МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Т. Ф. ГОРБАЧЕВА»
Филиал КузГТУ в г. Белово

Кафедра информационных технологий и гуманитарных дисциплин

# МДК 02.01 ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические указания к лабораторно-практическим работам
для обучающихся очной формы обучения

09.02.07 «Информационные системы и программирование»

Составитель Белугина С.В.

Рассмотрены и утверждены
на заседании кафедры
Протокол № 10 от 14.06.2022
Рекомендованы для использования в
образовательном процессе
учебно-методической комиссией по
специальности 09.02.07 «Информационные
системы и программирование»
Протокол № 7 от 21.06.2022

Белово 2022

Составил:

Ст. преподаватель кафедры информационных технологий и гуманитарных дисциплин
должность наименование кафедры

_____ Белугина С.В.
подпись инициалы, фамилия

Обсуждено на заседании кафедры
информационных технологий и гуманитарных дисциплин
наименование кафедры

Протокол № 10 от 14.06.2022

Зав. кафедрой информационных технологий и гуманитарных дисциплин
наименование кафедры

_____ Верчагина И.Ю
подпись инициалы, фамилия

Согласовано учебно-методической комиссией
по направлению подготовки (специальности)

09.02.07 «Информационные системы и программирование»

Председатель учебно-методической комиссии по
направлению специальности 09.02.07

_____ Макарчук Р.С.
Подпись инициалы, фамилия

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Профессиональный модуль ПМ.02 «Осуществление интеграции программных модулей» является частью профессионального цикла основной образовательной программы в соответствии с ФГОС по специальности 09.02.07 «Информационные системы и программирование». ПМ.02 «Осуществление интеграции программных модулей» обеспечивает формирование профессиональных и общих компетенций по всем видам деятельности ФГОС по специальности 09.02.07 «Информационные системы и программирование».

Целью освоения междисциплинарного курса МДК 02.01 «Технологии разработки программного обеспечения» является приобретение обучающимися знаний в области формирования требований к ИС и тестирования программного обеспечения (ПО).

Основными задачами изучения МДК 02.01 «Технологии разработки программного обеспечения», являются:

1. Изучение основных принципов процесса разработки программного обеспечения.

2. Изучение встроенных и основных специализированные инструментов анализа качества программных продуктов.

3. Использование специализированных графических средств построения и анализа архитектуры программных продуктов.

4. Умение выполнять ручное и автоматизированное тестирование программного модуля.

МДК 02.01 содержит три раздела. Сборник состоит из введения, описания четырех практических занятий и десяти лабораторных работ, списка используемой литературы.

Общее количество часов на практические работы – 40.

Методические указания к лабораторно-практическим работам соответствуют разделам и темам рабочей программы, содержат задания, технологию выполнения работы, вопросы для самоконтроля изученного материала.

Обучающиеся имеют возможность видеть конечную цель своего труда, планировать уровень его сложности и прогнозировать результат.

Практические работы проводятся в порядке изучения программного материала. Для более эффективного выполнения практических работ необходимо заранее повторить соответствующий теоретический материал по учебным пособиям, на занятии тщательно ознакомиться с содержанием работы и программным обеспечением компьютера.

## ОБЩИЕ МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ И УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ И ЛАБОРАТОРНЫХ РАБОТ

### 1.1 Подготовка к лабораторно-практическим работам

1. Внимательно ознакомиться с описанием соответствующей практической или лабораторной работы и установить, в чем состоит ее основная цель и задача;
2. По лекционному курсу и соответствующим литературным источникам изучить теоретическую часть, относящуюся к данному практическому занятию.

### 1.2 Выполнение лабораторно-практических работ

Успешное выполнение работы может быть достигнуто в том случае, если обучаемый представляет себе цель выполнения лабораторной или практической работы, поэтому важным условием является тщательная подготовка к лабораторно-практической работе.

### 1.3 Как пользоваться методическими указаниями

- внимательно прочтите текст задания, который часто представляет собой алгоритм будущих технологических действий; в каждом новом задании для наглядности описана технология выполнения для достижения поставленной в задании цели.

– по окончании выполнения предложенных заданий необходимо ответить на вопросы самоконтроля в конце каждой практической работы.

**1.4 Оформление отчетов по лабораторно-практическим работам**

Требования к структуре и оформлению отчета по лабораторно-практическим работам:

- отчет по лабораторно-практическим работам состоит из титульного листа, отчетов по выполненным работам
- каждая работа содержит цель, ход работы, ответы на вопросы самоконтроля и вывод о проделанной работе;
- отчет выполняют, руководствуясь следующими положениями:
  - отчет выполнять на листах формата А4;
  - записать на первом листе цель и постановку задачи;
  - оформлять работу шрифтом Times New Roman.

Форма отчетности – письменный отчет о лабораторной или практической работе на электронном носителе с устной защитой преподавателю.

## ПЕРЕЧЕНЬ ЛАБОРАТОРНО- ПРАКТИЧЕСКИХ РАБОТ

| № темы | № работы | Темы | Кол-во часов |
|---|---|---|---|
| Тема 2.1.1 | 1 | Практическое занятие «Анализ предметной области» | 2 |
| | 2 | Практическое занятие «Разработка и оформление технического задания». | 2 |
| | 3 | Практическое занятие «Построение архитектуры программного средства» | 2 |
| | 4 | Практическое занятие «Изучение работы в системе контроля версий» | 4 |
| Тема 2.1.2 | 1 | Лабораторная работа «Построение диаграммы Вариантов использования и диаграммы последовательности» | 4 |
| | 2 | Лабораторная работа «Построение диаграммы Кооперации и диаграммы развертывания» | 4 |
| | 3 | Лабораторная работа «Построение диаграммы Деятельности, диаграммы состояний и диаграммы классов» | 4 |
| | 4 | Лабораторная работа «Построение диаграммы компонентов» | 4 |
| | 5 | Лабораторная работа «Построение диаграмм потоков данных» | 4 |
| Тема 2.1.3 | 6 | Лабораторная работа «Разработка тестового сценария» | 2 |
| | 7 | Лабораторная работа «Оценка необходимого количества тестов | 2 |
| | 8 | Лабораторные работы «Разработка тестовых пакетов» | 2 |
| | 9 | Лабораторная работа «Оценка программных средств с помощью метрик». | 2 |
| | 10 | Лабораторная работа «Инспекция программного кода на предмет соответствия стандартам кодирования» | 2 |
| | | **Итого** | 40 |

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 1 (2 часа)

**Тема:** Анализ предметной области

**Цели:** Изучить методы анализа предметной области

**Технология выполнения:**

1. Ознакомиться с предложенным вариантом описания предметной области
2. Проанализировать предметную область, уточнив и дополнив ее, руководствуясь собственным опытом, консультациями и любыми источниками (книгами, учебниками, интернет-источниками).
3. Выполнить структурное разбиение предметной области на отдельные подразделения (подсистемы) согласно выполняемым ими функциям.
4. Определить задачи и функции системы в целом и функции каждого подразделения (подсистемы).
5. Продумать подробное описание работы каждого подразделения (подсистемы), алгоритмов и сценариев выполнения ими отдельных работ. Продумать виды входной и выходной информации для каждого подразделения (подсистемы).
6. Описать схему работы будущей информационной системы, учитывая описанные ранее подсистемы.

**Контрольные вопросы**

1. Что такое «требование к информационной системе»?
2. Кто занимается выявлением требований к ИС?
3. Перечислите этапы формулировки потребностей

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2 (2 часа)

**Тема:** Разработка и оформление технического задания

**Цели:** Изучить порядок разработки и оформления технического задания

**Задание** – Составить техническое задание предложенного варианта предметной области

**Технология выполнения работы**

При разработке технического задания необходимо решить **следующие задачи:**

- установить общую цель создания ИС, определить состав подсистем и функциональных задач;
- разработать и обосновать требования, предъявляемые к подсистемам;
- разработать и обосновать требования, предъявляемые к информационной базе, математическому и программному обеспечению, комплексу технических средств (включая средства связи и передачи данных);
- установить общие требования к проектируемой системе;
- определить перечень задач создания системы и исполнителей;
- определить этапы создания системы и сроки их выполнения;
- провести предварительный расчет затрат на создание системы и определить уровень экономической эффективности ее внедрения.

**Результат** - правильно оформленное техническое задание.

**Контрольные вопросы**

1. Что такое техническое задание?
2. Какой ГОСТ регламентирует содержание технического задания?
3. Какие пункты должно содержать техническое задание?

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 3 (2 часа)

**Тема:** Построение архитектуры программного средства

**Цель:** Изучить порядок проектирования архитектуры ПО, построения структурных и функциональных схем ПО

**Задание:** на основании технического задания к программному обеспечению в практическом занятии № 2 разработать структурные и функциональные схемы программного

обеспечения по своему варианту задания. Оформить результаты, используя MS Visio или встроенный векторный редактор MSWord.

**Способ 1.** Создание блок-схемы в MS Visio

**Технология выполнения:**

1. Запустить Visio.
2. Выбрать категорию **Блок-схема**.
3. Активизировать значок **Простая блок-схема**.


4. Для каждого этапа документируемого процесса перетащить в документ соответствующую фигуру блок-схемы.

5. Чтобы соединить элементы блок-схемы, наведите указатель мыши на первую фигуру, и щелкните стрелку, указывающую на фигуру, с которой требуется создать соединение. Если вторая фигура находится не рядом с первой, необходимо перетащить маленькую стрелку к центру второй фигуры.

6. Чтобы добавить текст для фигуры или соединительной линии, выделите ее и введите текст. По завершении ввода текста щелкните в пустой области страницы.

7. Чтобы изменить направление стрелки соединительной линии, выберите соединение, а затем на вкладке **Фигура** в группе **Стили фигур** щелкните пункт **Линия**, наведите указатель на пункт **Стрелки** и выберите нужное направление и вид стрелки.

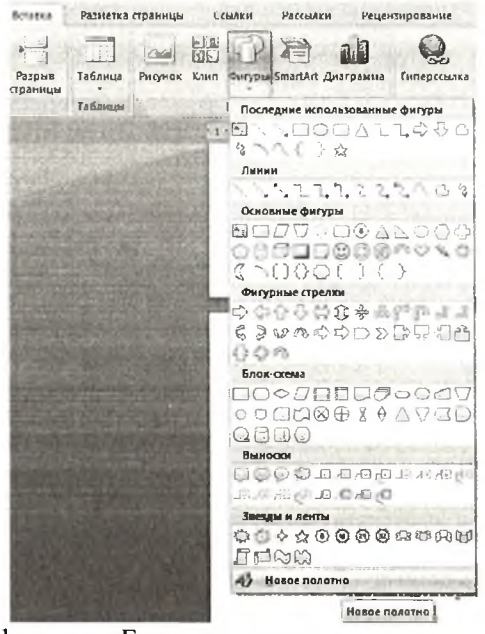
**Способ 2.** Создание блок-схемы в MS Word

**Технология выполнения:**

1.Составить блок-схему для задачи вашего варианта по номеру в журнале

2. Нарисовать блок-схему средствами программы MSWord:

- Вставить новое полотно




- Использовать автофигуры «Блок-схема»


- Добавить необходимый текст в блоки
- Подписать направления блоков решения и цикла с помощью инструмента «Надпись»

- Отформатировать рисунок и сделайте соответствующую подпись

**Контрольные вопросы**

1. Какие технологии для проектирования архитектуры ПО существуют?
2. Какие требования предъявляются к архитектуре и функциональности программного обеспечения?
3. Что такое структурная схема? Каково их назначение?
4. Для чего используют функциональные схемы?
5. В чем заключаются достоинства и недостатки использования функциональных схем?

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 4 (4 часа)

**Тема:** Изучение работы в системе контроля версий

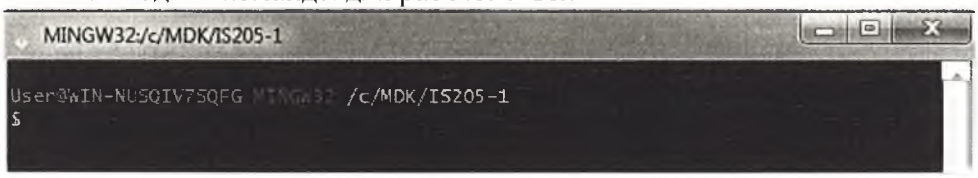
**Цель:** Изучить порядок работы с системой контроля версий GIT

**Задание 1** - Настройка пользователя и почты

**Технология выполнения работы:**

1) Создать папку MDK на диске **C:**, а в ней папку IS205-1 или IS205-2.

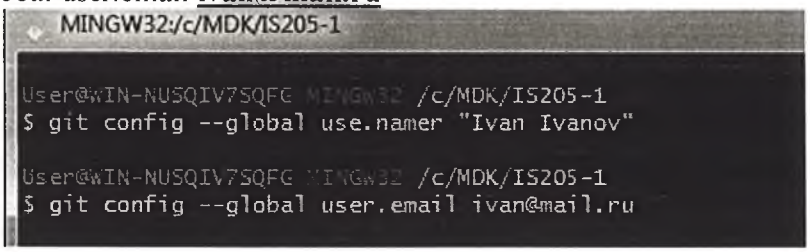
2) С помощью проводника открыть созданную папку (C:\MDK/IS205-1(2)), вызвать контекстное меню и выбрать пункт **Git Bash Here**, открыть окно оболочки с приглашением. В окно теперь можно вводить команды для работы с **Git**.




3) Настроить имя пользователя и адрес почты. Ввести свои данные, например, имя пользователя **IvanIvanov**, а почта - **ivan@mail.ru**

gitconfig --global user.name "Vasya Belov"

gitconfig --global user.email **ivan@mail.ru**




**Задание 2 - Разработка собственного проекта**

**Технология выполнения:**

1) В каталоге C:\MDK\IS205-1 (или C:\MDK\IS205-2) создать папку PR4:

2) Перейти туда в оболочке:

**mkdirPR4**

**cdPR4**

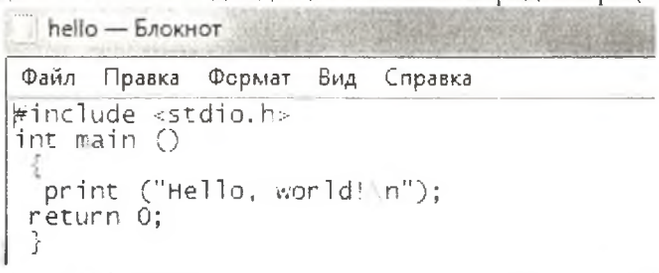

3) Создать новый (пустой) репозиторий: **git init**

Создание нового репозитория предполагает появление в каталоге PR4 скрытой папки с именем **.git**, в которой будут храниться все служебные файлы хранилища. В оболочке появляется надпись **master**, которая означает появление главной ветки репозитория.

4) Создайте файл-исходник **hello.c** с текстом программы на языке С для нашего проекта:

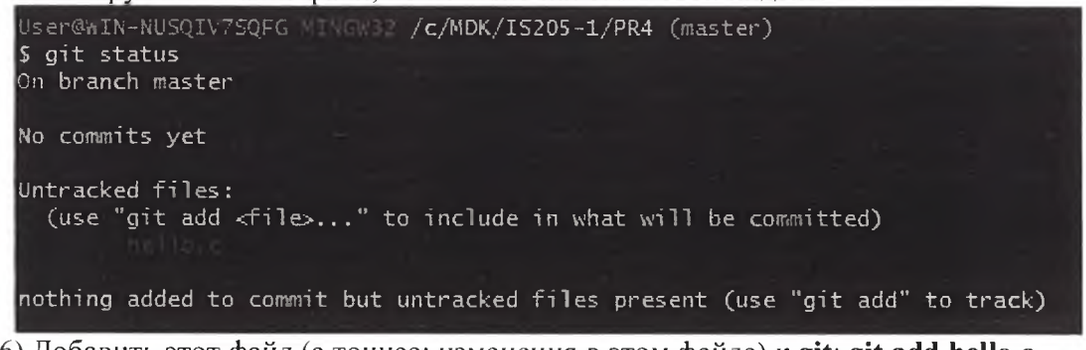
```
#include <stdio.h>
int main()
{
printf("Hello, world!\n");
return 0;
}
```

Файл можно создать в любом подходящем текстовом редакторе (**notepad**).



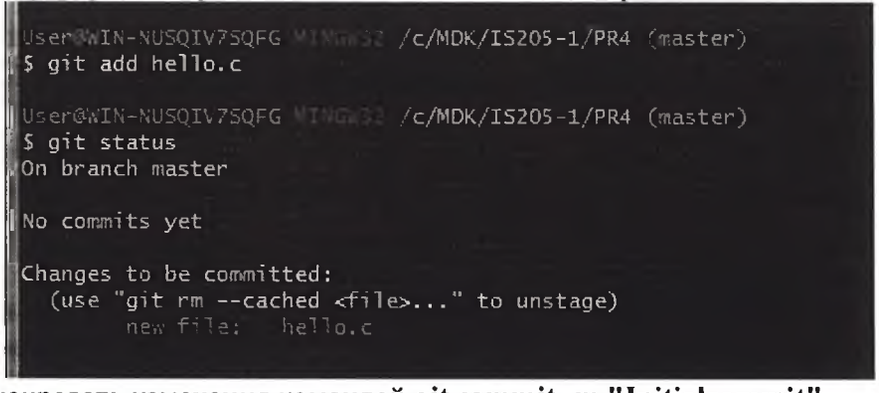

5) Выполнить команду **git status**, которая позволяет увидеть имя файла с программой, выделенное красным цветом.

Git обнаружил в папке файл, но не включил его в свой индекс.

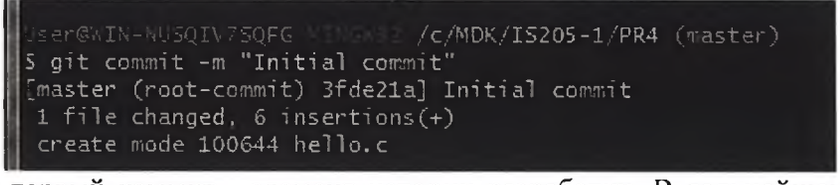



6) Добавить этот файл (а точнее: изменения в этом файле) к **git**: **git add hello.c**

Команда проверки статуса покажет имя добавленного файла зеленым цветом.




7) Зафиксировать изменения командой **git commit -m "Initialcommit"**




Появился первый коммит - элемент истории разработки. В дальнейшем на него можно ссылаться по семи первым символам хеш-кода (3fde21a).

**Хешкоммита** — это специальная метка, позволяющая отличать одни коммиты от других. Хешкоммита состоит из 40 символов, в составе могут быть как буквы, так и цифры.

Чтобы узнать хеш последнего коммита, требуется ввести команду: **$ gitrev-parse HEAD**

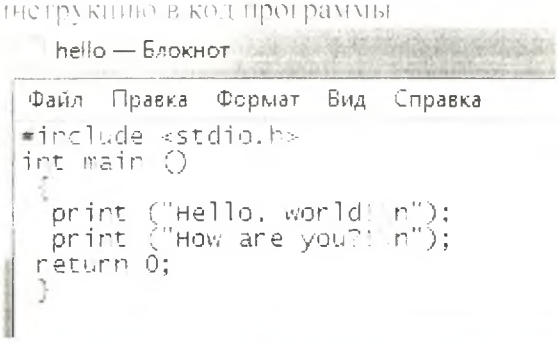
Разработка программы предполагает внесение изменений в исходный код. После некоторого количества правок, когда очередной элемент программы завершен, файл сохранен, необходимо снова дать последовательность команд:

**git add hello.c**

**git commit -m "Initial commit"**

Комментарий к коммиту должен отражать суть сделанных правок, быть лаконичным, но понятным.

8) Ввести еще одну инструкцию в код программы




9) Ввести команды

```
User@WIN-NU5QIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (master)
$ git add hello.c

User@WIN-NU5QIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (master)
$ git commit -m "Initial commit"
[master 15f1169] Initial commit
 1 file changed, 1 insertion(+)
```

10) Добавить в репозиторий текстовый файл

Создать файл demo.txt в папке PR4. Добавить его в репозиторий:

$ git add demo.txt

11) Проверить статус: **$ git status**

```
User@WIN-NU5QIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (master)
$ git add demo.txt

User@WIN-NU5QIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   demo.txt
```

Файл готов к коммиту. Сообщение о состоянии также говорит нам о том, какие изменения относительно файла были проведены в области подготовки – в данном случае это новый файл, но файлы могут быть модифицированы или удалены.

```
User@WIN-NU5QIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (master)
$ git commit -m "Initial commit"
[master d78671d] Initial commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demo.txt
```

Хеш-код текстового файла (d78671d).

**Задание 3 - Создание веток**

Работу над проектом принято вести в ветках. В каждом репозитории есть как минимум одна ветка. Это основная ветка, которую создаёт сам Git, она называется master. Обычно в ней находится стабильная версия программы без ошибок. Если требуется добавить новую функциональность в проект, попробовать какую-то технологию, но не хотите сломать код в основной ветке, вы ответвляетесь из master и трудитесь в своей новой ветке.

**Технология выполнения:**

1) Открыть терминал и ввести команду git branch. Она показывает список веток, с которыми мы работаем в проекте, и выделяет текущую.

2) Создать новую ветку: git checkout -b имя –новой - ветки (task).

```
User@WIN-NUSQIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (master)
$ git branch
* master

User@WIN-NUSQIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (master)
$ git checkout -b task
Switched to a new branch 'task'
```

3) Переключитесь с помощью команды git **checkout** между ветками, после git checkout надо указать название нужной ветки.

Это делается для того, чтобы новая ветка содержала свежую, на момент создания, рабочую версию проекта.

```
User@WIN-NUSQIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (task)
$ git checkout task
Already on 'task'

User@WIN-NUSQIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (task)
$ git checkout master
Switched to branch 'master'

User@WIN-NUSQIV7SQFG MINGW32 /c/MDK/IS205-1/PR4 (master)
$
```

4) Переключитесь между ветками.

Если вы ошиблись в названии, например, допустили опечатку, вы можете изменить название ветки с помощью команды: gitbranch -m старое-имя-ветки новое-имя-ветки.

После того как создали ветку, поработали в ней у себя локально — нужно сохранить результат, чтобы он не пропал и в итоге оказался в репозитории.

5) Если требуется сохранить изменения не во всех файлах, для начала можно ввести команду gitstatus. Она покажет текущее состояние в вашей ветке, а именно список с названиями изменённых файлов, если они есть, и укажет на те, которые ожидают записи и сохранения (обычно они выделены красным цветом).

**Задание 4 - Работа с удаленным репозиторием**

Для работы с серверной частью необходимо зарегистрироваться на https://github.com.


Активирование аккаунта происходит после подтверждающего письма по электронной почте.

**Технология выполнения:**

1) Создать репозиторий с именем PR4.

Тип репозитория надо оставить **public** для свободно распространяемых проектов и **private** для закрытых от внешнего мира.

2) Нажать кнопку **Создать репозиторий**

3) Затем выбрать команду

**git remote add origin https://github.com/имя_иользователя/hello.git**

и передать первый раз изменения на сервер: **git push -u origin master**

Слово **origin** в данном случае означает псевдоним серверного репозитория, а **master** имя ветки. Страницу репозитория можно обновить. Как только на локальной машине накапливается достаточно изменений (новых коммитов), осуществляется их передача на удаленный сервер командой **git push.**

Эта команда передает изменения из текущей ветки в удаленную. В процессе установления соединения, возможно, потребуется указать свои регистрационные данные для аккаунта **Git hub**.

**Контрольные вопросы**

1. Что такое Система контроля версий?
2. Какие состояния могут иметь проектные файлы в Git?
3. Приведите основные команды git.
4. Что такое репозиторий?

5. Как создать новую ветку в git?
6. Как переключиться в существующую ветку?
7. Как отправить изменения на сервер?

## ЛАБОРАТОРНАЯ РАБОТА № 1 (4 часа)

**Тема:** Построение диаграммы Вариантов использования и диаграммы последовательности

**Цель:** Изучить порядок построения диаграмм вариантов использования и диаграммы последовательности.

**Задание 1 - Построить диаграммы Вариантов использования и последовательности для модели системы управления банкоматом**

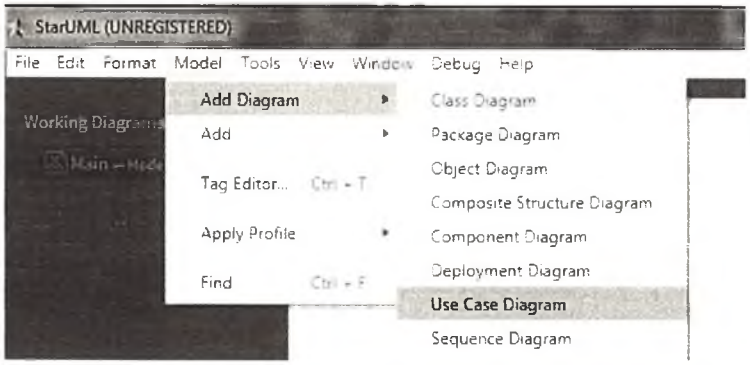
Задание 1.1- Построить диаграмму Вариантов использования

**Технология выполнения:**

1 Запустить программу **StarUML**. Новый проект будет создан автоматически.

2 Для разработки диаграммы вариантов использования (прецедентов, UseCase) в среде StarUML необходимо выполнить действия:

Способ 1. Активизировать команду главного меню Model→Add Diagram→Use Case Diagram.




Способ 2.С помощью команды контекстного меню Add Diagram → Use Case Diagram.

Способ 3. В навигаторе модели щелкнуть два раза по имени главной диаграммы **Main**, то откроется ее рабочее поле.

3 С помощью панели ToolBox требуется создать всех действующих лиц модели. Для того чтобы создать прецедент (UseCase), щелкните по овальному символу прецедента напанели элементов слева от рабочего поля диаграммы, а затем щелкните потому месту на рабочем поле диаграммы, в которое вы хотите поместить прецедент. Аналогичным образом создается актер (Actor).

4 Когда элемент помещается на поле диаграммы, он становится доступен для редактированияимени и некоторых свойств. В выделенное поле введите новое имя актера - *Клиент банк* и прецедента - *Снятие наличных по кредитной карте*.

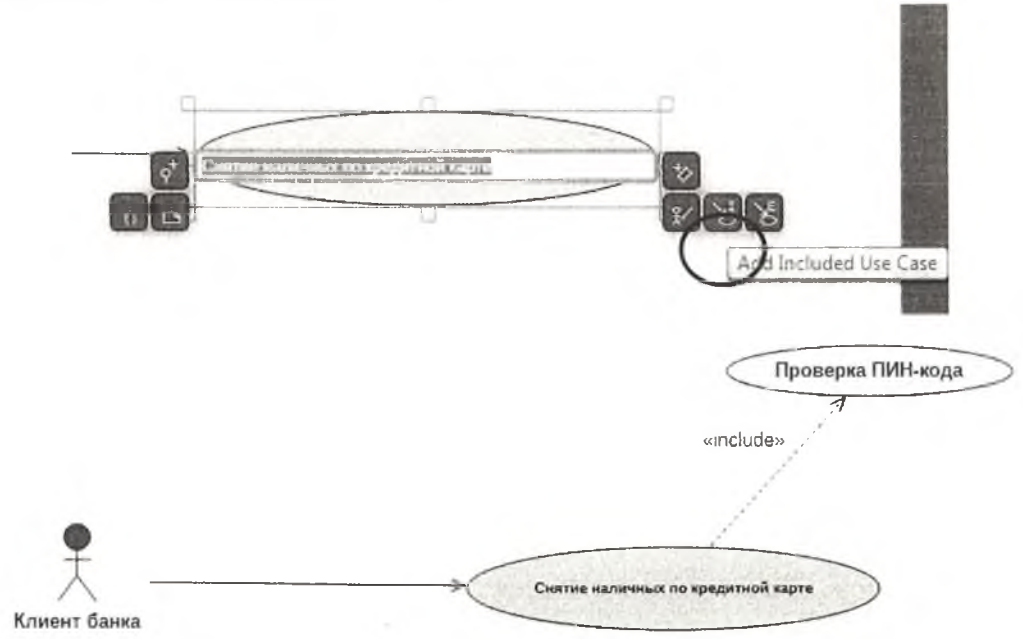

5 Оформить шрифт (Font), цвет заливки (Color), тип линии (LineStyle) можно с помощью команда панели **Styles** на панели элементов справа от рабочего поля диаграммы.


6 Для создания отношения между элементами диаграммы щелкнуть поизображению

соответствующего отношения на панели элементов справа, азатем проведите линию от одного элемента к другому, удерживая левуюкнопку мыши


7 Чтобы удалить элемент с диаграммы достаточно щелкнуть левой кнопкой мыши по этому элементу, а затем нажать кнопку Delete, либо щелкнуть правой кнопкой мыши по элементу и в контекстном меню выбрать Edit→Delete.

8 Добавить отношение зависимости со стереотипом <<include>>, направленное от варианта использования *Снятие наличных по кредитной карточке* к варианту использования *Проверка ПИН-кода.* Включение (include) говорит о том, что исходный прецедент явным образом включает в себя поведение целевого.




9 Для окончательного построения диаграммы варианта использования для рассматриваемой модели банкомата следует самостоятельно выполнить следующие действия:

1. Добавить актера с именем Банк.
2. Добавить прецедент *Получение справки о состоянии счета.*
3. Добавить прецедент *Блокирование кредитной карточки.*
4. Добавить направленную ассоциацию от актера *Клиент банкомата* к прецеденту *Получение справки о состоянии счета.*
5. Добавить направленную ассоциацию от прецедента *Снятие наличных по кредитной карточке* к прецеденту *Проверка ПИН-кода.*
6. Добавить направленную ассоциацию от прецедента *Получение справки о состоянии счета* к сервису *Банк.*
7. Добавить отношение зависимости со стереотипом <<include>>, направленное от прецедента *Получение справки о состоянии счета* к варианту использования *Проверка ПИН-кода.*
8. Добавить отношение зависимости со стереотипом <<extend>>, направленное от прецедента *Блокирование кредитной карточки* к прецеденту *Проверка ПИН-кода.*

Расширение (extend) показывает, что целевой прецедент расширяет поведение исходного. Используемый прецедент выполняется не всегда вместе с базовым, а только при выполнении дополнительных условий, таким образом, расширяя функциональность базового элемента. Изображается расширение пунктирной стрелкой с надписью <<extend>>, направленной от используемого варианта использования к базовому.

10 Результат моделирования представлен на рис. 1.

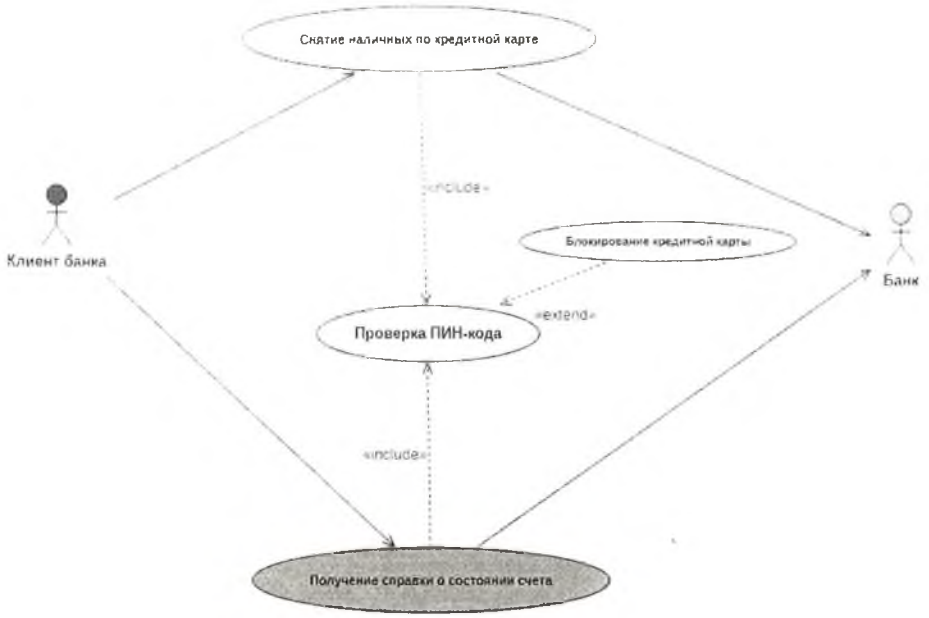



Рисунок 1 - Диаграмма варианта использования для разрабатываемой модели банкоматов

11 В модель нужно включить краткое описание каждого актера илипрецедента, делается это для того, чтобы между разработчиком и заказчикомсистемы не оставалось «белых пятен» и расхождений в пониманиифункциональности системы и ролей, взаимодействующих с ней актеров. Длякаждого актера описывается роль, которую он играет в системе, а длякаждого прецедента – его назначение и функциональность. Также можноуточнить, каким актером запускается прецедент.

Ввести описание элемента в окно документирования **Documentation**:

– для актера Клиента банкомата следует ввести текст: «Любое физическое лицо, пользующееся услугами банкомата»;

– для актера Банк следует ввести текст: «Все данные операций поступают в банк посредством шифрованного канала. Для получения денег нужно, чтобы банк одобрил это снятие»;

– для прецедента *Снятие наличных по кредитной карте* ввести текст: «Основной вариант использования для разрабатываемой модели банкомата»;

12 После окончания сеанса работы над проектом выполненную работу необходимо сохранить в файле проекта с помощью команды меню File→Save (Файл→Сохранить) или File→Save As (Файл - Сохранить как) с именем LR_1. MDL.

**Задание 2 - Построить диаграмму последовательности (сообщений)**

**Технология выполнения:**

1 Диаграмма последовательности может быть активизирована с помощью контекстного меню: New→Sequence Diagram (Новая→Диаграмма последовательности) для логического представления или представления вариантов использования в браузере проекта.

2 Назовите новую диаграмму «Банкомат» в поле «Properties» на правой панели в поле «Name».

3 Перетащить действующее лицо «Клиент банка» из диаграммы использования окна «Model1».

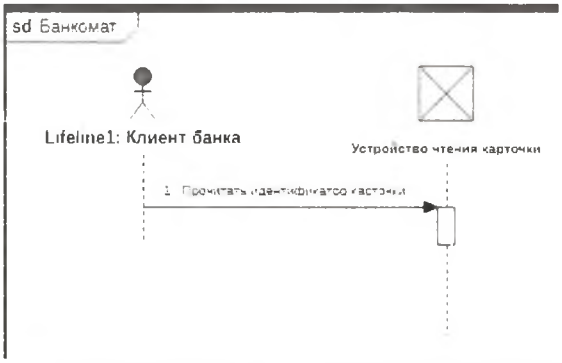
4 Добавить на диаграмму объект **LifeLine**, ввести имя *Выбор варианта заказа*

5 Добавить сообщений на диаграмму, для этого в окне «Toolbox» нажать кнопку «Message» (Сообщение объекта).

6 Для добавления сообщения между объектами нужно с помощью левой кнопки мыши нажать кнопку с изображением сообщения **Message** на специальной панели инструментов и протянуть ее между объектами.

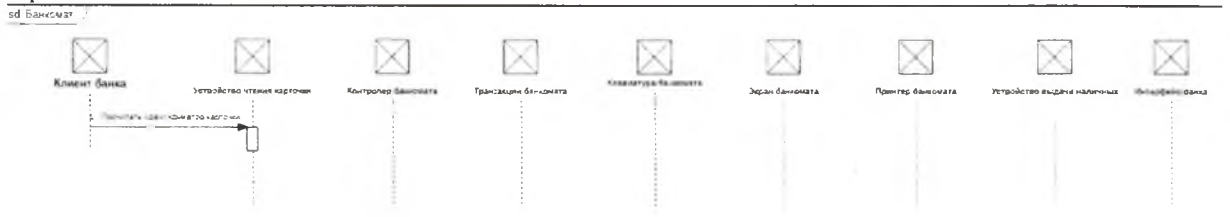
7 Ввести текст «Прочитать идентификатор карточки» для полученного сообщения.


8 Можно изменить вид объекта с помощью команды **Icon with Label** в меню Format.




9 Построение диаграммы последовательности сводится к добавлению и редактированию свойств отдельныхобъектовисообщений. С этой целью следует выполнить следующие действия:

1. Добавить объекты классов с именами: *Контроллер Банкомата, Транзакция Банкомата, Клавиатура Банкомата, Экран Банкомата, Принтер Банкомата, Устройство выдачи наличных и Интерфейс Банка.* Сделать одинаковую высоту объектов на листе диаграммы.




2. Добавить сообщение: *проверить идентификатор карточки*, направленное от объекта класса Контроллер Банкомата к объекту класса Интерфейс Банка.

3. Добавить сообщение: *ввести ПИН-код*, направленное от объекта класса-актера Клиент Банкомата к объекту класса Клавиатура Банкомата.

4. Добавить сообщение: *прочитать ПИН-код*, направленное от объекта класса Контроллер Банкомата к объекту класса Устройство чтения карточки.

5. Добавить сообщение: *создать новую транзакцию*, направленное от объекта класса Контроллер Банкомата к изображению объекта класса Транзакция Банкомата.

6. Добавить сообщение: *проверить правильность ПИН-кода*, направленное от объекта *класса Контроллер Банкомата к объекту класса Транзакция Банкомата.*

7. Добавить сообщение: *показать меню опций*, направленное от объекта класса Контроллер Банкомата к объекту класса Экран Банкомата.

8. Добавить сообщение: *ввести тип транзакции,* направленное от объекта класса-актера Клиент Банкомата к объекту класса Клавиатура Банкомата.

9. Добавить сообщение: *показать меню снятия суммы*, направленное от объекта класса Контроллер Банкомата к объекту класса Экран Банкомата.

10. Добавить сообщение: *ввести сумму снятия наличных*, направленное от объекта

класса-актера Клиент Банкомата к объекту класса Клавиатура Банкомата.

11. Последовательно добавить 3 сообщения: *открыть счет клиента*, *проверить баланс клиента* и *уменьшить счет клиента*, направленные от объекта класса Контроллер Банкомата к объекту класса Интерфейс Банка.

12. Добавить сообщение: *распечатать чек*, направленное от объекта класса Контроллер Банкомата к объекту класса Принтер Банкомата.

13. Добавить сообщение: *вернуть кредитную карточку*, направленное от объекта класса Контроллер Банкомата к объекту класса Устройство чтения карточки.

14. Добавить сообщение: *выдать наличные*, направленное от объекта класса Контроллер Банкомата к объекту класса Устройство выдачи наличных.

15. Добавить сообщение: *завершить транзакцию*, направленное от объекта класса Контроллер Банкомата к объекту класса Транзакция Банкомата.

10 Выполнить форматирование диаграммы (рис. 2).

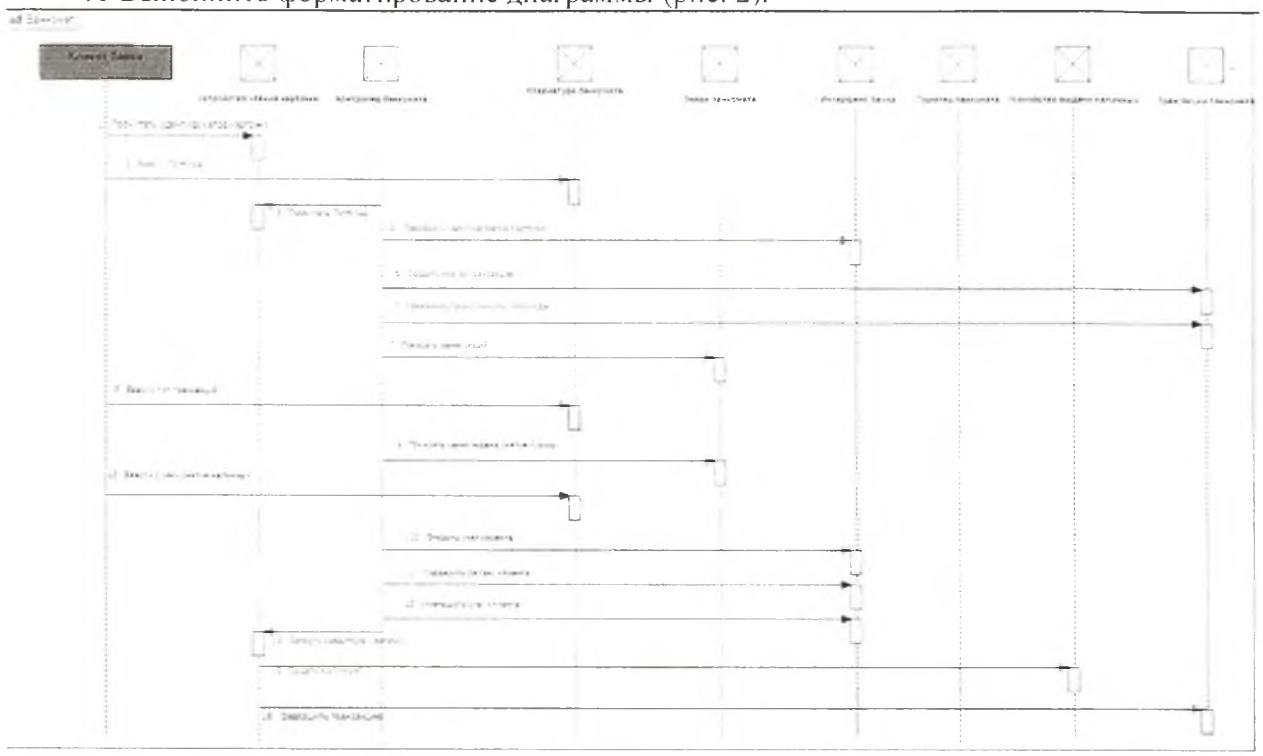


Рисунок 2 – Диаграмма последовательности, описывающей типичный ход событий варианта использования Снятие наличных по кредитной карточке

11 Сохранить проект

**Задание 3 -** Построить диаграммы Вариантов использованияи диаграммы последовательности для модели системы «Сдача экзамена» для предложенного варианта предметной области.

**Контрольные вопросы**

1. Что такое диаграмма последовательности действий?
2. Какие элементы содержит диаграмма последовательности действий?
3. Что такое диаграмма использования

## ЛАБОРАТОРНАЯ РАБОТА № 2 (4 часа)

**Тема:** Построение диаграммы Деятельности, диаграммы состояний и диаграммы классов

**Цель:** Изучить порядок построения различных диаграмм.

**Задание 1 Постройте диаграмму деятельности**

**Технология выполнения:**

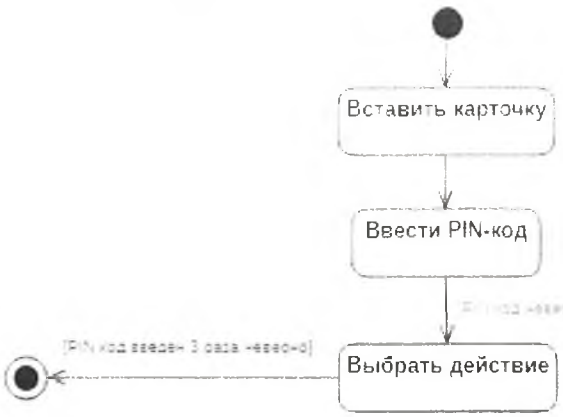
1. Выполнить операцию контекстного меню New - Activity Diagram (Новая - Диаграмма деятельности).

2. Добавить начальный узел или начальное состояние деятельности (Initial)

3. Добавить деятельности (Action) с именами: *Вставить карточку, Ввести PIN-код, Выбрать действие, Ввести сумму, Получить справку о состоянии счета.*

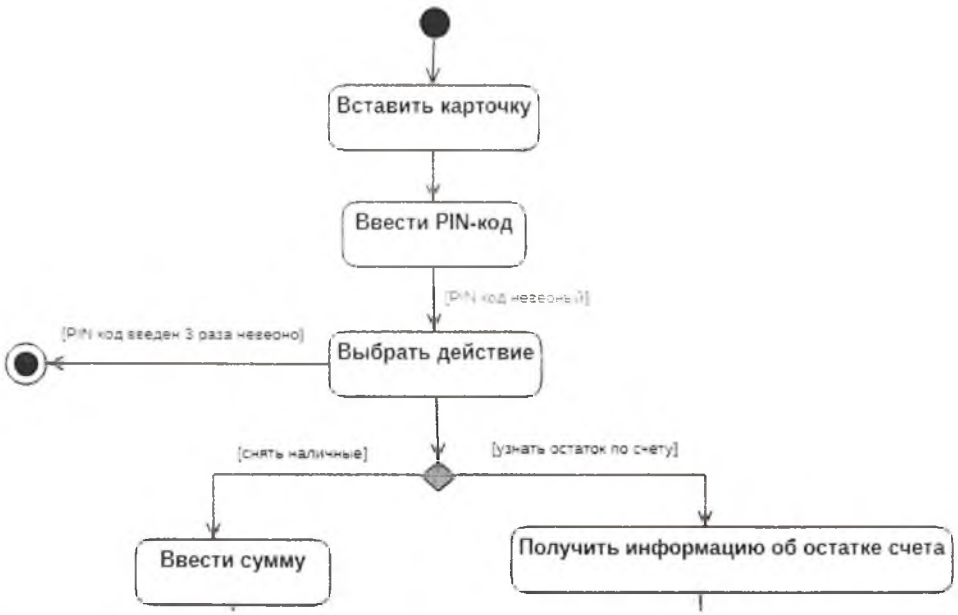
4. Выполнить переход со сторожевым условием [PIN-код неверный] между деятельностями *Ввести PIN-код* и *Выбрать действие*.

5. Добавить конечный узел деятельности (Final) и выполнить переход со сторожевым условием [PIN-код введен 3 раза неверно] от деятельности *Выбрать действие*.




6. Добавить символ ветвления **Decision** (узел решения), расположив его между деятельностями с именами: *Выбрать действие, Ввести сумму, Получить информацию об остатке счета о состоянии счета*.

7. Сделать между ними переходы со сторожевыми условиями [снять наличные] и [узнать остаток по счету].




8. Добавить символ ветвления **Decision** (узел решения), расположив их между деятельностями с именами: *Ввести сумму* и *Получить наличные*.

9. Добавить переход от деятельности Ввести сумму к узлу решения.

10. Ввести сторожевое условие [сумма превышает кредит], направленный отсимвола решениякузлу решения после деятельности *Получить справку*.

11. Добавить переход со сторожевым условием [сумма не превышает кредит], направленный от символа решения к деятельности *Получить наличные*.

12. Добавить символ ветвления после деятельности *Получить наличные* и ввести транзакции: [печать справки не выбрана] на деятельность *Получить карточку* и [выбрана печать справки] на деятельность *Получить справку*.

13. Добавить узел решения после деятельности *Получить справку* и сделать переход на деятельность *Получить карточку*.

14. Сделать переход от деятельности *Получить справку о состоянии счета* на узел решения после деятельности *Получить наличные*.

15. Добавить переход, направленный от деятельности *Получить карточку* к финальному состоянию (Final).

**Диаграмма деятельности** (рис. 3):

1. Клиент вставляет карточку;
2. Клиент вводит PIN код;

Если PIN код введен неверно 3 раза, то карта перемещается в хранилище карт и обслуживание клиента завершается

3. Клиент выбирает действие;
4. Если выбрана операция «Снять наличные», клиенту предлагается ввести сумму;
5. Если сумма не превышает остаток на карте, то происходит выдача наличных;
6. Печать справки, если было запрошено;
7. Клиент получает карту.

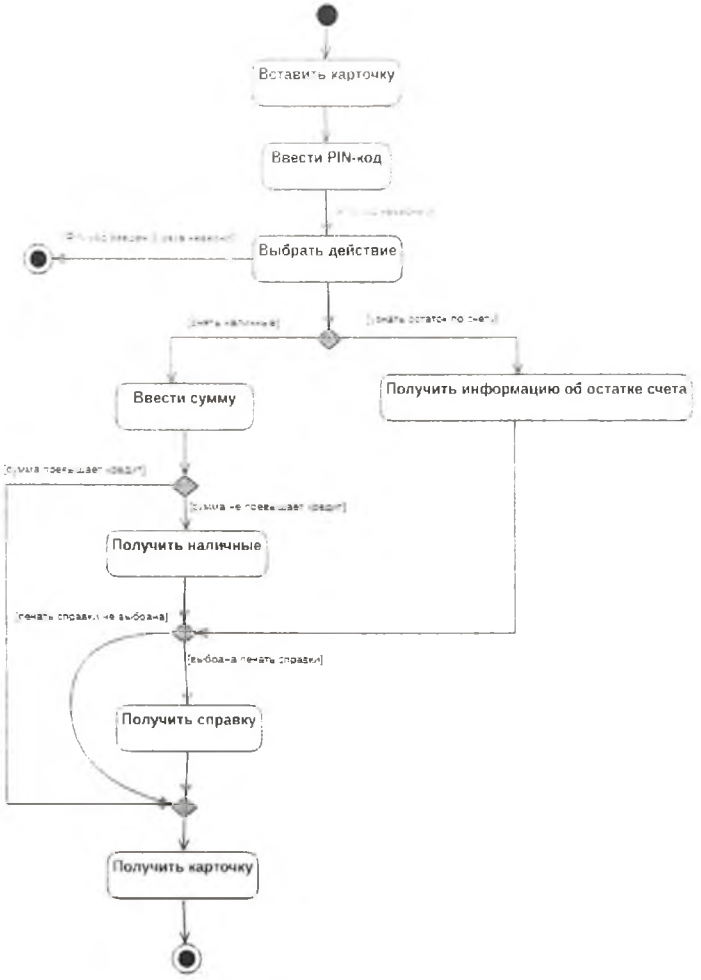



Рисунок 3 - Диаграмма деятельности

**Задание 2** - Постройте диаграмму состояний (рис.4)

**Технология выполнения:**

1. Для добавления диаграммы состояний в модель нужно в контекстном меню выбрать пункт **Add Diagram**, в списке выбрать диаграмму состояний **Statechart Diagram** (Диаграмма состояний).

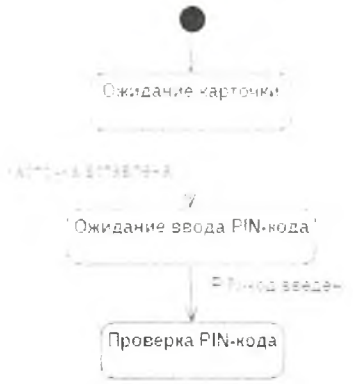
2. Добавить на диаграмму состояний начальное состояние (Initial State).

3. Добавить состояние *Ожидание карточки* (Simple State) и отразить на диаграмме переход между начальным и данным состояниями.

4. Добавить состояния с именами: Ожидание ввода ПИН-кода, Проверка ПИН-кода и финальное состояние (finalstate).

5. Добавить переход карточка вставлена, направленный от состояния *Ожидание карточки* к состоянию *Ожидание ввода PIN-кода.*

6. Добавить переход *PIN-код введен*, направленный от состояния *Ожидание ввода PIN-кода* к состоянию *Проверка PIN-кода.*




7. Для завершения построения диаграммы состояний добавить оставшиеся состояния и переходы. С этой целью следует выполнить следующие действия:

– Добавить состояния с именами: Ожидание выбора операции, Обработка запроса на снятие наличных, Отображение остатка по счету, Выдача наличных, Печать, Возврат карточки, Завершение транзакции.

– Добавить переход со сторожевым условием [PIN-код неверный], направленный от состояния *Проверка PIN-кода* к состоянию *Ожидание ввода PIN-кода.*

– Добавить переход со сторожевым условием [PIN-код верный], направленный от состояния *Проверка PIN-кода* к состоянию *Ожидание выбора операции.*

– Добавить переход со сторожевым условием выбор суммы [сумма введена], направленный от состояния *Ожидание выбора операции* к состоянию *Обработка запроса на снятие наличных.*

– Добавить переход, направленный от состояния *Ожидание выбора операции* к состоянию *Отображение остатка по счету.*

– Добавить переход со сторожевым условием [выбрана печать справки], направленный от состояния *Отображение остатка по счету* к состоянию *Печать.*

– Добавить переход со сторожевым условием [печать справки не выбрана], направленный от состояния *Отображение остатка по счету* к состоянию *Печать* и печать справки не выбрана к состоянию *Возврат карточки.*

– Добавить переход со сторожевым условием: [кредит превышен], направленный от состояния *Обработка запроса на снятие наличных* к состоянию *Возврат карточки.*

– Добавить переход со сторожевым условием: [кредит не превышен], направленный от состояния *Обработка запроса на снятие наличных* к состоянию *Выдача наличных.*

– Добавить переход со сторожевым условием наличные выданы/ [выбрана печать справки], направленный от состояния *Выдача наличных* к состоянию *Печать.*

– Добавить переход со сторожевым условием наличные выданы/ [печать справки не выбрана], направленный от состояния *Выдача наличных* к состоянию *Возврат карточки.*

– Добавить переход: закончена, направленный от состояния *Печать* к состоянию *Возврат карточки.*

– Добавить переход: карточка возвращена, направленный от состояния *Возврат карточки* к состоянию *Завершение транзакции.*

– Добавить переход: транзакция завершена, направленный от состояния *Завершение транзакции* к состоянию *Ожидание карточки.*

– Добавить переход, направленный от состояния *Ожидание карточки* к финальному или конечному состоянию

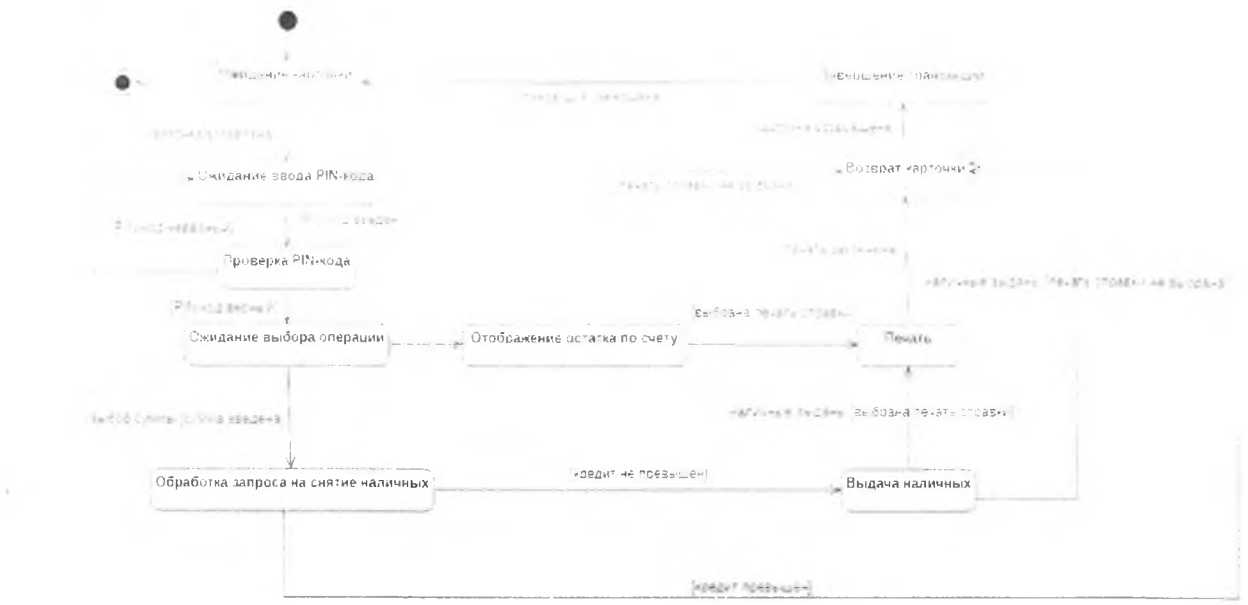



Рисунок 4 - Диаграмма состояний

**Банкомат может принимать следующие состояния:**

1. Ожидание карточки
2. Ожидание ввода PIN кода – после того, как карта вставлена
3. Проверка PIN кода – после ввода PIN кода
4. Ожидание выбора операции – после проверки PIN кода
5. Обработка запроса на снятие наличных – когда выбрана операция «Снять наличные»
6. Выдача наличных
7. Отображение остатка по счету – когда выбрана операция «Отобразить остаток по счету»
8. Печать – если запрошена печать справки
9. Возврат карточки
10. Завершение транзакции

**Задание 3 - Постройте диаграмму классов**

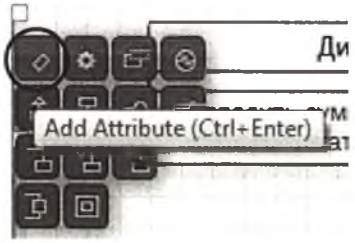
**Технология выполнения:**

1. Для создания новой диаграммы классов в контекстном меню выбрать пункт Add Diagram, в списке выбрать диаграмму классов Class Diagram (Диаграмма Классов).
2. Переименовать класс Class Diagram1 в Банкомат.
3. Добавить компоненты Классы (Class):

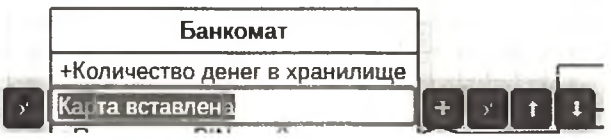
- **Банкомат**

1 Добавить атрибуты (поля, свойства) и методы (операции)

1.1 Добавить атрибут (поля, свойства) класса Банкомат с помощью команды Add Attribute - Количество денег в хранилище




1.2 Добавить атрибут *Карта вставлена* с помощью пиктограммы 



2 Добавить методы (операции) класса Банкомат с помощью команды Add Operation:


– Проверить PIN код – сверяет PIN код клиента с PIN кодом счета, в случае несовпадения предлагает клиенту ввести PIN код повторно;

– Поместить карту в хранилище – метод срабатывает при трехкратном неправильном вводе PIN кода;

– Выдать деньги – аргументом принимает сумму, введенную клиентом, выдает нужную сумму в случае достаточного количества денег в хранилище;

– Напечатать справку – принимает три аргумента: запрос на печать справки, операцию, выбранную клиентом и баланс;

– Выдать карту – завершает сеанс обслуживания.

| Банкомат |
| --- |
| +Количество денег в хранилище<br>+Карта вставлена |
| +Проверить PIN-код()<br>+Выдать карту()<br>+Выдать деньги()<br>+Напечатать справку()<br>+Поместить карту в хранилище() |

3 Аналогично добавить классы:

- **Клиент**

Поля класса Клиент:

– Номер карты;

– PIN-код клиента– PIN код, который вводит клиент;

– Печать – клиент подтверждает печать справки;

– Операция – выбранная клиентом операция.

- **Счет**

Поля класса Счет:

– Номер счета,

– Баланс,

– PIN код счета.

Методы класса Счет:

– Уменьшить баланс – уменьшает остаток по счету на сумму, введенную клиентом; аргументом принимает сумму

- **Дисплей**

Методы класса Дисплей:

– Запросить сумму – принимает аргументом сумму, запрошенную клиентом;

– Отобразить остаток по счету – принимает аргументом баланс счета.

4 Описание отношений классов:

4.1 Класс Дисплей связан с классом Банкомат отношением агрегации (Aggregation): дисплей является частью банкомата.

4.2 Класс Банкомат связан с классом Клиент отношением ассоциации (Association), причем у одного банкомата может быть много клиентов (1..*).


4.3 Класс Банкомат связан с классом Счет отношением ассоциации, причем один банкомат может работать со многими счетами (1..*).

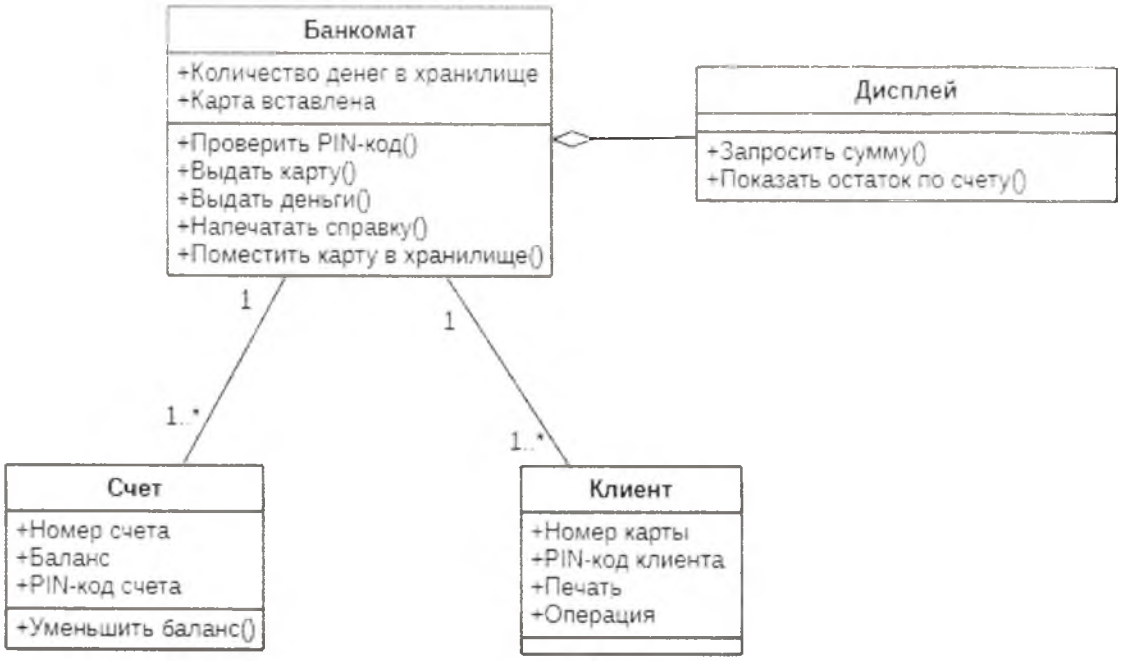



Рисунок 5 - Диаграмма классов

**Контрольные вопросы:**

1. Что такое диаграмма деятельности?
2. Что такое маркер на диаграмме деятельности?
3. Какие элементы содержит диаграмма деятельности?
4. Какие данные отражаются на диаграмме состояний?
5. Что такое сторожевое условие? Формат записи.
6. По каким принципам строится диаграмма классов?

## ЛАБОРАТОРНАЯ РАБОТА № 3 (4 часа)

**Тема:** Построение диаграммы Кооперации и диаграммы развертывания

**Цель:** Изучить порядок построения различных диаграмм.

**Задание 1** – Построить диаграмму кооперации для системы «Банкомат»

**Технология выполнения:**

1.1 Для добавления диаграммы кооперации в контекстном меню выбрать пункт **Add Diagram**, в списке выбрать диаграмму кооперации **Collaboration diagram**

1.2 Для завершения построения диаграммы кооперации рассматриваемого примера следует добавить оставшиеся объекты, связи сообщения.

С этой целью следует выполнить следующие действия:

1. Добавить объекты классов с именами: Контроллер Банкомата, Транзакция Банкомата, Клавиатура Банкомата, Экран Банкомата, Принтер Банкомата, Устройство выдачи наличных и Интерфейс Банка.

2. Добавить связи, соединяющие объекты классов с именами: Контроллер Банкомата с Устройством чтения карточки, Контроллер Банкомата с Транзакцией Банкомата, Контроллер Банкомата с Клавиатурой Банкомата, Контроллер Банкомата с Экраном Банкомата, Контроллер Банкомата с Принтером Банкомата, Контроллер Банкомата с Устройством выдачи наличных и Контроллер Банкомата с Интерфейсом Банка.

3. Добавить сообщение: проверить идентификатор карточки, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Интерфейс Банка*.

4. Добавить сообщение: *ввести ПИН-код()*, направленное от объекта класса-актера *Клиент Банкомата* к объекту класса *Клавиатура Банкомата*.

5. Добавить сообщение: *прочитать ПИН-код()*, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Устройство чтения карточки*.

6. Добавить сообщение: *создать новую транзакцию()*, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Транзакция Банкомата*.

7. Добавить сообщение: *проверить правильность ПИН-кода()*, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Транзакция Банкомата*.

8. Добавить сообщение: *показать меню опций()*, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Экран Банкомата*.

9. Добавить сообщение: *ввести тип транзакции()*, направленное от объекта класса-актера *Клиент Банкомата* к объекту класса *Клавиатура Банкомата*.

10. Добавить сообщение: *показать меню снятия суммы()*, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Экран Банкомата*.

11. Добавить сообщение: *ввести сумму снятия наличных()*, направленное от объекта класса-актера *Клиент Банкомата* к объекту класса *Клавиатура Банкомата*.

12. Последовательно добавить 3 сообщения: *открыть счет клиента, проверить баланс клиента* и *уменьшить счет клиента*, направленные от объекта класса *Контроллер Банкомата* к объекту класса *Интерфейс Банка*.

13. Добавить сообщение: *распечатать чек()*, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Принтер Банкомата*.

14. Добавить сообщение: *вернуть кредитную карточку()*, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Устройство чтения карточки*.

15. Добавить сообщение: *выдать наличные()*, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Устройство выдачи наличных*.

16. Добавить сообщение: *завершить транзакцию()*, направленное от объекта класса *Контроллер Банкомата* к объекту класса *Транзакция Банкомата*.

Диаграмма кооперации, описывающая реализацию типичного хода событий варианта использования Снятие наличных по кредитной карточкедля проекта системы управления банкоматом, показана на рис. 6.

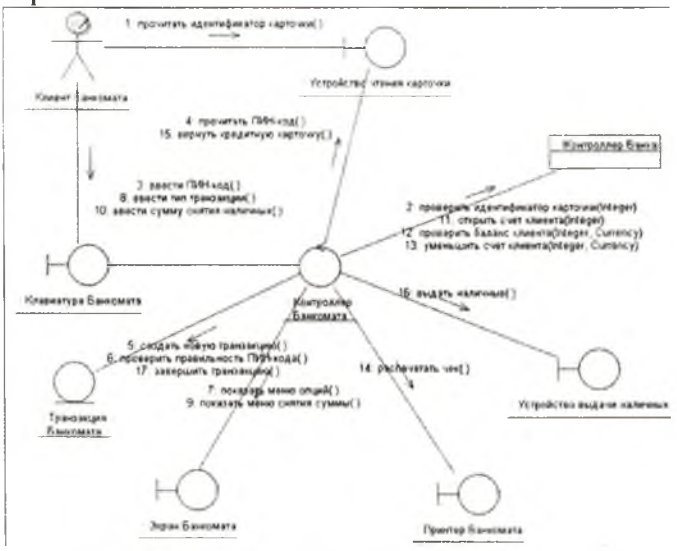
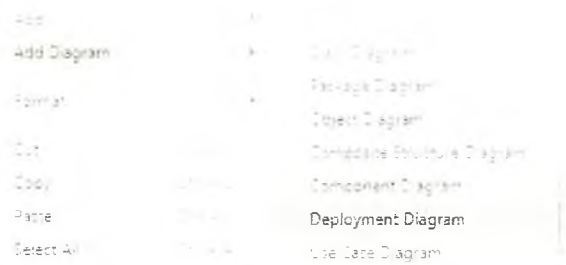



Рисунок 6 – Диаграмма кооперации системы «Банкомат»

**Задание 2** – Построить диаграмму развертываниедля описания аппаратной части системы «Банкомат»

**Технология выполнения:**

2.1 Добавить диаграмму развертывания с помощью контекстного меню **Add Diagram – Deployment Diagram**




2.2 Добавить узел (Node) на диаграмму развертывания и задать ему имя «Банкомат №1», для которого в форме примечания укажемпомеченное значение: {адрес = ул. Ленина, д. 12}. Это значение служит для спецификации конкретного адреса одного из банкоматов системы (рис. 7-8).

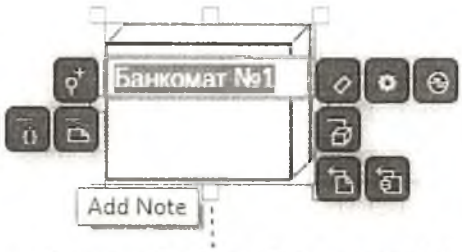



Рисунок 7 - Узел Банкомат № 1


Рисунок 8 - Диаграмма развертывания после добавления узла Банкомат № 1

2.3 Описать узел Банкомат №1 с помощью редактора Documentation

2.4 Добавить второй узел с именем Сеть со стереотипом <<закрытая сеть>>. При этом для задания стереотипа следует ввести его текст без угловых кавычек в строку с именем stereotype.


2.5 Для завершения построения диаграммы развертывания следует выполнить следующие действия:

1. Добавить узел с именем «Банкомат №2», для которого задать помеченное значение в форме примечания: {адрес = ул. Советская, д.53}.
2. Добавить узел с именем «Банкомат №3», для которого задать помеченное значение в форме примечания: {адрес = ул. Парковая, д.87}.
3. Добавить узел с именем «Сервер Банка».
4. Добавить соединение для узлов с именами «Банкомат №2» и «Сеть».
5. Добавить соединение для узлов с именами «Банкомат №3» и «Сеть».
6. Добавить соединение для узлов с именами «Сервер Банка» и «Сеть».
7. Диаграмма развертывания будет иметь следующий вид (рис. 9).


Рисунок 9- Диаграмма развертывания системы «Банкомат»

**Задание 3** - Построить диаграмму развертывания для системы управления банкоматом

**Технология выполнения:**

3.1 Данная диаграмма (рис.10-11) содержит семь компонентов, три из которых изображены в форме ресурсоемких узлов (NodeInstance), а четыре – в форме устройств.

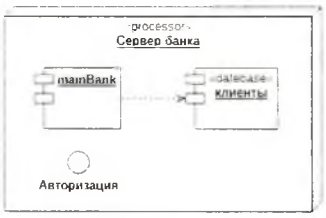



Рисунок 10 - Узел (NodeInstance) диаграммы развертывания

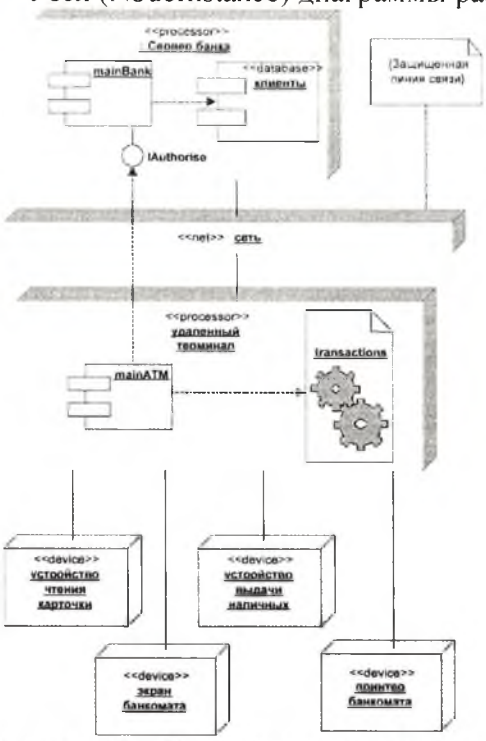



Рисунок 11 - Диаграмма развертывания

**Контрольные вопросы:**

1. Что такое диаграмма кооперации?
2. Чем отличаются представления кооперации на уровне спецификации и на уровне примеров?
3. Что такое диаграмма развертывания?
4. Какие элементы содержит диаграмма развертывания?

## ЛАБОРАТОРНАЯ РАБОТА № 4 (4 часа)

**Тема:** Построение диаграммы пакетов, диаграммы компонентов

**Цель:** Изучить порядок построения диаграммы компонентов и диаграммы пакетов

**Технология выполнения:**

**Задание 1** – Построить диаграмму пакетов

**Постановка задачи.** Задана предметная область: туристическое агентство. Клиент может выбрать тур на веб-сайте агентства. Описание бизнес-процессов туристического агентства. Клиент является потенциальным покупателем туристического продукта, взаимодействует с информационной системой через интернет. Турагент реализует клиенту, сформированный туроператором тур на тех условиях, которые предлагаются туроператором. Туроператор осуществляет деятельность по формированию, продвижению и реализации туристического продукта. Формирование туристического продукта складывается из бронирования и оплаты отеля, заказа авиарейса, обеспечения услуг по предоставлению транспорта, экскурсионных услуг и т. д. Кроме того, туроператор определяет цены на сформированный им тур и политику скидок. Информационная система предоставляет каталог всех туров. Каталог содержит полную информацию о туре (страна, дата вылета и прилета, количество дней пребывания, стоимость). Клиент может забронировать только тот тур, который присутствует в каталоге, а также оставить пожелания на сайте. Туроператор имеет возможность добавить новый тур в каталог. Турагент оформляет все необходимые документы с клиентом (составляет договор).

**Технология выполнения задания:**

1. Для туристического агентства создать пакеты **Entities** (сущности), **Boundaries** (границы) и **Control** (управление) (рис. 11) и в них разместить соответствующие классы.


Рисунок 11 - Диаграмма пакетов предметной области «Туристическое агентство»

2. Для туристического агентства создать диаграмму компонентов (рис. 12).

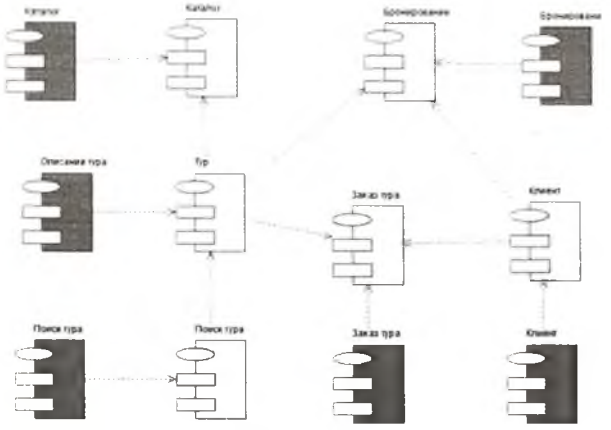


Рисунок 12 - Диаграмма компонентов предметной области «Туристическое агентство»

**Задание 2** – Построить диаграмму компонентов для системы «Банкомат»

**Технология выполнения задания:**

1. Добавить диаграмму компонентов (Component Diagram).
2. Изменить имя диаграммы, предложенное по умолчанию **Main**, на *Диаграмма компонентов АТМ.*

3. Добавить на диаграмму компонент **MainATM.exe** со стереотипом «EXE».

4. Добавить компоненты на диаграмму компонентов и выполнить редактирование его свойств. С этой целью следует выполнить следующие действия:

– Добавить компонент с именем: *Устройства Банкомата*, для которого задать стереотип Task Specification.

– Добавить компоненты с именами: *Устройство чтения карточки*, *Клавиатура Банкомата*, *Принтер Банкомата*, *Экран Банкомата*, *Устройство выдачи наличных*, для которых задать стереотип Task Body.

– Добавить зависимость от компонента с именем MainATM.exe к компоненту с именем *Устройства Банкомата*.

– Добавить зависимость от компонента с именем *Устройство чтения карточки* к компоненту с именем *Устройства Банкомата*.

– Добавить зависимость от компонента с именем *Клавиатура Банкомата* к компоненту с именем *Устройства Банкомата*.

– Добавить зависимость от компонента с именем *Принтер Банкомата* к компоненту с именем *Устройства Банкомата*.

– Добавить зависимость от компонента с именем *Экран Банкомата* к компоненту с именем *Устройства Банкомата*.

– Добавить зависимость от компонента с именем *Устройство выдачи наличных* к компоненту с именем *Устройства Банкомата*.

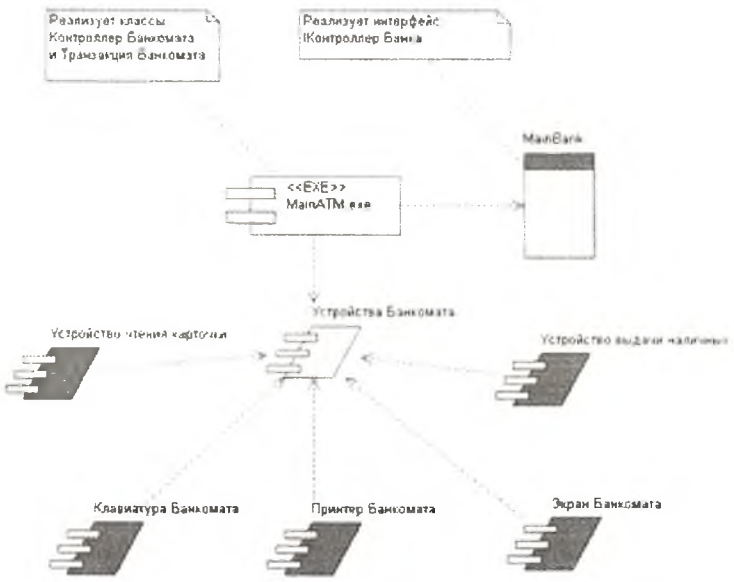



Рисунок 13 - Диаграмма компонентов модели управления банкоматом

**Задание 3** – Построить диаграмму пакетов для системы «Банкомат»

**Технология выполнения задания:**

1. Создать диаграмму пакетов Package

Пакет *Функционирование банкомата* зависит от пакета *Счета*, так как *Банкомат* использует в своих методах поля класса *Счет*: PIN код счета и Баланс.

Пакет *Функционирование банкомата* зависит от пакета *Клиенты*, так как использует в своих методах поля класса *Клиент*: номер карты, PIN код клиента, операция, печать.

**Контрольные вопросы**

1. Какие компоненты изображаются на диаграмме компонентов?
2. Каким символом изображается библиотека?
3. Как изображаются зависимости между компонентами?
4. Какие компоненты изображаются на диаграмме пакетов?
5. Какие подходы используют при группировке классов по пакетам?

## ЛАБОРАТОРНАЯ РАБОТА № 5 (4 часа)

**Тема:** Построение диаграммы потоков данных

**Цель:** Изучение порядка построения диаграммы потоков данных

**Задание 1** - Создать диаграмму DFD - контекстную диаграмму процесса «Изготовление мебели»

**Технология выполнения:**

1. Открыть программу **Process Modeler/ BPwin**.
2. Создать модели потоков данных с помощью CASE→средства→ Process Modeler BPwin
3. В диалоге **All Fusion Process Modeler** «кликните» по радио-кнопке **DFD** (рис. 14).

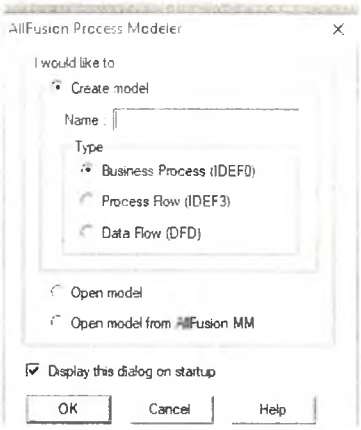



Рисунок 14 – Диалог All Fusion Process Modeler

4. В поле **Name** ввести имя контекстной диаграммы «Изготовление мебели»
5. В окне **Properties for New Models** ввести имя автора (свои ФИО)
6. Во вкладке **Definition** — описание предметной области.
7. Автоматически откроется палитра инструментов, соответствующая этой нотации (рис. 15). Ознакомьтесь с ее командами.

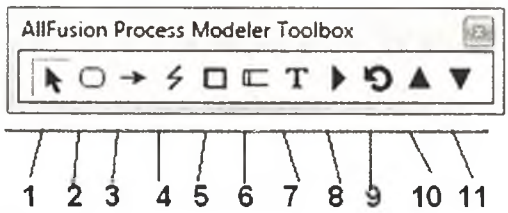
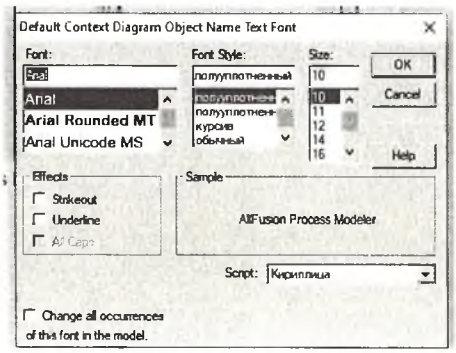



Рисунок 15 - Палитра инструментов в нотации DFD:
1 - указатель; 2 - новый процесс; 3 - поток; 4 - выноска;5 - внешний объект; 6 - хранилище данных; 7 - текст на диаграмме; 8 - словарь; 9 - дерево диаграмм; 10 - переход к родительской диаграмме; 7 - переход к дочерним диаграммам

8. Для ввода имени блока необходимо: щелкнуть ПКМ по блоку, затем выбрать команду **Name**. В диалоговом окне ввести название «Изготовление мебели».
9. Установить русский шрифт для объекта: выбрать **Model | Default Fonts**, выберите **Context Activity**. Установить шрифт *A*rial Unicode MS, 16 пт. Поставить флажок в нижней части окна.




10. Для изменения цвета текста выполнить команду контекстного меню **Color**.

11. Построить дугу управления с помощью кнопки . Для этого подвести курсор мыши к верхней стороне блока до образования темного треугольника и щелкнуть левой кнопкой мыши.

12. Для изменения стиля дуги выбрать в контекстно-зависимом меню команду **Style**.


13. Для идентификации дуги управления выбрать на панели редактирования кнопку . Щелкнуть правой кнопкой мыши по дуге. Выбрать команду Name.

14. В диалоговом окне ввести название дуги: «Нормативная документация»

15. Название дуги является независимым объектом, который можно перемещать относительно дуги. Текст может располагаться по отношению к дуге в свободной форме, либо соединяться с дугой символом тильды. Чтобы установить тильду следует: на панели инструментов нажать кнопку ; щелкнуть левой кнопкой мыши по тексту, а затем по дуге


15. В результате выполнения задания получится диаграмма (рис. 16).

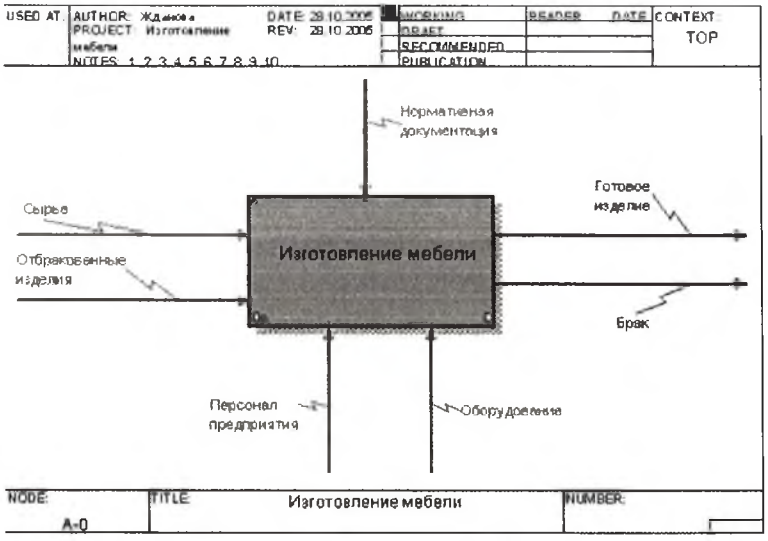



Рисунок 16 – Контекстная диаграмма «Изготовление мебели»

16. Сохранить полученную диаграмму:
   - в меню **File** выбрать **Save as**.

- указать путь к своей папке и имя файла Lab5.bp1.
- нажать OK.

**Задание 2**– Построить контекстную диаграмму своего варианта

**Задание 3** -Выполнить декомпозицию контекстной диаграммы

**Технология выполнения:**

1. Открыть файл Lab5.bp1

2. Следующим шагом является детализация контекстного процесса с помощью диаграммы верхнего уровня. Эта диаграмма содержит в себе четыре процесса:

Процесс 1.1 – ПЕРЕРАБОТКА СЫРЬЯ.

Процесс 1.2 – ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ.

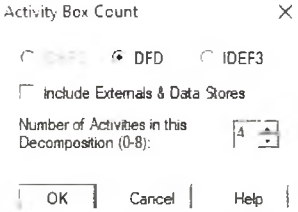
Процесс 1.3 – СБОРКА ИЗДЕЛИЯ.

Процесс 1.4 – КОНТРОЛЬ КАЧЕСТВА.

Произвести детализацию процесса «Изготовление мебели», задав нужное количество новых блоков. Для этого: щелкнуть по блоку «Изготовление мебели» и выбрать инструмент ▼ .

3. Указать тип DFD диаграммы и нажать OK.

4. В диалоговом окне ввести число, на которое будет произведена декомпозиция – 4.




5. Указать названия новых блоков: «Переработка сырья», «Изготовление деталей», «Сборка изделия», «Контроль качества».

6. При декомпозиции функции, входящие в нее и исходящие из нее дуги, автоматически появляются на диаграмме декомпозиции (миграция дуг), но при этом не касаются блоков. Такие стрелки называются несвязанными и воспринимаются в BPwin как синтаксическая ошибка (рис. 17).

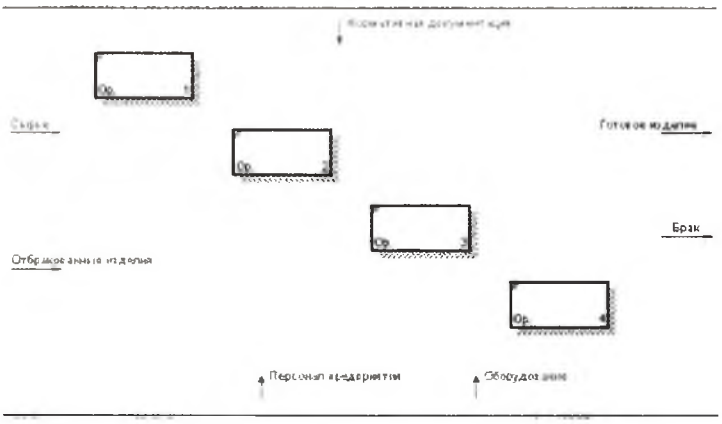


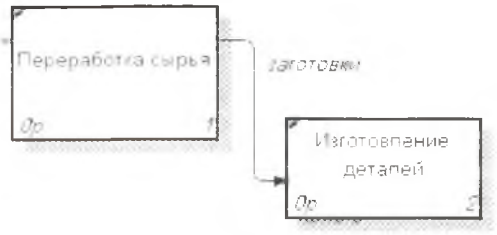

Рисунок 17 – Декомпозиция верхнего уровня

7. Выполнить детализацию процесса - Процесс 1.1. ПЕРЕРАБОТКА СЫРЬЯ:
Вход – СЫРЬЁ. Вход – ОТБРАКОВАННЫЕ ИЗДЕЛИЯ. Выход – ЗАГОТОВКИ.
Произвести процесс связывания мигрирующих дуг:

Выбрать инструмент → рисования дуг. Щелкнуть мышью по наконечнику входного потока СЫРЬЁ. Щелкнуть по входной стороне блока ПЕРЕРАБОТКА СЫРЬЯ.

8. Для построения выходного потока ЗАГОТОВКИ выполнить действия:

Выбрать инструмент → рисования дуг. Щелкнуть левой кнопкой мышки по выходной стороне блока ПЕРЕРАБОТКА СЫРЬЯ. Затем щелкнуть по входной стороне блока ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ. Выберите инструмент T текст, в контекстном меню – команду Name, укажите название дуги ЗАГОТОВКИ.




9. Самостоятельно выполнить детализацию процессов (рис. 108):
Процесс 1.2. ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ:
Вход – ЗАГОТОВКИ. Выход – ГОТОВЫЕ ДЕТАЛИ.
Процесс 1.3. СБОРКА ИЗДЕЛИЯ:
Вход – ГОТОВЫЕ ДЕТАЛИ. Выход – СОБРАННОЕ ИЗДЕЛИЕ.
Процесс 1.4. КОНТРОЛЬ КАЧЕСТВА:
Вход – СОБРАННОЕ ИЗДЕЛИЕ. Выход – ГОТОВОЕ ИЗДЕЛИЕ.
Выход – БРАК. Выход – ПРОИЗВОДСТВЕННЫЕ ОТХОДЫ

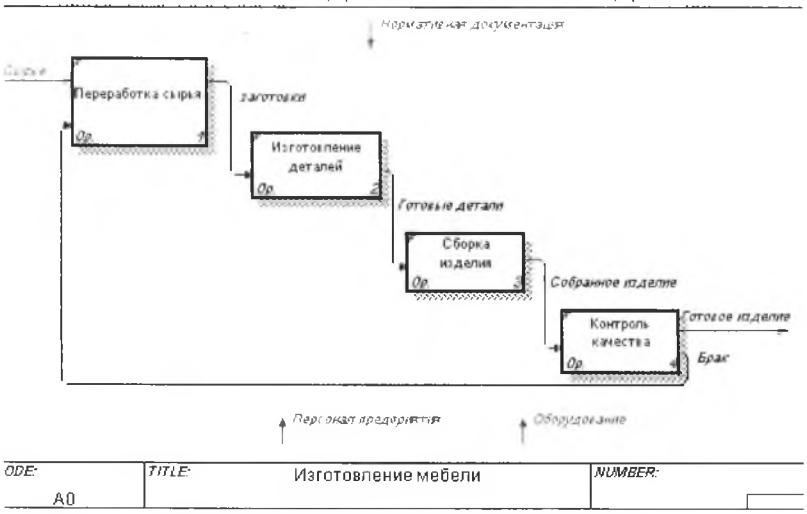



Рисунок 18 - Процесс декомпозиции

10. Построить ответвления дуг.

Управляющая стрелка *Нормативная документация* есть ответвления: Нормы переработки сырья, Чертеж детали, Сборочный чертеж, Стандарт качества.

Выбрать инструмент → рисование дуг. Щелкните мышью по наконечнику входного потока Нормативная документация. Щелкните по входной стороне блока *Переработка сырья.*

11. Самостоятельно выполнить ответвления дуги *Нормативная документация* на блоки Изготовление деталей, Сборка изделия, Контроль качества (рис.19)

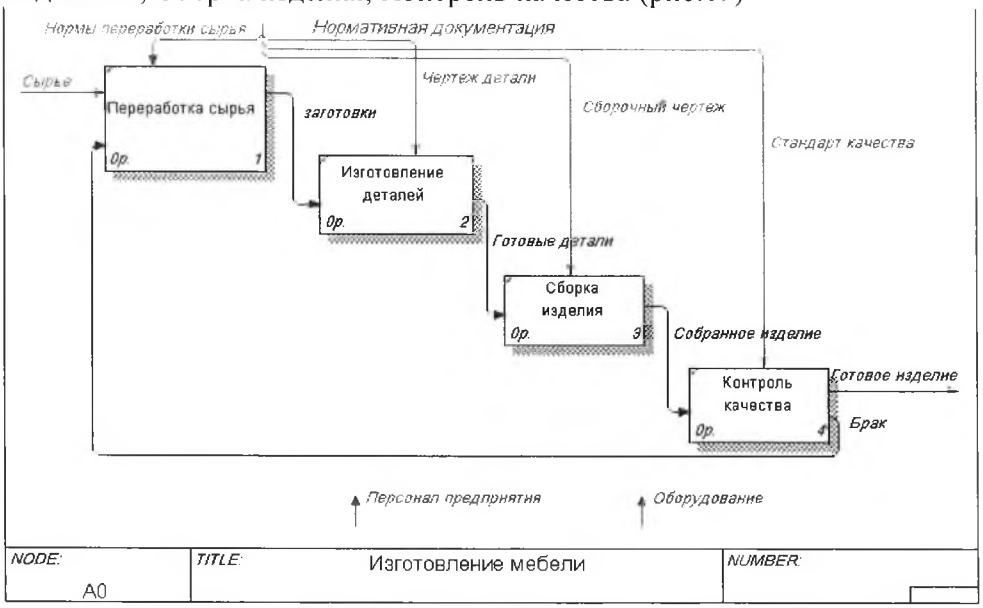



Рисунок 19 - Ответвления дуги Нормативная документация

12. Построить дуги Персонал предприятия, Оборудование (рис.20)

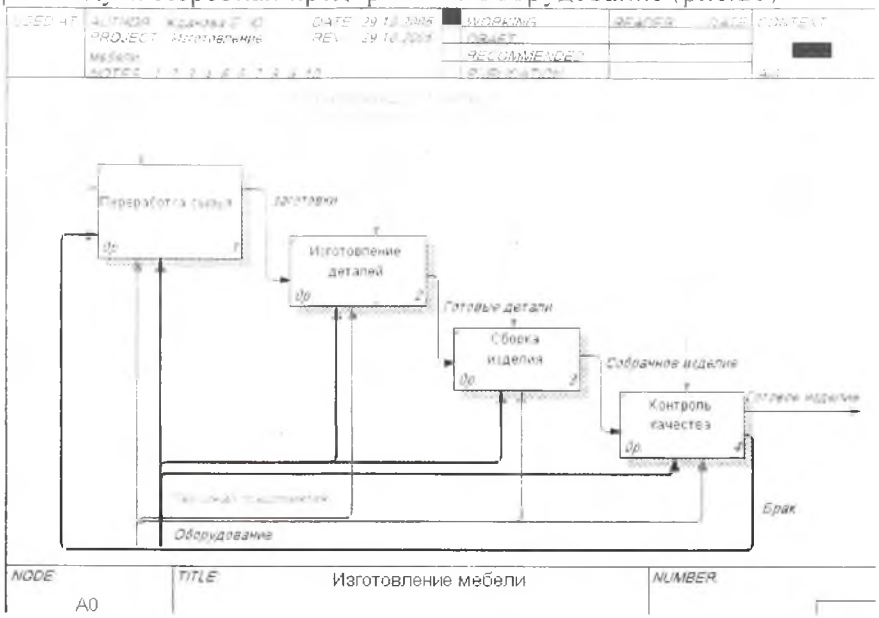



Рисунок 20 - Построение дуг Персонал предприятия и Оборудование

13. Выполнить «Тоннелирование» стрелок.

– в Процессе 1.2. ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ постройте новую граничную дугу, которой обозначьте Выход – ПРОИЗВОДСТВЕННЫЕ ОТХОДЫ.

– Выбрать инструмент редактирования. Кликнуть правой кнопкой мыши по квадратным скобкам. Выбрать в контекстном меню пункт **Arrow Tunnel**. В появившемся диалоге **Border Arrow Editor** (рис. 21) щелкнуть по кнопке **Resolve it to border arrow** для миграции стрелки на диаграмму верхнего уровня или по кнопке **Change it to resovled rounded** tunnel для «тоннелирования» дуги.

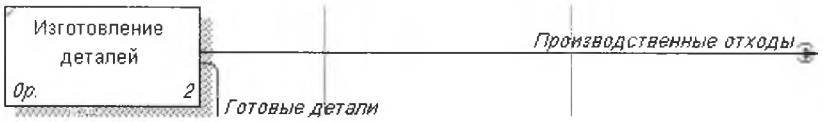



Рисунок 21 - Граничная дуга

– Отправить созданную дугу "Производственные отходы" в тоннель.

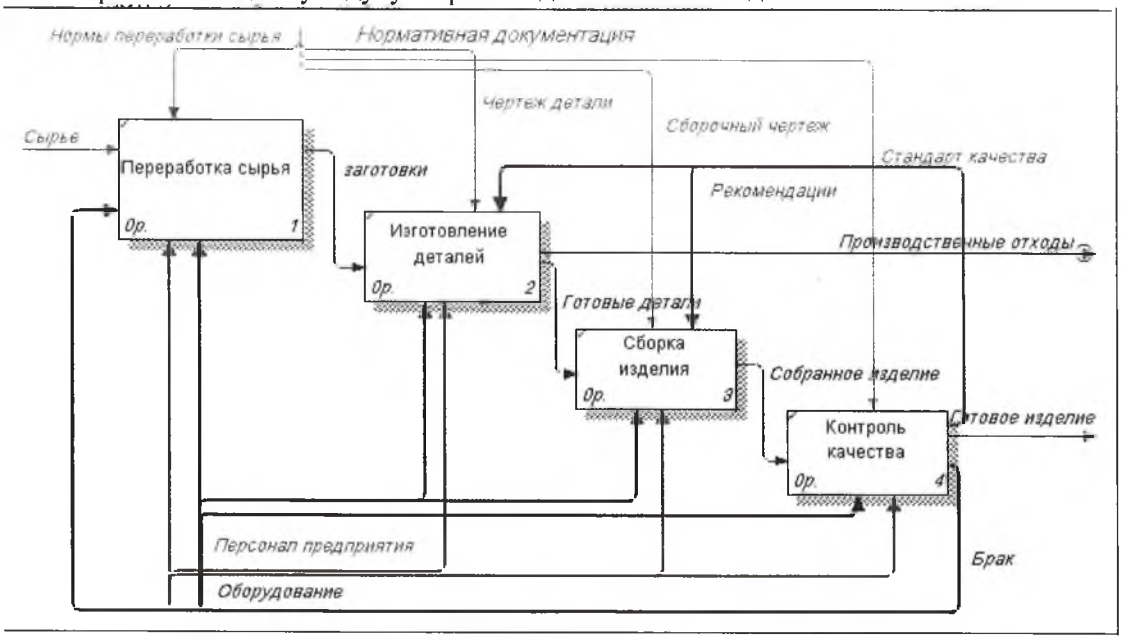



Рисунок 22 - Декомпозиция контекстной диаграммы

14. Сохранить полученную диаграмму

**Задание 4** - Создать диаграмму DFD-системы, организующей работу банкомата по обслуживанию клиента по его кредитной карте

**Технология выполнения:**

1. Создать новую модель: выбрать **File|New**, в поле **Name** ввести *Работа банкомата*, в области **Type** выбрать **DataFlow (DFD)**, нажать ОК, в поле **Autor** ввести ваше имя.
2. Ввести имя процесса и описание: выбрать процесс, нажать МП, выбрать **Name**, ввести *Обслужить*, выбрать вкладку **Defenition**, в поле **Defenition** ввести определение *Процесс обслуживания клиента по его кредитной карте с помощью банкомата*, выбрать вкладку **Color**, выбрать цвет - розовый, нажмите ОК.
3. Добавить внешнюю сущность: нажать кнопку **External Reference Tool**, выбрать мышью место размещения - в верхнем левом углу, ввести имя сущности Клиент, нажать ОК, нажать кнопку **Poiter Tool.**
4. Ввести описание: выбрать внешнюю сущность *Клиент*, нажать МП, выбрать **Name**, выбрать вкладку **Definition**, в поле *Defenition* ввести определение *Клиент банка с кредитной картой*, выбрать вкладку **Color**, цвет – синий.
5. Добавить внешнюю сущность *Компьютер банка* в нижнем правом углу и ее описание Хранит информацию о счетах всех клиентов.
6. Ввести поток данных: нажать кнопку **Precedence Arrow Tool**, выбрать правую грань внешней сущности *Клиент*, выбрать левую грань процесса *Обслужить*, нажать кнопку **PoiterTool**.
7. Ввести имя и тип потока данных: выбрать созданный поток данных, нажать МП, выбрать **Name**, в поле *Arrow Name* ввести *Кредитная карта*, выбрать вкладку **Style**, в области **Shape** выбрать двунаправленную стрелку.
8. Ввести описание потока данных: выбрать поток данных *Кредитная карта*, нажать МП, выбрать вкладку **Definition**, в поле **Definition** ввести *Для банковского обслуживания клиент должен предоставить кредитную карту для автоматического считывания с нее информации (пароль, лимит денег, детали клиента).*
9. Ввести поток данных *Ключевые данные* и его описание *Для банковского обслуживания клиент должен ввести ключевые данные-пароль, запрос на обслуживание.*

Банковское обслуживание должно:

a. Выдать сообщение, приглашающее клиента ввести ключевые данные,

b. Выдать клиенту деньги,

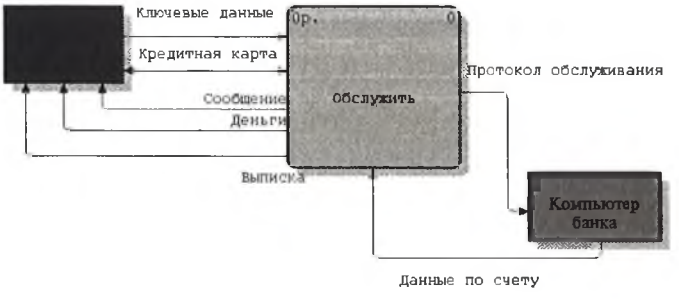
c. Выдать клиенту выписку по проведенному обслуживанию, включающую выписку о деньгах, выписку по балансу, выписку по операции, проведенной банком.

10. Ввести потоки данных *Сообщение, Деньги, Выписка.*
11. Ввести описание потоков *Сообщение, Деньги, Выписка.*
12. Процесс *Обслужить* и внешняя сущность *Компьютер банка* обмениваются следующей информацией:

a. Данные по счету клиента в банке,

b. Протокол обслуживания, включающий информацию об обработанной документации, изымаемой денежной сумме и данные по истории запроса.

13. Ввести потоки данных *Данные по счету, Протокол обслуживания.*
14. Ввести описание потоков *Данные по счету, Протокол обслуживания.*




15. Сохранить контекстную диаграмму

**Задание 5** - Создать диаграмму декомпозиции

1. Выберите процесс *Обслужить*, активизируйте на палитре инструментов кнопку декомпозиции, в окне **Activity Box Count** выберите **DFD**, в списке **Number of Activities...** выберите 4, выберите ОК.

2. Добавьте хранилище данных: нажмите кнопку **Date store tool**, введите имя хранилища *Данные кредитной карты*.

3. Добавьте еще раз хранилище данных *Данные кредитной карты* с целью избежать пересечений линий потоков данных.

4. Создать диаграмму согласно рисунка ниже (рис. 23).

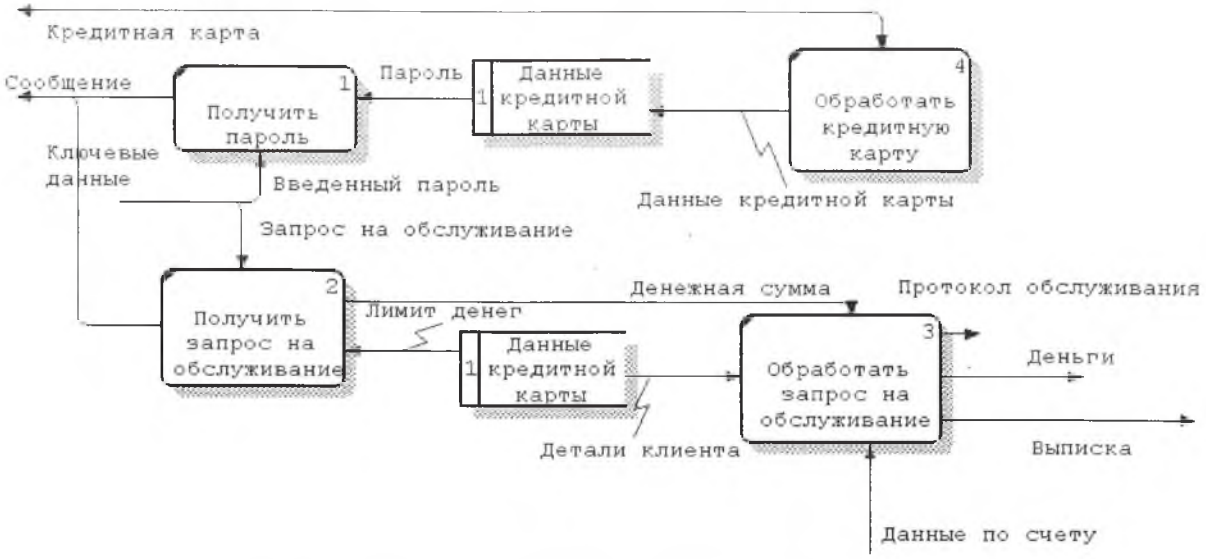



Рисунок 23 - Декомпозиция контекстной диаграммы

5. Сохранить диаграмму

**Контрольные вопросы**

1. Что такое поток данных?
2. Как изображается объект на DFD?
3. Как изображается накопитель на DFD?
4. Какие методологии поддерживаются в Process Modeler?
5. Какую роль играют в модели потоков данных цель и точка зрения?
6. Какие стрелки называют граничными?
7. Каково назначение разветвляющихся и сливающихся стрелок?

## ЛАБОРАТОРНАЯ РАБОТА № 6 (2 часа)

**Тема:** Разработка тестового сценария

**Цель:** Изучить порядок построения тестовых сценариев.

**Задание:**

Написать тест-кейсы, позволяющие детально протестировать функционал (по возможности используя техники тест дизайна), соответствующие вашей теме.

**Результат:**

- Заголовок тест-кейса
- Тестовые шаги и результаты (табл. 1)

Таблица 1 – Тест-кейс

| Номер шага | Действие | Ожидаемый результат |
|---|---|---|
| | | |
| | | |

**Варианты**

1. Проект «Реализация онлайн почтового клиента». Нужно описать тест-кейсы на функционал: Отправка сообщения. Путь к форме отправки сообщения: нажатие кнопки «Новое сообщение» на панели инструментов. Дизайн с комментариями прилагается.

2. Проект «Реализация системы библиотека». Нужно описать тест-кейсы на функционал «Регистрация пользователя». Путь к форме регистрации: главная страничка сайта, кнопка «Регистрация». Дизайн с комментариями прилагается.

3. Проект «Реализация системы поиска работы и сотрудников». Необходимо написать тест-кейсы на функционал «Поиск вакансии». Путь к форме поиска: главная страничка –кнопка поиск на панели инструментов. Дизайн с комментариями прилагается. В каждом выпадающем списке находится соответственно список городов, разделы сайта, зарплата от 10000 до 25000, от 25000 до 35000, от 35000 и выше.

4. Проект «Реализация сайта университета». Необходимо написать тест-кейсы на форму отправки пожелания о работе сайта, которую можно найти, пройдя по ссылке «Про нас» с главной странички сайта. Дизайн с комментариями прилагается.

**Контрольные вопросы**

1. Что такое тестирование?
1. Что такое тестовый сценарий?
2. Что используется в качестве основы для разработки тестового сценария?
2. Что такое тест кейс? Из чего состоит тест-кейс? Принципы написания тест-кейсов.

## ЛАБОРАТОРНАЯ РАБОТА № 7 (2 часа)

**Тема:** Оценка необходимого количества тестов

**Цель:** Изучить порядок оценки необходимого количества тестов

**Задание 1.** Даны длины сторон треугольника, определить вид треугольника и его площадь.

**1.1** Выполнить контроль вводимых чисел.

1. Разносторонний треугольник
2. Равнобедренный треугольник
3. Равносторонний треугольник

**Ограничения:**

- три числа не могут быть определены как стороны треугольника;
- если хотя бы одно из них меньше или равно 0;
- сумма двух из них меньше третьего.

**1.2** Требуется подготовить набор тестовых вариантов для обнаружения ошибок в программе.

Результат оформить в следующем виде:

| A | B | C | Ожидаемый результат | Объект проверки |
|---|---|---|---|---|
| Значение | Значение | Значение | Что должно получится | Значения вводимых данных, либо ожидаемый результат |
| ... | ... | ... | ... | ... |

**1.3** На основании проведенных тестов составьте рекомендации по исправлению ошибок, выявленных в ходе тестирования в виде отчета.

**Задание № 2** Найти минимальный набор тестов для программы вычисления функции с проверкой области допустимых значений (ОДЗ)

| Вариант | Функция |
|---|---|
| 1 | $y = \begin{cases} x - 1, & x \in (-\infty;-5) \\ x^2 + 2, & x \in [5;+\infty) \end{cases}$ |
| 2 | $y = \begin{cases} x^3 - 2x, & x \in [-20;0) \\ x - x^2, & x \in (20;40] \end{cases}$ |

| | |
|---|---|
| 3 | $x = \begin{cases} \frac{x}{2} + 5, & x \in [-15;1] \cup (15;30) \\ x^2 + \frac{1}{x}, & x \in [5;10] \cup (40;+\infty) \end{cases}$ |
| 4 | $y = \begin{cases} x^2 - 2, \text{ если } x > 5, \\ x - x^2, \text{ если } x \le 5. \end{cases}$ |
| 5 | $y = \begin{cases} x^3, \text{ если } x > 0, \\ 1, \text{ если } x = 0, \\ 1/x, \text{ если } x < -1. \end{cases}$ |
| 6 | $y = \begin{cases} x + 1, \text{ если } x > 1, \\ 0, \text{ если } x = 1, \\ x - 1, \text{ если } x < -1. \end{cases}$ |
| 7 | $y = \frac{1}{1 - x^2} + \sqrt{x + 2}$, x≠1 илих≠-1 и x≥-2 |
| 8 | $y = \begin{cases} \ln x, \text{если } x > 1, \\ x + \sin x, \quad \text{если } 0 < x \le 1, \\ x^2, \text{если } x \le 0, \end{cases}$ |
| 9 | $y = \left\{ \begin{matrix} x + 1, \ x \in (-\infty;-10) \\ \sqrt{x^3 - 2}, x \in [2;+\infty) \end{matrix} \right\}$ |
| 10 | $y = \begin{cases} x^3, \text{ если } x > 0, \\ 10, \text{ если } x = 0, \\ 1/x, \text{ если } x < 0. \end{cases}$ |
| 11 | $y = \frac{\ln x}{x} + \frac{\sqrt{x+2}}{1+x}$ x≠0и x≠-1 и x≥-2и x>0 |
| 12 | $y = \begin{cases} \sin x, \text{если } x > 1, \\ e, \text{если } x = 1, \\ \lvert x \rvert, \text{если } x < -1. \end{cases}$ |
| 13 | $y = \frac{2x}{x-3} + \frac{\sqrt{x+2}}{x}$ x≠3и x≠0 и x≥-2 |
| 14 | $y = \begin{cases} x^2, \text{если } x > 0, \\ 0, \text{если } x = 0, \\ x^3, \text{если } x < 0. \end{cases}$ |
| 15 | $y = \frac{x^3 - 2}{x^4 - 16} + \frac{x+1}{x}$ x≠4или x≠-4 и x≠0 |
| 16 | $y = \begin{cases} \sqrt{x} + 2, \text{если } x > 7, \\ x - x^2, \text{если } x \le 7. \end{cases}$ |

**Технология выполнения задания:**

1. Составить программу средствами VBAв программе Excel
2. Разработайте набор тестовых сценариев (как позитивных, так и негативных)

Результат оформить в следующем виде:

| Номер шага | X | Ожидаемый результат Y | Фактический результат | Объект проверки |
|---|---|---|---|---|
| 1 | Значение | Что должно получится | Что получилось | Значения вводимых данных, либо ожидаемый результат |
| 2 | … | … | | … |

**Контрольные вопросы**

1. Что такое покрытие кода тестами?
2. Как правильно оценивать покрытие кода тестами?
3. Что используется для определения покрытия тестами

## ЛАБОРАТОРНАЯ РАБОТА № 8 (2 часа)

**Тема:** Разработка тестовых пакетов

**Цель:** Изучить порядок разработки тестовых пакетов

**Задание - Разработка тестовых пакетов**

**Технология выполнения задания:**

**1. Создайте проект для тестирования**

1. Запустите **Visual Studio.**

2.Создать проект.

5.В поле *Имя* введите *Bank* и нажмите *OK*.

Будет создан новый проект Bank. Этот проект отобразится в обозревателе решений, а его файл Class1.cs откроется в редакторе кода.

6.Замените содержимое файласледующими кодом на C#, который определяет класс *BankAccount*:

```
using System;
namespace BankAccountNS
{
   /// <summary>
   /// Bank account demo class.
   /// </summary>
public class BankAccount
   {
private read only string m_customerName;
private doublem_balance;
private BankAccount() { }
public BankAccount(string customerName, double balance)
      {
m_customerName = customerName;
m_balance = balance;
      }
public string CustomerName
      {
get { return m_customerName; }
      }
public double Balance
      {
get { return m_balance; }
      }
public void Debit(double amount)
      {
if (amount >m_balance)
         {
throw new ArgumentOutOfRangeException("amount");
         }
if (amount < 0)
         {
throw new ArgumentOutOfRangeException("amount");
         }
m_balance += amount; // intentionally incorrect code
}
public void Credit(double amount)
      {
if (amount < 0)
         {
throw new ArgumentOutOfRangeException("amount");
         }
m_balance += amount;
      }
public static void main()
```

```
    }
    BankAccountba = new BankAccount("Mr. Bryan Walton", 11.99);
    ba.Credit(5.77);
    ba.Debit(11.22);
    Console.WriteLine("Current balance is ${0}", ba.Balance);
    }
}
```

7.Сохранить файл как BankAccount.cs.

8. В меню *Сборка* выбрать *Собрать решение*.

Будет создан проект с именем «Bank». Он содержит исходный код, подлежащий тестированию, и средства для его тестирования.

**2. Создание проекта модульного теста**

1. В меню Файл выбрать команду Добавить →Создать проект.

2. В диалоговом окне *Новый проект* развернуть узлы *Установленные и Visual C#* и выберите **Тест**.

3. В списке шаблонов выбрать **Проект модульного теста**.

4. В поле Имя ввести BankTests, а затем нажать кнопку ОК. Проект BankTests добавляется в решение Банк.

5. В проекте BankTests добавить ссылку на проект Банк.

В меню **Проект** выбрать команду **Добавить ссылку**.

6. В диалоговом окне **Диспетчер ссылок** развернуть **Решение** и проверить элемент *Банк*.

7.Создание тестового класса

Создание тестового класса, чтобы проверить класс BankAccount. Можно использовать UnitTest1.cs, созданный в шаблоне проекта, но лучше дать файлу и классу более описательныеимена. Можно сделать это за один шаг, переименовав файл в обозревателе решений.

8.Переименование файла класса

В обозревателе решений выберите файл UnitTest1.cs в проекте BankTests. В контекстном меню выберите команду *Переименовать*, а затем переименуйте файл в BankAccountTests.cs. Выберите *Да* в диалоговом окне, предлагающем переименовать все ссылки на элемент кода UnitTest1 в проекте.

Этот шаг изменяет имя класса на BankAccountTests. Файл BankAccountTests.cs теперь содержит следующий код:

9. Добавление оператора using в тестируемый проект

Можно добавить оператор using в класс, чтобы тестируемый проект можно было вызывать без использования полных имен. Вверху файла класса добавьте:

using BankAccountNS;

10. Требования к тестовому классу

Минимальные требования к тестовому классу следующие:

– Атрибут [TestClass] является обязательным для платформы модульных тестов Microsoft для управляемого кода в любом классе, содержащем методы модульных тестов, которые необходимо выполнить в обозревателе тестов.

– Каждый метод теста, предназначенный для запуска в обозревателе тестов, должен иметь атрибут [TestMethod].

Можно иметь другие классы в проекте модульного теста, которые не содержат атрибута [TestClass], а также иметь другие методы в тестовых классах, у которых атрибут – [TestMethod]. Можно использовать эти другие классы и методы в методах теста.

11. Создание первого тестового метода

В этой процедуре написать методы модульного теста для проверки поведения метода Debit класса BankAccount. Метод Debit.

Существует по крайней мере три поведения, которые требуется проверить:

– Метод создает исключение ArgumentOutOfRangeException, если сумма по дебету превышает баланс.

– Метод создает исключение ArgumentOutOfRangeException, если сумма по дебету меньше нуля.

– Если значение дебета допустимо, то метод вычитает сумму дебета из баланса счета.

12. Создание метода теста

Первый тест проверяет, снимается ли со счета нужная сумма при допустимом размере кредита (со значением меньшим, чем балане счета, и большим, чем ноль). Добавьте следующий метод в этот класс BankAccountTests:

```
[TestMethod]
public void Debit_WhenAmountIsMoreThanBalance_ShouldThrowArgumentOutOfRange()
{
// Arrange
double beginningBalance = 11.99;
double debitAmount = 20.0;
BankAccountaccount = newBankAccount(«Mr. BryanWalton», beginningBalance);
// Act
try
{ account.Debit(debitAmount);
}
catch (ArgumentOutOfRangeException e)
{
// Assert
StringAssert.Contains(e.Message, BankAccount.DebitAmountExceedsBalanceMessage);
return;
}
Assert.Fail(«The expected exception was not thrown.»);
}
```

13. Требования к методу теста

Метод теста должен удовлетворять следующим требованиям:

- Он декорируется атрибутом [TestMethod].
- Он возвращает void.
- Он не должен иметь параметров.

14. Сборка и запуск теста

1. В меню Сборка нажмите **Построить решение** (CTRL + SHIFT + B).

2. Откройте Обозреватель тестов, выбрав Тест – Окна - Обозреватель тестов в верхней строке меню (или нажмите клавиши CTRL + E, T).

3. Выберите *Запустить все*, чтобы выполнить тест (CTRL + R, V).

Если ошибок нет, появится обозреватель тестов с элементом Debit_WithValidAmount_UpdatesBalance в группе **Незапускавшиеся тесты**.

Если обозреватель тестов не откроется после успешной сборки, выберите в меню пункт Тест, щелкните Окна, а затем – Обозреватель тестов.

2. Выберите Запустить все, чтобы выполнить тест. Во время выполнения теста в верхней части окна отображается анимированная строка состояния. По завершении тестового запуска строка состояния становится зеленой, если все методы теста успешно пройдены, или красной, если какие-либо из тестов не пройдены.

3. В данном случае тест пройден не будет. Метод теста будет перемещен в группу Неудачные тесты. Выберите этот метод в обозревателе тестов для просмотра сведений в нижней части окна.

15. Исправление кода и повторный запуск тестов

1. Анализ результатов теста

Результат теста содержит сообщение, описывающее возникшую ошибку. Для метода AreEqual сообщение отражает ожидаемый результат (параметр Ожидается <значение>) и фактически полученный (параметр Фактическое<значение>). Ожидалось, что баланс уменьшится, а вместо этого он увеличился на сумму списания.

Модульный тест обнаружил ошибку: сумма списания добавляется на баланс счета, вместо того чтобы вычитаться.

2. Исправление ошибки

Чтобы исправить эту ошибку, в файле BankAccount.cs замените строку:

```
m_balance += amount;
```

на:

```
m_balance -= amount;
```

16. Повторный запуск теста

В обозревателе тестов выберитеЗапустить все, чтобы запустить тест повторно. Красно-

зеленая строка состояния станет зеленой, сигнализируя о том, что тест пройден, а сам тест будет перемещен в группу Пройденные тесты.

17.Создание методов теста

Создадим метод теста для проверки правильного поведения в случае, когда сумма по дебету меньше нуля:

18. Запуск тестов

Запуск теста показывает, что тест пройден.

19. Использование модульных тестов для улучшения кода

Рассмотреть, как последовательный процесс анализа, разработки модульных тестов и рефакторинга может помочь сделать рабочий код более надежным и эффективным.

20. Анализ проблем

Создали тестовый метод для подтверждения того, что допустимая сумма правильно вычитается в методе Debit.

**Контрольные вопросы**

1. Расскажите порядок создания модульного теста в VS.
2. Что такое рефакторинг кода?
3. Как провести рефакторинг, используя модульные тесты?

## ЛАБОРАТОРНАЯ РАБОТА № 9 (2 часа)

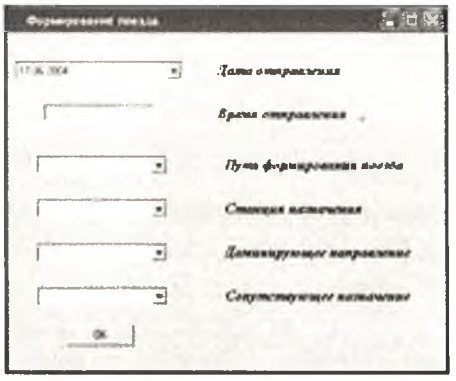
**Тема:** Оценка программных средств с помощью метрик

**Цель:** Изучить порядок оценки программных средств с помощью метрик

**Задание 1** - Рассчитать функциональность пользовательских форм, которая будет использована в конечном продукте.

**Типовой пример:**

1) Рассчитать функциональность одной из пользовательских форм, которая будет использована в конечном продукте.



При расчетах по функционально-ориентированной метрике используется 5 информационных характеристик:

1. Количество внешних вводов: 1 (кнопка ОК); данный элемент ввода состоит из 7 элементов данных (1 поле ввода, 5 полей со списком, 1 командная кнопка).

2. Количество внешних выводов: 1 (сообщение уведомления); элемент вывода состоит из 1 элемента данных (командная кнопка).

3. Количество внешних запросов: 0.

4. Количество внутренних логических файлов: 4 (справочник Доминирующее направление, справочник сопутствующее Направление, таблица Станции, таблица Пути); таблица Станции состоит из 6 элементов данных, справочникДоминирующееНаправление, справочниксопутствующееНаправлениеитаблицаСтанции- из 3.

5. Количество внешних интерфейсных файлов: 0.

Для каждой информационной характеристики по таблицам определяются ранг и оценка сложности.

После сбора всей необходимой информации подсчитаем **общую функциональную метрику** (см. табл. 2).

Таблица 2 – Общая функциональная метрика

| Имя характеристики | Ранг, сложность, количество | | | |
|---|---|---|---|---|
| | Низкий | Средний | Высокий | Итого |
| Внешние вводы | 0 * 3 = 0 | 1 * 4 = 4 | 0 * 6 = 0 | 4 |
| Внешние выводы | 1 * 4 = 4 | 0 * 5 = 0 | 0 * 7 = 0 | 4 |
| Внешние запросы | 0 * 3 = 0 | 0 * 4 = 0 | 0 * 6 = 0 | 0 |
| Внутренние логические файлы | 4 * 7 = 28 | 0 * 10 = 0 | 0 * 15 =0 | 28 |
| Внутренние интерфейсные файлы | 0 * 5 = 0 | 0 * 7 = 0 | 0 * 10 = 0 | 0 |
| | | | **Общее количество** | =36 |

2) Аналогичным образом рассчитаем функциональность второго типа пользовательской формы.
3) Результаты расчета в таблице 3.

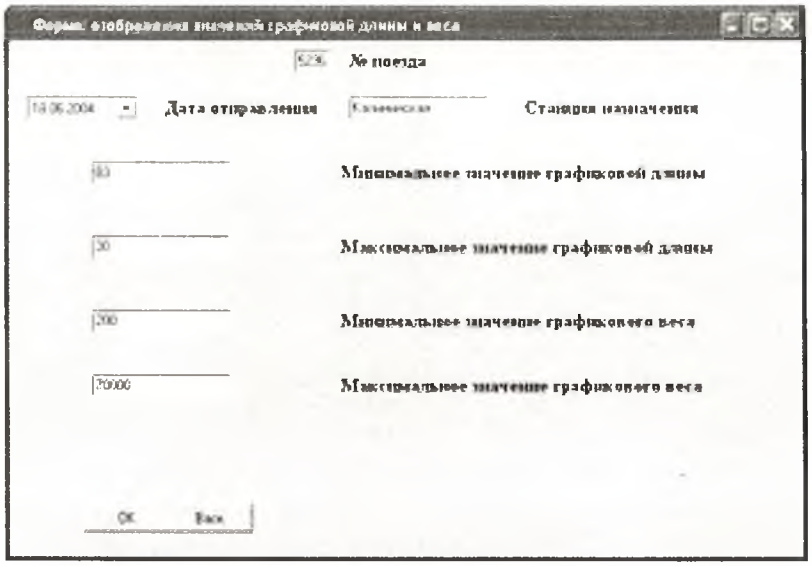



Таблица 3 – Общая функциональная метрика

| **Имя характеристики** | **Ранг, сложность, количество** | | | |
|---|---|---|---|---|
| | **Низкий** | **Средний** | **Высокий** | **Итого** |
| Внешние вводы | 0 * 3 = 0 | 1 * 4 = 4 | 0 * 6 = 0 | 4 |
| Внешние выводы | 1 * 4 = 4 | 0 * 5 = 0 | 1 * 7 = 7 | 11 |
| Внешние запросы | 1 * 3 = 3 | 0 * 4 = 0 | 0 * 6 = 0 | 3 |
| Внутренние логические файлы | 2 * 7 = 14 | 0 * 10 = 0 | 0 * 15 =0 | 14 |
| Внутренние интерфейсные файлы | 0 * 5 = 0 | 0 * 7 = 0 | 0 * 10 = 0 | 0 |
| | | | **Общее количество** | **=32** |

С учетом того, что в проекте предполагается использование 3 пользовательских форм первого типа и 2 пользовательских форм второго типа, подсчитаем общую функциональную метрику для всего проекта:

FP = 3 * 36 + 2 * 32 = **172**

Таблица 4 - Определение системных параметров приложения

| № | Системный параметр | Описание | Коэф. |
|---|---|---|---|
| 1 | Передача данных | Сколько средств связи требуется для передачи или обмена информацией с приложением или системой? | 2 |
| 2 | Распределенная обработка данных | Как обрабатываются распределенные данные и функции обработки? | 3 |
| 3 | Производительность | Нуждается ли пользователь в фиксации времени ответа или производительности? | 3 |
| 4 | Распространенность используемой конфигурации | Насколько распространена текущая аппаратная платформа, на которой будет выполняться приложение? | 0 |
| 5 | Скорость транзакций | Как часто выполняются транзакции? (каждый день, каждую неделю, каждый месяц) | 5 |
| 6 | Оперативный ввод данных | Какой процент информации надо вводить в режиме онлайн? | 4 |
| 7 | Эффективность работы конечного пользователя | Приложение проектировалось для обеспечения эффективной работы конечного пользователя? | 5 |
| 8 | Оперативное обновление | Как много внутренних файлов обновляется в онлайновой транзакции? | 3 |
| 9 | Сложность обработки | Выполняет ли приложение интенсивную логическую или математическую обработку? | 2 |
| 10 | Повторная используемость | Приложение разрабатывалось для удовлетворения требований одного или многих пользователей? | 0 |
| 11 | Легкость инсталляции | Насколько трудны преобразование и инсталляция приложения? | 0 |
| 12 | Легкость эксплуатации | Насколько эффективны и/или автоматизированы процедуры запуска, резервирования и восстановления? | 2 |
| 13 | Разнообразные условия размещения | Была ли спроектирована, разработана и поддержана возможность инсталляции приложения в разных местах для различных организаций? | 0 |
| 14 | Простота изменений | Была ли спроектирована, разработана и поддержана в приложении простота изменений? | 0 |
| | | $\sum_{i=1}^{14} F_i =$ | **29** |

Каждый коэффициент может принимать следующие значения: 0 - нет влияния, 1 - случайное, 2 - небольшое, 3 - среднее, 4 - важное, 5 - основное.

Значения выбираются эмпирически в результате ответа на 14 вопросов, которые характеризуют системные параметры приложения (таблица 3).

В результате количество функциональных указателей равно:

FP = **Общее количество** x $(0{,}65+ 0{,}01 \times \sum_{i=1}^{14} F_i)$,

**FP = 172 * (0.65 + 0.01*29) = 162**

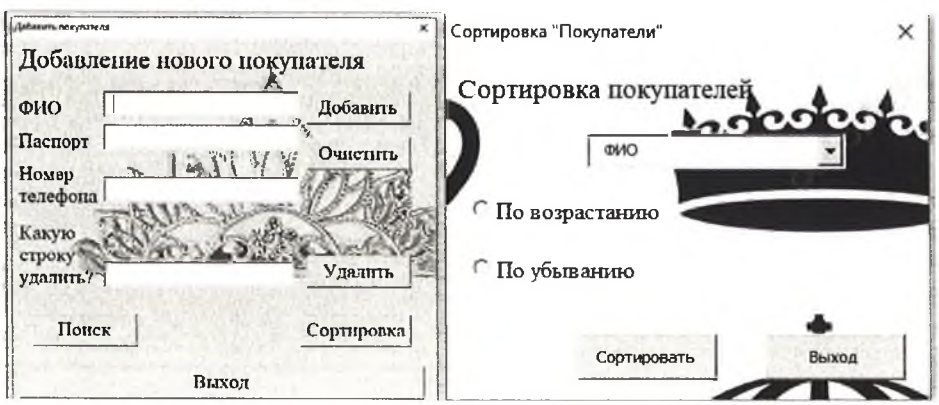
Используя таблицу перевода, а также учитывая, что реализация ИС предполагается с использованием языка Visual Basic, получим LOC-оценку проекта:

**162 * 32 = 5184 (строк кода).**

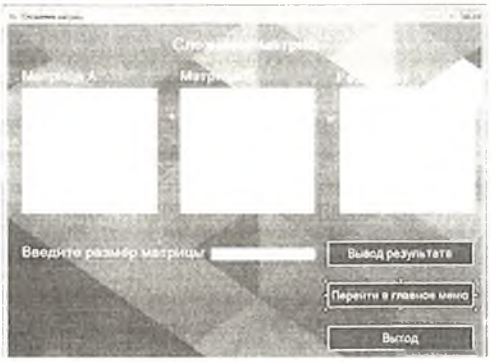
**Задание -** Рассчитать функциональность пользовательских форм одного варианта

**Варианты проектов:**

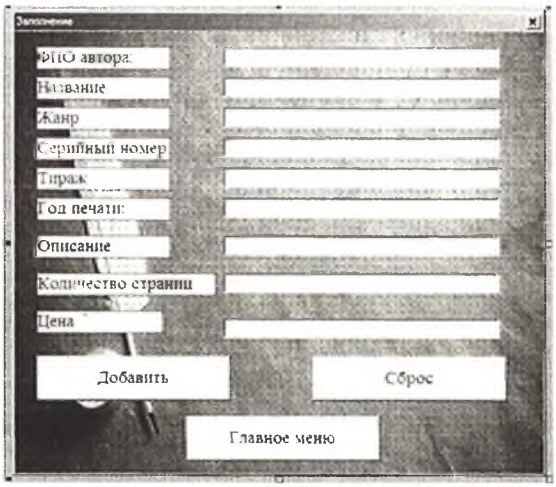
**1 вариант**– 2 формы 1 типа, 3 формы 2 типа

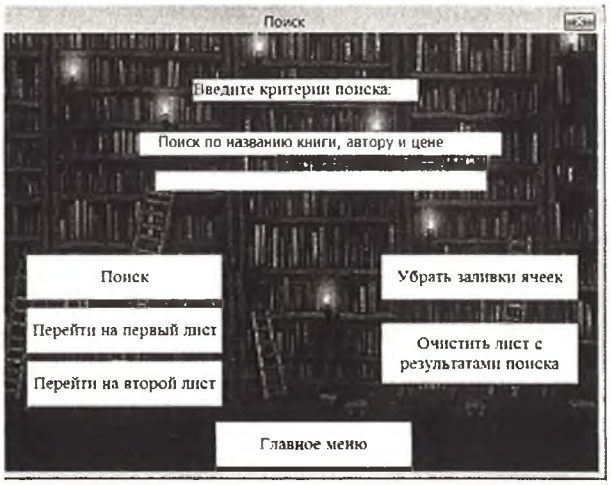


**2 вариант** – 5 форм 1 типа

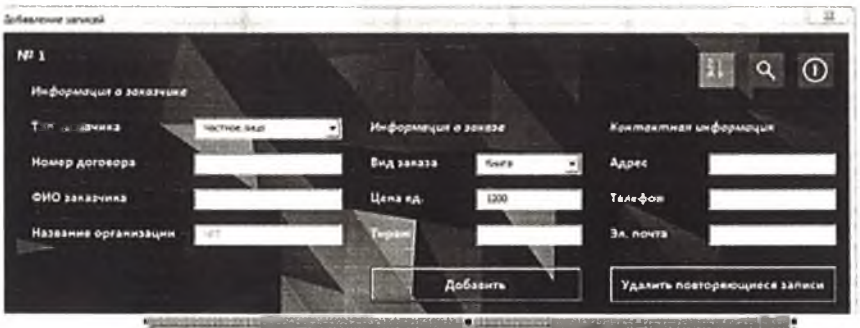


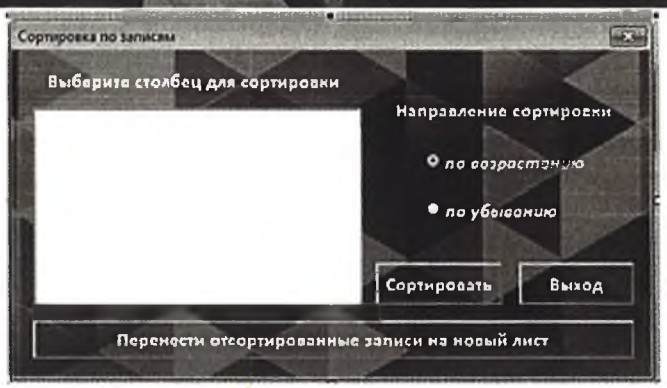

**3 вариант** – 2 формы 1 типа, 3 формы 2 типа



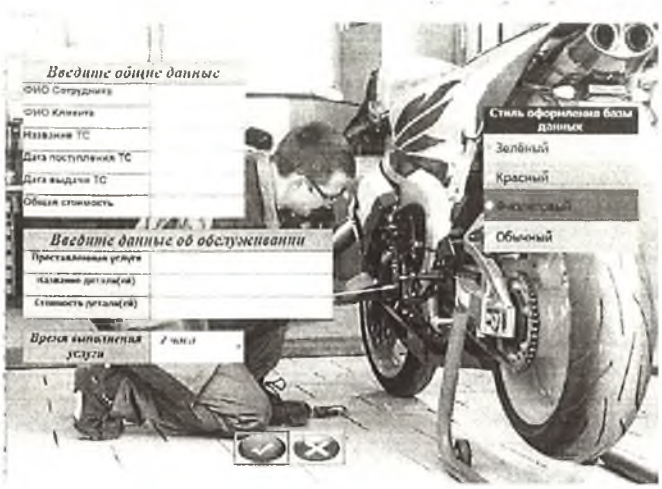



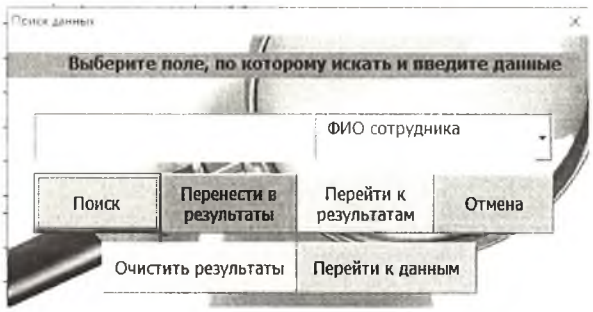

**4 вариант** – 2 формы 1 типа, 3 формы 2 типа



**5 вариант** – 3 формы 1 типа, 2 формы 2 типа




**6 вариант** – 1 форма 1 типа


**7 вариант** – 1 форма 1 типа

**Контрольные вопросы**

1. Что такое метрика качества кода?
2. Как определить качество кода, используя метрики?
3. Приведите порядок действия для оценки качества кода

## ЛАБОРАТОРНАЯ РАБОТА № 10 (2 часа)

**Тема:** Инспекция программного кода на предмет соответствия стандартам кодирования

**Цель:** Изучить порядок инспекции программного кода на предмет соответствия стандартам кодирования

**Задание 1** – Обзор инструментов для код-ревью.

**Технология выполнения:**

1.1 Рассмотреть 5 наиболее популярных инструменты для код-ревью.

1.2 Заполнить таблицу.

| № | Наименование | Описание | Область применения |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**Задание 2** - Выполнить анализ кода

**Технология выполнения:**

2.1 Запустить Visual Studio.

2.2 Открыть проект «Банк» (лаб. работа №8).

2.4 Выполнить анализ кода в решении.

2.5 Рассчитать метрики кода для решения.

2.6 Сделать анализ полученных результатов.

**Контрольные вопросы:**

1. Что такое CodeReview?
2. Какие плюсы ревьирования кода?
3. Как проводить CodeReview?
4. Почему требуется использовать инструменты для код-ревью?

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

Основные источники:

1. Заботина, Н. Н. Проектирование информационных систем : учебное пособие / Н. Н. Заботина. — Москва :ИНФРА-М, 2020. — 331 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-004509-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/1036508 (дата обращения: 10.02.2022). – Режим доступа: по подписке.

2. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учеб.пособие / Г.Н. Федорова. — М. :КУРС : ИНФРА-М, 2019. — 336 с. (Среднее Профессиональное Образование). - ISBN 978-5-906818-41-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/989682 (дата обращения: 12.02.2022). – Режим доступа: по подписке.

Дополнительные источники:

3. 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: 20.04.2022). – Режим доступа: по подписке.

Интернет-ресурсы:

1. Жизненный цикл программного продукта: [сайт]. – URL: https://qaevolution.ru/zhiznennyj-cikl-programmnogo-obespecheniya (дата обращения 10.05.2021). - Текст : электронный.

2. Моделирование на UML [сайт]. – URL: http://book.uml3.ru/content (дата обращения 10.03.2022). - Текст : электронный.

3. НОУ «ИНТУИТ» [сайт]. – Москва, 2003-2020.- URL: http://www.intuit.ru/studies/courses/ 617/473/lecture/20968 (дата обращения 02.03.2022). -Текст : электронный.

Составитель
Белугина Светлана Викторовна

**НАИМЕНОВАНИЕ КУРСА**
Методические указания по выполнению лабораторно-практических работ
для студентов очной формы обучения
по направлению специальности
09.02.07 «Информационные системы и программирование»

Публикуется в авторской редакции