МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Индустриальный институт (филиал)
федерального государственного бюджетного образовательного учреждения
высшего образования «Югорский государственный университет»
(ИндИ (филиал) ФГБОУ ВО «ЮГУ»)

УТВЕРЖДАЮ
Заместитель директора по УВР
_____ Гарбар О.В.
«09» сентября 2021 г.

ПМ.02. ОСУЩЕСТВЛЕНИЕ ИНТЕГРАЦИИ ПРОГРАММНЫХ МОДУЛЕЙ

МДК 02.01 ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические указания
по выполнению практических работ

09.02.07 Информационные системы и программирование

г. Нефтеюганск
2021

РАССМОТРЕНО:
Предметной цикловой
Комиссией специальных технических
дисциплин
Протокол №1 от 09.09.2021
_____Шарипова И.А.

СОГЛАСОВАНО:
заседанием Методсовета
протокол №1 от 16.09.2021
Председатель Методсовета
_____Н.И. Савватеева

Методические указания по выполнению практических работ по МДК 02.01 Технология разработки программного обеспечения разработаны в соответствии с рабочей программой ПМ.02. Осуществление интеграции программных модулей по специальности 09.02.07 Информационные системы и программирование.

Разработчик: Игнатенко Е.С., преподаватель ИндИ (филиала) ФГБОУ ВО «ЮГУ».

СОДЕРЖАНИЕ

Пояснительная записка

Методические указания по выполнению лабораторных и практических работ по МДК 02.01 Технология разработки программного обеспечения разработаны в соответствии с рабочей программой профессионального модуля и предназначены для приобретения необходимых практических навыков и закрепления теоретических знаний, полученных обучающимися при изучении профессионального модуля, обобщения и систематизации знаний перед экзаменом.

Методические указания предназначены для обучающихся специальности 09.02.07 Информационные системы и программирование.

МДК 02.01 Технология разработки программного обеспечения относится к профессиональному циклу, изучается на 2 курсе и при его изучении отводится значительное место выполнению практических работ.

Освоение содержания МДК 02.01 Технология разработки программного обеспечения во время выполнения практических работ обеспечивает достижение обучающимися следующих **результатов:**

Код	Наименование общих компетенций
ОК 1.	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам
ОК 2.	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.
ОК 3	Планировать и реализовывать собственное профессиональное и личностное развитие.
ОК 4	Планировать и реализовывать собственное профессиональное и личностное развитие.
ОК 5	Планировать и реализовывать собственное профессиональное и личностное развитие.
ОК 6	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, применять стандарта антикоррупционного поведения.
ОК 7	Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.
ОК 8	Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности
ОК 9	Использовать информационные технологии в профессиональной деятельности.
ОК 10	Пользоваться профессиональной документацией на государственном и иностранном языках
ОК 11	Использовать знания по финансовой грамотности, планировать предпринимательскую деятельность в профессиональной сфере
ВД 2	Осуществление интеграции программных модулей
ПК 2.1.	Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент
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

В результате освоения профессионального модуля обучающийся должен:

Иметь практический опыт	Модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации

	и аттестации программного обеспечения
Уметь	Использовать выбранную систему контроля версий; использовать методы для получения кода с заданной функциональностью и степенью качества
Знать	Модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации и аттестации программного обеспечения

В соответствии с рабочей программой ПМ.02. Осуществление интеграции программных модулей практические работы по МДК 02.01 Технология разработки программного обеспечения разбиты на два семестра – третий и четвертый семестры. Целесообразность данной группировки обусловлена необходимостью обобщения и систематизации знаний перед экзаменом.

Рабочая программа профессионального модуля предусматривает проведение практических работ в объеме 90 часов.

Порядок выполнения практической работы

- записать название работы, ее цель в тетрадь;
- выполнить основные задания в соответствии с ходом работы;
- выполнить индивидуальные задания.

Рекомендации по оформлению практической работы

Задания выполняются обучающимися по шагам. Необходимо строго придерживаться порядка действий, описанного в практической работе

Результаты выполнения практических работ необходимо сохранять в своей папке на компьютере или USB – накопителе.

В случае пропуска занятий обучающийся осваивает материал самостоятельно в свободное от занятий время и сдает практическую работу с пояснениями о выполнении.

Критерии оценки практической работы

- наличие цели выполняемой работы, выполнение более половины основных заданий (удовлетворительно);
- наличие цели выполняемой работы, выполнение всех основных и более половины дополнительных заданий (хорошо);
- наличие цели выполняемой работы, выполнение всех основных и индивидуальных заданий (отлично).

Перечень практических работ

№	Наименование разделов и тем профессионального модуля (ПМ)	Наименование лабораторных работ и практических занятий	Объем часов
	Раздел 1. Разработка программного обеспечения		
	МДК 02.01 Технология разработки программного обеспечения		
1	Тема 2.1.1 Основные понятия и стандартизация требований к программному обеспечению	Практическая работа №1 «Анализ предметной области»	2
2		Практическая работа №2 «Разработка и оформление технического задания»	2
3		Практическая работа №3 «Построение архитектуры программного средства»	2
4		Практическая работа №4 «Изучение работы в системе контроля версий»	2
5	Тема 2.1.2. Описание и анализ требований. Диаграммы IDEF	Лабораторная работа №1 «Построение диаграммы Вариантов использования и диаграммы. Последовательности»	2
6		Лабораторная работа №2 «Построение диаграммы Кооперации и диаграммы Развертывания»	2
7		Лабораторная работа №3 «Построение диаграммы Деятельности, диаграммы Состояний и диаграммы Классов»	2
8		Лабораторная работа №4 «Построение диаграммы компонентов»	2
9		Лабораторная работа №5 «Построение диаграмм потоков данных»	2
10	Тема 2.1.3. Оценка качества программных средств	Лабораторная работа №6 «Разработка тестового сценария»	2
11		Лабораторная работа №7«Оценка необходимого количества тестов»	2
12		Лабораторные работы №8 «Разработка тестовых пакетов»	2
13		Лабораторные работы №9 «Оценка программных средств с помощью метрик»	2
14		Лабораторные работы №10 «Инспекция программного кода на предмет соответствия стандартам кодирования»	2
	Итого		28

Практическая работа №1 «Анализ предметной области»

Цель работы: описать и проанализировать ИС, определить необходимые элементы КТС ИС и системного ПО ИС.

Теоретические сведения

Проблемы управления программными проектами впервые появились в 60-х– начале70-хгодов прошлого века, когда провалились многие большие проекты по разработке программных продуктов. Были зафиксированы задержки в создании ПО, программное обеспечение было ненадежным, затраты на разработку в несколько раз превосходили первоначальные оценки и т.д. Провалы этих проектов обуславливались не только некомпетентностью руководителей и программистов. Напротив, в этих больших поисковых проектах принимали участие люди, уровень квалификации которых был явно выше среднего. Причины провалов коренились в тех подходах, которые использовались в управлении проектами. Применяемая методика была основана на опыте управления техническими проектами и оказалась неэффективной при разработке программных проектов.

Руководители программных проектов выполняют такую же работу, что и руководители технических проектов. Вместе с тем процесс разработки ПО существенно отличается от процессов реализации технических проектов. Ниже приведен небольшой список этих отличий.

1. Программный продукт нематериален. Менеджер судостроительного проекта или проекта постройки здания видит результат выполнения своего проекта. Если реализация проекта отстает от графика,то это видно по незавершенности конструкции. В противоположность этому процент незавершенности программного проекта нельзя увидеть или потрогать. Менеджер программного проекта может полагаться только на документацию, которая фиксирует процесс разработки программного продукта.

2. Не существует стандартных процессов разработки программного обеспечения. На сегодняшний день не существует четкой зависимости между процессом создания ПО и типом создаваемого программного продукта. Другие технические дисциплины имеют длительную историю, процессы разработки технических изделий многократно опробованы и проверены. Изучением же процессов создания ПО специалисты занимаются только последние несколько лет. Поэтому прока нельзя точно предсказать, на каком этапе процесса разработки ПО могут возникнуть проблемы, угрожающие всему проекту.

3. Большие программные проекты – это часто одноразовые проекты. Большие программные проекты, как правило, значительно отличаются от проектов, реализованных ранее. Поэтому, чтобы уменьшить неопределенность в планировании проекта, руководители проектов должны обладать очень большим практическим опытом. Но постоянные технологические изменения в компьютерной технике обесценивают предыдущий опыт. Перечисленные особенности могут привести к тому, что реализация проекта выйдет за рамки временного графика или бюджетных ассигнований. Об этом всегда нужно помнить.

Процессы управления программными проектами

Невозможно описать и стандартизировать все работы, выполняемые менеджером проекта по созданию ПО, но в большинстве случаев к ним относятся.

- Написание предложений по созданию ПО.
- Планирование и составление графика работ проекта.
- Оценивание стоимости проекта.
- Контроль процессов выполнения работ.
- Подбор персонала.
- Написание отчетов и представлений.

Время выполнения больших программных проектов может занимать несколько лет. В течение этого времени цели и намерения организации, оказавшей программный проект, могут существенно измениться. Может оказаться, что разрабатываемый программный продукт стал уже ненужным либо исходные требования к ПО устарели и их нужно кардинально менять. В такой ситуации руководство организации-разработчика может принять решение о прекращении разработки ПО или об изменении проекта в целом.

Планирование проекта

Эффективное управление проектами напрямую зависит от правильного планирования работ. План, разработанный на начальном этапе проекта, рассматривается всеми его участниками как руководящий документ, выполнение которого должно привести к успешному завершению

проекта. Этот первоначальный план должен максимально подробно описывать все этапы реализации проекта.

План проекта должен показать ресурсы, необходимые для реализации проекта, разделение работ на этапы и временной график выполнения этих этапов. Детализация планов проектов очень разнится в зависимости от типа разрабатываемого программного продукта и организации-

Разработчика. Но в любом случае большинство планов содержит следующие разделы.

1. Введение. Краткое описание целей проекта и проектных ограничений (бюджетных, временных и т.д.).

2. Организация выполнения проекта. Описание способа подбора команды разработчиков и распределение обязанностей между членами команды.

3. Анализ рисков. Описание возможных проектных рисков, вероятность их проявления и стратегий, направленных на их уменьшение.

4. Аппаратные и программные ресурсы для реализации проекта. Перечень аппаратных средств и программного обеспечения, необходимого для разработки программного продукта.

5. Разбиение работ на этапы. Проект разбивается на отдельные процессы, определяются этапы выполнения проекта, приводится описание результатов каждого этапа и контрольные отметки.

6. График работ. В графике работ отображаются зависимости между отдельными этапами разработки по, оценки времени их выполнения и распределение членов команды проекта по отдельным этапам.

7. Механизмы контроля и мониторинга за ходом выполнения проекта. Описываются механизмы и сроки предоставления отчетов о ходе работ, а также механизмы мониторинга всего проекта.

При планировании проекта разработки ПО определяются контрольные точки – ***вехи***, отмечающие окончание определенного этапа работ. Для каждой вехи создается отчет, который предоставляется руководству проекта.

При определении контрольных точек весь процесс создания ПО должен быть разбит на отдельные этапы с указанным «выходом» (результатом) каждого этапа. Например, на рис. 1 показаны этапы разработки спецификации требований в случае, когда для ее проверки используется прототип системы.

Рисунок 1. Этапы процесса разработки спецификации

Информационный процесс - это осуществление всей совокупности следующих элементарных информационных актов: прием или создание информации, ее хранение, передача и использование. Информационная система - это совокупность механизмов, обеспечивающих полное осуществление информационного процесса.

Вне ИС информация может лишь сохраняться в виде записей на тех или иных физических носителях, но не может быть ни принятой, ни переданной, ни использованной.

Информационная система - организационно-техническая система, которая предназначена для выполнения информационно-вычислительных работ или предоставления информационно-вычислительных работ или предоставления информационно-вычислительных услуг, удовлетворяющих потребности системы управления и ее пользователей - управленческого персонала, внешних пользователей путем использования и/или создания информационных продуктов. Информационные системы существуют в рамках системы управления и полностью подчинены целям функционирования этих систем.

Информационно-вычислительная работа - деятельность, связанная с использованием информационных продуктов. Типичным примером информационной работы является поддержка информационных технологий управления.

Информационно-вычислительная услуга - это разовая информационно-вычислительная работа.Под информационным продуктом понимается вещественный или нематериальный результат интеллектуального человеческого труда, обычно материализованный на определенном носителе, например разнообразных программных продуктов, выходной информации в виде документов управления, баз данных, хранилищ данных, баз знаний, проектов ИС и ИТ.

Методологическую основу изучения ИС составляет системный подход, в соответствии с которым любая система представляет собой совокупность взаимосвязанных объектов, функционирующих совместно для достижения общей цели.

Информационная система представляет собой совокупность функциональной структуры, информационного, математического, технического, организационного и кадрового обеспечения, которые объединены в единую системы в целях сбора, хранения, обработки и выдачи необходимой информации для выполнения функций управления. Она обеспечивает информационные потоки:

I-1 - информационный поток из внешней среды в систему управления, который, с одной стороны, представляет собой поток нормативной информации, создаваемый государственными учреждениями в части законодательства, а с другой стороны - поток информации о конъюнктуре рынка, создаваемый конкурентами, потребителями, поставщиками;

I-2 - информационный поток из системы управления во внешнюю среду (отчетная информация, прежде всего финансовая в государственные органы, инвесторам, кредиторам, потребителям; маркетинговая информация потенциальным потребителям);

I-3 - информационный поток из системы управления на объект, представляет собой совокупность плановой, нормативной и распорядительной информации для осуществления хозяйственных процессов;

I-4 - информационный поток от объекта в систему управления, который отражает учетную информацию о состоянии объекта управления экономической системой (сырья, материалов, денежных, энергетических, трудовых ресурсов, готовой продукции и выполненных услугах) в результате выполнения хозяйственных процессов.

Задачи информационных систем

Корпоративные системы позволяют решить следующие задачи:

- гарантировать требуемое качество управления предприятием;
- повысить оперативность и эффективность взаимодействия между подразделениями;
- обеспечить управляемость качеством выпускаемой продукции;
- увеличить экономическую эффективность деятельности предприятия;
- создать систему статистического учета на предприятии;
- осуществлять прогноз развития предприятия;
- создать систему стратегического и оперативного планирования, систему прогнозирования.

Задания для практической работы

1. Выберите предметную область
2. Выберите название ИС в рамках предметной области.
3. Определите цель ИС
4. Проведите анализ осуществимости ИС

4.1. Что произойдет с организацией, если система не будет введена в эксплуатацию?

4.2. Какие текущие проблемы существуют в организации и как новая система поможет их решить?

4.3. Каким образом (и будет ли) ИС способствовать целям бизнеса?

4.4. Требует ли разработка ИС технологии, которая до этого раньше не использовалась в организации?

5. Где будет размещена ИС? Кто является пользователем ИС?
6. Комплекс технических средств ИТ

6.1. Какие средства компьютерной техники необходимы для ИС?

6.2. Какие средства коммуникационной техники необходимы для ИС?

6.3. Какие средства организационной техники необходимы для ИС?

6.4. Какие средства оперативной полиграфии необходимы для ИС?

7. Опишите системное ПО ИТ.

Таблица 1. Варианты предметных областей

Предметная область	Сущность задачи
Страховая медицинская компания	Страховая медицинская компания (СМК) заключает договоры добровольного медицинского страхования с населением и договоры с лечебными учреждениями на лечение застрахованных клиентов. При возникновении страхового случая клиент подает заявку на оказание медицинских услуг по условиям договора инспектору, который работает с данным клиентом. Инспектор направляет данного клиента в лечебное учреждение. Отчеты о своей деятельности инспектор предоставляет в бухгалтерию. Бухгалтерия проверяет оплату договоров, перечисляет денежные средства за оказанные услуги лечебным учреждениям, производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. СМК не только оплачивает лечение застрахованного лица при возникновении с ним страхового случая, но и, при возникновении каких-либо осложнений после лечения, оплачивает лечение этих осложнений
Агентство недвижимости	Агентство недвижимости занимается покупкой, продажей, сдачей в аренду объектов недвижимости по договорам с их собственниками. Агентство управляет объектами недвижимости как физических, так и юридических лиц. Собственник может иметь несколько объектов. В случае покупки или аренды клиент может произвести осмотр объекта. В качестве одной из услуг, предлагаемых агентством, является проведение инспектирования текущего состояния объекта для адекватного определения его рыночной цены. По результатам своей деятельности агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики
Кадровое агентство	Кадровое агентство способствует трудоустройству безработных граждан. Агентство ведет учет и классификацию данных о безработных на основании резюме от них. От предприятий города поступают данные о свободных вакансиях, на основании которых агентство предлагает различные варианты трудоустройства соискателям. В случае положительного исхода поиска вакансия считается заполненной, а безработный становится трудоустроенным. По результатам своей деятельности кадровое агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики
Компания по разработке программных продуктов	Компания заключает договор с клиентом на разработку программного продукта согласно техническому заданию. После утверждения технического задания определяется состав и объем работ, составляется предварительная смета. На каждый проект назначается ответственный за его выполнение – куратор проекта, который распределяет нагрузку между программистами и следит за выполнением технического задания. Когда программный продукт готов, то его внедряют, производят обучение клиента и осуществляют дальнейшее сопровождение. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики

Туроператор	Туроператор предоставляет возможность своим клиентам осуществить туристическую или деловую поездку в различные города России и мира. При разработке нового тура сначала анализируется текущая ситуация на рынке туризма и выбирается направление тура. После этого определяется статус тура, бронируются места в гостиницах и билеты на переезд к месту тура, разрабатывается культурная/ деловая/ развлекательная программа, утверждаются сроки тура. На каждый тур назначается ответственное лицо от туроператора, которое будет вести данный тур для улаживания проблем в случае возникновения каких-нибудь чрезвычайных или форс-мажорных ситуаций. Клиент приходит в офис туроператора, где вместе с менеджером выбирает уже разработанный тур и оформляет путевку. После возвращения из тура клиент может высказать свои замечания или пожелания, которые будут учтены при доработке существующих туров или при разработке новых. Также, для дальнейшего улучшения тура, туроператор проводит анализ отчетов от посредников (гостиница, гиды и т.д.). По результатам своей деятельности туроператор производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики

Практическая работа №2 «Разработка и оформление технического задания»

Цель работы: приобретение навыков разработки технического задания на программный продукт, ознакомиться с правилами написания технического задания

Теоретические сведения

Техни́ческое зада́ние (ТЗ, техзада́ние) - исходный документ для проектирования сооружения или промышленного комплекса, конструирования технического устройства (прибора, машины, системы управления и т. д.), разработки информационных систем, стандартов либо проведения научно-исследовательских работ (НИР).

ТЗ содержит основные технические требования, предъявляемые к сооружению, изделию или услуге и исходные данные для разработки. В ТЗ указываются назначение объекта, область его применения, стадии разработки конструкторской (проектной, технологической, программной и т.п.) документации, её состав, сроки исполнения и т. д., а также особые требования, обусловленные спецификой самого объекта либо условиями его эксплуатации. Как правило, ТЗ составляют на основе анализа результатов предварительных исследований, расчётов и моделирования. Типовые требования к составу и содержанию технического задания приведены в таблице 1.

Таблица 1. Состав и содержание технического задания (ГОСТ 34.602- 89)

№ пп	Раздел	Содержание
1	Общие сведения	- полное наименование системы и ее условное обозначение - шифр темы или шифр (номер) договора; - наименование предприятий разработчика и заказчика системы, их реквизиты - перечень документов, на основании которых создается ИС - плановые сроки начала и окончания работ - сведения об источниках и порядке финансирования работ - порядок оформления и предъявления заказчику результатов работ по созданию системы, ее частей и отдельных средств
2	Назначение и цели создания (развития) системы	- вид автоматизируемой деятельности - перечень объектов, на которых предполагается использование системы - наименования и требуемые значения технических, технологических, производственно-экономических и др. показателей объекта, которые должны быть достигнуты при внедрении ИС
3	Характеристика объектов автоматизации	- краткие сведения об объекте автоматизации - сведения об условиях эксплуатации и характеристиках окружающей среды
4	Требования к системе	Требования к системе в целом: - требования к структуре и функционированию системы (перечень подсистем, уровни иерархии, степень централизации, способы информационного обмена, режимы функционирования, взаимодействие со смежными системами, перспективы развития системы) - требования к персоналу (численность пользователей, квалификация, режим работы, порядок подготовки) - показатели назначения (степень приспособляемости системы к изменениям процессов управления и значений параметров) - требования к надежности, безопасности, эргономике, транспортабельности, эксплуатации, техническому обслуживанию и ремонту, защите и сохранности

		информации, защите от внешних воздействий, к патентной чистоте, по стандартизации и унификации Требования к функциям (по подсистемам) : - перечень подлежащих автоматизации задач - временной регламент реализации каждой функции - требования к качеству реализации каждой функции, к форме представления выходной информации, характеристики точности, достоверности выдачи результатов - перечень и критерии отказов Требования к видам обеспечения: - математическому (состав и область применения мат. моделей и методов, типовых и разрабатываемых алгоритмов) - информационному (состав, структура и организация данных, обмен данными между компонентами системы, информационная совместимость со смежными системами, используемые классификаторы, СУБД, контроль данных и ведение информационных массивов, процедуры придания юридической силы выходным документам) - лингвистическому (языки программирования, языки взаимодействия пользователей с системой, системы кодирования, языки ввода- вывода) - программному (независимость программных средств от платформы, качество программных средств и способы его контроля, использование фондов алгоритмов и программ) - техническому - метрологическому - организационному (структура и функции эксплуатирующих подразделений, защита от ошибочных действий персонала) - методическому (состав нормативно- технической документации
5	Состав и содержание работ по созданию системы	- перечень стадий и этапов работ - сроки исполнения - состав организаций — исполнителей работ - вид и порядок экспертизы технической документации - программа обеспечения надежности - программа метрологического обеспечения
6	Порядок контроля и приемки системы	- виды, состав, объем и методы испытаний системы - общие требования к приемке работ по стадиям - статус приемной комиссии
7	Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие	- преобразование входной информации к машиночитаемому виду - изменения в объекте автоматизации - сроки и порядок комплектования и обучения персонала
8	Требования к документированию	- перечень подлежащих разработке документов - перечень документов на машинных носителях
9	Источники разработки	- документы и информационные материалы, на основании которых разрабатывается ТЗ и система

Порядок разработки технического задания

Разработка технического задания выполняется в следующей последовательности. Прежде всего, устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных.

Затем определяют перечень результатов, их характеристики и способы представления.

Далее уточняют среду функционирования программною обеспечения: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту.

В случаях, когда разрабатываемое программное обеспеченно собирает и хранит некоторую информацию или включается в управление каким-либо техническим процессом, необходимо также четко регламентировать действия программы в случае сбоев оборудования и энергоснабжения.

1. Общие положения

1.1 Техническое задание оформляют в соответствии с ГОСТ 19.106-78 на листах формата А4 и А3 по ГОСТ 2.301-68, как правило, без заполнения полей листа. Номера листов (страниц) проставляют в верхней части листа над текстом.

1.2 Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104-78. Информационную часть (аннотацию и содержание), лист регистрации изменений допускается и в документ не включать.

1.3 Для внесения изменений и дополнений в техническое задание на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

1.4. Техническое задание должно содержать следующие разделы:

- введение;
- наименование и область применения;
- основание для разработки;
- назначение разработки;
- технические требования к программе или программному изделию;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;
- приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них. При необходимости допускается в техническое задание включать приложения.

2. Содержание разделов

2.1 Введение должно включать краткую характеристику области применения программы или программного продукта, а также объекта (например, системы), в котором предполагается их использовать. Основное назначение введения - продемонстрировать актуальность данной разработки и показать, какое место эта разработка занимает в ряду подобных.

2.2 В разделе «Наименование и область применения» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

2.3 В разделе «Основание для разработки» должны быть указаны:

- документ (документы), на основании которых ведется разработка. Таким документом может служить план, приказ, договор и т. п.;
- организация, утвердившая этот документ, и дата его утверждения;
- наименование и (или) условное обозначение темы разработки.

2.4 В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

2.5 Раздел «Технические требования к программе или программному изделию» должен содержать следующие подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;

- требования к информационной и программной совместимости;
- требования к маркировке и упаковке;
- требования к транспортированию и хранению;
- специальные требования.

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

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

2.5.3 В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации (температура окружающего воздуха, относительная влажность и т. п. для выбранных типов носителей данных), при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

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

2.5.5 В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования. При необходимости должна обеспечиваться защита информации и программ.

2.5.6 В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

2.5.7 В подразделе «Требования к транспортированию и хранению» должны быть указаны для программного изделия условия транспортирования, места хранения, условия хранения, условия складирования, сроки хранения в различных условиях.

2.5.8 В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

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

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

2.8 В приложениях к техническому заданию при необходимости приводят:
- перечень научно-исследовательских и других работ, обосновывающих разработку;
- схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке;
- другие источники разработки.

В случаях, если какие-либо требования, предусмотренные техническим заданием, заказчик не предъявляет, следует в соответствующем месте указать «Требования не предъявляются».

Задания для практической работы

1. Разработать техническое задание по варианту выбранному в практической работе №1
2. Оформить отчет

Порядок выполнения отчета по практической работе

1.Разработать техническое задание на программный продукт

2. Оформить работу в соответствии с ГОСТ 19.106-78. При оформлении использовать MS Office.

3. Сдать и защитить работу

Защита отчета по практической работе заключается в предъявлении преподавателю полученных результатов (на экране монитора и печатном виде), демонстрации полученных навыков и ответах на вопросы преподавателя

Практическая работа №3 «Построение архитектуры программного средства»

Цель работы: приобретение навыков создания формальных моделей и на их основе определение спецификаций разрабатываемого программного обеспечения, приобретение навыков проектирования программного обеспечения

Теоретические сведения

Эскизный проект

Эскизный проект предусматривает разработку предварительных проектных решений по системе и ее частям.

Выполнение стадии эскизного проектирования не является строго обязательной. Если основные проектные решения определены ранее или достаточно очевидны для конкретной ИС и объекта автоматизации, то эта стадия может быть исключена из общей последовательности работ.

Содержание эскизного проекта задается в ТЗ на систему. Как правило, на этапе эскизного проектирования определяются:

- функции ИС;
- функции подсистем, их цели и ожидаемый эффект от внедрения;
- состав комплексов задач и отдельных задач;
- концепция информационной базы и ее укрупненная структура;
- функции системы управления базой данных;
- состав вычислительной системы и других технических средств;
- функции и параметры основных программных средств.

По результатам проделанной работы оформляется, согласовывается и утверждается документация в объеме, необходимом для описания полной совокупности принятых проектных решений и достаточном для дальнейшего выполнения работ по созданию системы.

На основе технического задания (и эскизного проекта) разрабатывается технический проект.

Разработка эскизного проекта программы. Этапы выполнения эскизного проекта.

.Эскизный проект	Разработка эскизного проекта	Предварительная разработка структуры входных и выходных данных.
		Уточнение методов решения задачи.
		Разработка общего описания алгоритма решения задачи
		Разработка технико-экономического обоснования.
	Утверждение эскизного проекта	Разработка пояснительной записки.
		Согласование и утверждение эскизного проекта.

Основная задача эскизного проекта – создать прообраз будущей автоматизированной системы. При разработке эскизного проекта разработчик определяет основные контуры будущей системы, а заказчик в свою очередь получает представление об основных чертах будущего объекта автоматизации и анализирует их возможную применимость в последующей работе.

При разработке эскизного проекта составляются:

- Ведомость эскизного проекта. Общая информация по проекту.
- Пояснительная записка к эскизному проекту. Вводная информация, позволяющая ее потребителю быстро освоить данные по конкретному проекту.
- Схема организационной структуры. Описание организационной структуры организации, которая будет использовать создаваемую автоматизированную систему в практической работе.
- Структурная схема комплекса технических средств. Техническая составляющая автоматизированной системы, включающая в себя набор серверов, рабочих станций, схему локальной вычислительной сети и структурированной кабельной системы.
- Схема функциональной структуры. Описание задач, которые будут использоваться в работе подсистем. Видение участков информационной системы и порядок и их взаимодействия.
- Схема автоматизации. Логический процесс создания автоматизированной системы от начала до конца.

- Согласно ГОСТ 34.201-89, дополнительно в эскизный проект по необходимости может быть включено техническое задание на разработку новых технических средств.

Эскизный проект чаще всего не разделяют, он выполняется в рамках общего (первоначального) этапа всего проекта. Перечень работ, составляющих эскизный проект, может варьироваться в зависимости от конкретного технического задания заказчика (его пожеланий) и сложности проектируемого проекта. Соответственно варьируется и цена этого этапа.

Эскизный проект не всегда создается под конкретного заказчика. Нередко разработчики с помощью эскизного проекта стремятся показать свой творческий потенциал и найти потенциальных заказчиков. Не случайно на различные конкурсы представляются именно эскизные проекты.

Разработка спецификаций

Разработка программного обеспечения начинается с анализа требований к нему. В результате анализа получают спецификации разрабатываемого программного обеспечения, строят общую модель его взаимодействия с пользователем или другими программами и конкретизируют его основные функции.

При структурном подходе к программированию на этапе анализа и определения спецификаций разрабатывают три типа моделей: модели функций, модели данных и модели потоков данных. Поскольку разные модели описывают проектируемое программное обеспечение с разных сторон, рекомендуется использовать сразу несколько моделей, разрабатываемых в виде диаграмм, и пояснять их текстовыми описаниями, словарями и т. п.

Структурный анализ предполагает использование следующих видов моделей:

- диаграмм потоков данных (DFD - Data Flow Diagrams), описывающих взаимодействие источников и потребителей информации через процессы, которые должны быть реализованы в системе;
- диаграмм «сущность-связь» (ERD Entity-Relationship Diagrams), описывающих базы данных разрабатываемой системы;
- диаграмм переходов состояний (STD - State Transition Diagrams), характеризующих поведение системы во времени;
- функциональных диаграмм (методика SADT);
- спецификаций процессов;
- словаря терминов.

Спецификации процессов

Спецификации процессов обычно представляют в виде краткого текстового описания, схем алгоритмов, псевдокодов, Flow-форм или диаграмм Насси - Шнейдермана.

Словарь терминов

Словарь терминов представляет собой краткое описание основных понятий, используемых при составлении спецификации. Он должен включать определение основных понятий предметной области, описание структур элементов данных, их типом и форматов, а также всех сокращений и условных обозначении.

Диаграммы переходов состояний

С помощью диаграмм переходов состояний можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования. Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены.

Функциональные диаграммы

Функциональные диаграммы отражают взаимосвязи функций разрабатываемого программного обеспечения.

Они создаются на ранних этапах проектирования систем, для того чтобы помочь проектировщику выявить основные функции и составные части проектируемой системы и, по возможности, обнаружить и устранить существенные ошибки. Для создания функциональных диаграмм предлагается использовать методологию SADT.

Диаграммы потоков данных

Для описания потоков информации в системе применяются диаграммы потоков данных (DFD – Data Flow Diagrams). DFD позволяет описать требуемое поведение системы в виде

совокупности процессов, взаимодействующих посредством связывающих их потоков данных. DFD показывает, как каждый из процессов преобразует свои входные потоки данных в выходные потоки данных и как процессы взаимодействуют между собой.

Диаграммы «сущность - связь»

Диаграмма сущность-связь - инструмент разработки моделей данных, обеспечивающий стандартный способ определения данных и отношений между ними. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют требованиям, предъявляемым к ИС.

Разработка документации. Стадия «Технический проект».

Проект технический - образ намеченного к созданию объекта, представленный в виде его описания, схем, чертежей, расчетов, обоснований, числовых показателей.

Цель технического проекта - определение основных методов, используемых при создании информационной системы, и окончательное определение ее сметной стоимости.

Техническое проектирование подсистем осуществляется в соответствии с утвержденным техническим заданием.

Технический проект программной системы подробно описывает:

- выполняемые функции и варианты их использования;
- соответствующие им документы;
- структуры обрабатываемых баз данных;
- взаимосвязи данных;
- алгоритмы их обработки.

Технический проект должен включать данные об объемах и интенсивности потоков обрабатываемой информации, количестве пользователей программной системы, характеристиках оборудования и программного обеспечения, взаимодействующего с проектируемым программным продуктом.

При разработке технического проекта оформляются:

- ведомость технического проекта. Общая информация по проекту;
- пояснительная записка к техническому проекту. Вводная информация, позволяющая ее потребителю быстро освоить данные по конкретному проекту;
- описание систем классификации и кодирования;
- перечень входных данных (документов). Перечень информации, которая используется как входящий поток и служит источником накопления;
- перечень выходных данных (документов). Перечень информации, которая используется для анализа накопленных данных;
- описание используемого программного обеспечения. Перечень программного обеспечения и СУБД, которые планируется использовать для создания информационной системы;
- описание используемых технических средств. Перечень аппаратных средств, на которых планируется работа проектируемого программного продукта;
- проектная оценка надежности системы. Экспертная оценки надежности с выявлением наиболее благополучных участков программной системы и ее узких мест;
- ведомость оборудования и материалов. Перечень оборудования и материалов, которые потребуются в ходе реализации проекта.

Структурная схема

Структурной называют схему, отражающую состав и взаимодействие по управлению частями разрабатываемого программного обеспечения. Структурная схема определяется архитектурой разрабатываемого ПО.

Функциональная схема

Функциональная схема - это схема взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств.

Разработка алгоритмов

Метод пошаговой детализации реализует нисходящий подход к программированию и предполагает пошаговую разработку алгоритма.

Структурные карты

Методика структурных карт используется на этапе проектирования ПО для того, чтобы продемонстрировать, каким образом программный продукт выполняет системные требования. Структурные карты Константайна предназначены для описания отношений между модулями.

Техника структурных карт Джексона основана на методе структурного программирования Джексона, который выявляет соответствие между структурой потоков данных и структурой программы. Основное внимание в методе сконцентрировано на соответствии входных и выходных потоков данных.

Задания для практической работы

Задание №1

1. На основе технического задания из практической работы №2 выполнить анализ функциональных и эксплуатационных требований к программному продукту.
2. Определить основные технические решения (выбор языка программирования, структура программного продукта, состав функций ПП, режимы функционирования) и занести результаты в документ, называемый «Эскизным проектом».
3. Определить диаграммы потоков данных для решаемой задачи.
4. Определить диаграммы «сущность-связь», если программный продукт содержит базу данных.
5. Добавить словарь терминов.
6. Оформить результаты, используя MS Office или MS Visio в виде эскизного проекта.
7. Сдать и защитить работу.

Порядок выполнения отчета по практической работе

Отчет по лабораторной работе должен состоять из:

1. Постановки задачи.
2. Документа «Эскизный проект», содержащего:

- выбор метода решения и языка программирования;
- спецификации процессов;
- все полученные диаграммы;
- словарь терминов.

Защита отчета по практической работе заключается в предъявлении преподавателю полученных результатов (на экране монитора), демонстрации полученных навыков и ответах на вопросы преподавателя.

Задание №2

1. Разработать функциональную схему программного продукта.
2. Представить структурную схему в виде структурных карт Константайна.
3. Представить структурную схему в виде структурных карт Джексона.
4. Оформить результаты, используя MS Office или MS Visio в виде технического проекта.
5. Сдать и защитить работу.

Порядок выполнения отчета по практической работе

Отчет по практической работе должен состоять из:

1. Структурной схемы программного продукта.
2. Функциональной схемы.
3. Алгоритма программы.
4. Структурной карты Константайна.
5. Структурной карты Джексона.
6. Законченного технического проекта программного модуля.

Защита отчета по практической работе заключается в предъявлении преподавателю полученных результатов (на экране монитора и в печатном виде), демонстрации полученных навыков и ответах на вопросы преподавателя.

Практическая работа №4 «Изучение работы в системе контроля версий»

Цель работы: применение методов объектно-ориентированного проектирования

Теоретические сведения

Сущность объектно-ориентированного подхода к программированию заключается в том, что основные идеи объектно-ориентированного подхода опираются на следующие положения:

- программа представляет собой модель некоторого реального процесса, части реального мира;
- модель реального мира или его части может быть описана как совокупность взаимодействующих между собой объектов;
- объект описывается набором параметров, значения которых определяют состояние объекта, и набором операций (действий), которые может выполнять объект;
- взаимодействие между объектами осуществляется посылкой специальных сообщений от одного объекта к другому. Сообщение, полученное объектом, может потребовать выполнения определенных действий, например, изменения состояния объекта;
- объекты, описанные одним и тем же набором параметров и способные выполнять один и тот же набор действий представляют собой класс однотипных объектов.

С точки зрения языка программирования класс объектов можно рассматривать как тип данного, а отдельный объект - как данное этого типа. Определение программистом собственных классов объектов для конкретного набора задач должно позволить описывать отдельные задачи в терминах самого класса задач (при соответствующем выборе имен типов и имен объектов, их параметров и выполняемых действий).

Таким образом, объектно-ориентированный подход предполагает, что при разработке программы должны быть определены классы используемых в программе объектов и построены их описания, затем созданы экземпляры необходимых объектов и определено взаимодействие между ними.

Классы объектов часто удобно строить так, чтобы они образовывали иерархическую структуру. Например, класс «Студент», описывающий абстрактного студента, может служить основой для построения классов «Студент 1 курса», «Студент 2 курса» и т.д., которые обладают всеми свойствами студента вообще и некоторыми дополнительными свойствами, характеризующими студента конкретного курса. При разработке интерфейса с пользователем программы могут использовать объекты общего класса «Окно» и объекты классов специальных окон, например, окон информационных сообщений, окон ввода данных и т.п. В таких иерархических структурах один класс может рассматриваться как базовый для других, производных от него классов. Объект производного класса обладает всеми свойствами базового класса и некоторыми собственными свойствами, он может реагировать на те же типы сообщений от других объектов, что и объект базового класса и на сообщения, имеющие смысл только для производного класса. Обычно говорят, что объект производного класса наследует все свойства своего базового класса.

Некоторые параметры объекта могут быть локализованы внутри объекта и недоступны для прямого воздействия извне объекта. Например, во время движения объекта-автомобиля объект-водитель может воздействовать только на ограниченный набор органов управления (рулевое колесо, педали газа, сцепления и тормоза, рычаг переключения передач) и ему недоступен целый ряд параметров, характеризующих состояние двигателя и автомобиля в целом.

Очевидно, для того, чтобы продуктивно применять объектный подход для разработки программ, необходимы языки программирования, поддерживающие этот подход, т.е. позволяющие строить описание классов объектов, образовывать данные объектных типов, выполнять операции над объектами. Одним из первых таких языков стал язык SmallTalk в котором все данные являются объектами некоторых классов, а общая система классов строится как иерархическая структура на основе предопределенных базовых классов.

Опыт программирования показывает, что любой методический подход в технологии программирования не должен применяться слепо с игнорированием других подходов. Это относится и к объектно-ориентированному подходу. Существует ряд типовых проблем, для которых его полезность наиболее очевидна, к таким проблемам относятся, в частности, задачи имитационного моделирования, программирование диалогов с пользователем. Существуют и задачи, в которых применение объектного подхода ни к чему, кроме излишних затрат труда, не приведет. В связи с этим наибольшее распространение получили объектно-ориентированные языки программирования, позволяющие сочетать объектный подход с другими методологиями. В

некоторых языках и системах программирования применение объектного подхода ограничивается средствами интерфейса с пользователем (например, Visual FoxPro ранних версий).

Наиболее используемыми в настоящее время объектно-ориентированными языками являются Паскаль с объектами и Си++, причем наиболее развитые средства для работы с объектами содержатся в Си++.

Объектно-базирующееся программирование - это методология разработки программ, основанная на использовании совокупности объектов, каждый из которых является реализацией определенного класса. Программный код и данные структурируются так, чтобы имитировалось поведение фактически существующих объектов. Содержимое объекта защищено от внешнего мира посредством инкапсуляции. Благодаря наследованию уже запрограммированные функциональные возможности можно использовать и для других объектов. Объекты являются программным представлением физических и/или логических сущностей реального мира. Они необходимы для моделирования поведения физических или логических объектов, которые они представляют. Для изменения поведения и состояния элементов управления используются их свойства, методы, поля и события. Классы задают структуру объектов. При программировании создаются объекты - представители классов. С другой стороны, классы составляют группы одноименных объектов. Внутренняя структура класса в Visual Basic передается объекту с использованием модуля класса. С использованием команды Project ⟶Add Class Module модуль класса можно добавить в проект. После добавления модуля класса выводится окно кода, в котором можно реализовать компоненты (свойства, поля, методы, события) класса.

Задания для практической работы

Пример использования методики объектно–ориентированного программирования

Создать в предметной области «Автомобили» класс с требуемой функциональностью (использовать компоненты класса: методы, поля и т.д.). Создать объект - экземпляр класса. Создать пример использования объектом компонентов класса.

Реализация задания

Приводится проект, дающий справку желающим приобрести автомобиль. Создан класс Class1, содержащий компоненты, определяющие название фирмы-изготовителя, модель автомобиля, его стоимость, изображение автомобиля и следующие технические характеристики:

- тип двигателя (бензин/дизель),
- число цилиндров/рабочий объём,
- система питания (карбюратор/впрыскивание),
- мощность (л.с),
- максимальная скорость (км/час),
- разгон 0 - 100 (км/час)/сек,
- привод (передний/задний/4х4).

Далее создаётся экземпляр класса: Dim av As New Class1, использующий компоненты класса.

Пользователю предлагается решить вопрос о необходимости покупки, выбрать фирму-изготовителя, ответить на вопрос о выводе изображения покупаемого автомобиля, либо его технических характеристик, либо обеих категорий одновременно (используются процедуры Property Get и Property Let, созданные в классе Class1), после чего программа адекватно реагирует: либо выводятся вышеперечисленные данные, либо выводится некоторое сообщение.

Для реализации проекта нужно выполнить следующую последовательность действий:

1. добавить в стандартный проект модуль класса (Project ⟶Add Class Module ⟶Class Module ⟶Открыть),
2. создать:

- методы класса. Четыре метода создаются в процедурах: Public Function Met1(), Public Function Met2(), Public Function Met3(), Public Function Met4() (Tools ⟶Add Procedure ⟶ввести имя { Met1, Met2, Met3, Met4}⟶ выбрать Function ⟶выбрать Public ⟶OK),
- свойства класса. Свойства задаются с использованием процедур Property Get и Property Let (Tools ⟶Add Procedure ?ввести имя (здесь - varian) ⟶выбрать Property ⟶выбрать Public ⟶OK),
- поля класса - avto, firma, model, stoim, pict, var, см. ниже.

3. создать на форме:

- два элемента управления ComboBox с именами Combo1 и Combo2,
- два элемента управления CommandButton с именами Command1 и Command2; значению свойства Caption объекта Command1 присвоить значение "OK", Command2 - "Exit",
- элементы управления Label1 - Label4, значениям свойств Caption присвоить: Label1 - "Хотите ли Вы купить машину?", Label2 - "Выберите фирму-изготовитель", Label3 - "Хотите ли Вы увидеть изображение выбранного автомобиля или его технические характеристики ?", Label4- "", свойству Visible объекта Label4 присвоить False,
- массив элементов управления OptionButton (присвоить значения свойствам - Option1(0).Caption= "да", Option1(1).Caption= "нет"),
- массивы элементов управления PictureBox: Picture1(0) - Picture1(12) и Picture2(0) - Picture(12). Свойству Visible всех элементов управления присвоить значение False. Свойству Picture каждого элемента управления присвоить значение изображения соответствующего автомобиля и списка технических характеристик (эти списки создаются в приложении Excel, далее таблицы передаются в приложение Paint и сохраняются как рисунки).

4. ввести код в область класса (см. ниже "область проекта Class1"),
5. ввести код, данный ниже, в области:

- General Declarations формы,
- Combo1, событие Click,
- Combo2, событие Click,
- Command1, событие Click,
- Command, событие Click,
- Form, событие Load,
- Form, событие Unload,

6. стартовать проект, получить справку о предполагаемой покупке.

```
//////////////////////////область проекта Class1////////////////////////////////
Public avto As Boolean
Public firma As String
Public model As String
Public stoim As String
Public pict As String
Dim var As String
Private Sub Class_Initialize() ' инициализация полей класса
avto = False: firma = "": model = "": stoim = "": var = ""
End Sub
Public Function Met1()
' Если пользователь нажал кнопку (OptionButton) - Да, то выполнить процедуры
' Met2, Met3, Met4, результатом выполнения которых является вывод данных:
' марка, стоимость, изображение и технические характеристики, иначе
' Met1 = False и выводится сообщение "Приносим свои извинения, мы даем
' информацию для желающих купить автомобиль"
If avto = True Then
model = Met2()
stoim = Met3()
pict = Met4() ' поле pict определяет номера элементов массивов
' PictureBox, см. Met4
Met1 = True
Else
Met1 = False
End If
End Function
' после щелчка на кнопках Да/Нет (два переключателя OptionButton) и выбора
' фирмы из списка ComboBox с именем Combo1 определить марку автомобиля
Public Function Met2()
Select Case firma
```

```
Case "AUDI": Met2 = "A6"
Case "CITROEN": Met2 = "C5"
Case "FORD": Met2 = "Focus"
Case "HONDA": Met2 = "Accord"
Case "HYUNDAI": Met2 = "Elanta"
Case "JEEP": Met2 = "Grand Cherokee LTD"
Case "LAND ROVER": Met2 = "Land Rover Discovery"
Case "LEXSUS": Met2 = "RX330"
Case "MITSUBISHI": Met2 = "Pajero III"
Case "NISSAN": Met2 = "Primera(1.8)"
Case "PEUGEOT": Met2 = "307 XR"
Case "PORSCHE": Met2 = "Cayenne Turbo"
Case "RENAULT": Met2 = "Laguna II"
End Select
End Function
' определить стоимость автомобиля в долларах США
Public Function Met3()
Select Case firma
Case "AUDI": Met3 = "41500"
Case "CITROEN": Met3 = "20100"
Case "FORD": Met3 = "12430"
Case "HONDA": Met3 = "33900"
Case "HYUNDAI": Met3 = "13790"
Case "JEEP": Met3 = "41690"
Case "LAND ROVER": Met3 = "40850"
Case "LEXSUS": Met3 = "65500"
Case "MITSUBISHI": Met3 = "56640"
Case "NISSAN": Met3 = "25100"
Case "PEUGEOT": Met3 = "13808"
Case "PORSCHE": Met3 = "140500"
Case "RENAULT": Met3 = "22900"
End Select
End Function
Public Function Met4()
' при выборе данных из списка ComboBox с именем Combo2
' (после щелчка на кнопке "OK" ) определяется номер элемента массива
' PictureBox, соответствующий выбранной фирме-изготовителю и
' на экран позднее выводится соответствующая фотография
' и/или технические характеристики автомобиля
Select Case firma
Case "AUDI": Met4 = "0"
Case "CITROEN": Met4 = "1"
Case "FORD": Met4 = "2"
Case "HONDA": Met4 = "3"
Case "HYUNDAI": Met4 = "4"
Case "JEEP": Met4 = "5"
Case "LAND ROVER": Met4 = "6"
Case "LEXSUS": Met4 = "7"
Case "MITSUBISHI": Met4 = "8"
Case "NISSAN": Met4 = "9"
Case "PEUGEOT": Met4 = "10"
Case "PORSCHE": Met4 = "11"
Case "RENAULT": Met4 = "12"
End Select
End Function
' процедура Property Let используется для задания значения свойства,
' Property Get - для считывания значения свойства
```

```
Public Property Get varian() As String
Select Case var
Case Is = 0: varian = "pict"
Case Is = 1: varian = "texn"
Case Is = 2: varian = "all"
End Select
End Property
Public Property Let varian(ByVal vNewValue As String)
Select Case vNewValue
Case "изображение": var = 0
Case "технические параметры": var = 1
Case Else: var = 2
End Select
End Property
////////////////////////////////область проекта Form1/////////////////////////////
Dim av As Class1 ' av - экземпляр класса
Dim v As String
Dim i As Integer, j As Integer
Private Sub Combo1_Click()
' сделать невидимыми элементы управления Label и Picture
' (формирующие фотографии, технические характеристики, фирму,
' марку и стоимость), для того, чтобы впоследствии на форму
' выводились только те из них, которые определяет своими
' действиями покупатель
Label5.Visible = False
For i = 0 To 12
Picture1(i).Visible = False
Picture2(i).Visible = False
Next
Dim ot As String ' переменная для хранения сообщения программы
av.firma = Combo1.Text ' значение поля firma объекта av взять из
' списка ComboBox с именем Combo1
av.avto = Option1(0).Value ' значение поля avto объекта av взять
' из поля массива OptionButton
If av.Met1 = True Then
ot = " " & CStr(av.firma) & vbCrLf: ot = ot & " " & vbCrLf
ot = ot & " модель " & CStr(av.model) & vbCrLf: ot = ot & " " & vbCrLf
ot = ot & " цена в $ " & CStr(av.stoim) & vbCrLf: ot = ot & " " & vbCrLf
ot = ot & "Для получения более полной информации обращайтесь_
по телефону 7077888"
MsgBox Title:="Мы можем предложить", Prompt:=ot
Else
Label5.Visible = False
Picture1(Val(av.pict)).Visible = False ' аргумент Picture1: (av.pict)
' определяет индекс элемента массива PictureBox
ot = "Приносим свои извинения, мы даем информацию для желающих_
купить автомобиль"
MsgBox Title:="Автосалон START", Prompt:=ot
End If
End Sub
Private Sub Combo2_Click()
av.varian = Combo2.Text ' см. процедуру Property Let. Присваиваем
' свойству varian значение выбранные из списка ComboBox с именем Combo2
End Sub
Private Sub Command1_Click()
Label5.Visible = False
Label5.Caption = ""
```

```
For i = 0 To 12
Picture1(i).Visible = False
Picture2(i).Visible = False
Next
v = av.varian ' см. процедуру Property Get. Переменной v присваиваем
' значение свойства varian объекта av
av.avto = Option1(0).Value
If av.Met1 = True Then
Select Case v
Case "pict"
Picture1(Val(av.pict)).Visible = True
Case "texn"
Picture2(Val(av.pict)).Visible = True ' технические характеристики
' хранятся как изображения в соответствующих элементах
' массива PictureBox2
Case "all"
Picture1(Val(av.pict)).Visible = True
Picture2(Val(av.pict)).Visible = True
Label5.Visible = True
Label5.Caption = CStr(av.firma) & " " & CStr(av.model) & _
vbCrLf & "цена в $ " & CStr(av.stoim)
End Select
Else
Picture1(Val(av.pict)).Visible = False
Picture2(Val(av.pict)).Visible = False
MsgBox Title:="Автосалон START", Prompt:="Приносим свои_
извинения, мы даем информацию для желающих купить автомобиль"
End If
End Sub
Private Sub Command2_Click()
MsgBox Title:="Автосалон START", Prompt: = "Мы всегда рады помочь!_
Будем рады новой встрече!"
End ' выход из программы после сообщения MsgBox.
End Sub
Private Sub Form_Load()
Set av = New Class1 ' описание объектной переменной дано выше
' заполнение списка ComboBox с именем Combo1 названиями фирм
Combo1.AddItem "AUDI"
Combo1.AddItem "CITROEN"
Combo1.AddItem "FORD"
Combo1.AddItem "HONDA"
Combo1.AddItem "HYUNDAI"
Combo1.AddItem "JEEP"
Combo1.AddItem "LAND ROVER"
Combo1.AddItem "LEXSUS"
Combo1.AddItem "MITSUBISHI"
Combo1.AddItem "NISSAN"
Combo1.AddItem "PEUGEOT"
Combo1.AddItem "PORSCHE"
' заполнение списка ComboBox с именем Combo2 предложениями для
' выбора данных в процедурах Property Get и Property Let
Combo2.AddItem "изображение"
Combo2.AddItem "технические параметры"
Combo2.AddItem "все данные"
End Sub
Private Sub Form_Unload(Cancel As Integer)
Set av = Nothing ' удалить объект из памяти
```

End Sub

Инструкция пользователя

После старта проекта при отрицательном ответе на вопрос «Хотите ли Вы купить машину?» выводится сообщение: «Приносим свои извинения, мы даем информацию для желающих купить автомобиль».

При положительном ответе на вопрос и выборе фирмы-изготовителя из списка на экран выводится сообщение о фирме, марке и стоимости автомобиля.

Далее покупатель может просмотреть или внешний вид, или технические характеристики, или одновременно обе категории, выбрав соответствующую строку во втором списке и сделав щелчок на кнопке ОК (используются процедуры Property Get и Property Let), где дан результат работы программы при выборе строки «все данные».

При щелчке на кнопке Exit выводится сообщение: «Мы всегда рады помочь! Будем рады новой встрече!» и проводится выход из программы.

Заключение (выводы)

Созданный программный продукт позволяет клиенту получить справочные данные при покупке автомобиля. Представленная программа является лишь небольшим примером использования классов, в реальности же сфера применения свойств объектно-базирующегося программирования гораздо шире.

Задания для выполнения:

1. Для предметной области (выбранной в практической работе №1) выполнить объектно-ориентированное проектирование программного продукта.

2. Оформить отчет.

Отчет по практической работе должен быть оформлен на основании требований и состоять из следующих структурных элементов:

- Анализа предметной области
- Определения функций предметной области
- Схемы документопотока
- Выделенных сущностей, атрибутов и установленных связей
- Концептуальной модели
- Описания выходных и входных данных

Лабораторная работа №1 «Построение диаграммы Вариантов использования и диаграммы. Последовательности»

Цель работы: исследование процесса построения диаграмм прецедентов и диаграмм взаимодействий в заданной предметной области с помощью пакета Rational Rose

Теоретические сведения

Постановка задачи (описание предметной области).

Магазин осуществляет продажу товаров клиенту путем оформления документов «Заказ». Директор магазина- Антон, принял решение автоматизировать документооборот продаж товара и пригласил для выполнения работ программиста Павла. Поговорив с Антоном, в соответствие с концепцией жизненного цикла (ЖЦ) программы Павел приступил к описанию бизнес процессов, сопровождающих продажу товара. Взяв за основу язык UML, он начал с построения контекстной диаграммы процессов- Use Case diagram. Диаграмма должна ответить на вопрос-«что должно делаться в системе и кто участник этих процессов».

Задание 1. Создание диаграммы вариантов использования и действующих лиц.

Окончательный вид диаграммы показан на рис. 1.

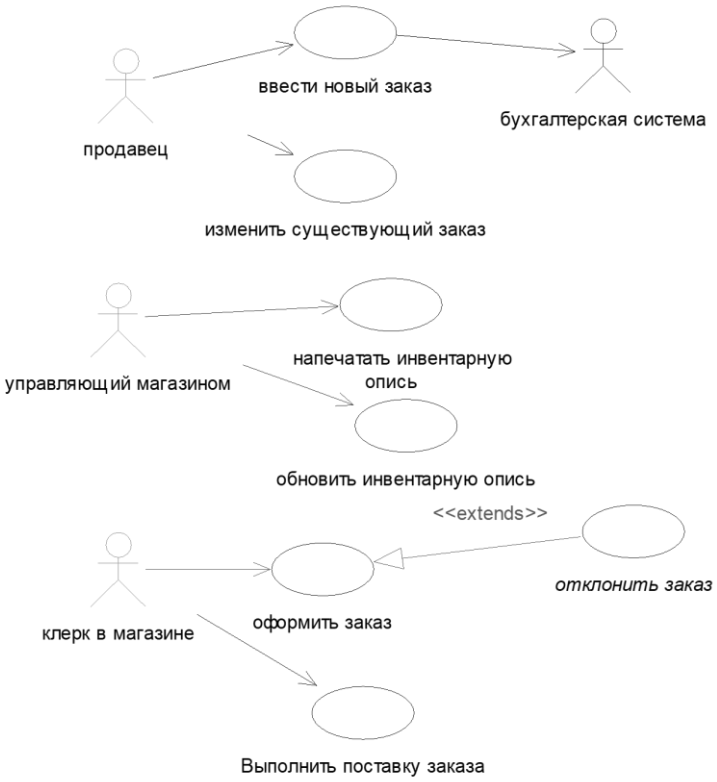

Рис. 1 Диаграмма вариантов использования задачи о заказе товара.

Этапы выполнения

1. Дважды щелкнув мышью на Главной диаграмме Вариантов Использования (Main) в браузере, откройте ее.

2.С помощью кнопки Use Case (Вариант использования) панели инструментов поместите на диаграмму новый вариант использования. Назовите его "Ввести новый заказ".

3. Повторив этапы 2 и 3, поместите на диаграмму остальные варианты использования:

Изменить существующий заказ

Напечатать инвентарную опись

Обновить инвентарную опись

Оформить заказ

Отклонить заказ

Выполнить поставку заказа

4. С помощью кнопки Actor (Действующее лицо) панели инструментов поместите на диаграмму новое действующее лицо.

5. Назовите его "*Продавец*".

6. Повторив шаги 4 и 5, поместите на диаграмму остальных действующих лиц:

Управляющий магазином

Клерк магазина

Бухгалтерская система

7. Создание абстрактного варианта использования (не требующего дальнейшей декомпозиции).

Щелкните правой кнопкой мыши на варианте использования "*Отклонить заказ*" на диаграмме.

В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

Установите флажок Abstract (Абстрактный), чтобы сделать этот вариант использования абстрактным.

Добавление ассоциаций

1. С помощью кнопки Unidirectional Association (Однонаправленная ассоциация) панели инструменте нарисуйте ассоциацию между действующим лицом *Продавец* и вариантом использования "*Ввести заказ*".

2. Повторив шаг 1, поместите на диаграмму остальные ассоциации, согласно рис. 1.

Добавление связи расширения

С помощью кнопки Generalization (Обобщение) панели инструментов нарисуйте связь между вариантом использования "*Отклонить заказ*" и вариантом использования "*Оформить заказ*". Стрелка должна быть направлена от первого варианта использования ко второму. Связь расширения означает, что вариант использования "*Отклонить заказ*" при необходимости дополняет функциональные возможности варианта использования "*Оформить заказ*".

Щелкните правой кнопкой мыши на новой связи между вариантами использования "*Отклонить заказ*" и "*Оформить заказ*".

В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

В раскрывающемся списке стереотипов введите слово extends (расширение), затем нажмите ОК.

Надпись «extends» появится на линии данной связи.

Добавление описаний к вариантам использования

Выделите в браузере вариант использования "*Ввести новый заказ*".

В окне документации введите следующее описание: " Этот вариант использования дает клиенту возможность ввести новый заказ в систему".

С помощью окна документации добавьте описания ко всем остальным вариантам использования.

Добавление описаний к действующему лицу

Выделите в браузере действующее лицо *Продавец.*

В окне документации введите следующее описание: "Продавец — это служащий, старающийся продать товар".

С помощью окна документации добавьте описания к остальным действующим лицам.

Задание 2. Создание диаграммы Последовательности.

Согласовав основные бизнес процессы с Антоном, Павел приступил к построению модели бизнес - процессов, что бы ответить на вопрос - «как это должно делаться в системе». Для начала он выбрал наиболее важный Вариант использования - «Ввод нового заказа» и построил для него диаграммы взаимодействия.

Диаграммы взаимодействия включают в себя два типа диаграмм - Последовательности и Кооперативную.

Этапы выполнения

Настройка программной среды

1. В меню модели выберите пункт Tools >- Options (Инструменты >- Параметры).

2. Перейдите на вкладку Diagram (Диаграмма).

3. Установите флажки Sequence numbering, Collaboration numbering и Focus of control.

4. Нажмите ОК, чтобы выйти из окна параметров.

Создание диаграммы Последовательности

1. Щелкните правой кнопкой мыши на Логическом представлении браузера.

2. В открывшемся меню выберите пункт New > Sequence Diagram (Создать >Диаграмма Последовательности).

3. Назовите новую диаграмму: Ввод заказа.

4. Дважды щелкнув на этой диаграмме, откройте ее.

Добавление на диаграмму действующего лица и объектов

1. Перетащите действующее лицо *Продавец* из браузера на диаграмму.

2. Нажмите кнопку Object (Объект) панели инструментов.

3. Щелкните мышью в верхней части диаграммы, чтобы поместить туда новый объект.

4. Назовите объект *Выбор варианта заказа*.

5. Повторив шаги 3 и 4, поместите на диаграмму объекты:

- *Форма деталей заказа*
- *Заказ №1234*

Добавление сообщений на диаграмму

1. На панели инструментов нажмите кнопку Object Message (Сообщение объекта).

2. Проведите мышью от линии жизни действующего лица *Продавец* к линии жизни объекта *Выбор варианта заказа*.

3. Выделив сообщение, введите его имя — *Создать новый заказ*.

4. Повторив шаги 2 и 3, поместите на диаграмму сообщения:

- *Открыть форму* — между *Выбор Варианта Заказа* и *Форма деталей Заказа*
- *Ввести номер заказа, заказчика и число заказываемых предметов* — между *Продавец* и *Форма Деталей Заказа*
- *Сохранить заказ* — между *Продавец* и *Форма Деталей Заказа*
- *Создать пустой заказ* — между *Форма Деталей Заказа* и *Заказ N1234*
- *Ввести номер заказа, заказчика и число заказываемых предметов* — между *Форма Деталей Заказа* и *Заказ N1234*
- *Сохранить заказ* — между *Форма Деталей Заказа* и *Заказ N1234*

Завершен первый этап работы. Готовая диаграмма Последовательности представлена на рис. 2.

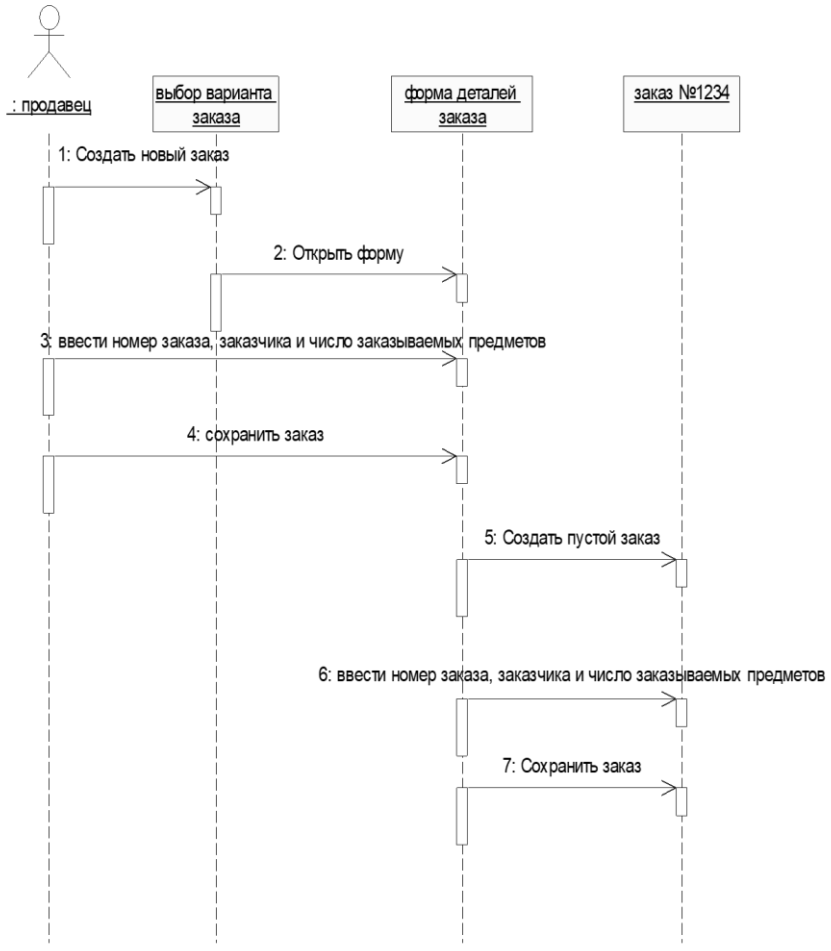

Рис. 2. Диаграмма последовательности без управляющих элементов.

Теперь нужно позаботиться об управляющих объектах и о взаимодействии с базой данных. Как видно из диаграммы, объект *Форма Деталей Заказа* имеет множество ответственностей, с которыми лучше всего мог бы справиться управляющий объект. Кроме того, новый заказ должен сохранять себя в базе данных сам. Вероятно, эту обязанность лучше было бы переложить на другой объект.

Добавление на диаграмму дополнительных объектов

1. Нажмите кнопку Object панели инструментов.

2. Щелкните мышью между объектами *Форма Деталей Заказа* и *Заказ №1234*, чтобы поместить туда новый объект.

3. Введите имя объекта — *Управляющий заказами*.

4. Нажмите кнопку Object панели инструментов.

5. Новый объект расположите справа от *Заказ №1234*.

6. Введите его имя- *Управляющий транзакциями*.

Назначение ответственностей объектам

1. Выделите сообщение 5: *Создать пустой заказ*.

2. Нажав комбинацию клавиш CTRL+D, удалите это сообщение.

3. Повторите шаги 1 и 2 для удаления двух последних сообщений:

- *Вести номер заказа, заказчика и число заказываемых предметов*
- *Сохранить заказ*

4. Нажмите кнопку Object Message панели инструментов.

5. Поместите на диаграмму новое сообщение, расположив его под сообщением 4 между *Форма деталей заказа* и *Управляющий заказами*.

6. Назовите его *Сохранить заказ*.

7. Повторите шаги 4 — 6, добавив сообщения с шестого по девятое и назвав их:

- *Создать новый заказ* — между *Управляющий заказами* и *Заказ №1234*
- *Ввести номер заказа, заказчика и число заказываемых предметов*- между *Управляющий заказами* и *Заказ №1234*
- *Сохранить заказ*- между *Управляющий заказами* и *Управляющий транзакциями*
- *Информация о заказе* —между *Управляющий транзакциями* и *Заказ №1234*

8. На панели инструментов нажмите кнопку Message to Self (Сообщение себе).

9. Щелкните на линии жизни объекта *Управляющий транзакциями* ниже сообщения 9, добавив туда рефлексивное сообщение.

10. Назовите его *Сохранить информацию о заказе в базе данных*.

Соотнесение объектов с классами

1. Щелкните правой кнопкой мыши на объекте *Выбор варианта заказа*.

2. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

В раскрывающемся списке классов выберите пункт <New> (Создать). Появится окно спецификации классов.

4. В поле Name введите *Выбор заказа*.

5. Щелкните на кнопке ОК. Вы вернетесь в окно спецификации объекта.

6. В списке классов выберите класс *Выбор Заказа*.

7. Щелкните на кнопке ОК, чтобы вернуться к диаграмме. Теперь объект называется *Выбор варианта заказа: Выбор Заказа*

8. Для соотнесения остальных объектов с классами повторите шаги с 1 по 7:

- Класс *Детали заказа* соотнесите с объектом *Форма деталей заказа*
- Класс *Упр_заказами* — с объектом *Управляющий заказами*
- Класс *Заказ* — с объектом *Заказ N 1234*
- Класс *Упр_ транзакциями* — с объектом *Управляющий транзакциями*

Соотнесение сообщений с операциями

1. Щелкните правой кнопкой мыши на сообщении 1: *Создать новый заказ.*

2. В открывшемся меню выберите пункт <new operation> (создать операцию). Появится окно спецификации операции.

3. В поле Name введите имя операции —*Создать*.

4. Нажмите на кнопку ОК, чтобы закрыть окно спецификации операции и вернуться к диаграмме.

5. Еще раз щелкните правой кнопкой мыши на сообщении 1.

6. В открывшемся меню выберите новую операцию *Создать()*.

7. Повторите шаги с 1 по 6, чтобы соотнести с операциями все остальные сообщения:

- Сообщение 2: *Открыть форму* соотнесите с операцией *Открыть()*

- Сообщение 3: *Ввести номер заказа, заказчика и число заказываемых предметов* — с операцией *Ввести номер заказа, заказчика и число заказываемых предметов()*

- Сообщение 4: *Сохранить заказ* — с операцией *Сохранить заказ()*

- Сообщение 5: *Сохранить заказ* — с операцией *Сохранить заказ()*

- Сообщение 6: *Создать пустой заказ* – с операцией *Создать пустой заказ()*

- Сообщение 7: *Ввести номер заказа, заказчика и число заказоваемых предметов*- с одноименной операцией.

- Сообщение 8 *Сохранить заказ* – с операцией *Сохранить заказ()*

- Сообщение 9 *Информация о заказе* – с одноименной операцией

- Сообщение 10 *Сохранить информацию о заказе* с одноименной операцией.

Диаграмма должна выглядеть, как на рис. 3.

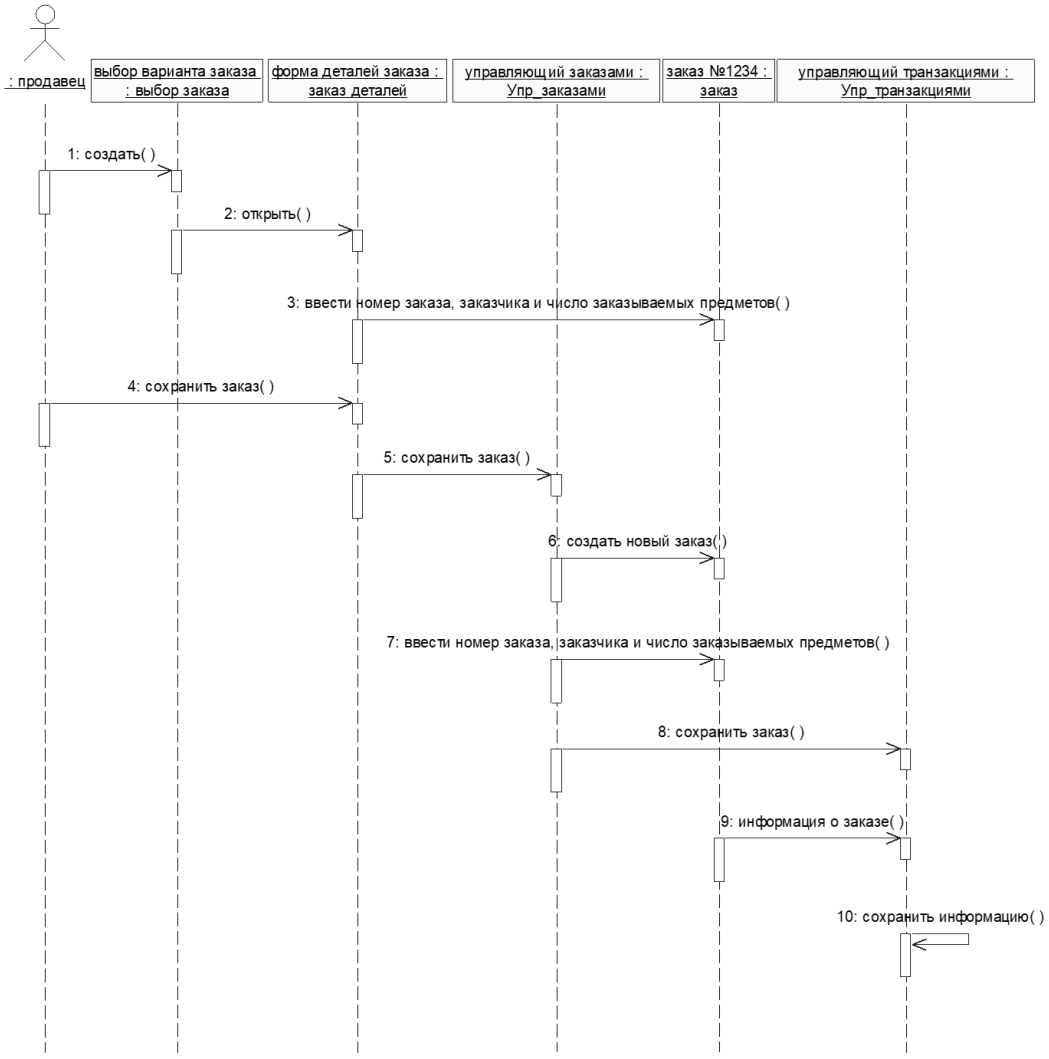

Рис. 3. Окончательный вид диаграммы последовательности

Лабораторная работа №2 «Построение диаграммы Кооперации и диаграммы Развертывания»

Цель работы: исследование моделирования процессов, описывающих взаимодействие объектов в диаграмме кооперации и диаграмме развертывания в заданной предметной области с помощью пакета Rational Rose

Задание 1. Построение Создание Кооперативной диаграммы

Конечный вид диаграммы представлен на рис. 4.

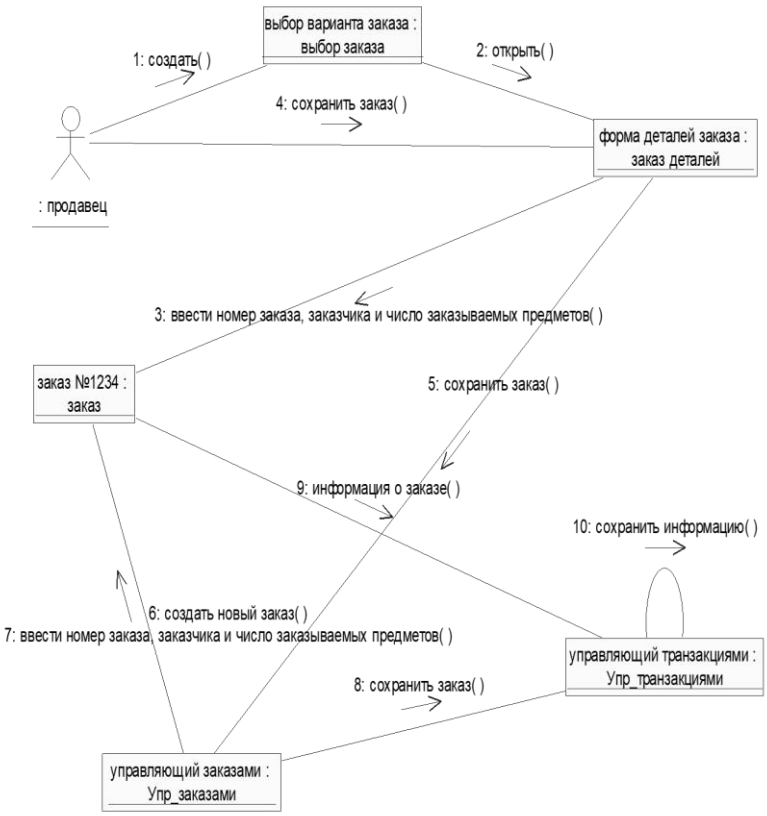

Рис. 4. Окончательный вид кооперативной диаграммы.

1. Щелкните правой кнопкой мыши на Логическом представлении в браузере.

2. В открывшемся меню выберите пункт New > Collaboration Diagram (Создать > Кооперативная диаграмма).

3. Назовите эту диаграмму *Ввод заказа*.

4. Дважды щелкнув мышью на диаграмме, откройте ее.

Добавление действующего лица и объектов на диаграмму

1. Перетащите действующее лицо *Продавец* из браузера на диаграмму.

2. Нажмите кнопку Object (Объект) панели инструментов.

3. Щелкните мышью где-нибудь внутри диаграммы, чтобы поместить туда новый объект.

4. Назовите объект *Выбор варианта заказа*.

5. Повторив шаги 3 и 4, поместите на диаграмму объекты:

- *Форма деталей заказа*
- *Заказ №1234*

Добавление сообщений на диаграмму

1. На панели инструментов нажмите кнопку Object Link (Связь объекта).

2. Проведите мышью от действующего лица *Продавец* к объекту *Выбор варианта заказа*.

3. Повторите шаги 1 и 2, соединив связями следующие объекты:

- Действующее лицо *Продавец* и объект *Форма деталей Заказа*
- Объект *Форма деталей Заказа* и объект *Выбор Варианта Заказа*
- Объект *Форма деталей Заказа* объект *Заказ N1234*

4. На панели инструментов нажмите кнопку Link Message (Сообщение связи).

5. Щелкните мышью на связи между *Продавец* и *Форма деталей Заказа.*

6. Выделив сообщение, введите его имя —*Создать новый заказ*;

7. Повторив шаги с 4 по 6, поместите на диаграмму сообщения:

- *Открыть форму* — между *Выбор Варианта Заказа* и *Форма Деталей Заказа*.

- *Ввести номер заказа, заказчика и число заказываемых предметов* — между *Продавец* и *Форма Деталей Заказа*

- *Сохранить заказ* — между *Продавец* и *Форма деталей Заказа*

- *Создать пустой заказ* — между *Форма деталей Заказа* и *Заказ №1234*

- Ввести номер заказа, заказчика и число заказываемых предметов — между *Форма деталей Заказа* и *Заказ №1234*

- Сохранить заказ — между *Форма деталей Заказа* и *Заказ №1234*

Теперь нужно поместить на диаграмму дополнительные элементы, а также рассмотреть ответственности объектов.

Добавление на диаграмму дополнительных объектов

1. Нажмите кнопку Object панели инструментов.

2. Щелкните мышью где-нибудь на диаграмме, чтобы поместить туда новый объект.

3. Введите имя объекта — *Управляющий заказами*.

4. На панели инструментов нажмите кнопку Object.

5. Поместите на диаграмму еще один объект.

6. Введите его имя — *Управляющий транзакциями*.

Назначение ответственностей объектам

1. Выделите сообщение 5: *Создать пустой заказ*. Выделяйте слова, а не стрелку.

2. Нажав комбинацию клавиш CTRL+D, удалите это сообщение.

3. Повторите шаги 1 и 2 для удаления сообщений 6 и 7:

- *Ввести номер заказа, заказчика и число заказываемых предметов*

- *Сохранить заказ*

4. Выделите связь между объектами *Форма деталей Заказа* и *Заказ №1234*

5. Нажав комбинацию клавиш CTRL+D, удалите эту связь

6. На панели инструментов нажмите кнопку Object Link (Связь объекта).

7. Нарисуйте связь между *Форма деталей Заказа* и *Управляющий Заказами*.

8. На панели инструментов нажмите кнопку Object Link (Связь объекта).

9. Нарисуйте связь между *Управляющий Заказами* и *Заказ №1234*

10. На панели инструментов нажмите кнопку Object Link (Связь объекта).

11. Нарисуйте связь между *Заказ №1234* и *Управляющий Транзакцией*.

12. На панели инструментов нажмите кнопку Object Link (Связь объекта).

13. Нарисуйте связь между *Управляющий Заказами* и *Управляющий Транзакцией*.

14. На панели инструментов нажмите кнопку Link Message (Сообщение связи).

15. Щелкните мышью на связи между объектами *Форма деталей Заказа* и *Управляющий Заказами*, чтобы ввести новое сообщение.

16. Назовите это сообщение Сохранить заказ.

17. Повторите шаги 14 — 16, добавив сообщения с шестого по девятое, и назвав их:

- *Создать новый заказ* — между *Управляющий Заказами* и *Заказ №1234*

- *Ввести номер заказа, заказчика и число заказываемых предметов* — между *Управляющий Заказами* и *Заказ №1234*

- *Сохранить заказ* — между *Управляющий Заказами* и *Управляющий Транзакцией*

- *Информация о заказе* — между *Управляющий Транзакцией* и *Заказ №1234*

18. На панели инструментов нажмите кнопку Link to Self (Связь с собой).

19. Щелкнув на объекте *Управляющий Транзакцией*, добавьте к нему рефлексивное сообщение.

20. На панели инструментов нажмите кнопку Link Message (Сообщение связи).

21. Щелкните мышью на рефлексивной связи *Управляющий Транзакциями*, чтобы ввести туда сообщение.

22. Назовите новое *Сохранить информацию о заказе в базе данных*.

Соотнесение объектов с классами (если классы были созданы при разработке описанной выше диаграммы Последовательности)

1. Найдите в браузере класс *Выбор Заказа*.

2. Перетащите его на объект *Выбор варианта заказа* на диаграмме.

3. Повторите шаги 1 и 2 соотнеся остальные объекты и соответствующие им классы:

- Класс *заказ деталей* соотнесите с объектом *Форма деталей заказа*

- Класс *Упр_заказами* — с объектом *Управляющий Заказами*

- Класс *Заказ* — с объектом *Заказ №1234*

- Класс *Упр_транзакциями* — с объектом *Управляющий транзакциями*

Соотнесение объектов с классами (если вы не создавали описанную выше диаграмму Последовательности)

1. Щелкните правой кнопкой мыши на объекте *Форма деталей Заказа*.

2. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

3. В раскрывающемся списке классов выберите пункт <New> (Создать). Появится окно спецификации классов.

4. В поле имени введите *Выбор заказа*.

5. Щелкните на кнопке ОК. Вы вернетесь в окно спецификации объекта.

6. В списке классов выберите класс *Выбор заказа.*

7. Щелкните на кнопке ОК, чтобы вернуться к диаграмме. Теперь объект называется *Выбор варианта заказа: Выбор Заказа*

8. Для соотнесения остальных объектов с классами повторите шаги с 1 по 7:

- Класс *Детали заказа* соотнесите с объектом *Форма деталей заказа*

- Класс *Упр_заказами* — с объектом *Управляющий заказами*

- Класс *Заказ* — с объектом *Заказ N 1234*

- Класс *Упр_ транзакциями* — с объектом *Управляющий транзакциями*

Соотнесение сообщений с операциями (если операции были созданы при разработке описанной выше диаграммы Последовательности)

1. Щелкните правой кнопкой мыши на сообщении 1: Создать новый заказ.

2. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

3. В раскрывающемся списке имен укажите имя операции — *Создать()*.

4. Нажмите на кнопку ОК.

5. Повторите шаги 1—4 для соотнесения с операциями остальных сообщений:

- Сообщение 2: *Открыть форму* соотнесите с операцией *Открыть()*

- Сообщение 3: *Ввести номер заказа, заказчика и число заказываемых предметов* — с операцией *Ввести номер заказа, заказчика и число заказываемых предметов()*

- Сообщение 4: *Сохранить заказ* — с операцией *Сохранить заказ()*

- Сообщение 5: *Сохранить заказ* — с операцией Сохранить заказ()

Сообщение 6: *Создать пустой заказ* – с операцией Создать пустой заказ()

Сообщение 7: *Ввести номер заказа, заказчика и число заказоваемых предметов*- с одноименной операцией.

Сообщение 8 Сохранить заказ – с операцией Сохранить заказ()

Сообщение 9 *Информация о заказе* – с одноименной операцией

Сообщение 10 *Сохранить информацию о заказе* с одноименной операцией

Соотнесение сообщений с операциями (если вы не создавали описанную выше диаграмму Последовательности)

1. Щелкните правой кнопкой мыши на сообщении 1: *Создать новый заказ()*.

2. В открывшемся меню выберите пункт <new operation> (создать операцию). Появится окно спецификации операции.

3. В поле имени введите имя операции —*Создать().*

4. Нажмите на кнопку ОК, чтобы закрыть окно спецификации операции и вернуться к диаграмме.

5. Еще раз щелкните правой кнопкой мыши на сообщении 1.

6. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

7. В раскрывающемся списке Name (Имя) укажите имя новой операции.

8. Нажмите на кнопку ОК.

9. Повторите шаги 1—8, чтобы создать новые операции и соотнести с ними остальные сообщения:

- Сообщение 2: *Открыть форму* соотнесите с операцией *Открыть()*

- Сообщение 3: *Ввести номер заказа, заказчика и число заказываемых предметов* — с операцией *Ввести номер заказа, заказчика и число заказываемых предметов()*

- Сообщение 4: *Сохранить заказ* — с операцией *Сохранить заказ()*

- Сообщение 5: *Сохранить заказ* — с операцией *Сохранить заказ()*

Сообщение 6: *Создать пустой заказ* – с операцией *Создать пустой заказ()*

Сообщение 7: *Ввести номер заказа, заказчика и число заказоваемых предметов*- с одноименной операцией.

Сообщение 8 Сохранить заказ – с операцией *Сохранить заказ()*

Сообщение 9 *Информация о заказе* – с одноименной операцией

Сообщение 10 *Сохранить информацию о заказе* с одноименной операцией

Ваша диаграмма должна выглядеть, как показано на рис. 4

Лабораторная работа №3 «Построение диаграммы Деятельности, диаграммы Состояний и диаграммы Классов»

Цель работы: исследование процесса построения диаграммы состояний и диаграммы классов в заданной предметной области с помощью пакета Rational Rose 2000.

Задание 1. Постройте диаграмму Состояний

Постройте диаграмму Состояний для класса *Заказ*, показанную на рис. 5.

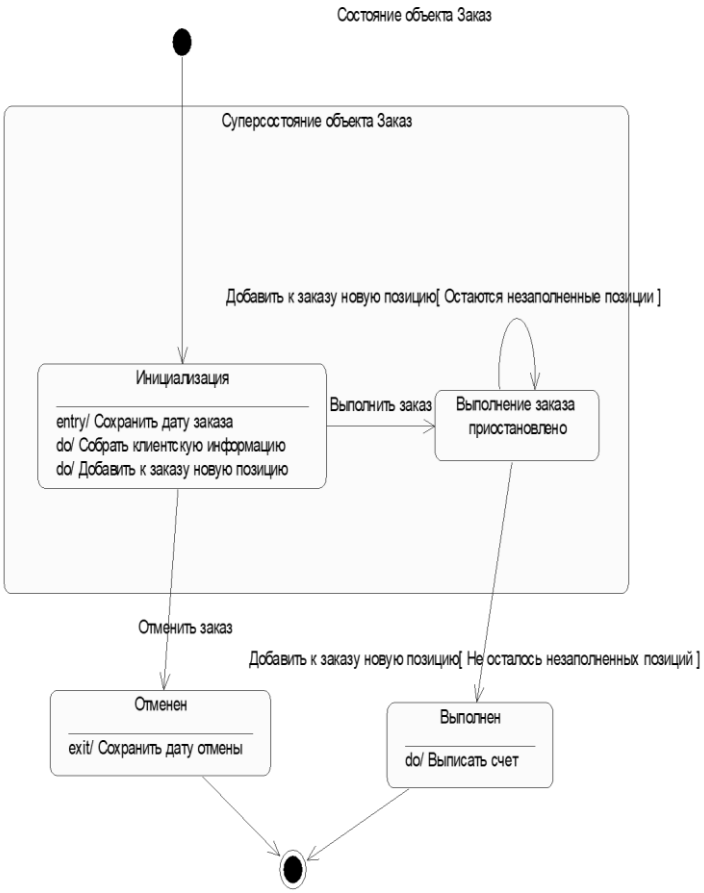

Рис 5. Диаграмма состояний для класса *Заказ*

Этапы выполнения

Создание диаграммы

1. Найдите в браузере класс *Заказ*.
2. Щелкните на классе правой кнопкой мыши и в открывшемся меню укажите пункт New > Statechart Diagram (Создать диаграмму состояний).

Добавление начального и конечного состояний

1. Нажмите кнопку Start State (Начальное состояние) панели инструментов.
2. Поместите это состояние на диаграмму.
3. Нажмите кнопку End State (Конечное состояние) панели инструментов.
4. Поместите это состояние на диаграмму.

Добавление суперсостояния

1. Нажмите кнопку State (Состояние) панели инструментов.
2. Поместите это состояние на диаграмму.

Добавление оставшихся состояний

1. На панели инструментов нажмите кнопку State (Состояние).
2. Поместите состояние на диаграмму.
3. Назовите состояние *Отменен*.
4. На панели инструментов нажмите кнопку State(Состояние).
5. Поместите состояние на диаграмму.
6. Назовите состояние *Выполнен*.

7. На панели инструментов нажмите кнопку State(Состояние).

8. Поместите состояние на диаграмму внутрь суперсостояния.

9. Назовите состояние *Инициализация*.

10. На панели инструментов нажмите кнопку State (Состояние).

11. Поместите состояние на диаграмму внутрь суперсостояния.

12. Назовите состояние *Выполнение заказа приостановлено*.

Описание состояний

1. Дважды щелкните мышью на состоянии *Инициализация*.

2. Перейдите на вкладку Detail (Подробно).

3. Щелкните правой кнопкой мыши в окне Actions(Действия).

4. В открывшемся меню выберите пункт Insert(Вставить).

5. Дважды щелкните мышью на новом действии.

6. Назовите его *Сохранить дату заказа*.

7. Убедитесь, что в окне When (Когда) указан пункт On Entry (На входе).

8. Повторив шаги 3—7, добавьте следующие действия:

- *Собрать клиентскую информацию*, в окне When укажите DO (Выполнять между входом и выходом)

- *Добавить к заказу новые позиции*, укажите DO

9. Нажмите два раза на ОК, чтобы закрыть спецификацию.

10. Дважды щелкните мышью на состоянии *Отменен*.

11. Повторив шаги 2—7, добавьте действия:

Сохранить дату отмены, укажите On Exit (На выходе)

12. Нажмите два раза на ОК, чтобы закрыть спецификацию.

13. Дважды щелкните мышью на состоянии *Выполнен*.

14. Повторив шаги 2—7, добавьте действие:

- *Выписать счет*, укажите On Exit

15. Нажмите два раза на ОК, чтобы закрыть спецификацию.

Добавление переходов

1. Нажмите кнопку Transition (Переход) панели инструментов.

2. Щелкните мышью на начальном состоянии.

3. Проведите линию перехода к состоянию *Инициализация*.

4. Повторив шаги с первого по третий, создайте следующие переходы:

- От состояния Инициализация к состоянию *Выполнение заказа приостановлено*

- От состояния *Выполнение заказа приостановлено* к состоянию *Выполнен*

- От суперсостояния к состоянию *Отменен*

- От состояния *Отменен* к конечному состоянию

- От состояния *Выполнен* к конечному состоянию

5. На панели инструментов нажмите кнопку Transition to Self (Переход к себе).

6. Щелкните мышью на *состоянии Выполнение заказа приостановлено*

Описание переходов

1. Дважды щелкнув мышью на переходе от состояния *Инициализация* к состоянию *Выполнение заказа приостановлено*, откройте окно спецификации перехода.

2. В поле Event (Событие) введите фразу *Выполнить заказ*.

3. Щелкнув на кнопке ОК, закройте окно спецификации.

4. Повторив шаги с первого по третий, добавьте событие *Отменить заказ* к переходу между суперсостоянием и состоянием *Отменен*.

5. Дважды щелкнув мышью на переходе от состояния *Выполнение заказа приостановлено* к состоянию *Выполнен*, откройте окно его спецификации.

6. В поле Event (Событие) введите фразу *Добавить к заказу новую позицию*.

7. Перейдите на вкладку Detail (Подробно).

8. В поле Guard Condition (Сторожевое Условие) введите *Не осталось незаполненных позиций*.

9. Щелкнув на кнопке ОК, закройте окно спецификации.

10. Дважды щелкните мышью на рефлексивном переходе (Transition to Self) состояния *Выполнение заказа приостановлено*.

11. В поле Event (Событие) введите фразу *Добавить к заказу новую позицию*.

12. Перейдите на вкладку Detail (Подробно).

13. В поле Guard Condition (Сторожевое Условие) введите *Остаются незаполненные позиции*.

14. Щелкнув на кнопке ОК, закройте окно спецификации.

Задание 2. Построение диаграммы Активности для варианта использования «Выполнить поставку Заказа».

Побеседовав с Павлом, Антон понял, что необходимо согласовать логику реализации еще одного варианта использования «Выполнить поставку заказа». Стало ясно, что здесь возможны несколько альтернативных потоков управления. Для таких ситуаций более удобно использовать не диаграммы взаимодействия, приспособленные для единственного потока, а диаграмму активности.

Описание варианта использования.

При оформлении заказа проверяют каждую содержащуюся в нем позицию, чтобы убедиться в наличии соответствующих товаров на складе. После этого выписываются товары для реализации заказа. Во время выполнения этих процедур одновременно проверяется прохождение платежа. Если платеж прошел, и товары имеются на складе, то осуществляется их поставка. Если платеж прошел, но товары на складе отсутствуют, то заказ ставится в ожидание. Если платеж не прошел, то заказ аннулируется.

Этапы выполнения

1. Найдите в браузере вариант использования «Выполнить поставку заказа»

2. Щелкните на классе правой кнопкой мыши и в открывшемся меню укажите пункт New > Activity Diagram (Создать диаграмму активности).

3. Назвите диаграмму «Выполнить поставку» и откройте ее двукратным щелчком мыши

4. На панели инструментов щелкните мышкой на элементе Swimline, затем на поле диаграммы. На диаграмме появится разделительная линия («водная дорожка»).

5. Установите курсор на заголовок NewSwinlane и нажмите правую клавишу мыши. В выпадающем списке нажмите Select in brousere. В браузере выделится этот объект. Нажав правую клавишу мыши в выпадающем списке выберете Open Specefication и откройте спецификацию. Измените поле Name на *Клерк.* Выберите в поле Class *Клерк в магазине*.

6. Выполните заново пункты 5-6 и присвойте полю Name *Система*, Class- *Бухгалтерская система.*

7. Найдите в браузере сплошной черный кружок (начальное состояние). Перенесите его на дорожку *Клерк.*

8. Выберете из панели инструментов объект Activity и поместите его на диаграмму в “дорожку” *Клерк*. Измените имя объекта на ”*Получить заказ*”.

9. Повторите предыдущий этап, создайте на «дорожке» *Клерк* 4 новых Activity и присвойте им имена *Проверить позицию заказа, закрепить позицию за заказом, Поставить заказ в ожидание, Скомплектовать заказ*

10. Поместите на «дорожку» 2 новых объекта End State (конечное состояние). Одному из них измените поле Name на «*Выполнить поставку*»

11. На дорожку *Система* поместите новый объект Activity и присвойте полю Name “*Проверить платеж*”. На эту же дорожку поместите новый объект End State и измените в его спецификации поле Name на «*Отменить заказ*».

12. Поместить на «дорожку» Клерк 2 объекта Horisontal Sinhronization (горизонтальная синхронизация). Присвойте полю Name спецификации одного объекта «*1*», другого- «*2*».

13. Поместить на «дорожку» *Клерк* объект Dicision (выбор). Через спецификацию присвойте полю Name «*Позиция имеется?*».

14. Поместить на «дорожку» *Система* объект Dicision. Присвойте полю Name «*Деньги поступили?*».

15. Щелкните мышкой на панели инструментов объекте- стрелке State Transition (состояние перехода). Затем щелкните мышкой на диаграмме объекта начальное состояние. Удерживая кнопку мыши перенесите курсор на активность ”*Получить заказ*” и лишь затем отпустить курсор. В результате два объекта будут соединены стрелкой.

16. Выполните этап 14, соединив стрелкой объект Активность «*Получить заказ*» с объектом Horisontal Sinhronization 1.

17. Соедините этими же стрелками объекты 1 и «Проверить платеж», 1 и «Проверить позицию заказа», «Проверить заказ» и «Деньги подступили?», «Деньги поступили?» и «Отменить заказ», «Проверить позицию заказа» и «Позиция имеется», «Позиция имеется» и «Закрепить

позицию за заказом», «Деньги получены?» и 2, «Закрепить позицию за заказом» и 2, «Позиция имеется?» и «Поставить заказ в ожидание», 2 и «Скомплектовать заказ», «Скомплектовать заказ» и «Выполнить поставку», «Поставить заказ в ожидание» и объект Конечное состояние (без имени).

18. Присвоим некоторым стрелкам наименование полю Event (условие перехода).Для этого, установим курсор на стрелке, соединяющей «Деньги получены?» и «Отменить заказ». Двукратным щелчком мыши откроем окно спецификации. В поле Event введем «Нет».

19. Выполним пункт 18 для стрелки, соединяющей «Деньги получены?» и 2 и присвойте Event «Да». Аналогично для стрелки соединяющей «Позиция имеется?» и «Закрепить позицию за заказом» присвоить Event «Да». Стрелке, соединяющей «Позиция имеется?» и «Поставить заказ в ожидание» - «Нет».

20. Добавим элементарные действия (Actions) к активности “Проверить позицию заказа». Установим курсор на «Проверить позицию заказа» и двукратным щелчком мыши откроем окно спецификации. Откроем закладку Actions. Установим курсор на свободное поле и нажмем правую клавиши мыши. В выпадающем меню нажмем Insert. В появившейся заставке в поле When выберем Entry(на входе в активность), В поле Name введем «Просмотреть спецификацию к заказу». Нажать Ok. Вновь нажмем курсор правой мыши и введем новое действие. Полю When прсвоим Do(промежуток между входом и выходом), а полю Name «Найти новую позицию». При вводе третьей активности полю When присвойте Exit (выход), а полю Name «Передать результаты поиска».

21. Путем перемещения объектов (установить курсор мыши- нажать- тащить- отпустить) привести диаграмму к виду, показанному на рис. 6.

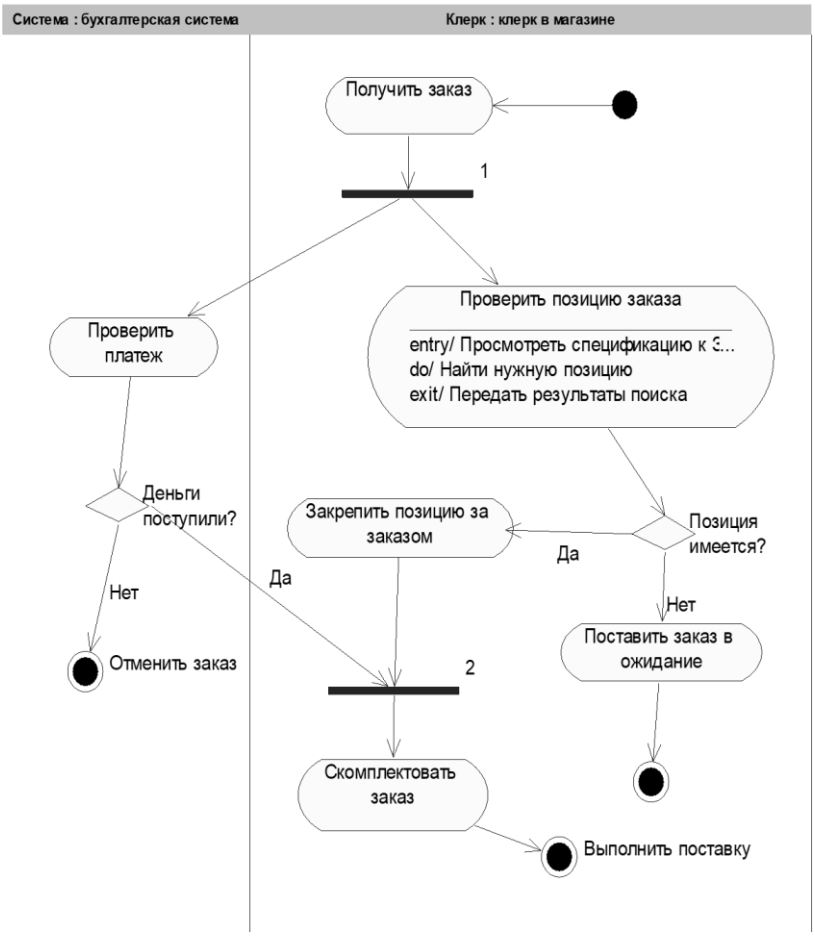

Рис. 6 Диаграмма активности для варианта использования «Выполнить поставку заказа»

Задание 3. Пакеты и классы

В этом упражнении необходимо сгруппировать в пакеты классы, созданные при выполнении предыдущих работ. Затем нужно будет построить несколько диаграмм Классов и показать на них классы и пакеты системы.

Создание диаграммы Классов

Объедините обнаруженные классы в пакеты. Создайте диаграмму Классов для отображения пакетов, диаграммы Классов, для представления классов в каждом пакете и диаграмму Классов для представления всех классов варианта использования "Ввести новый заказ".

Этапы выполнения

Создание пакетов

1. Щелкните правой кнопкой мыши на Логическом представлении браузера.
2. В открывшемся меню выберите пункт New > Package (Создать >Пакет).
3. Назовите новый пакет Сущности.
4. Повторив шаги 1—3, создайте пакеты Границы и Управление.

Создание Главной диаграммы Классов

1. Дважды щелкнув мышью на Главной диаграмме Классов, находящейся под Логическим представлением браузера, откройте ее.
2. Перетащите пакет *Сущности* из браузера на диаграмму.
3. Перетащите пакеты *Границы* и *Управление* из браузера на диаграмму.

Главная диаграмма Классов должна выглядеть, как показано на рис. 7

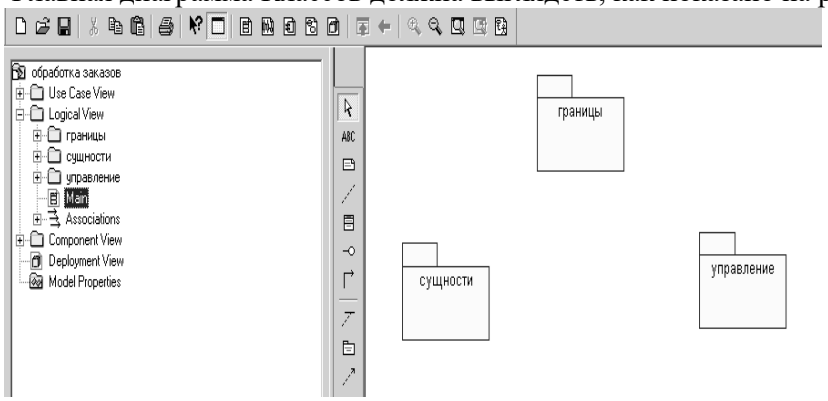

Рис. 7 Главная диаграмма классов в логическом представлении браузера.

Создание диаграммы Классов для сценария "Ввести новый заказ" с отображением всех классов

1. Щелкните правой кнопкой мыши на Логическом представлении браузера.
2. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмма Классов).
3. Назовите новую диаграмму Классов: Ввод нового заказа.
4. Дважды щелкнув мышью на этой диаграмме в браузере, откройте ее.
5. Перетащите из браузера все классы (*Выбор_заказа, Заказ_деталей, упр_заказами, Заказ, Упр_транзакциями.*

Объединение классов в пакеты

1. В браузере перетащите класс *выбор_заказа* на пакет Границы.
2. Перетащите класс *заказ_деталей* на пакет Границы.
3. Перетащите классы *Упр_заказами* и *Упр-транзакциями* на пакет Управление.
4. Перетащите класс *Заказ* на пакет Сущности.

Классы и пакеты в браузере показаны на рис. 9

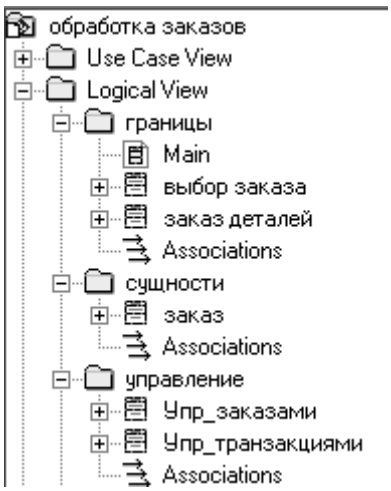

Рис. 8 Представление пакетов и классов

Добавление диаграмм Классов к каждому пакету

1. В браузере щелкните правой кнопкой мыши на пакете *Границы*.

2. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмма Классов).

3. Введите имя новой диаграммы — Main (Главная).

4. Дважды щелкнув мышью на этой диаграмме, откройте ее.

5. Перетащите на нее из браузера классы *выбор_заказа* и *заказ_деталей*.

6. Закройте диаграмму.

В браузере щелкните правой кнопкой мыши на пакете *Сущности.*

8. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмма Классов).

9. Введите имя новой диаграммы — Main (Главная).

10. Дважды щелкнув мышью на этой диаграмме, откройте ее.

11. Перетащите на нее из браузера класс *Заказ*.

12. Закройте диаграмму

13. В браузере щелкните правой кнопкой мыши на пакете *Управление*

14. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмма Классов).

15. Введите имя новой диаграммы — Main (Главная).

16. Дважды щелкнув мышью на этой диаграмме, откройте ее.

17. Перетащите на нее из браузера классы *Упр_заказами* и *Упр_транзакциями*

18. Закройте диаграмму

Задание 4. Уточнение методов и свойств классов.

В этом упражнении к описаниям операций будут добавлены детали, включая параметры и типы возвращаемых значений, и определены атрибуты классов

Постановка проблемы

Для определения атрибутов классов был проанализирован поток событий. В результате к классу *Заказ* диаграммы Классов были добавлены атрибуты *Номер заказа* и *Имя клиента*. Так как в одном заказе можно указать большое количество товаров и у каждого из них имеются свои собственные данные и поведение, было решено моделировать товары как самостоятельные классы, а не как атрибуты класса *Заказ*.

Добавление атрибутов и операций

Добавим атрибуты и операции к классам диаграммы Классов "Ввод нового заказа". При этом используем специфические для языка особенности. Установим параметры так, чтобы показывать все атрибуты, все операции и их сигнатуры. Применим нотацию UML.

Этапы выполнения

Настройка

1. В меню модели выберите пункт Tools > Options (Инструменты >Параметры).

2. Перейдите на вкладку Diagram.

3. Убедитесь, что флажок Show visibility (Показать видимость) установлен.

4. Убедитесь, что флажок Show stereotyps (Показать стереотипы) установлен.

5. Убедитесь, что флажок Show operation signatures (Показать сигнатуры операций) установлен.

6. Убедитесь, что флажки Show all attributes (Показать все атрибуты) и Show all operations (Показать вое операции) установлены.

7. Убедитесь, что флажки Suppress attributes (Подавить атрибуты) и Suppress operations (Подавить операции) сброшены.

8. Перейдите на вкладку Notation (Нотация).

9. Убедитесь, что флажок Visibility as icons (Отображать пиктограммы) сброшен.

Добавление нового класса

1. Найдите в браузере диаграмму Классов варианта использования "Ввести новый заказ".

2. Дважды щелкнув мышью на диаграмме, откройте ее.

3. Нажмите кнопку C1ass панели инструментов.

4. Щелкните мышью внутри диаграммы, чтобы поместить туда новый класс.

5. Назовите его *Позиц_заказа*.

6. Назначьте этому классу стереотип Entity.

7. В браузере перетащите класс в пакет *Сущности.*

Добавление атрибутов

1. Щелкните правой кнопкой мыши на классе *Заказ*.

2. В открывшемся меню выберите пункт New Attribute (Создать атрибут),

3. Введите новый атрибут:

OrderNumber : Integer

4. Нажмите клавишу Enter

5. Введите следующий атрибут:

CustomerName : String.

6. Повторив шаги 4 и 5, добавьте атрибуты:

OrderDate : Date

OrderFillDate : Date

Если тип атрибута не появляется в выпадающем списке, то введите его от руки и он далее будет появляться.

7. Щелкните правой кнопкой мыши на классе *Позиц_заказа*.

8. В открывшемся меню выберите пункт New Attribute (Создать атрибут).

9. Введите новый атрибут:

ItemID : Integer.

10. Нажмите клавишу Enter.

11. Введите следующий атрибут:

ItemDescription : String.

Добавление операций к классу *Позиц_заказа*

1. Щелкните правой кнопкой мыши на классе *Позиц_заказа*.

2. В открывшемся меню выберите пункт New Opration (Создать операцию).

3. Введите новую операцию:

Создать()

4. Нажмите клавишу Enter.

5. Введите следующую операцию:

Взять_информацию()

6. Нажмите клавишу Enter.

7. Введите операцию:

Дать_информацию()

Подробное описание операций с помощью диаграммы Классов

1. Щелкнув мышью на классе *Заказ*, выделите его.

2. Щелкните на этом классе еще раз, чтобы переместить курсор внутрь.

3. Отредактируйте операцию *Создать()*, чтобы она выглядела следующим образом:

Создать() : Boo1ean

4. Отредактируйте операцию *Взять_информацию*:

Взять_информацию (OrderNum : Integer, Customer : String, OrderDate : Date, FillDate : Date): Boolean

5. Отредактируйте операцию *Дать_информацию*;

Дать_информацию(): String

Подробное описание операций с помощью браузера

1. Найдите в браузере класс *Позиц_заказа*.

2. Раскройте этот класс, щелкнув на значке "+" рядом с ним. В браузере появятся атрибуты и операции класса.

3. Дважды щелкнув мышью на операции *Дать_информацию()* , откройте окно ее спецификации:

4. В раскрывающемся списке Return class (Возвращаемый класс) укажите String.

5. Щелкнув мышью на кнопке ОК, закройте окно спецификации операции.

6. Дважды щелкните в браузере на операции *Дать_информацию* () класса *Позиц_заказа*, чтобы открыть окно ее спецификации.

7. В раскрывающемся списке Return class укажите Boo1ean.

8. Перейдите на вкладку Detail(Подробно).

9. Щелкните правой кнопкой мыши в области аргументов, чтобы добавить туда новый параметр:

10. В открывшемся меню выберите пункт Insert (Вставить). Rose добавит аргумент под названием argname.

11. Щелкнув один раз на этом слове, выделите его и измените имя аргумента на ID.

12. Щелкните на колонке Type (Тип). В раскрывающемся списке типов выберите Integer (Если этого либо иного необходимого типа не будет- введите его вручную).

13. Щелкните на колонке Default (По умолчанию), чтобы добавить значение аргумента по умолчанию. Введите число 0.

14. Нажав на кнопку ОК, закройте окно спецификации операции.

15. Дважды щелкните на операции *Создать()* класса *Позиц_заказа*, чтобы открыть окно ее спецификации.

16. В раскрывающемся списке Return class укажите Boo1ean.

17. Нажав на кнопку ОК, закройте окно спецификации операции.

Подробное описание операций

1. Используя браузер или диаграмму Классов, введите следующие сигнатуры операций класса *Заказ_деталей*:

Открыть() : Boolean

Сохранить заказ() : Boolean

2. Используя браузер или диаграмму Классов, введите сигнатуру операций класса *Выбор_заказа*:

Создать() : Boo1ean

3. Используя браузер или диаграмму Классов, введите сигнатуру операций класса *Упр_заказами*:

Сохранить заказ(OrderID : Integer) : Boo1ean

4. Используя браузер или диаграмму Классов, введите сигнатуры операций класса *Упр_транзакциями*:

Сохранить заказ(OrderID : Integer) : Boolean

Сохранить информацию() : Integer

Задание 5. Описание связей между классами

В этом упражнении определяются связи между классами, участвующими в варианте использования "Ввести новый заказ".

Постановка задачи

Чтобы найти связи, были просмотрены диаграммы Последовательности. Все взаимодействующие там классы нуждались в определении соответствующих связей на диаграммах Классов. После обнаружения связи были добавлены на диаграммы классов.

Добавление связей

Добавим связи к классам, принимающим участие в варианте использования "Ввести новый заказ".

Этапы выполнения упражнения

Настройка

1. Найдите в браузере диаграмму Классов "Ввод нового заказа",

2. Дважды щелкнув на диаграмме, откройте ее.

3. Проверьте, имеется ли в панели инструментов диаграммы кнопка Unidirectional Association (Однонаправленная ассоциация). Если ее нет, продолжите настройку, выполнив шаги 4 и 5. Если есть, приступайте к выполнению самого упражнения.

4. Щелкните правой кнопкой мыши на панели инструментов диаграммы и в открывшемся меню выберите пункт Customize(Настроить),

5. Добавьте на панель кнопку Creates A Unidirectional Association (Создать однонаправленную ассоциацию).

Добавление ассоциаций

1. Нажмите кнопку Unidirectional Association панели инструментов.

2. Проведите ассоциацию от класса *выбор_заказа* к классу *заказ_деталей*.

3. Повторите шаги 1 и 2, создав ассоциации:

• От класса *заказ_деталей* к классу *упр_заказами*

• От класса *упр_заказами* к классу *Заказ*

• От класса *упр_заказами* к классу *упр_транзакциями*

• От класса *упр_транзакциями* к классу Заказ

• От класса *упр_транзакциями* к классу *Позиц_заказа*

• От класса *Заказ* к классу *Позиц_заказа*

4. Щелкните правой кнопкой мыши на однонаправленной ассоциации между классами *выбор_заказа* и *заказ_деталей* класса *выбор_заказа*.

5. В открывшемся меню выберите пункт Multiplicity > Zero or One (Множественность >- Нуль или один),

6. Щелкните правой кнопкой мыши на другом конце однонаправленной ассоциации.

7. В открывшемся меню выберите пункт Multiplicity > Zero or One (Множественность > Нуль или один),

8. Повторите шаги 4—7, добавив на диаграмму значения множественности для остальных ассоциаций, как показано на рис. 10

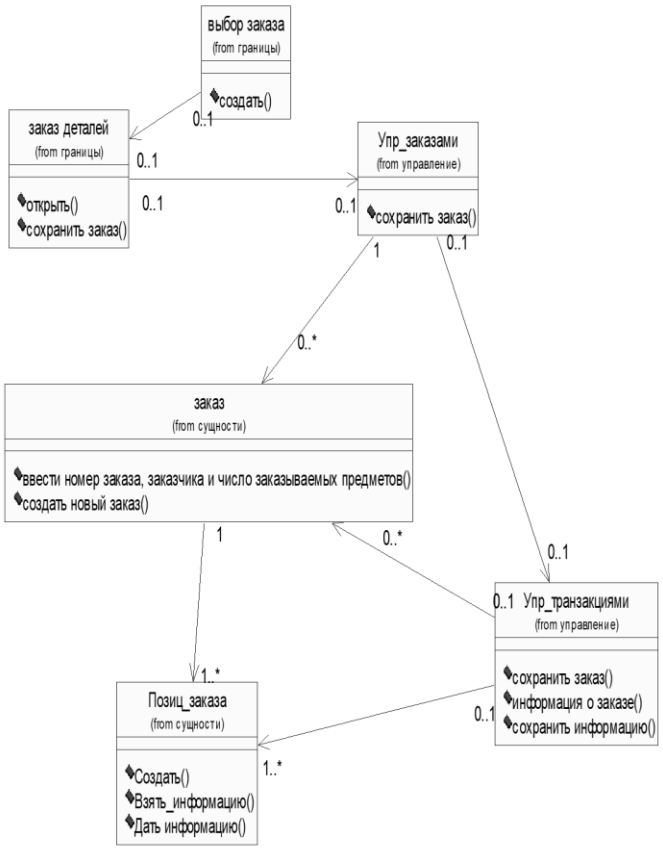

Рис. 10 Ассоциации сценария "Ввести новый заказ"

Задание 6. Исключение кириллизованного текста в информации классов.

Разработанные ранее модели, предназначенные для описания предметной области используют кириллизованную символику, недопустимую для большинства языков программирования. Выполните замену русского текста на латинский шрифт. Для этой цели сохраните предыдущую модель под другим именем и далее работайте с новым файлом (что бы при необходимости можно было бы вернуться к бизнес- процессам, описанным русским шрифтом).

Этапы выполнения

Этап 1. Используя меню (Файл-> Сохранить как) сохраните данную модель под другим именем (например Заказ1) в той же папке, что и исходная модель.

Работайте далее с копией модели (то есть Заказ1).

Этап 2. Переименуйте классы и их спецификации таким образом, чтобы использовался только латинский шрифт. Замените имя класса

Заказ_деталей на *OrderDetail*

Выбор_заказа на *OrderOptions*

Заказ на *Order*

Упр_заказами на *OrderMgr*

Позиц_заказа на *OrderItem*

Упр_транзакциями на *TransactionMgr*

Измените имена операций таким образом, чтобы рис.10 преобразовался в рис. 11. Для этого, измените операцию класса *OrderOptions*

Открыть() на *Open()*

Класса *OrderDetail*

Открыть() на *Open()*

Сохранить заказ() на *Save()*

Класса *Order*

Ввести номер заказа, заказчика и число заказываемых предметов() на *SetInfo()*

Сохранить_заказ() на *Save()*

Класса *OrderMgr*

Сохранить заказ() на *SaveOrder()*

Класса *TransactionMgr*

Сохранить заказ() на *SaveOrder()*

Сохранить информацию о заказе() на *Commit()*

Создать_заказ() на *SubmitInfo()*

Класса *OrderItem*

Создать() на *Create()*

Взять_информацию() на *GetInfo()*

Дать_информацию на *SetInfo()*

Переименуйте имена пакетов

Границы на *Boundaries*

Сущности на *Entity*

Контроль на *Control*

Добавление стереотипов к классам

1. Щелкните правой кнопкой мыши на классе OrderOptions диаграммы.

2. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

3. В поле стереотипа выберите из выпадающего списка слово Boundary (если его нет, то введите).

4. Нажмите на кнопку ОК.

5. Повторив шаги 1—4, свяжите классы OrderDetail со стереотипом Boundary, OrderMgr и TransactionMgr со стереотипом Control, а класс Order и OderItem— со стереотипом Entity.

Теперь диаграмма Классов должна иметь вид, показанный на рис. 11.

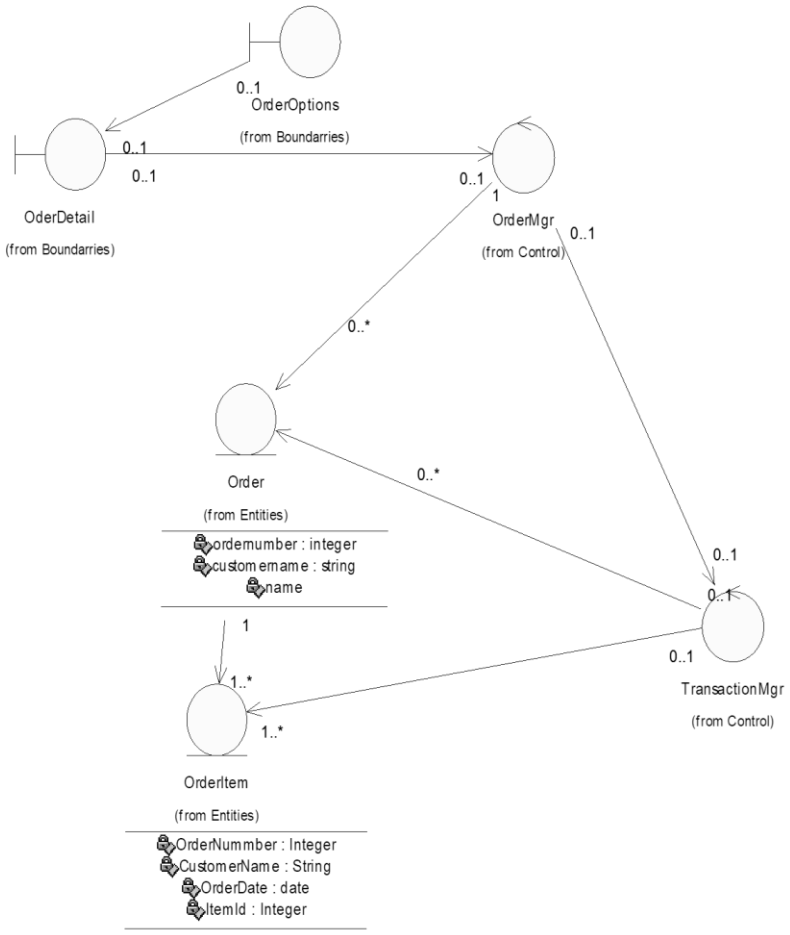

Рис. 11 Основная диаграмма классов

Замечание. На диаграмме рис. 11 возможно визуальное представление классов не в виде иконок, а в виде дополнительной строки текста с именем стереотипа. За этот вид отвечает метка установленная либо на icon либо на label (Class> Open Specefication> Options> Label)

Лабораторная работа №4 «Построение диаграммы компонентов»

Цель работы: исследование процесса построения диаграммы компонентов и в заданной предметной области с помощью пакета Rational Rose

Задание 1. Построение диаграммы Компонентов

Этапы выполнения

Так как эта модель связана с конкретным языком программирования, то в настройках это необходимо отметить. Выполнить Tools>Options>Notations>Default Language и из выпадающего списка языков программирования выбрать Delphi.

Создание пакетов компонентов

1. Щелкните правой кнопкой мыши на представлении компонентов в браузере.
2. В открывшемся меню выберите пункт New > Package (Создать > Пакет}.
3. Назовите пакет Entities (Сущности).
4. Повторив шаги с первого по третий, создайте пакеты Boundaries (Границы) и Control (Управление).

Добавление пакетов на Главную диаграмму Компонентов

1. Откройте Главную диаграмму Компонентов, дважды щелкнув на ней мышью,
2. Перетащите пакеты Entities, Boundary и Control из браузера на Главную диаграмму.

Отображение зависимостей между пакетами

1. Нажмите кнопку Dependency (Зависимость) панели инструментов.
2. Щелкните мышью на пакете Boundary Главной диаграммы Компонентов.
3. Проведите линию зависимости к пакету Control.
4. Повторив шаги 1 — 3, проведите зависимость от пакета Control к пакету Entities.

В результате диаграмма примет вид рис. 12

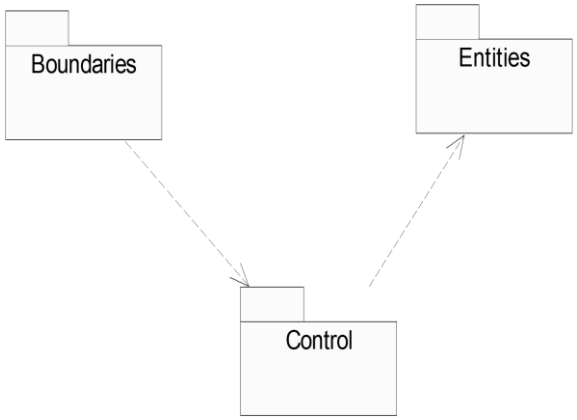

Рис. 12 Зависимости между пакетами

Добавление компонентов к пакетам и отображение зависимостей

1. Дважды щелкнув мышью на пакете Entities Главной диаграммы Компонентов, откройте Главную диаграмму Компонентов этого пакета.
2. Нажмите кнопку Package Specification (Спецификация пакета) панели инструментов.
3. Поместите спецификацию пакета на диаграмму.
4. Введите имя спецификации пакета — *OrderItem_*.
5. Повторив шаги 2—4, добавьте спецификацию пакета *Order_*.
6. Нажмите кнопку Dependency (Зависимость) панели инструментов.
7. Щелкните мышью на спецификации пакета *OrderItem_*.
8. Проведите линию зависимости к спецификации пакета *OrderItem_*.
9. С помощью описанного метода создайте следующие компоненты и зависимости:

Для пакета Boundaries:

• Спецификацию пакета *Orderoptions_*
• Спецификацию пакета *OrderDetail_*

Зависимости в пакете Boundaries:

• От спецификации пакета *Orderoptions_*
к спецификации пакета *OrderDetai_l*

Для пакета Control:

• Спецификацию пакета *OrderMgr_*

• Спецификацию пакета *TransactionMgr_*

Зависимости в пакете Control:

• От спецификации пакета *OrderMgr_*

к спецификации пакета *TransactionMg_*

Создание диаграммы Компонентов системы

1. Щелкните правой кнопкой мыши на представлении Компонентов в браузере.

2. В открывшемся меню выберите пункт New > Component Diagram (Создать > Диаграмма Компонентов).

3. Назовите новую диаграмму System.

4. Дважды щелкните на этой диаграмме мышью.

Размещение компонентов на диаграмме Компонентов системы

1. Разверните в браузере пакет компонентов *Entities,* чтобы открыть его.

2. Щелкните мышью на спецификации пакета *Order_* в пакете компонентов *Entities*.

3. Перетащите эту спецификацию на диаграмму.

4. Повторив шаги 2 и 3, поместите на диаграмму спецификацию пакета *OrderItem_*.

5. С помощью этого метода поместите на диаграмму следующие компоненты:

Из пакета компонентов *Boundaries*:

• Спецификацию пакета *Orderoptions_*

• Спецификацию пакета *OrderDetail_*

Из пакета компонентов Control:

• Спецификацию пакета *OrderMgr_*

• Спецификацию пакета *TransactionMgr_*

6. Нажмите кнопку Task Specification (Спецификация задачи) панели инструментов.

7. Поместите на диаграмму спецификацию задачи и назовите ее *OrderClientExe*.

8. Повторите шаги 6 и 7 для спецификации задачи *OrderServerExe*.

Добавление оставшихся зависимостей на диаграмму Компонентов системы

Уже существующие зависимости будут автоматически показаны на диаграмме Компонентов системы после добавления туда соответствующих компонентов. Теперь нужно добавить остальные зависимости.

1. Нажмите кнопку Dependency (Зависимость) панели инструментов.

2. Щелкните мышью на спецификации пакета *OrderDetail_*

3. Проведите линию зависимости к спецификации пакета *OrderDetail_*

4. Повторив шаги 1 — 3, создайте следующие зависимости:

• От спецификации пакета *OrderMgr_*

к спецификации пакета *Order_*

• От спецификации пакета *TransactionMgr_*

к спецификации пакета *OrderItem_*

• От спецификации пакета *TransactionMgr_*

к спецификации пакета *Order_*

• От спецификации задачи *OrderClientExe* к спецификации пакета *Orderoptions_*

От спецификации задачи *OrderServerExe* к спецификации пакета *OrderMg_r*

Соотнесение классов с компонентами

1. В Логическом представлении браузера найдите класс *Order* пакета *Entities*.

2. Перетащите этот класс на спецификацию пакета компонента *Order_* в представлении Компонентов браузера, В результате класс *Order* будет соотнесен со спецификацией пакета компонента *Order_*.

3. Повторив шаги 1 — 2, соотнесите классы со следующими компонентами:

• Класс *OrderItem* со спецификацией пакета *OrderItem_*

• Класс *Orderoptions* со спецификацией пакета *Orderoptions_*

• Класс *OrderDetail* со спецификацией пакета *OrderDetai_l*

• Класс *OrderMgr* со спецификацией пакета *OrderMgr_*

• Класс *TransactionMgr* со спецификацией пакета *TransactionMgr_*

В результате в браузере после имени класса, в скобках появятся имена компонентов, с которыми этот класс связан (рис. 13)

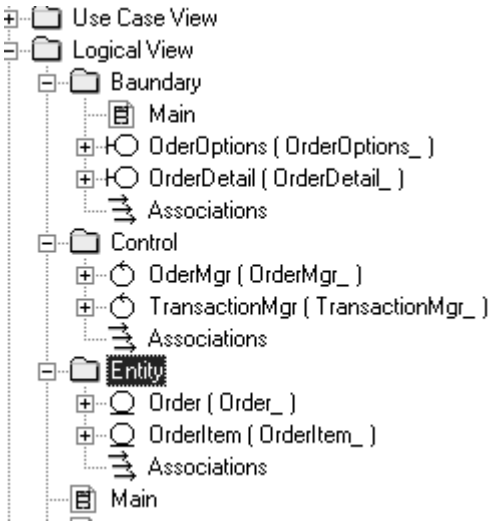

Рис. 13 Представление компонентов и классов в браузере

Лабораторная работа №5 «Построение диаграмм потоков данных»

Цель работы: исследование процесса построения диаграммы потоков данных и диаграммы размещения в заданной предметной области с помощью пакета Rational Rose и Delphi

Задание 1. Кодогенерация проекта в Delphi.

Теперь вся информация подготовлена к тому, чтобы запрограммировать классы с их методами и операциями.

Для выполнения кодогенерации в среде Delphi необходимо выполнить следующую последовательность действий:

- протестировать модель на логические непротиворечия;
- настроить (или проверить настройки) среду на законы кодогенерации (соответствие элемента модели Rose элементу кода Delphi);
- создать имя проекта Delphi и выполнить кодогенерацию.

Этапы выполнения упражнения.

1) Протестируйте модель Tools->Ceck Model. Просмотрите log файл на наличие ошибок. Если файл не виден- выполните команду file->Save Log As и введите имя файла (по умолчанию error.log). Затем его просмотрите и, при необходимости, исправьте ошибки. К наиболее распространенным ошибкам относятся такие, как неотображение сообщений на операции или несоотнесение объектов с классами на диаграммах взаимодействия. С помощью пункта меню Check Model можно выявить большую часть неточностей и ошибок в модели.

2) Пункт меню Access Violations позволяет обнаружить нарушения правил доступа, возникающие тогда, когда существует связь между двумя классами разных пакетов. При этом связи между самими пакетами нет. Например, если существует связь между классами Order пакета Entities и OrderManager пакета Control, то обязательно должна существовать и связь между пакетами Entities и Control. Если последняя связь не установлена, Rose выявит нарушение правил доступа. Чтобы обнаружить нарушение правил доступа:

Выберите в меню Report > Show Access Violations.

Проанализируйте все нарушения правил доступа.

3) Выполните Tools>Options>Notation>Default Language и из выпадающего списка выберите язык программирования Delphi.

4) Проверьте правильность установок кодогенерации по умолчанию (default). Для этого выполните Tools->Options->Delphi и последовательно переберите из выпадающего списка поля Type все элементы. Сравните установки в поле Model Propities с данными (default) из таблицы Приложения А. В случае несоответствия- исправьте.

5) Выполните Tools> Ensemble Tools>Rose Delphi Link (рис.14)

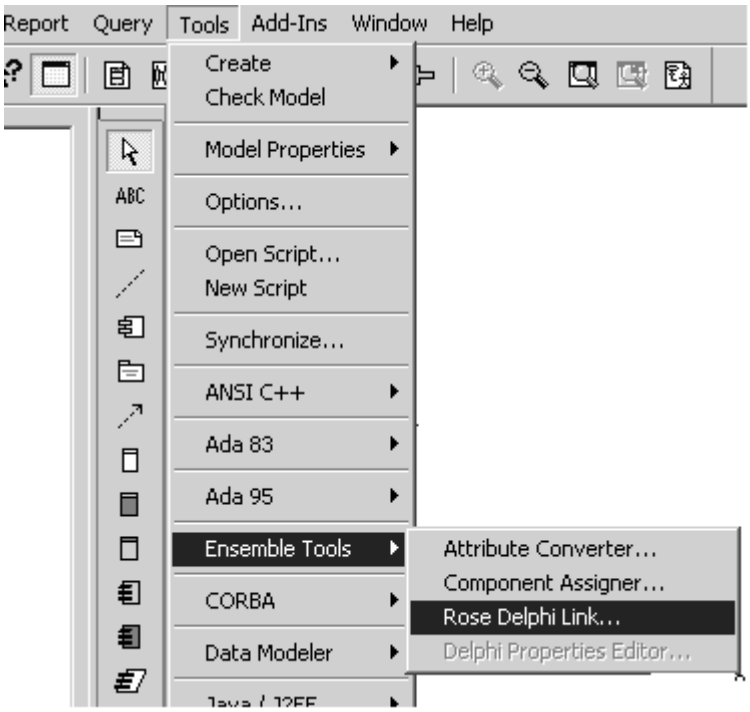

Рис. 14. Меню для выбора процесса кодогенерации

В результате появится соответствующая экранная форма. Выполните на этой форме File>New Proect. Появится форма с браузером. Введите имя файла и место на диске, куда будет сохранено имя сгенерированного проекта в Delphi. Например, NewProect.dpr и нажмите Открыть. В результате форма примет вид (рис. 15)

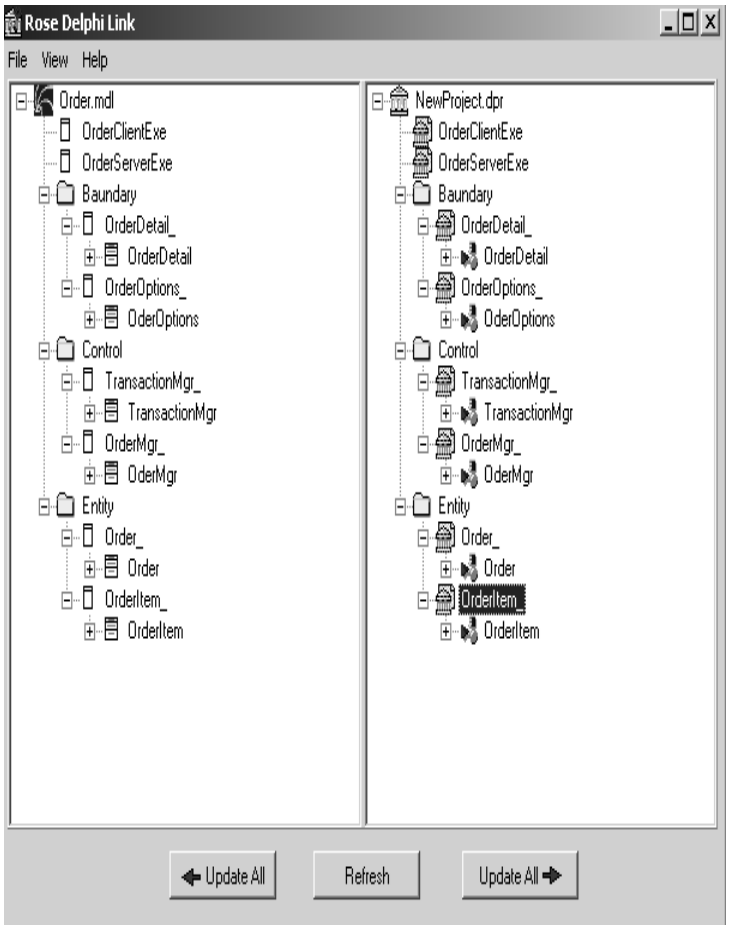

Рис. 15. Представление результатов кодогенерации в окне Rose Delphi Link

5. Через проводник Windows найдите папку проекта Delphi. С помощью программы Блокнот просмотрите содержимое всех файлов. В приложении В к руководству приведено содержимое всех файлов проекта.

Задание 2. Анализ Delphi проекта, добавление визуальных объектов, реинжениринг в Rose

1) Запустите на выполнение программу Delphi и загрузите сгенерированный проект (Proect1.dpr). Проверьте, что проект содержит все модули и присмотрите их содержимое через редактор Delphi.

2) Создайте в проекте Delphi новую форму с Name Form1. Поместите на форму компонент MainMenu (главное меню)

3) С помощью Menu Disigner введите две позиции горизонтального меню с названиями (полями Caption) Oder и OderItem.

4) Для Oder введите две строки вертикального меню с Caption Create и SubmitInfo. Для OderItem введите одну строку вертикального меню с названием GetInfo.

5) Сохраните проект в Delphi.

Реинжениринг Delphi проекта в модель Rose

1) Вернитесь в проект Rose и откройте окно проектов Rose Delphi Link. Проверьте, что открыт именно тот проект, для которого выполнялась кодогенерация.

2) Курсором мыши нажмите клавишу Update ALL со стрелкой влево (обновление модели Rose на основе изменений проекта Delphi). В результате в моделе Rose должны произойти определенные изменения (рис. 16):

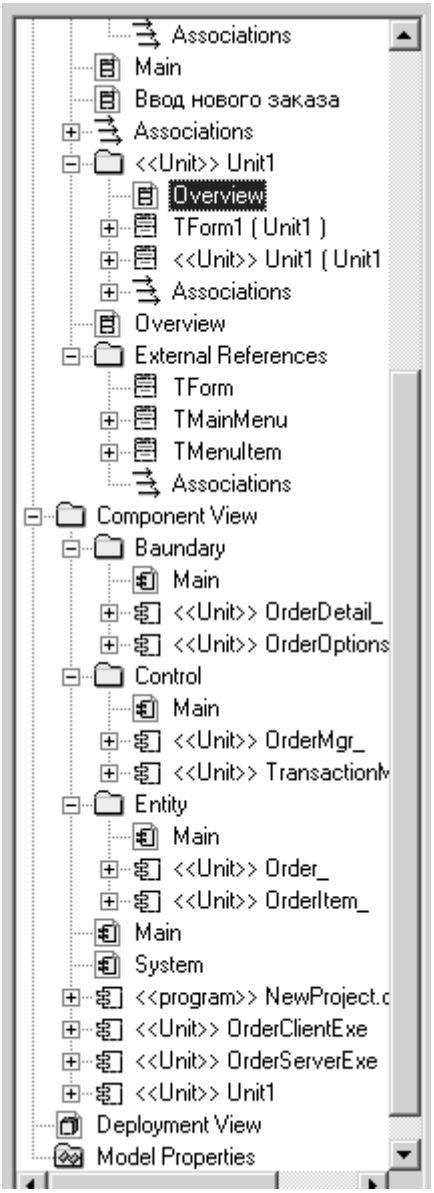

Рис. 16. Окно Rose Delphi Link после кодогенерации

- в представление Logic View создался новый пакет Unit1 и External References (Внешние ссылки). Внутри второго пакета созданы три класса TForm, TMainMenu и TMenuItem, которые использовались при развитии проекта Delphi. Отметим, что эта папка не создалась бы, если бы мы при первоначальном создании проекта включили в него пакет классов Delphi FreimWork.

- в этом же представлении в пакете Unit1 создан класс TForm1 и Unit1 оба соотнесенные с вновь созданным компонентом Unit1. Кроме того, в этом же пакете создалась диаграмма классов Overview, содержимое которой показано на рис.17

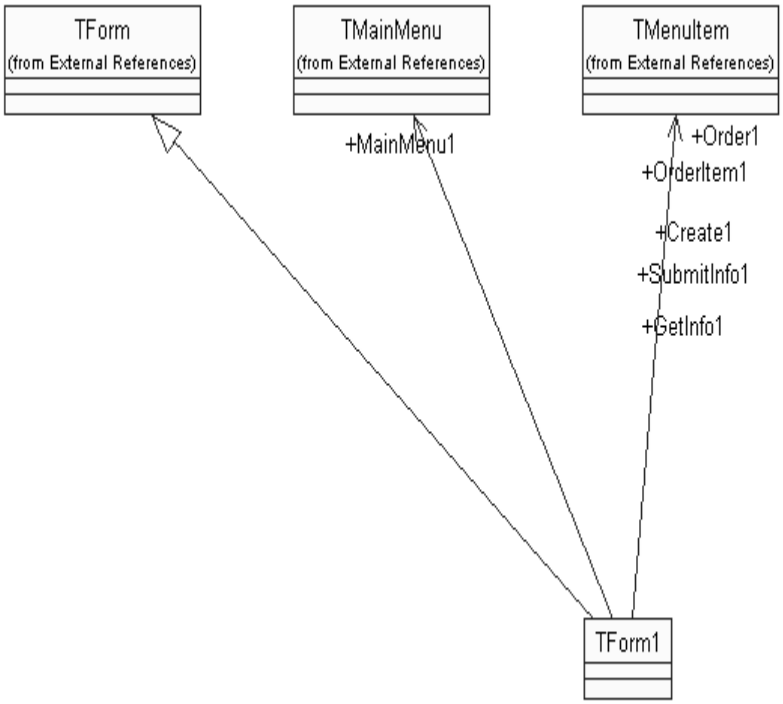

Рис. 17. Результаты реинжиниринга проекта Delphi в Rose

Задание 3. Построение диаграммы размещения

В этом задании создается диаграмма Размещения для системы обработки заказов.

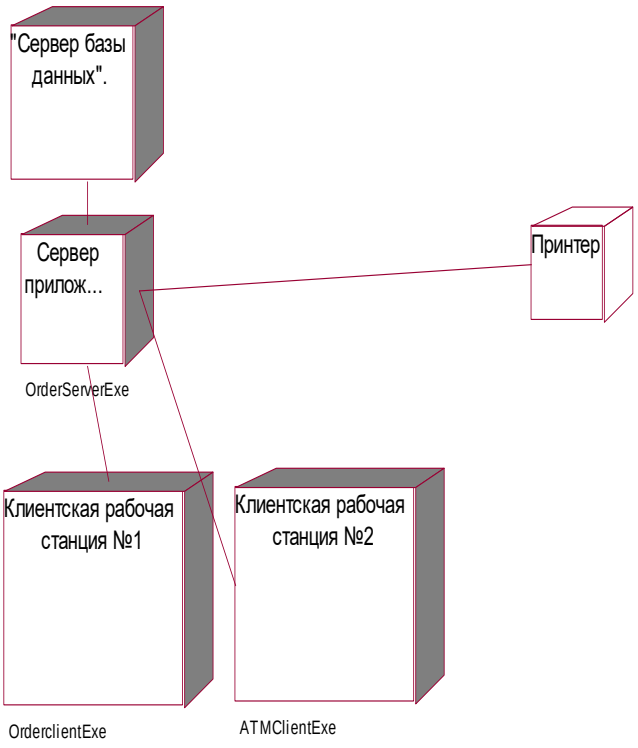

Рис. 18 Диаграмма размещения для модельной задачи

Этапы выполнения

Добавление узлов к диаграмме Размещения

1. Дважды щелкнув мышью на представлении Размещения в браузере, откройте диаграмму Размещения.

2. Нажмите кнопку Processor (Процессор) панели инструментов.

3. Щелкнув мышью на диаграмме, поместите туда процессор.

4. Введите имя процессора "Сервер базы данных".

5. Повторив шаги 2—4, добавьте следующие процессоры:

-Сервер приложения

- Клиентская рабочая станция №1

- Клиентская рабочая станция №2

6. На панели инструментов нажмите кнопку Devices (Устройство).

7. Щелкнув мышью на диаграмме, поместите туда устройство.

8. Назовите его "Принтер".

Добавление связей

1. Нажмите кнопку Connection (Связь) панели инструментов.

2. Щелкните мышью на процессоре "Сервер базы данных".

3. Проведите линию связи к процессору "Сервер приложения".

4. Повторив шаги 1 — 3, добавьте следующие связи;

- От процессора "Сервер приложения" к процессору "Клиентская рабочая станция №1"

- От процессора "Сервер приложения" к процессору "Клиентская рабочая станция №2"

- От процессора "Сервер приложения" к устройству "Принтер"

Добавление процессов

1. Щелкните правой кнопкой мыши на процессоре "Сервер приложения" в браузере.

2. В открывшемся меню выберите пункт New > Process (Создать > Процесс),

3. Введите имя процесса — OrderServerExe.

4. Повторив шаги 1 —- 3, добавьте процессы:

- Процесс OrderclientExe на процессоре "Клиентская рабочая станция №1"

- Процесс ATMClientExe на процессоре "Клиентская рабочая станция №2"

Показ процессов на диаграмме

1. Щелкните правой кнопкой мыши на процессоре "Сервер приложения".

2. В открывшемся меню выберите пункт Show Process (Показать процессы).

3. Повторив шаги 1 и 2, покажите процессы на следующих процессорах:

- Клиентская рабочая станция №1

- Клиентская рабочая станция №2

Лабораторная работа №6 «Разработка тестового сценария»

Цели: усвоить знание о видах тестирования; освоить способы обнаружения и фиксирования ошибок.

Теоретические сведения

Общепринятая практика состоит в том, что после завершения продукта и до передачи его заказчику независимой группой тестировщиков проводится тестирование ПО.

Уровни тестирования:

Модульное тестирование. Тестируется минимально возможный для тестирования компонент, например отдельный класс или функция;

Интеграционное тестирование. Проверяется, есть ли какие-либо проблемы в интерфейсах и взаимодействии между интегрируемыми компонентами, например, не передается информация, передается некорректная информация;

Системное тестирование**.** Тестируется интегрированная система на ее соответствие исходным требованиям.

Таблица 1. Виды некоторых ошибок и способы их обнаружения

Виды программных ошибок	Способы их обнаружения
Ошибки выполнения, выявляемые автоматически: а) переполнение, защита памяти; б) несоответствие типов; в) зацикливание	Динамический контроль: аппаратурой процессора; run-time системы программирования; операционной системой – по превышению лимита времени

Тест – это набор контрольных входных данных совместно с ожидаемыми результатами. Тесты должны обладать определенными свойствами.

Детективность: тест должен с большой вероятностью обнаруживать возможные ошибки.

Покрывающая способность: один тест должен выявлять как можно больше ошибок.

Воспроизводимость: ошибка должна выявляться независимо от изменяющихся условий.

С помощью тестирования разных видов обнаруживаются ошибки в разрабатываемом программном обеспечении. После обнаружения ошибок проводится их устранение.

Задание

Создать приложение Простой калькулятор, в котором реализовать выполнение простых операций с вводимыми двумя операндами. Выполнить тестирование приложения на различных данных, отличающихся по типу и значению.

Программа работы

1. Разработать интерфейс приложения и написать программные коды для событий кнопок.
2. Сохранить проект в отдельной папке, скопировать исполняемый файл на рабочий стол.
3. Составить тесты для проверки работы приложения.
4. Провести тестирование исполняемого файла

Составить отчет по итогам тестирования и рекомендации по устранению выявленных ошибок

Лабораторная работа №7«Оценка необходимого количества тестов»

Цель: получить навыки разработки тестовых сценариев.

Теоретические вопросы

- Оценка стоимости и причины ошибок в программном обеспечении.
- Виды и методы тестирования.
- Понятие теста.
- Требования к разработке тестовых сценариев.
- Правила разработки тестовых сценариев.

Задание № 1

Написать программу решения квадратного уравнения ax2 + bx + c = 0.

Задание № 2

Найти минимальный набор тестов для программы нахождения веще-ственных корней квадратного уравнения ax2 + bx + c = 0. Решение представлено в таблице.

Номер теста	a	b	c	Ожидаемый результат	Что проверяется
1	2	-5	2	x_1=2, x_2=0,5	Случай вещественных корней
2	3	2	5	Сообщение	Случай комплексных корней
3	3	-12	0	x_1=4, x_2=0	Нулевой корень
4	0	0	10	Сообщение	Неразрешимое уравнение
5	0	0	0	Сообщение	Неразрешимое уравнение
6	0	5	17	Сообщение	Неквадратное уравнение
7	9	0	0	x_1=x_2=0	Нулевые корни

Таким образом, для этой программы предлагается минимальный набор функциональных тестов, исходя из 7 классов выходных данных.

Заповеди по отладки программного средства, предложенные Г. Майерсом.

Заповедь 1. Считайте тестирование ключевой задачей разработки ПС, поручайте его самым квалифицированным и одаренным программистам, нежелательно тестировать свою собственную программу.

Заповедь 2. Хорош тот тест, для которого высока вероятность обнаружить ошибку, а не тот, который демонстрирует правильную работу программы.

Заповедь 3. Готовьте тесты как для правильных, так и для неправильных данных.

Заповедь 4. Документируйте пропуск тестов через компьютер, детально изучайте результаты каждого теста, избегайте тестов, пропуск которых нельзя повторить. Заповедь 5. Каждый модуль подключайте к программе только один раз, никогда не изменяйте программу, чтобы облегчить ее тестирование.

Заповедь 6. Пропускайте заново все тесты, связанные с проверкой работы какой-либо программы ПС или ее взаимодействия с другими программами, если в нее были внесены изменения (например, в результате устранения ошибки).

Задание № *3*

Разработайте набор тестовых сценариев (как позитивных, так и негативных) для следующей программы:

Имеется консольное приложение (разработайте самостоятельно). Ему на вход подается 2 строки. На выходе приложение выдает число вхождений второй строки в первую. Например:

Строка 1	Строка 2	Вывод
абвгабвг	аб	2
стстсап	стс	2

Набор тестовых сценариев запишите в виде таблицы, приведенной выше.

Задание № 4

Оформить отчет.

Лабораторные работы №8 «Разработка тестовых пакетов»

Цель: получить навыки разработки тестовых пакетов.

Теоретические вопросы

- Системные основы разработки требований к сложным комплексам программ.
- Формализация эталонов требований и характеристик комплекса программ.
- Формирование требований компонентов и модулей путем декомпозиции функций комплексов программ.
- Тестирование по принципу «белого ящика».

Задание № 1

В Древней Греции (II в. до н.э.) был известен шифр, называемый "квадрат Полибия". Шифровальная таблица представляла собой квадрат с пятью столбцами и пятью строками, которые нумеровались цифрами от 1 до 5. В каждую клетку такого квадрата записывалась одна буква. В результате каждой букве соответствовала пара чисел, и шифрование сводилось к замене буквы парой чисел. Для латинского алфавита квадрат Полибия имеет вид:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I, J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Пользуясь изложенным способом создать программу, которая:

а) зашифрует введенный текст и сохранит его в файл;

б) считает зашифрованный текст из файла и расшифрует данный текст.

Задание № 2

Спроектировать тесты по принципу «белого ящика» для программы, разработанной в задании № 1. Выбрать несколько алгоритмов для тестирования и обозначить буквами или цифрами ветви этих алгоритмов. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования. Записать тесты, которые позволят пройти по путям алгоритма. Протестировать разработанную вами программу. Результаты оформить в виде таблиц:

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
…	…	…	…

Задание № 3

Проверить все виды тестов и сделать выводы об их эффективности

Задание № 4

Оформить отчет.

Лабораторные работы №9 «Оценка программных средств с помощью метрик»

Цель: знакомство с ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения»; определить способы получения информации о ПС; формирование информационно - правовых компетенции обучающихся.

Теоретические сведения

Необходимая документация**:** ГОСТ 28.195-89

Одной из важнейших проблем обеспечения качества программных средств является формализация характеристик качества и методология их оценки. Для определения адекватности качества функционирования, наличия технических возможностей программных средств к взаимодействию, совершенствованию и развитию необходимо использовать стандарты в области оценки характеристик их качества.

Показатели качества программного обеспечения устанавливают ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения» и ГОСТ Р ИСО/МЭК 9126 «Информационная технология. Оценка программной продукции. Характеристика качества и руководства по их применению». Одновременное существование двух действующих стандартов, нормирующих одни и те же показатели, ставит вопрос об их гармонизации. Ниже рассмотрим каждый из перечисленных стандартов.

ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения» устанавливает общие положения по оценке качества программных средств, номенклатуру и применяемость показателей качества.

Оценка качества ПС представляет собой совокупность операций, включающих выбор номенклатуры показателей качества оцениваемого ПС, определение значений этих показателей и сравнение их с базовыми значениями.

Методы определения показателей качества ПС различаются: по способам получения информации о ПС – измерительный, регистрационный, органолептический, расчетный; по источникам получения информации – экспертный, социологический.

Измерительный метод основан на получении информации о свойствах и характеристиках ПС с использованием инструментальных средств. Например, с использованием этого метода определяется объем ПС - число строк исходного текста программ и число строк - комментариев, число операторов и операндов, число исполненных операторов, число ветвей в программе, число точек входа (выхода), время выполнения ветви программы, время реакции и другие показатели.

Регистрационный метод основан на получении информации во время испытаний или функционирования ПС, когда регистрируются и подсчитываются определенные события, например, время и число сбоев и отказов, время передачи управления другим модулям, время начала и окончания работы.

Органолептический метод основан на использовании информации, получаемой в результате анализа восприятия органов чувств (зрения, слуха), и применяется для определения таких показателей как удобство применения, эффективность и т. п.

Расчетный метод основан на использовании теоретических и эмпирических зависимостей (на ранних этапах разработки), статистических данных, накапливаемых при испытаниях, эксплуатации и сопровождении ПС. При помощи расчетного метода определяются длительность и точность вычислений, время реакции, необходимые ресурсы.

Определение значений показателей качества ПС *экспертным методом* осуществляется группой экспертов-специалистов, компетентных в решении данной задачи, на базе их опыта и интуиции. Экспертный метод применяется в случаях, когда задача не может быть решена никаким другим из существующих способов или другие способы являются значительно более трудоемкими. Экспертный метод рекомендуется применять при определении показателей наглядности, полноты и доступности программной документации, легкости освоения, структурности.

Социологические методы основаны на обработке специальных анкет-вопросников.

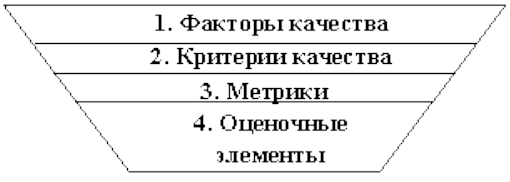

Рис. 1 – Уровни системы показателей качества

Показатели качества объединены в систему из четырех уровней. Каждый вышестоящий уровень содержит в качестве составляющих показатели нижестоящих уровней (рисунок 1).

Стандарт ИСО 9126 (ГОСТ Р ИСО/МЭК 9126) «Информационная технология. Оценка программной продукции. Характеристика качества и руководства по их применению».

Определенные настоящим стандартом характеристики дополнены рядом требований по выбору метрик и их измерению для различных проектов ПС. Они применимы к любому типу ПС, включая компьютерные программы и данные, содержащиеся в программируемом оборудовании. Эти характеристики обеспечивают согласованную терминологию для анализа качества ПС. Кроме того, они определяют схему для выбора и специфицирования требований к качеству ПС, а также для сопоставления возможностей различных программных продуктов, таких как функциональные возможности, надежность, практичность и эффективность.

Все множество атрибутов качества ПС может быть классифицировано в структуру иерархического дерева характеристик и субхарактеристик. Самый высший уровень этой структуры состоит из характеристик качества, а самый нижний уровень – из их атрибутов. Эта иерархия не строгая, поскольку некоторые атрибуты могут быть связаны с более чем одной субхарактеристикой. Таким же образом, внешние свойства (такие, как пригодность, корректность, устойчивость к ошибкам или временная эффективность) влияют на наблюдаемое качество. Недостаток качества в использовании (например, пользователь не может закончить задачу) может быть прослежен к внешнему качеству (например, функциональная пригодность или простота использования) и связанным с ним внутренним атрибутам, которые необходимо изменить.

Внутренние метрики могут применяться в ходе проектирования и программирования к неисполняемым компонентам ПС (таким, как спецификация или исходный программный текст). При разработке ПС промежуточные продукты следует оценивать с использованием внутренних метрик, которые измеряют свойства программ, и могут быть выведены из моделируемого поведения. Основная цель внутренних метрик – обеспечивать, чтобы было достигнуто требуемое внешнее качество. Внутренние метрики дают возможность пользователям, испытателям и разработчикам оценивать качество ЖЦ программ и заниматься вопросами технологического обеспечения качества задолго до того, как ПС становится готовым исполняемым продуктом.

Внутренние метрики позволяют измерять внутренние атрибуты или формировать признаки внешних атрибутов путем анализа статических свойств промежуточных или поставляемых программных компонентов. Измерения внутренних метрик используют категории, числа или характеристики элементов из состава ПС, которые, например, имеются в процедурах исходного программного текста, в графе потока управления, в потоке данных и в представлениях изменения состояний памяти. Документация также может оцениваться с использованием внутренних метрик.

Внешние метрики используют меры ПС, выведенные из поведения системы, частью которых они являются, путем испытаний, эксплуатации или наблюдения исполняемого ПС или системы. Перед приобретением или использованием ПС его следует оценить с использованием метрик, основанных на деловых и профессиональных целях, связанных с использованием, эксплуатацией и управлением продуктом в определенной организационной и технической среде. Внешние метрики обеспечивают заказчикам, пользователям, испытателям и разработчикам возможность определять качество ПС в ходе испытаний или эксплуатации.

Когда требования к качеству ПС определены, в них должны быть перечислены характеристики и субхарактеристики, которые составляют полный набор показателей качества. Затем определяются подходящие внешние метрики и их приемлемые диапазоны значений, устанавливающие количественные и качественные критерии, которые подтверждают, что ПС удовлетворяет потребностям заказчика и пользователя. Далее определяются и специфицируются внутренние атрибуты качества, чтобы спланировать удовлетворение требуемых внешних характеристик качества в конечном продукте и обеспечивать их в промежуточных продуктах в ходе разработки. Подходящие внутренние метрики и приемлемые диапазоны специфицируются для получения числовых значений или категорий внутренних характеристик качества, чтобы их можно было использовать для проверки того, что промежуточные продукты в процессе разработки удовлетворяют внутренним спецификациям качества. Рекомендуется использовать внутренние метрики, которые имеют наиболее сильные связи с целевыми внешними метриками, чтобы они могли помогать при прогнозировании значений внешних метрик.

Метрики качества в использовании измеряют, в какой степени продукт удовлетворяет потребности конкретных пользователей в достижении заданных целей с результативностью,

продуктивностью и удовлетворением в заданном контексте использования. При этом результативность подразумевает точность и полноту достижения определенных целей пользователями при применении ПС; продуктивность соответствует соотношению израсходованных ресурсов и результативности при эксплуатации ПС, а удовлетворенность – психологическое отношение к качеству использования продукта. Эта метрика не входит в число шести базовых характеристик ПС, регламентируемых стандартом ИСО 9126, однако рекомендуется для интегральной оценки результатов функционирования комплексов программ.

Оценивание качества в использовании должно подтверждать его для определенных сценариев и задач, оно составляет полный объединенный эффект характеристик качества ПС для пользователя. *Качество в использовании* – это восприятие пользователем качества системы, содержащей ПС, и оно измеряется скорее в терминах результатов использования комплекса программ, чем собственных внутренних свойств ПС. Связь качества в использовании с другими характеристиками качества ПС зависит от типа пользователя, так, например, для конечного пользователя качество в использовании обусловливают, в основном, характеристики функциональных возможностей, надежности, практичности и эффективности, а для персонала сопровождения ПС качество в использовании определяет сопровождаемость. На качество в использовании могут влиять любые характеристики качества, и это понятие шире, чем практичность, которая связана с простотой использования и привлекательностью. Качество в использовании, в той или иной степени, характеризуется сложностью применения комплекса программ, которую можно описать трудоемкостью использования с требуемой результативностью. Многие характеристики и субхарактеристики ПС обобщенно отражаются неявными технико-экономическими показателями, которые поддерживают функциональную пригодность конкретного ПС. Однако их измерение и оценка влияния на показатели качества, представляет сложную проблему.

Задание №1. Провести сравнение понятий «качество» государственным и международным стандартами. Выписать документы, в которых даны данные определения.

Задание №2. Опишите методы получения информации о ПС по ГОСТу. Для каждого метода выделите источник информации.

Задание №3. Выберите стандарты для оценки качества ПС. Перечислите критерии надежности ПС по ГОСТу.

Задание №4. Методика оценки качественных показателей ПП основана на составлении метрики ПП. В лабораторной работе необходимо выполнить следующее:

1. Выбрать показатели качества (не менее 5) и сформулировать их сущность. Каждый показатель должен быть существенным, т. е. должны быть ясны потенциальные выгоды его использования. Показатели представить в виде таблицы (таблица 1).

Показатели качества	Сущность показателя	Экспертная оценка (вес) wi	Оценка, установленная экспериментом ri

2. Установить веса показателей wi (?wi =1).

3. Для каждого показателя установить конкретную численную оценку ri от 0 до 1, исходя из следующего:

§ 0 – свойство в ПП присутствует, но качество его неприемлемо;

§ 0.5 — 1 – свойство в ПП присутствует и обладает приемлемым качеством;

§ 1 – свойство в ПП присутствует и обладает очень высоким качеством.

§ Возможно, присвоение промежуточных значений в соответствии с мнением оценивающего лица относительно полезности того или иного свойства ПП.

$$K = \frac{\sum w_i \cdot r_i}{общее\_количество\_показателей}$$

Результатом выполнения данной работы является отчет

Лабораторные работы №10 «Инспекция программного кода на предмет соответствия стандартам кодирования»

Цель: научиться выполнять реорганизацию программного кода на основании шаблонов рефакторинга.

Задание №1

Выполнить анализ программного кода для разрабатываемого ПО и модульных тестов с целью плохо организованного кода.

Задание №2

Используя шаблоны рефакторинга, выполнить реорганизацию программного кода разрабатываемого ПО.

Задание №3

Выполнить описание произведенных операций рефакторинга (было-стало-шаблон рефакторинга).

Задание №4

В случае необходимости скорректировать проектную документацию.

Задание №5

Сделать выводы по результатам выполнения работ.

Класс Main

Было:

```
пакетная игра;
импорт игры. Персонажи. *;
импорт игры. Персонажи. Персонажи;
импорт игры. Энергетика. Энергетика;
импорт игры. Энергетика. Освещение;
импорт игры. Уровни. Блок;
импорт игры. Уровни. Уровень;
импортировать game.Levels.Level_data;
импорт игры. Weapon.Bullet;
импорт игры. Оружие. Оружие;
import javafx.animation.AnimationTimer;
импорт javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
import java.io.DataInputStream;
import java.io.FileInputStream;
импорт java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
публичный класс Main расширяет приложение {
public static ArrayList <Block> blocks = new ArrayList <> ();
public static ArrayList <Bullet> bullets = new ArrayList <> ();
public static ArrayList <Bullet> врагаBullets = новый ArrayList <> ();
public static ArrayList <EnemyBase> враги = новый ArrayList <> ();
static HashMap <KeyCode, Boolean> keys = new HashMap <> ();
публичная статическая сцена этапа;
публичная статическая сцена;
public static Pane gameRoot = new Pane ();
public static Pane appRoot = new Pane ();
публичное статическое меню;
публичный статический Персонаж букера;
публичная статическая HUD HUD;
статическое оружие общего назначения;
публичная статика елизавета елизавета;
статический VendingMachine vendingMachine;
статическое учебное пособие;
```

```
частные статические CutScenes cutScene;
публичная статика Энергетика энергетика;
публичная статическая молния молнии;
public static int levelNumber;
уровень статического уровня;
public static AnimationTimer timer = new AnimationTimer () {
@Override
public void handle (давно) {
Обновить();
}
};
приватное статическое void update () {
для (EnemyBase враг: враги) {
enemy.update ();
if (врага.getDelete ()) {
enemies.remove (враг);
перемена;
}
}
Bullet.update ();
Controller.update ();
booker.update ();
if (! energetic.getName (). equals (""))
energetic.update ();
if (levelNumber> 0)
elizabeth.update ();
если (молния! = ноль) {
lightning.update ();
if (lightning.getDelete ())
молния = ноль;
}
menu.update ();
hud.update ();
weapon.update ();
if (booker.getTranslateX ()> Level_data.BLOCK_SIZE * 295)
cutScene = новые CutScenes (levelNumber);
}
@Override
public void start (Stage primaryStage) выдает исключение {
stage = primaryStage;
сцена = новая сцена (appRoot, 1280, 720);
. AppRoot.getChildren () добавить (gameRoot);
уровень = новый уровень ();
try (DataInputStream dataInputStream = new DataInputStream (новый FileInputStream ("C:
/DeadShock/saves/data.dat"))) {
levelNumber = dataInputStream.readInt ();
level.createLevels (levelNumber);
level.changeImageView (levelNumber);
vendingMachine = new VendingMachine ();
букер = новый персонаж ();
booker.setMoney (dataInputStream.readInt ());
booker.setSalt (dataInputStream.readByte ());
booker.setCountLives (2);
оружие = новое оружие ();
weapon.setWeaponClip (dataInputStream.readInt ());
weapon.setBullets (dataInputStream.readInt ());
hud = новый HUD ();
```

```
Елизавета = новая Елизавета ();
энергичный = новый Энергетический ();
} catch (IOException e) {
levelNumber = 0;
level.createLevels (levelNumber);
vendingMachine = new VendingMachine ();
букер = новый персонаж ();
оружие = новое оружие ();
hud = новый HUD ();
энергичный = новый Энергетический ();
tutorial = new Tutorial (levelNumber);
}
switch (levelNumber) {
случай 0:
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 127, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 148, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 161, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 171, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 185, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 204, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 215, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 228, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 233, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 243, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 252, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 262, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 280, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 286, 200));
перемена;
Дело 1:
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 57, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 67, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 74, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 87, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 104, 150));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 133, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 156, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 177, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 193, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 201, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 216, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 224, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 246, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 260, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 100));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 34,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 36,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 60,
Level_data.BLOCK_SIZE * 9));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 61,
Level_data.BLOCK_SIZE * 9));
```

```
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 106,
Level_data.BLOCK_SIZE * 7));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 107,
Level_data.BLOCK_SIZE * 7));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 168,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 170,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 196,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 197,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 232,
Level_data.BLOCK_SIZE * 8));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 233,
Level_data.BLOCK_SIZE * 8));
перемена;
}
меню = новое меню ();
appRoot.getChildren () добавить (menu.menuBox).
booker.translateXProperty (). addListener (((наблюдаемый, oldValue, newValue) -> {
int offset = newValue.intValue ();
if (offset> 600 && offset <gameRoot.getWidth () - 680) {
gameRoot.setLayoutX (- (смещение - 600));
level.getBackground (). setLayoutX ((смещение - 600) / 1,5);
}
если (смещение <= 100)
level.getBackground () setLayoutX (0).
}));
vendingMachine.createButtons ();
stage.getIcons (). add (новое изображение ("файл: / C: /DeadShock/images/icon.jpg"));
stage.setTitle ( "DeadShock");
stage.setResizable (ложь);
stage.setWidth (scene.getWidth ());
stage.setHeight (scene.getHeight ());
stage.setScene (сцены);
stage.show ();
timer.start ();
}
public static void main (String [] args) {
запуск (арг);
}
}
```

Стало:

```
пакетная игра;
импорт игры. Персонажи. *;
импорт игры. Персонажи. Персонажи;
импорт игры. Энергетика. Энергетика;
импорт игры. Энергетика. Освещение;
импорт игры. Уровни. Блок;
импорт игры. Уровни. Уровень;
импортировать game.Levels.Level_data;
импорт игры. Weapon.Bullet;
импорт игры. Оружие. Оружие;
import javafx.animation.AnimationTimer;
импорт javafx.application.Application;
import javafx.scene.Scene;
```

```
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
import java.io.DataInputStream;
import java.io.FileInputStream;
импорт java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
публичный класс Main расширяет приложение {
public static ArrayList <Block> blocks = new ArrayList <> ();
public static ArrayList <Bullet> bullets = new ArrayList <> ();
public static ArrayList <Bullet> врагаBullets = новый ArrayList <> ();
public static ArrayList <EnemyBase> враги = новый ArrayList <> ();
static HashMap <KeyCode, Boolean> keys = new HashMap <> ();
публичная статическая сцена этапа;
публичная статическая сцена;
public static Pane gameRoot = new Pane ();
public static Pane appRoot = new Pane ();
публичное статическое меню;
публичный статический Персонаж букера;
публичная статическая HUD HUD;
статическое оружие общего назначения;
публичная статика елизавета елизавета;
статический VendingMachine vendingMachine;
статическое учебное пособие;
частные статические CutScenes cutScene;
публичная статика Энергетика энергетика;
публичная статическая молния молнии;
public static int levelNumber;
уровень статического уровня;
public static AnimationTimer timer = new AnimationTimer () {
@Override
public void handle (давно) {
Обновить();
}
};
private void initContent () {
. AppRoot.getChildren () добавить (gameRoot);
уровень = новый уровень ();
try (DataInputStream dataInputStream = new DataInputStream (новый FileInputStream ("C:
/DeadShock/saves/data.dat"))) {
levelNumber = dataInputStream.readInt ();
level.createLevels (levelNumber);
level.changeImageView (levelNumber);
vendingMachine = new VendingMachine ();
букер = новый персонаж ();
booker.setMoney (dataInputStream.readInt ());
booker.setSalt (dataInputStream.readByte ());
booker.setCountLives (2);
оружие = новое оружие ();
weapon.setWeaponClip (dataInputStream.readInt ());
weapon.setBullets (dataInputStream.readInt ());
hud = новый HUD ();
Елизавета = новая Елизавета ();
энергичный = новый Энергетический ();
} catch (IOException e) {
```

```
levelNumber = 0;
level.createLevels (levelNumber);
vendingMachine = new VendingMachine ();
букер = новый персонаж ();
оружие = новое оружие ();
hud = новый HUD ();
энергичный = новый Энергетический ();
tutorial = new Tutorial (levelNumber);
}
createEnemies ();
меню = новое меню ();
appRoot.getChildren () добавить (menu.menuBox).
booker.translateXProperty (). addListener (((наблюдаемый, oldValue, newValue) -> {
int offset = newValue.intValue ();
if (offset> 600 && offset <gameRoot.getWidth () - 680) {
gameRoot.setLayoutX (- (смещение - 600));
level.getBackground (). setLayoutX ((смещение - 600) / 1,5);
}
если (смещение <= 100)
level.getBackground () setLayoutX (0).
}));
vendingMachine.createButtons ();
}
public static void createEnemies () {
switch (levelNumber) {
случай 0:
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 127, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 148, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 161, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 171, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 185, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 204, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 215, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 228, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 233, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 243, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 252, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 262, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 280, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 286, 200));
перемена;
Дело 1:
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 57, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 67, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 74, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 87, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 104, 150));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 133, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 156, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 177, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 193, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 201, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 216, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 224, 200));
```

```
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 246, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 260, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 100));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 34,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 36,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 60,
Level_data.BLOCK_SIZE * 9));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 61,
Level_data.BLOCK_SIZE * 9));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 106,
Level_data.BLOCK_SIZE * 7));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 107,
Level_data.BLOCK_SIZE * 7));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 168,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 170,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 196,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 197,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 232,
Level_data.BLOCK_SIZE * 8));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 233,
Level_data.BLOCK_SIZE * 8));
перемена;
}
}
приватное статическое void update () {
для (EnemyBase враг: враги) {
enemy.update ();
If (enemy.GetDelete()) {
enemies.remove(enemy);
break;
}
}
Bullet.update();
Controller.update();
booker.update();
If (!energetic.GetName().Equals(""))
energetic.update();
if (levelNumber > 0)
elizabeth.update();
if (lightning != null) {
lightning.update();
If (lightning.GetDelete())
lightning = null;
}
menu.update();
hud.update();
weapon.update();
if (booker.getTranslateX() > Level_data.BLOCK_SIZE * 295)
cutScene = new CutScenes(levelNumber);
}
@Override
```

```
public void start(Stage primaryStage) throws Exception {
stage = primaryStage;
scene = new Scene(appRoot, 1280, 720);
initContent();
stage.getIcons().add(new Image("file:/C:/DeadShock/images/icon.jpg"));
stage.setTitle("DeadShock");
stage.setResizable(false);
stage.setWidth(scene.getWidth());
stage.setHeight(scene.getHeight());
stage.setScene(scene);
stage.show();
timer.start();
}
public static void main(String[] args) {
launch(args);
}
}
```

Шаблон рефакторинга: Выделение метода(Extract Method)

Список литературы

Основные источники

1. Гагарина, Л. Г. Разработка и эксплуатация автоматизированных информационных систем: учебное пособие / Л. Г. Гагарина. — Москва: ФОРУМ: ИНФРА-М, 2021. — 384 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0735-1. - Текст: электронный. - URL: https://znanium.com/catalog/product/1214882 (дата обращения: 21.05.2021). – Режим доступа: по подписке.
2. Гагарина, Л. Г. Технология разработки программного обеспечения: учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул; под ред. Л.Г. Гагариной. — Москва: ФОРУМ: ИНФРА-М, 2021. — 400 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0812-9. - Текст: электронный. - URL: https://znanium.com/catalog/product/1189951 (дата обращения: 21.05.2021). – Режим доступа: по подписке.
3. Исаченко, О. В. Программное обеспечение компьютерных сетей: учебное пособие / О.В. Исаченко. — 2-е изд., испр. и доп. — Москва: ИНФРА-М, 2020. — 158 с. — (Среднее профессиональное образование). - ISBN 978-5-16-015447-3. - Текст: электронный. - URL: https://znanium.com/catalog/product/1033087 (дата обращения: 21.05.2021). – Режим доступа: по подписке.
4. Математическое моделирование технических систем [Электронный ресурс]: учебник / В.П. Тарасик. — Минск: Новое знание; Москва: ИНФРА-М, 2019. — 592 с. - Режим доступа: http://znanium.com/catalog/product/1019246

Дополнительные источники

1. Гниденко, И. Г. Технология разработки программного обеспечения: учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва: Издательство Юрайт, 2021. — 235 с. — (Профессиональное образование). — ISBN 978-5-534-05047-9. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/472502 (дата обращения: 21.05.2021).
2. Лисьев, Г. А. Программное обеспечение компьютерных сетей и web-серверов: учеб. пособие / Г. А. Лисьев, П. Ю. Романов, Ю. И. Аскерко. — Москва: ИНФРА-М, 2020. — 145 с. — (Среднее профессиональное образование). - ISBN 978-5-16-014514-3. - Текст: электронный. - URL: https://znanium.com/catalog/product/1069176 (дата обращения: 21.05.2021). – Режим доступа: по подписке.
3. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: учебное пособие / Г. Н. Федорова. — Москва: КУРС: ИНФРА-М, 2021. — 336 с. — (Среднее профессиональное образование). - ISBN 978-5-906818-41-6. - Текст: электронный. - URL: https://znanium.com/catalog/product/1138896 (дата обращения: 21.05.2021). – Режим доступа: по подписке.

Интернет-источники

1. Электронная библиотечная система Znanium: сайт.- URL: https://znanium.com/ – Текст: электронный.
2. Электронная библиотечная система Юрайт: сайт. - URL: https://urait.ru/ -Текс: электронный.