щелкните по значку объектного сообщения (стрелке) на панели инструментов;
  - щелкните по пунктирной линии под актером Student и переместите стрелку на пунктирную линию под объектом registration form;
  - введите имя сообщения — fill in information, пока стрелка остается выделенной;
  - повторите предыдущие шаги пункта 9 для создания сообщения submit.

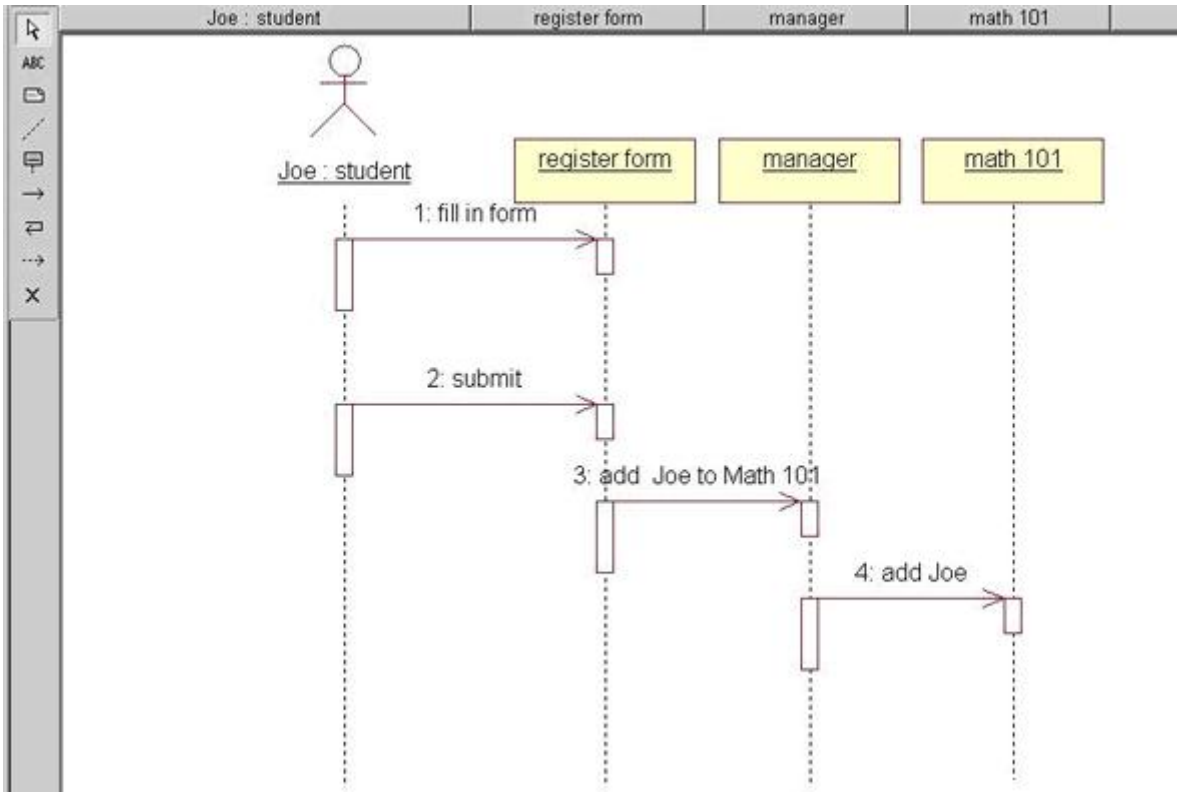


Рис. 7. Добавление на диаграмму объектов и сообщений

Объект Регистрационная форма является промежуточным звеном в цепи передачи информации;

- добавьте в поле диаграммы объект-менеджер (Manager);
- создайте сообщение менеджеру (manager) (*Рис. 7*). Менеджер узнает, что студент должен быть добавлен к какому-то курсу, (например, Joe хочет подписаться на курс math 101):
  - щелкните по значку объекта (прямоугольнику) на панели инструментов;
  - щелкните в нужном месте диаграммы для добавления объекта в диаграмму;
  - введите имя manager, пока объект остается выделенным;
  - щелкните по значку объектного сообщения (стрелке) на панели инструментов;
  - щелкните по пунктирной линии под объектом registration form и переместите стрелку на пунктирную линию под объектом manager;
  - введите имя сообщения — add Joe to Math 101 (*рис.7*), пока стрелка остается выделенной.

Менеджер — один из объектов системы. Он взаимодействует как с регистрационной формой, так и с набором объектов — учебных курсов. Для обслуживания студента Joe должен быть объект Math 101. Если такой объект существует, то Менеджер обязан передать объекту-курсу Math 101, что Joe должен быть добавлен к курсу;

- добавьте на диаграмму объекты и сообщения таким же образом, как в п. 8 - 11 в соответствии с *рис. 7*.

Объект-курс не принимает самостоятельных решений о возможности добавления студентов. Этим занимается экземпляр класса Предложение курса (course offering). Назовем такой экземпляр (объект) Section 1.

Объект-курс обращается к объекту Section 1, если он открыт (в этом сценарии — ответ "да") с предложением добавить студента Joe (*рис. 8*).

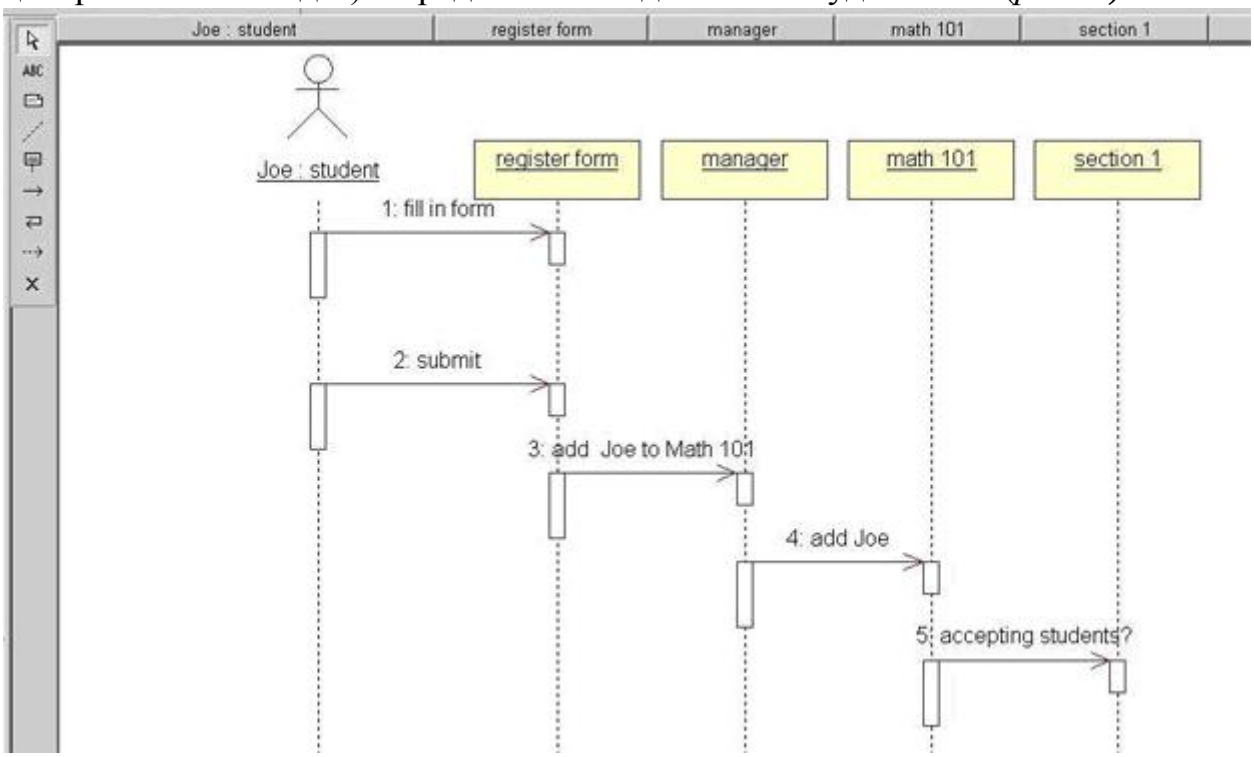


Рис. 8. Создание диаграммы последовательностей

- щелкните по значку объекта (прямоугольнику) на панели инструментов;
- щелкните в нужном месте диаграммы для добавления объекта;
- введите имя — section 1, пока объект остается выделенным;
- щелкните по значку объектного сообщения (стрелке) на панели инструментов;
- щелкните по пунктирной линии под объектом math 101 и переместите стрелку на пунктирную линию под объектом section 1;
- введите имя сообщения — accepting students, пока стрелка остается выделенной;
- повторите шаги 4-6 для создания сообщения add Joe.

После того, как Менеджер убеждается в том, что Студенту Joe предоставляется возможность изучать курс math 101, он уведомляет Учетную систему (billing system) о необходимости выписки счета.

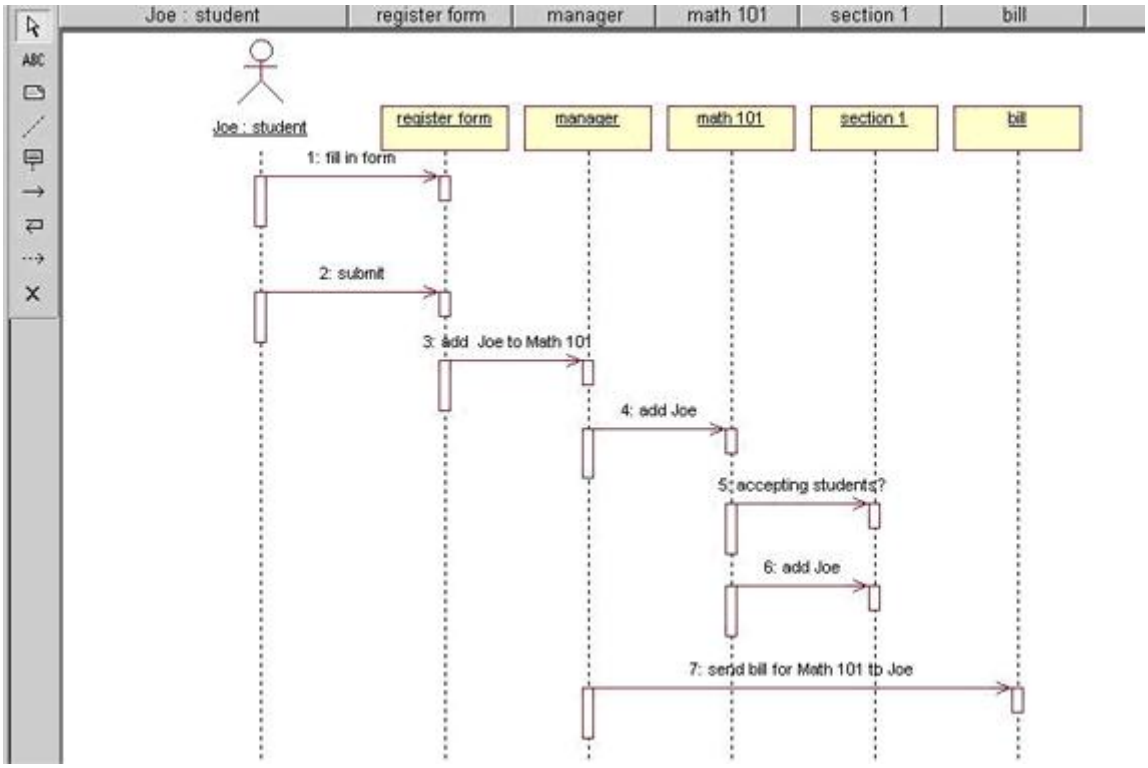


Рис. 9. Диаграмма последовательностей Add a Course

- щелкните по значку объекта (прямоугольнику) на панели инструментов;
- щелкните в нужном месте диаграммы для добавления объекта в диаграмму;
- введите имя — bill, пока объект остается выделенным;
- щелкните по значку объектного сообщения (стрелке) на панели инструментов;
- щелкните по пунктирной линии под объектом manager и переместите стрелку на пунктирную линию под объектом bill;
- пока стрелка остается выделенной, введите имя сообщения — send bill for Math 101 to Joe,

- сравните результат своей работы с *рис. 9*.

Диаграмма классов

Объекты из диаграмм последовательности группируются в классы. Из созданной в предыдущих заданиях диаграммы последовательности можно идентифицировать следую щие объекты и классы:
registration form - является объектом класса RegForm;
manager является объектом класса Manager;
math 101 является объектом класса Course;
section 1 является объектом класса CourseOffering;
bill является интерфейсом к внешней учетной системе, имя его класса BillingSystem.

Классы создаются в логическом представлении системы.

- В окне браузера щелкните правой кнопкой по значку пакета Logical View (рис. 10).

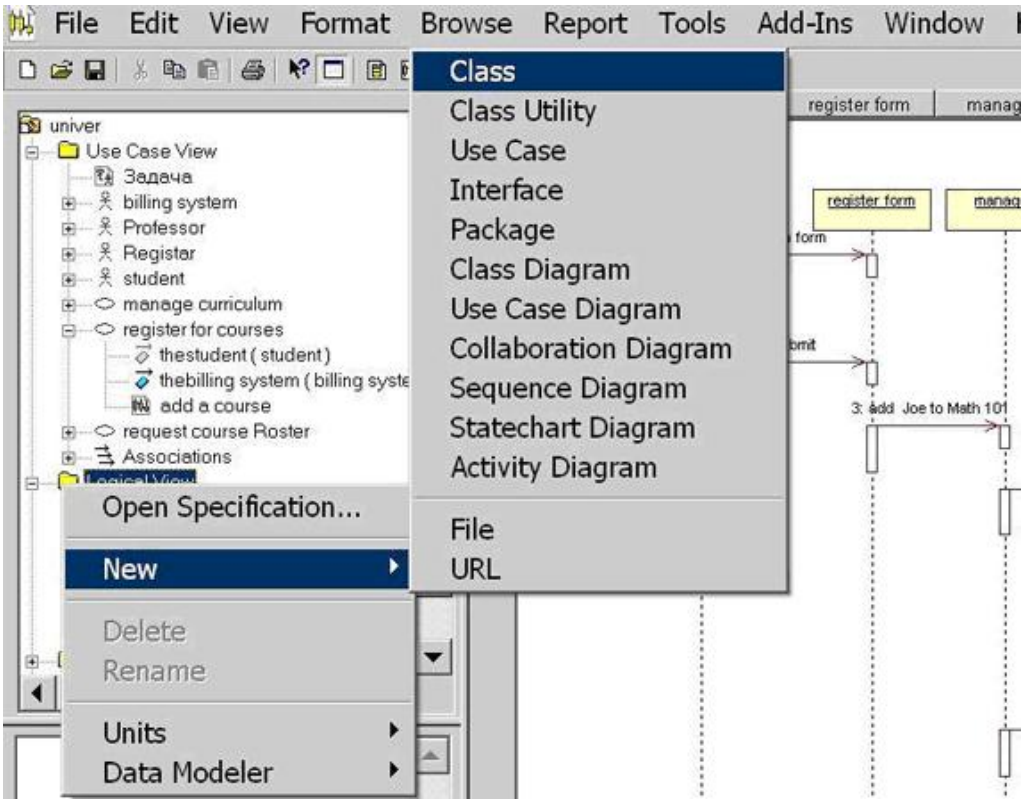


Рис. 10. Окно создания класса

- Выберите команду New Class в появившемся контекстном меню. В результате в дерево окна браузера будет добавлен класс с именем NewClass.
- Введите имя RegForm, пока значок класса остается выделенным.
- Повторите предыдущие шаги для добавления других классов: Manager, Course, CourseOffering и BillingSystem.

После создания классов они описываются (документируются). Описания добав ляются с помощью Documentation Window (*рис. 11*).

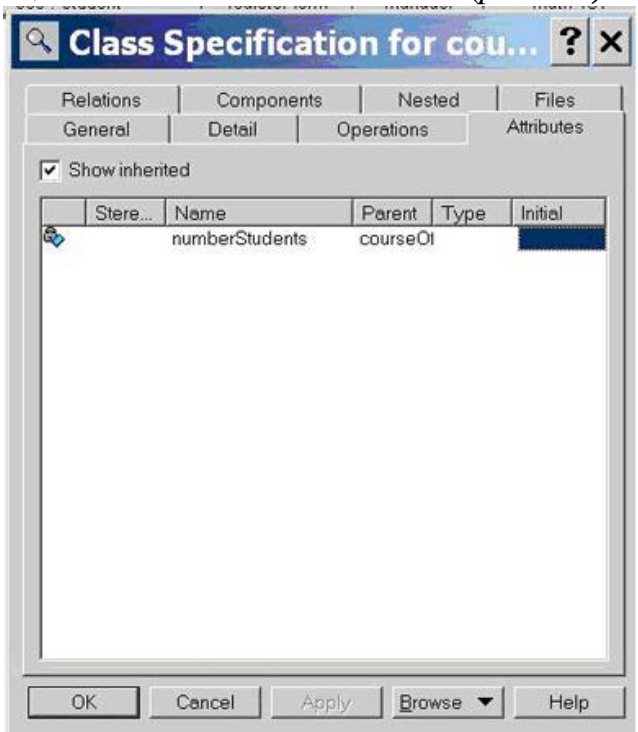


Рис. 11. Описание классов

- В окне браузера щелкните по значку класса CourseOffering, в ведите описание класса в Documentation Window.

Добавление классов:

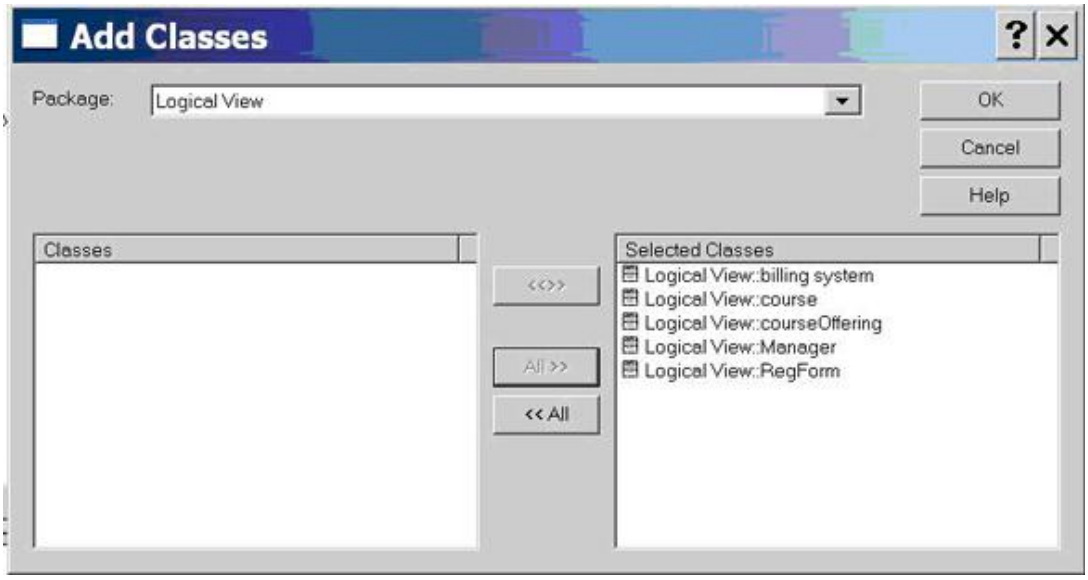


Рис. 12. Добавление классов.

- Откройте главную диаграмму классов двойным щелчком по значку Main в окне браузера.
- Добавьте на нее классы, выбрав в главном меню команду Query: Add Classes (*рис. 12*). Для этого нажмите кнопку “ All ” (выбрать все) (*рис. 12*).
- Закройте окно нажав, кнопку ОК.
- Переупорядочите классы в диаграмме, выделяя конкретный класс и перетаскивая его на новое место (*рис. 13*).

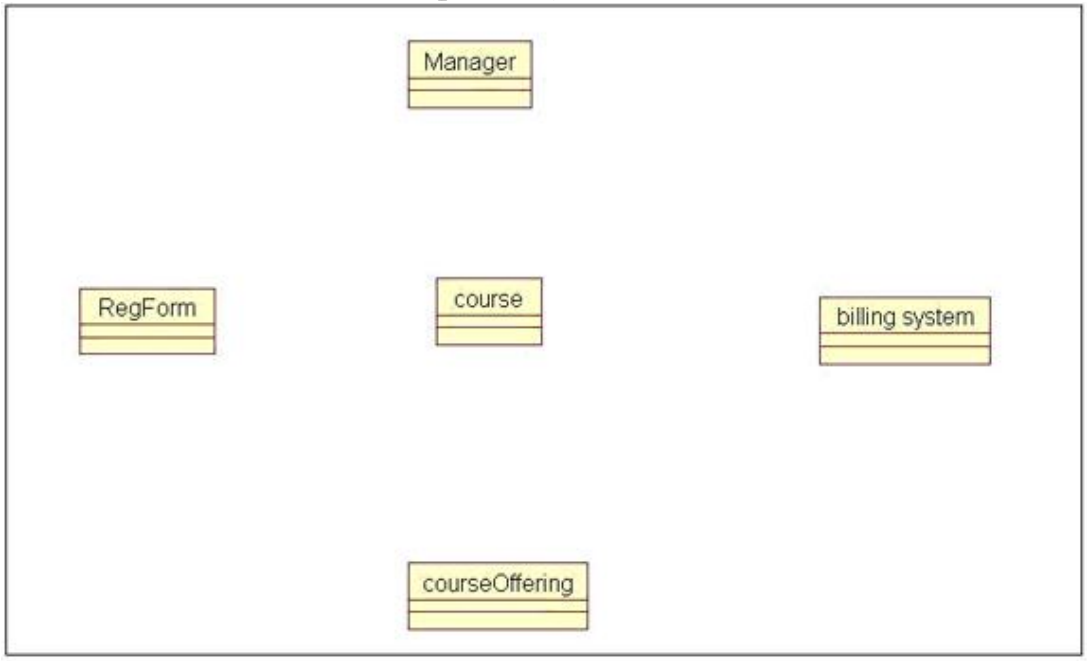


Рис. 13. Добавление классов в диаграмму

ПРИМЕЧАНИЕ.

Классы можно добавлять в диаграмму перетаскиванием их из окна браузера (по одному классу). Для создания новых типов моделирующих элементов

в UML используется поня тие стереотипа. С помощью стереотипа можно «нагрузить» элемент новым смыс лом.

- Использование предопределенного класса

Класс BillingSystem определяет только интерфейс к внешней учетной системе (billing system). Для и спользования предопределенного стереотипа Interface для этого класса (*рис. 14*):

- выполните двойной щелчок по значку класса BillingSystem в главной диаграмме классов. Появляется окно спецификации класса (Class Specification);
- щелкните по стрелке раскрывающегося списка Stereotype (Рис. 14);
- наберите на клавиатуре слово-стереотип Interface;
- закройте окно спецификации, нажав кнопку OK.

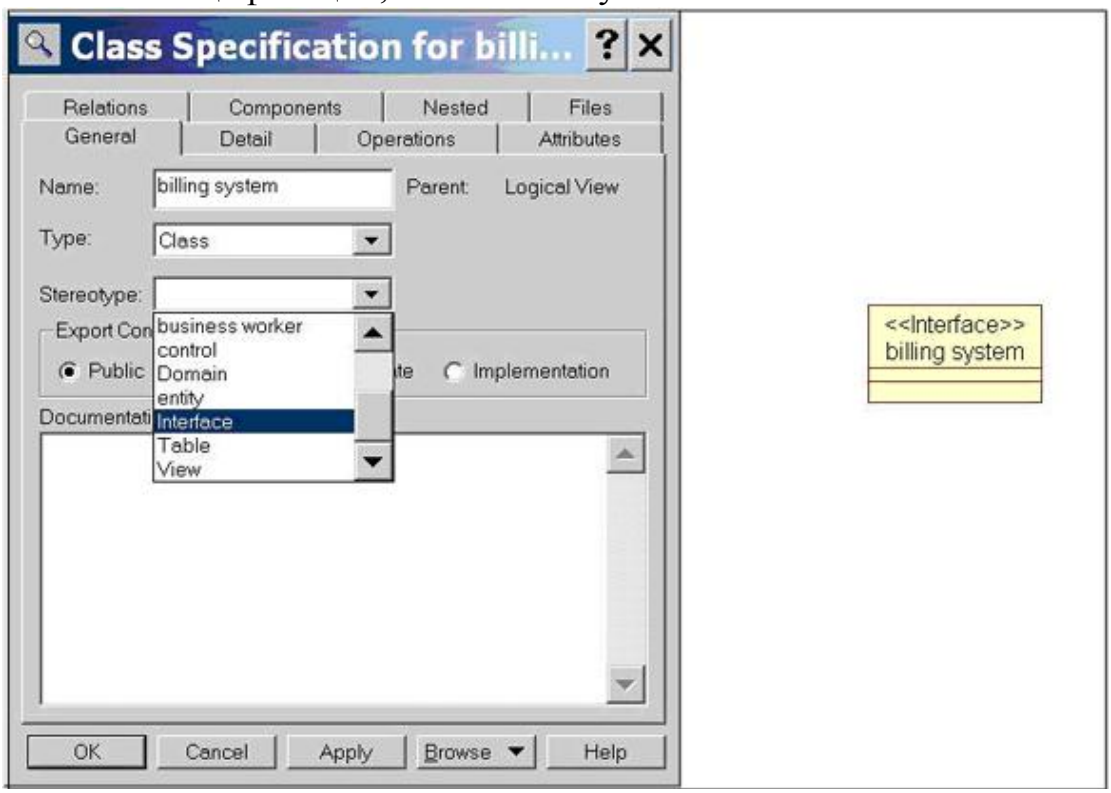


Рис. 14. Предопределенный класс

Если есть взаимодействия между объектами, определенные на диаграмме последовательностей, то должен быть определен путь для коммуникации между их классами.

Структурные отношения определяются двумя типами:

1. *ассоциациями*
2. *агрегациями*.

Ассоциация определяет соединение между классами.

- Создайте на диаграмме последова тельности Add a Course следующие ассоциа ции: от RegForm к Manager, от Manager Course и от Manager к BillingSystem (*рис. 16*). Для этого:

- щелкните по значку однонаправленной ассоциации (стрелке) на панели инструментов;
- щелкните по классу RegForm и переместите линию ассоциации на класс Manager;

- повторите предыдущие шаги для ввода следующих отношений:
- от Manager к Course;
- от Manager к BillingSystem.

Ассоциации задают пути между объектами-партнерами одинакового уровня.

Агрегация фиксирует неравноправные связи. Она показывает отношение между целым и его частями.

- Создайте отношение агрегации между классом Course и клас сом CourseOffering (*рис. 15*), т. к. предложение Курса CourseOfferings является частью агрегата — класса Course:
- щелкните по значку агрегации (линии с ромбиком) на панели инструментов;
- щелкните по классу, представляющему целое — Course;
- переместите линию агрегации на класс, представляющий часть — CourseOffering.

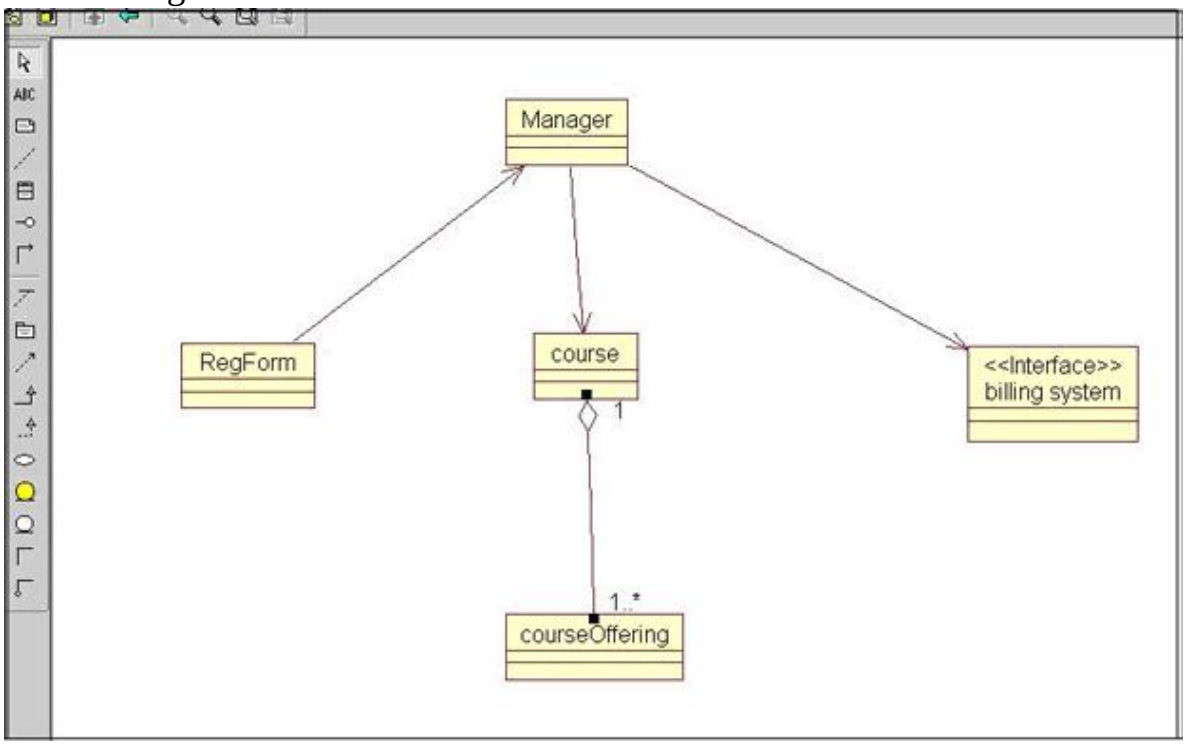


Рис. 15. Агрегация и ассоциация

- Добавьте на диаграмму индикаторы мощности (*рис. 15*) для отображения того, какое количество объектов участвует в отношении:
- щелкните правой кнопкой по линии агрегации возле класса CourseOffering;
- выберите из контекстного меню команду Multiplicity One or More;
- щелкните правой кнопкой по линии агрегации возле класса Course;
- из контекстного меню выберите команду Multiplicity.

Определение класса начинается с задания имени. Любой класс должен инкапсулировать в себе структуру данных и поведение, определяющее возможности обработки этой структуры. Второй шаг ориентируется на фиксацию структуры, а третий — на фиксацию поведения.

Структура класса представляется набором его свойств. Структура находится путем исследования проблемных требований и соглашений между разработчиками и заказчиками.

Каждое предложение курса (CourseOffering) в нашей модели явля ется свойством (attribute) класса-агрегата Course. Класс CourseOffering тоже имеет свойства.

- Определите одно из свойств класса CourseOffering — количество студентов:
  - щелкните правой кнопкой по классу CourseOffering в диаграмме классов;
  - выберите из контекстного меню команду Insert New Attribute. Это приведет к добавлению в класс свойства;
  - введите его имя — numberStudents, пока новое свойство остается выделенным.
- Задайте поведение класса (третий шаг).

Поведение класса представляется набором его операций. Исходная информация об операциях класса находится в диаграммах последовательности. В операции ото бражаются сообщения из диаграмм последовательности. Необходимо привязать объекты (из диаграмм после довательности) к конкретным классам.

Для этого:

- откройте диаграмму последовательности Add a Course двукратным щелчком по ее значку в окне браузера;
- щелкните в окне браузера по значку класса CourseOffering;
- переместите класс CourseOffering на объект section 1;
- отметьте, что имя объекта удлинилось, в нем появились две части, разделенные двоеточием. Слева от двоеточия записывается имя объекта, а справа — имя класса (*рис. 16*).

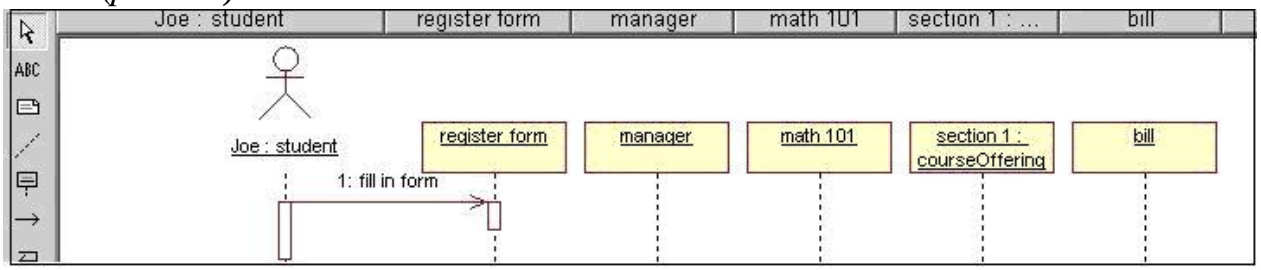


Рис. 16. Привязка классов

- Выполните второе действие — наполнение клас са операциями. Как правило, в операции класса превращаются сообщения, полу чаемые его объектом. При этом сообщения переименовываются — произ водится согласование имени сообщения и имени операции:
  - имя операции должно отражать ее принадлежность к классу (а не к источнику соответствующего сообщения);
  - имя операции должно указывать на ее обязанность, а не на способ ее реализации;
  - имя должно быть допустимым с точки зрения синтаксиса языка программирования, который будет использоваться для кодирования класса.

Создайте новую операцию класса и свяжите с ней сообщение — оно автомати чески поменяет имя;

- щелкните правой кнопкой по сообщению "add Joe". В результате станет видимым контекстное меню;
- выберите команду new operation. В результате станет видимой спецификация операции Operation Specification;

- введите имя новой операции — add;
- перейдите на вкладку Detail;
- щелкните правой кнопкой мышки по полю Arguments;
- выберите в контекстном меню команду Insert. В появившейся рамке наберите имя аргумента — Joe. Щелкните вне рамки;
- закройте окно спецификации, нажав кнопку ОК.

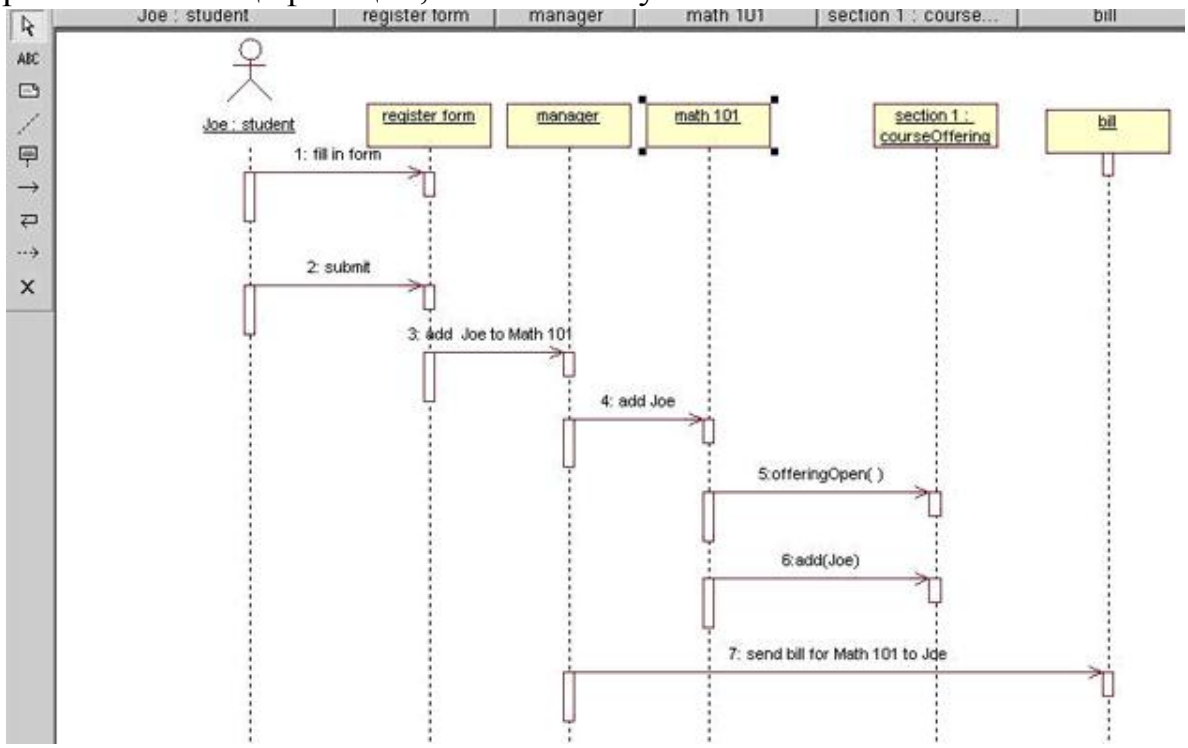


Рис. 17. Отображение операции на сообщение

- Получите имя сообщения обратным действием — отталкиваясь от имени операции (реализуется последовательность: отдельно создается новая операция класса, а затем она отображается на существующее сообщение) (рис. 17):
- щелкните правой кнопкой по классу в браузере;
- выберите в появившемся контекстном меню команду New: Operation. Появляется рамка с надписью opname;
- наберите имя новой операции класса — offeringOpen вместо надписи opname;
- щелкните правой кнопкой по сообщению "accepting students?" на диаграмме последовательности. В результате станет видимым контекстное меню;
- выберите в меню операцию offeringOpen( )— сообщение переименовывается (на нем отображается операция класса).
- Настройте описания классов на конкрет ный язык программирования:
- откройте диалоговое окно командой Tools: Options;
- перейдите на вкладку Notation (*рис. 18*);
- выберите название языка Ada 95 из раскрывающегося списка Default Language.

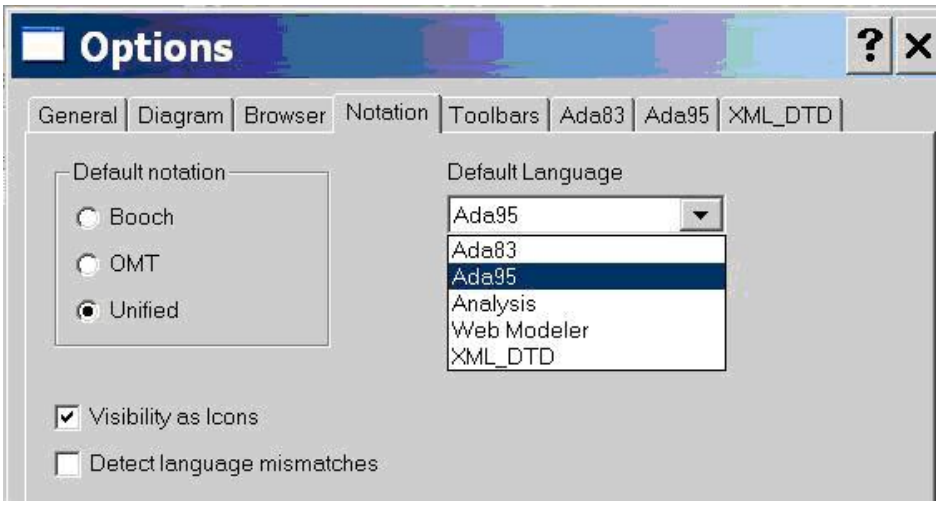


Рис. 18. Выбор языка программирования

- Определите типы данных, типы возвращаемых значений:
  - выполните двукратный щелчок по значку класса CourseOffering в окне браузера или диаграмме классов. В результате станет видимым окно спецификации класса;
  - выберите страницу Attributes (свойства) (*рис. 19*);
  - щелкните по полю Type. В результате станет видимым раскрывающийся список;
  - введите требуемый тип данных (Integer);
  - закройте окно спецификации, нажав кнопку ОК.

ПРИМЕЧАНИЕ:

Типы данных для свойств можно устанавливать, используя спецификацию Attribute или вводя их в строчке отображения свойства на диаграмме классов (формат attribute: data type).

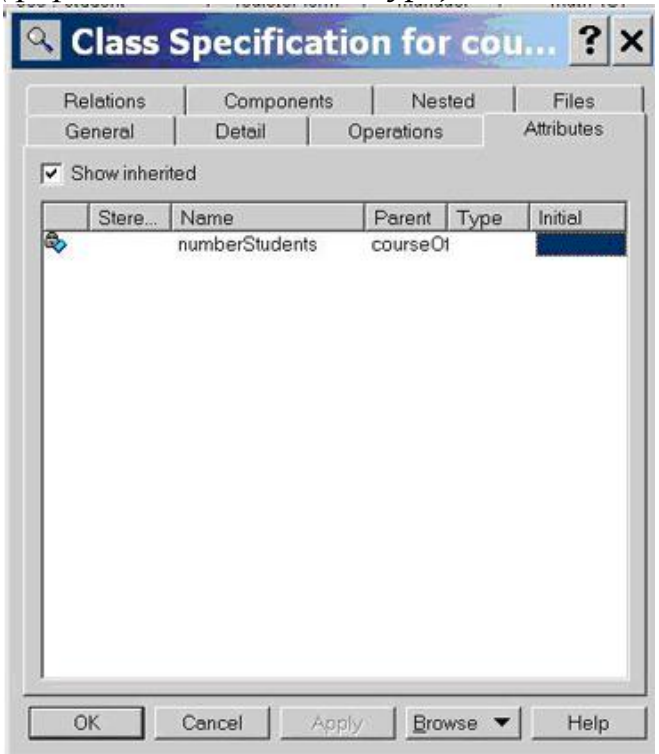


Рис. 19. Определение атрибутов

- Задайте тип возвращаемого результата для операции offeringOpen (*рис. 20*):
  - выполните двукратный щелчок по значку класса CourseOffering в окне браузера или диаграмме классов. В результате станет видимым окно спецификации класса;
  - выберите страницу Operations;

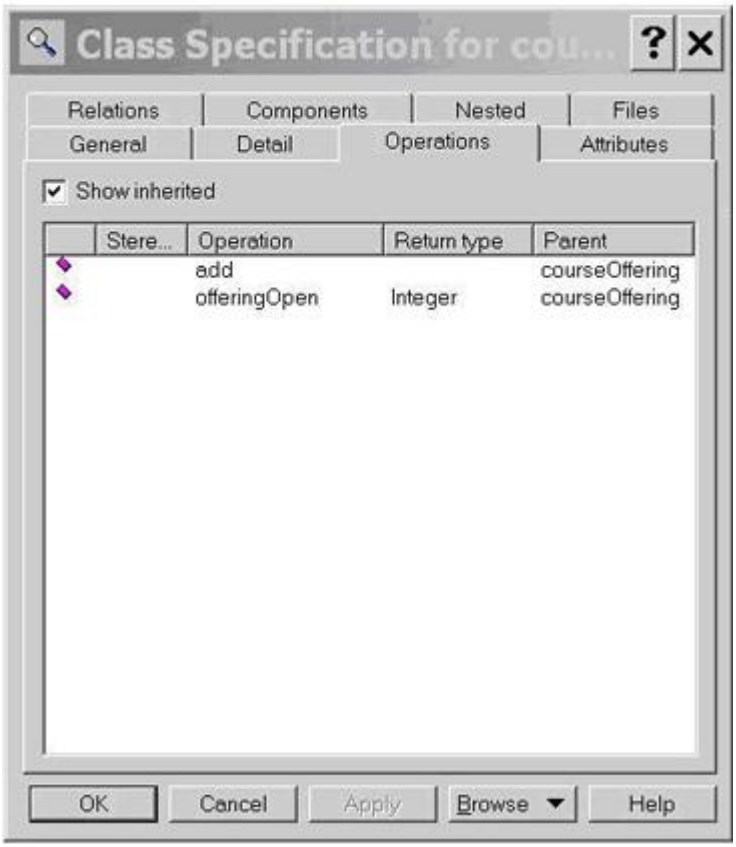


Рис. 20. Задание типа возвращаемого результата

  - щелкните по полю Return type. В результате станет видимым раскрывающийся список;
  - введите требуемый возвращаемый тип (Integer);
  - закройте окно спецификации, нажав кнопку OK.

Примечания:

Аргументы операции устанавливают с помощью диалогового окна Operation Spe cification. Для перехода к этому окну нужно на вкладке (странице) Operations щелкнуть правой кнопкой по имени операции и в появившемся контекстном меню выбрать ко манду Specification. Далее в появившемся диалоговом окне следует перейти на вкладку Detail. Аргументы операции и возвращаемый тип можно также установить, вводя их в строчке отображения операции на диаграмме классов (формат operation (argument name: data type): return type).

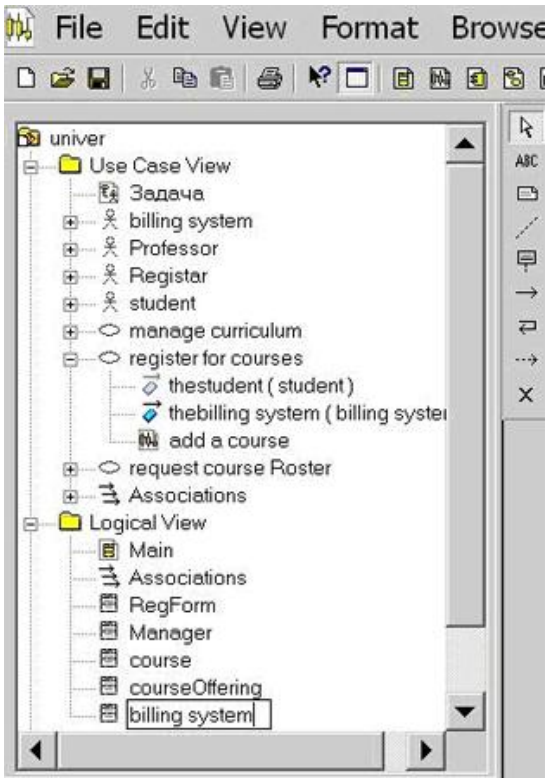


Рис. 21. Отображение классов в браузере

Для генерации кодов классов модели нужно создать компонентную диаграмму. Для определения компонентов исходного кода используют компонентное представление (Component View) (*рис. 22*). Среда Rational Rose автоматически создает главную компонентную диаграмму.

- Щелкните в окне браузера по значку + слева от пакета Component View.
- Откройте главную компонентную диаграмму, выполнив двойной щелчок по значку Main.

В общем случае каждому классу должны соответствовать два компонента — компо нент спецификации и компонент реализации. Для каждого компонента будет создан свой файл. Например, в языке C++ классу соответствуют два файла-компонента: h-файл (файл спецификации) и cpp-файл (файл реализации).

- Создайте один компонент для представления файла специфи кации по классу CourseOffering (расширение. ads и стереотип Package Specification) и один компонент для представления файла реали зации по классу CourseOffering (расширение.adb и стереотип Package Body) (*Рис. 22*).

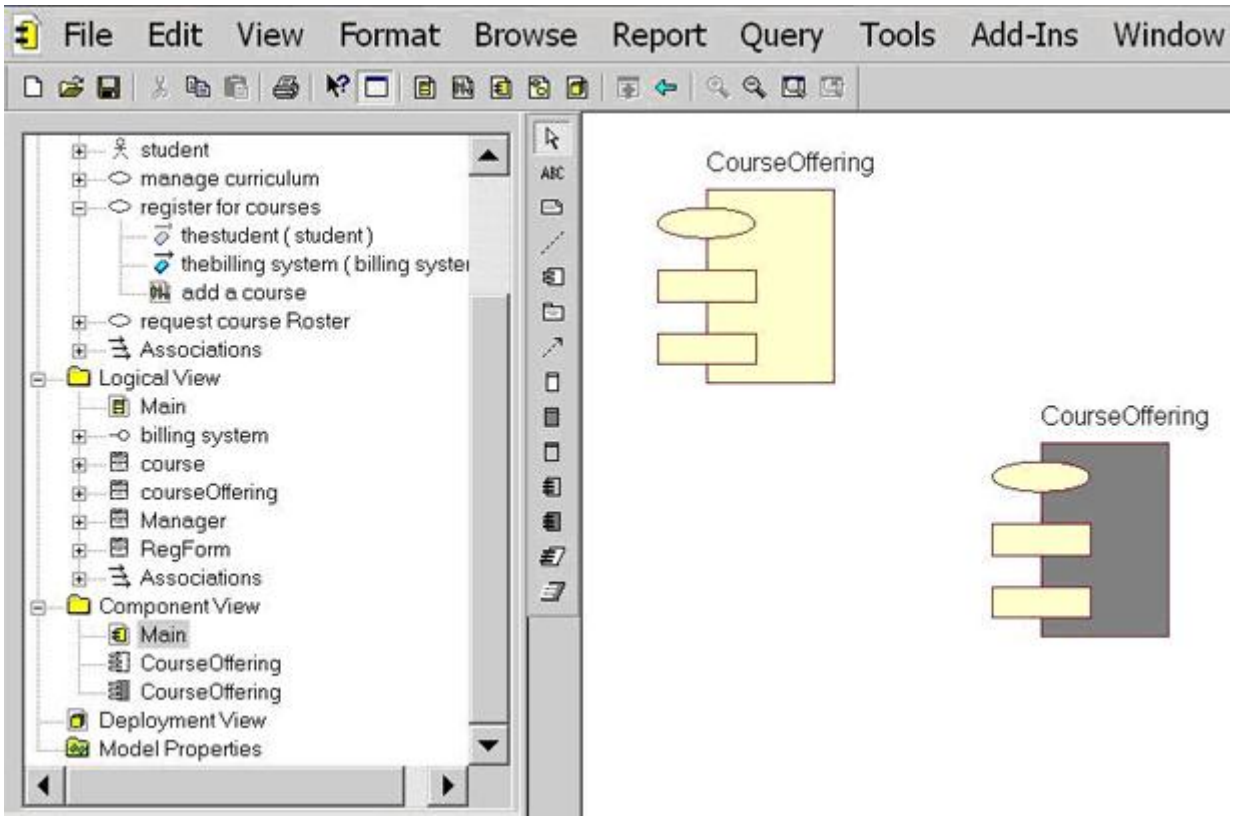


Рис. 22. Диаграмма компонентов

- щелкните на панели инструментов по значку спецификации пакета Package Specification;
- щелкните в нужном месте диаграммы для добавления компонента в диаграмму;
- введите его имя — CourseOffering, пока новый компонент остается выделенным;
- повторите предыдущие шаги с использованием значка тела пакета Package Body;
- щелкните на панели инструментов по значку отношения зависимости;
- щелкните по компоненту, представляющему. adb-файл (тело пакета), и перета щите стрелку на компонент, представляющий.ads-файл (спецификация пакета).

- Назначьте классы модели созданным компонентам (*рис. 23*).

- выполните двукратный щелчок по значку компонента CourseOffering, пред ставляющего.ads-файл (спецификацию пакета), в окне браузера или ком понентной диаграмме. В результате станет видимым окно спецификации компонента (*рис. 23*);
- выберите страницу (вкладку) Realizes. На ней представлен список классов модели;
- щелкните правой кнопкой по классу CourseOffering. В результате станет видимым контекстное меню;
- выберите команду Assign;

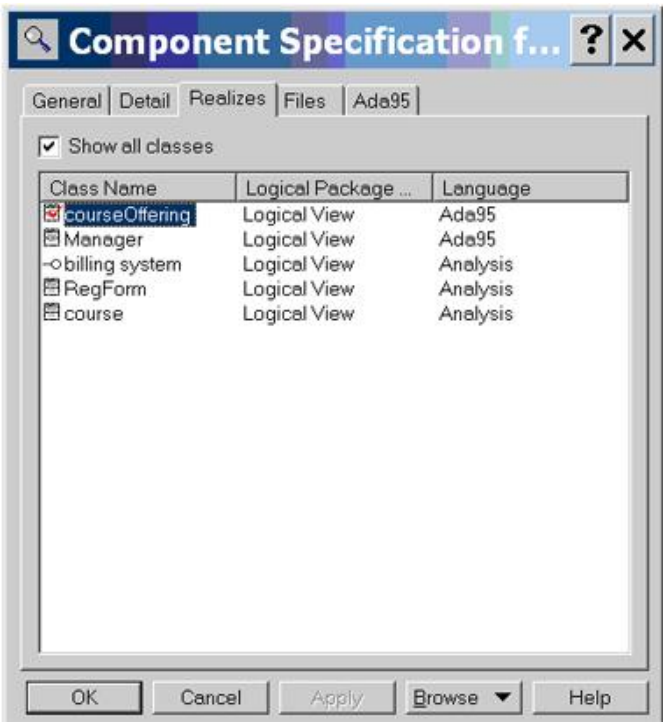


Рис. 23. Назначение классов компоненту

- закройте окно спецификации, нажав кнопку ОК;
- выполните аналогичные действия для тела пакета, представляющего.adb-файл.

Генерация кода:

- На компонентной диаграмме выделите оба компонента CourseOffering.
- Выберите команду Tools/Ada95/Code Generation из главного меню *(рис. 24*).
- Ознакомьтесь с появившимся окном итогов генерации кода Code Generation Status (*рис. 25*). Все ошибки заносятся в log -окно.
- Для завершения процесса генерации кода нажмите кнопку Close.

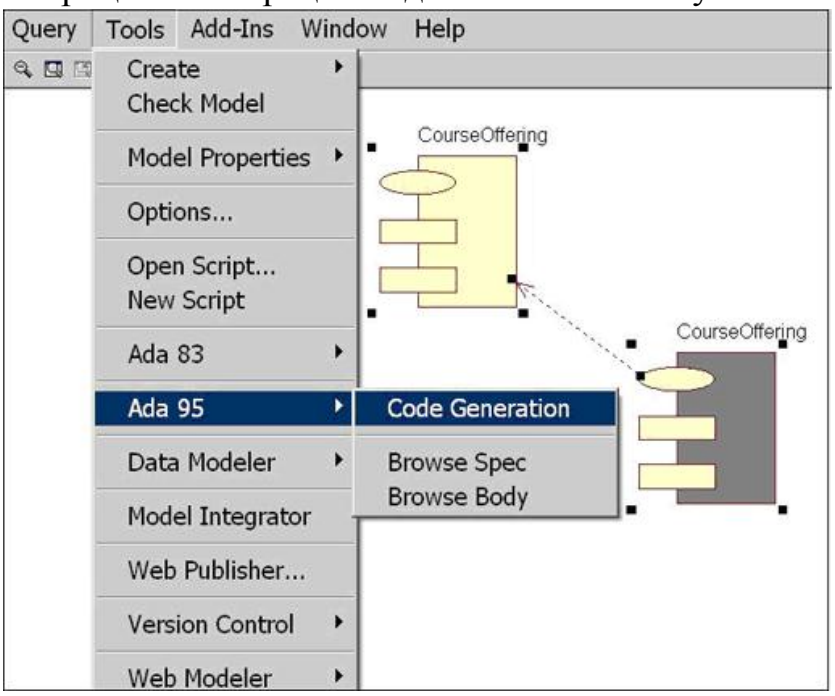


Рис. 24. Генерация кода

В процессе генерации Rational Rose отображает логическое описание класса в кар кас программного кода — в коде появляются языковые описания имени класса, свойств класса и заголовки методов. Для описания тела каждого метода формируется программная заготовка.

Рис. 25. Окно итогов генерации кода

Появляются и программные связи классов.

Программист будет дополнять этот код, работая в конкретной среде программирования, имеющей мост связи с системой Rational Rose. После каждого существенного дополнения программист с помощью возвратного проектирования, основанного на использовании моста связи, будет модифицировать диаграммы классов, вводя в них изменения, соответствующие результатам программирования.

- Просмотрите код, сгенерированный средой Rational Rose.

Приложение Г
(рекомендуемое)

Оформление задания для деловой (ролевой) игры

Деловая (ролевая) игра

по дисциплине _____
(наименование дисциплины)

1 Тема (проблема)
……
……
……
……

2 Концепция игры
……
……
……

3 Роли:
-
……
…;
-
……
…;

4 Ожидаемый (е) результат (ы)……..……
……
……
Критерии оценки:

- оценка «отлично» выставляется, если ..……;
- оценка «хорошо» ……
……;
- оценка «удовлетворительно»
…….……;
- оценка «неудовлетворительно»
…….
- оценка «зачтено» выставляется, если ……..……;
- оценка «не зачтено»

Составитель _____ И.О. Фамилия
(подпись)

«___»_____20 г.

Приложение Д
(рекомендуемое)

Оформление задания для кейс-задачи

Кейс-задача

по дисциплине _____
(наименование дисциплины)

Задание (я):

-
……;
-
……;
-
……;
-
……

Критерии оценки:

- оценка «зачтено» выставляется учащемуся, если ……..……;
- оценка «не зачтено»
……..

Составитель _____ И.О. Фамилия
(подпись)

«____»_____20 г.

Приложение Е
(рекомендуемое)

Оформление вопросов для собеседования

Вопросы собеседования

по дисциплине _____
(наименование дисциплины)

Раздел ……
1 ……
2 ……
… ……
n ……

Раздел ……
1 ……
2 ……
… ……
n ……

Критерии оценки:

- оценка «отлично» выставляется, если ……;
- оценка «хорошо» ……;
- оценка «удовлетворительно» ……;
- оценка «неудовлетворительно» ……
- оценка «зачтено» выставляется, если ……;
- оценка «не зачтено»

……

Составитель _____ И.О. Фамилия
(подпись)

«___»_____20 г.

Приложение Ж
(рекомендуемое)

Оформление комплекта заданий для контрольной работы

Комплект заданий для контрольной работы

по дисциплине _____
(наименование дисциплины)

Тема

……

Вариант 1

……

Задание 1

……..

Задание n

……

Вариант 2

……

Задание 1

……

Задание n

……

Тема

……

Вариант 1

……

Задание 1

……

Задание n

…….

Вариант 2

……..……
……

Задание 1

……..……..……
…….

Задание n

……..……..……
……

Критерии оценки:

- оценка «отлично» выставляется, если ..……..……;
- оценка «хорошо»…… ……..;
- оценка «удовлетворительно» ……..;
- оценка «неудовлетворительно» ……..…….;
- оценка «зачтено» выставляется, если ……..……..;
- оценка «не зачтено» ……

Составитель _____ И.О. Фамилия
(подпись)

«____»_____20 г.

Приложение 3
(рекомендуемое)

Оформление тем для круглого стола
(дискуссии, полемики, диспута, дебатов)

Перечень дискуссионных тем для круглого стола
(дискуссии, полемики, диспута, дебатов)

по дисциплине _____
(наименование дисциплины)

1
……..
2……..
…
……...
n
…….…….

Критерии оценки:

- оценка «отлично» выставляется, если ..……;
- оценка «хорошо» ……
……;
- оценка «удовлетворительно» ……..……;
- оценка «неудовлетворительно» …….
- оценка «зачтено» выставляется , если ……;
- оценка «не зачтено» ……

Составитель _____ И.О. Фамилия
(подпись)

«___»_____20 г.

Приложение И
(рекомендуемое)

Оформление задания для портфолио

Портфолио[1]

по дисциплине _____
(наименование дисциплины)

1 Название портфолио …….

2 Структура портфолио (инвариантные и вариативные части)**:**
2.1 …….
2.2 …….
… ……
n …….……...

Критерии оценки портфолио содержатся в методических рекомендациях по составлению портфолио.

Составитель _____ И.О. Фамилия
(подпись)

«____»_____20 г.

[1] Данное оценочное средство должно сопровождаться разработанными методическими рекомендациями по его составлению и использованию

Приложение К
(рекомендуемое)

Оформление групповых и/или индивидуальных творческих заданий/проектов

Темы групповых и/или индивидуальных творческих заданий/проектов**

по дисциплине _____

(наименование дисциплины)

Групповые творческие задания (проекты):

1
……
2
……
…
……..
n
……

Индивидуальные творческие задания (проекты):

1
……
2
……
…
…….
n
……

Критерии оценки:

- оценка «отлично» выставляется, если ……;
- оценка «хорошо» …… ……;
- оценка «удовлетворительно» ……;
- оценка «неудовлетворительно» …….

** Кроме курсовых проектов (работ)

- оценка «зачтено» выставляется, если ……..……;
- оценка «не зачтено»

……

Составитель _____ И.О. Фамилия
(подпись)

Приложение Л
(рекомендуемое)

Оформление комплекта разноуровневых задач (заданий)
Комплект разноуровневых задач (заданий)
по дисциплине_____
(наименование дисциплины)

1 Задачи репродуктивного уровня
Задача (задание) 1
…….…….…….
Задача (задание) 2
…….…….…….
Задача (задание) n
…….…….
2 Задачи реконструктивного уровня
Задача (задание) 1
…….……..…….
Задача (задание) 2
…….……..…….
Задача (задание) n
…….
3 Задачи творческого уровня
Задача (задание) 1
……...……...…….
Задача (задание) 2
……..…….
Задача (задание) n
…….…….

Критерии оценки:

- оценка «отлично» выставляется, если ..……;
- оценка «хорошо» …… ……;
- оценка «удовлетворительно» ……;
- оценка «неудовлетворительно» …….……
-оценка «зачтено» выставляется студенту, если ……..……;
- оценка «не зачтено»……..……

Составитель _____ И.О. Фамилия
(подпись)

«___»_____20 г.

Приложение М
(рекомендуемое)

Оформление комплекта заданий по видам работ
Комплект заданий для выполнения
расчетно-графической работы, работы на тренажере

по дисциплине _____
(наименование дисциплины)

Задача (задание) 1
……
Задача (задание) 2
……
Задача (задание) 3
……
Задача (задание) 4
……
Задача (задание) 5
……
Задача (задание) n
……

Критерии оценки:

- оценка «отлично» выставляется, если ..……..……;
- оценка «хорошо» …… ……..……;
- оценка «удовлетворительно» ……..……;
- оценка «неудовлетворительно» ……..…….
-оценка «зачтено» выставляется, если ……;
- оценка «не зачтено»……

Составитель _____ И.О. Фамилия
(подпись)

«___»_____20 г.

Приложение Н
(рекомендуемое)

Оформление тем для эссе
(рефератов, докладов, сообщений)

Темы эссе
(рефератов, докладов, сообщений)

по дисциплине _____
(наименование дисциплины)

1
……..
2
……..
3
……..
…
……
n
…….…….

Критерии оценки:

- оценка «отлично» выставляется, если ..……;
- оценка «хорошо» …… ……;
- оценка «удовлетворительно» ……;
- оценка «неудовлетворительно» …….
- оценка «зачтено» выставляется, если ……..……;
- оценка «не зачтено» ……..……..

Составитель _____ И.О. Фамилия
(подпись)

«____»_____20 г.