

Министерство науки и высшего образования РФ
Федеральное государственное образовательное учреждение
высшего образования
«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

Направление подготовки
09.03.03 Прикладная информатика

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе по дисциплине «Информационные системы»

«Разработка кроссплатформенного программного продукта на языке JAVA с
использованием системы контроля версий»

Выполнили:
Ст. гр. ПИ-223
Пупков Е.А.
Леонтьев В.А.
Денисов Н.А.
Шамсутдинов Р.М.

Проверил:
Преподаватель
Казанцев А.В.

Уфа – 2021

ЗАДАНИЕ

на курсовую работу по дисциплине «Информационные системы»

Студент	<u>Пупков Е.А.</u> Фамилия И.О.	Группа	<u>ПИ-223</u> номер группы	Консультант	<u>Казанцев А.В.</u> Фамилия И.О.
Студент	<u>Шамсутдинов Р.М.</u> Фамилия И.О.	Группа	<u>ПИ-223</u> номер группы	Консультант	<u>Казанцев А.В.</u> Фамилия И.О.
Студент	<u>Денисов Н.А.</u> Фамилия И.О.	Группа	<u>ПИ-223</u> номер группы	Консультант	<u>Казанцев А.В.</u> Фамилия И.О.
Студент	<u>Леонтьев В.А.</u> Фамилия И.О.	Группа	<u>ПИ-223</u> номер группы	Консультант	<u>Казанцев А.В.</u> Фамилия И.О.

1.Тема курсового проекта:	<u>Разработка кроссплатформенного программного продукта на языке JAVA с использованием системы контроля версий.</u>
	наименование темы

2.Основное содержание:

1. Пояснительная записка с необходимыми материалами.
2. Репозиторий системы контроля версий содержащий программный код с комментариями и необходимую документацию.

3. Требования к оформлению:

3.1. Пояснительная записка должна быть оформлена в текстовом процессоре LibreOffice Writer в соответствии с требованиями СТО УГАТУ. Минимальные требования к оформлению: размер шрифта 14 пунктов; отступы от края листа: отступ слева 2 см. и остальные отступы 0.5 см. В бумажном виде оформляются: титульный лист, задание, календарный план и аннотация, которая содержит ссылку на репозиторий с программным кодом и документацией.

3.2. В пояснительной записке должны содержаться следующие разделы:

Раздел 1. Описание предметной области.

Раздел 2. Техническое задание на создание программного продукта.

Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux.

Раздел 4. Настройка среды разработки для подключения к системе контроля версий.

Раздел 5. Реализация исходного кода по зонам ответственности.

Раздел 6. Сборка и тестирование программного продукта.

Раздел 7. Настройка программной среды для развертывания и запуска программного продукта.

Раздел 8. Руководство пользователя программного продукта.

3.3. В приложение выносится программный код и код тестов.

4. Графическая часть должна включать:

- мнемосхема рассматриваемого процесса;
- диаграммы UML;
- экранные формы инструментальных средств;
- экранные формы, разрабатываемого программного продукта.

Дата выдачи 6 марта 2021 г.

Дата окончания 29 мая 2021 г.

Руководитель Казанцев А.В.

ФИО

План-график выполнения курсовой работы по дисциплине «Информационные системы»

Наименование этапа работ	Трудоемкость выполнения, час	Процент к общей трудоемкости выполнения	Срок предъявления консультанту	Отметка о выполнении
Получение и согласование задания	1,7	1,7%	27 неделя	
Раздел 1. Описание предметной области	20	20%	29 неделя	
Раздел 2. Техническое задание на создание программного продукта	10	10%	30 неделя	
Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux	10	10%	31 неделя	
Раздел 4. Настройка среды разработки для подключения к системе контроля версий	7	7%	32 неделя	
Раздел 5. Реализация исходного кода по зонам ответственности	23	23%	34 неделя	
Раздел 6. Сборка и тестирование программного продукта	8	8%	35 неделя	
Раздел 7. Настройка программной среды для развертывания и запуска программного продукта	10	10%	36 неделя	
Раздел 8. Руководство пользователя программного продукта	10	10%	37 неделя	
Защита	0,3	0,3%	38 неделя	

АННОТАЦИЯ

В курсовой работе рассматривается разработка кроссплатформенного программного продукта на языке JAVA с использованием системы контроля версий.

В первом разделе приведено описание рассматриваемой предметной области «Производство мягкой мебели».

Во втором разделе представлено техническое задание.

В третьем разделе описан процесс настройки среды разработки для ОС семейств Windows и Linux.

В четвертом разделе описан процесс настройки среды разработки для подключения к системе контроля версий.

В пятом разделе представлена таблица с распределением зон ответственности для реализации программного кода.

В шестом разделе представлено описание UNIT-тестов, процесса сборки проекта и структуры проекта, а также некоторых файлов проекта.

В седьмом разделе представлен файл .travis.yml, а также описание процесса настройки Travis-CI.

В восьмом разделе представлено руководство пользователя.

Ссылка на репозиторий: <https://github.com/PupkovEgor/kr24>

СОДЕРЖАНИЕ

АННОТАЦИЯ.....	4
Раздел 1. Описание предметной области.....	6
Раздел 2. Техническое задание на создание программного продукта.....	13
Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux.....	14
Раздел 4. Настройка среды разработки для подключения к системе контроля версий.....	23
Раздел 5. Реализация исходного кода по зонам ответственности.....	27
Раздел 6. Сборка и тестирование программного продукта.....	28
Раздел 7. Настройка программной среды для развертывания и запуска программного продукта.....	32
Раздел 8. Руководство пользователя программного продукта.....	34
ПРИЛОЖЕНИЕ 1.....	35
ПРИЛОЖЕНИЕ 2.....	36
ПРИЛОЖЕНИЕ 3.....	37
ПРИЛОЖЕНИЕ 4.....	38
ПРИЛОЖЕНИЕ 5.....	39
ПРИЛОЖЕНИЕ 6.....	40
ПРИЛОЖЕНИЕ 7.....	41
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	42

Раздел 1. Описание предметной области.

В данной курсовой работе была рассмотрена предметная область «Мебельное производство».

Мебельное производство — это сфера деятельности человека, направленная на производство мебели.

Мебельное производство подразделяется на множество видов, но в данной работе мы рассмотрим производство мягкой мебели.

Мягкая мебель — собирательное название комфортабельных мебельных изделий для сидения и лежания, имеющих мягкий элемент. К мягкой мебели относятся: диваны-кровати, диваны, кресла-кровати, кресла для отдыха, кушетки, тахты, скамьи, банкетки.

Наша компания «Стул Стульч» будет производить мягкие стулья, некоторые виды которых представлены на рисунке 1.



Рисунок 1- Виды стульев

Процесс изготовления начинается с поступления заказа от клиента, в качестве которого могут выступать физические и юридические лица. Затем этот заказ обрабатывается дизайнером, который работает с заказчиком, учитывает все его требования и пожелания. С учетом всего этого, а также данных по стандартам и размерам изделия создается индивидуальная модель (чертеж). Для того чтобы заказ был выполнен, необходима договоренность с поставщиками на поставку сырья на производство, где оно сортируется по материалу каркаса (металл, дерево, пластик) и материалу обивки (ткань, замша, велвет и т.д.). Затем сырье подлежит определенной обработке. После обработки из сырья получают детали для изготовления изделий. После того как готовы все части каркаса, их собирают. Из обивочного материала готовят чехлы, которые в будущем будут устанавливаться на каркас. После того как всё готово к сборке, готовые чехлы устанавливают на каркас-основу изделия с мягким наполнителем (поролон, синтепон, войлок). Проверка качества касается как деталей, изделий, так и готовой продукции. Обычный заказ выполняется, в среднем, за 2 недели, срочный - за 1 неделю. Мнемосхема данного бизнес-процесса показана на рисунке 2.

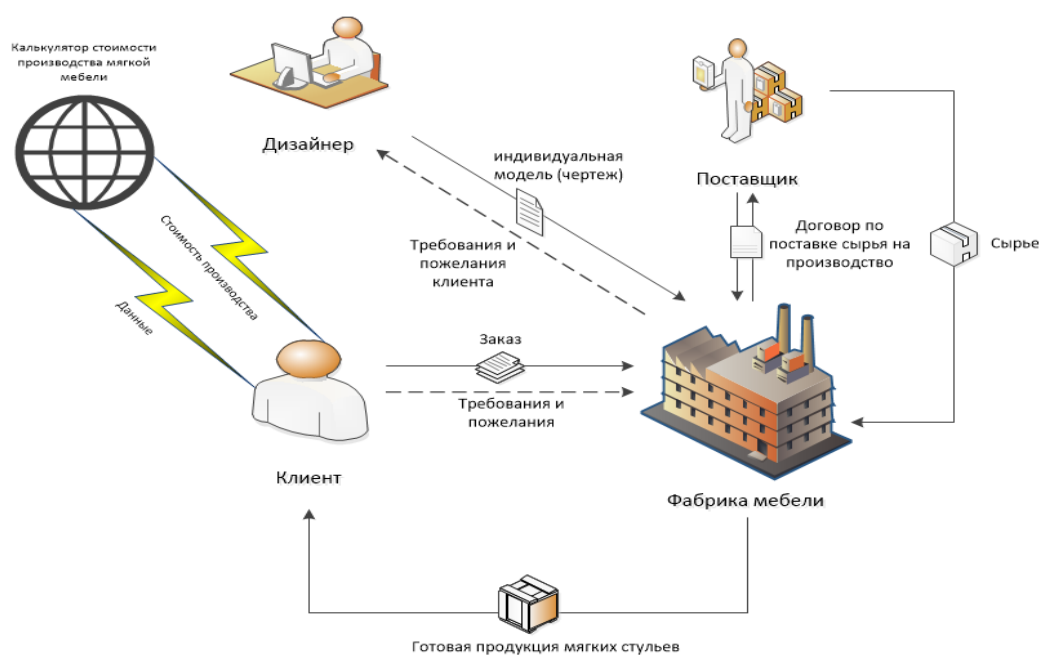


Рисунок 2 – Мнемосхема бизнес-процесса «Мебельное производство»

Цены на все материалы представлены в таблице 1.

Таблица 1 - Таблица цен.

Вид стула	Цена, руб	Материал каркаса	Цена, руб	Материал обивки	Цена, руб	Материал наполнителя	Цена, руб
Игровое кресло	3000	Металл	500	Искусственная кожа	500	Поролон	400
Офисное кресло	2000	Дерево	250	Эко-кожа	800	Синтепон	130
Пуфик	500	Пластик	100	Ткань	550	Войлок	2700
Кресло-качалка	1500	-	-	Замша	1800	-	-
Стул с мягкой спинкой	1000	-	-	Вельвет	830	-	-

В соответствии с предметной областью система строится с учетом следующих особенностей:

- изготовление каждого изделия состоит из нескольких стадий;
- стадии заключаются в изготовлении деталей и собирании их в готовое изделие;
- определенная стадия выполняется на определенном участке, подразделяющемся по номерам;
- приход и расход сырья определяет наименование поставщика и складирование;
- каждый участок состоит из бригад, бригады — соответственно из рабочих.

Основными документами регулирующие производственные процессы и предъявляющие требования к качеству являются:

- ГОСТ 21640-91 - Методы определения мягкости мебели для сидения и лежания
- ГОСТ 19917-2014 - Технические условия для мебели для сидения и лежания.
- ГОСТ 19120-93 Методы испытаний мебели для сидения и лежания
- ГОСТ 16371-2014 Мебель. Общие технические условия
- ГОСТ 13025.2-85 Функциональные размеры мебели для сидения и лежания
- Закон РФ от 07.02.1992 N 2300-1 (ред. от 31.07.2020) "О защите прав потребителей"

Математическая модель работы программы «Мебельный калькулятор».

Общая стоимость заказа рассчитывается по формуле:

$$C_{\text{общ}} = (C_{\text{к}} + C_{\text{об}} + C_{\text{р}}) * n \quad (1.1)$$

где:

$C_{\text{к}}$ - стоимость каркаса,

$C_{\text{об}}$ - стоимость обивки,

$C_{\text{р}}$ - стоимость работы,

n - количество стульев.

Стоимость работы рассчитывается по формуле:

$$C_{\text{р}} = C_{\text{вида}} * k \quad (1.2)$$

где:

$C_{\text{вида}}$ - цена производства определённого вида мебели

k – коэффициент скорости производства (если выбран пункт «срочность», равен 1,5, иначе 1.).

Стоимость обивки рассчитывается по формуле

$$C_{\text{об}} = C_{\text{м}} + C_{\text{нап}} \quad (1.3)$$

где:

$C_{\text{м}}$ - стоимость материала обивки,

$C_{\text{нап}}$ - стоимость наполнителя.

Для представления описания процесса в системе была построена диаграмма прецедентов (рис. 3)

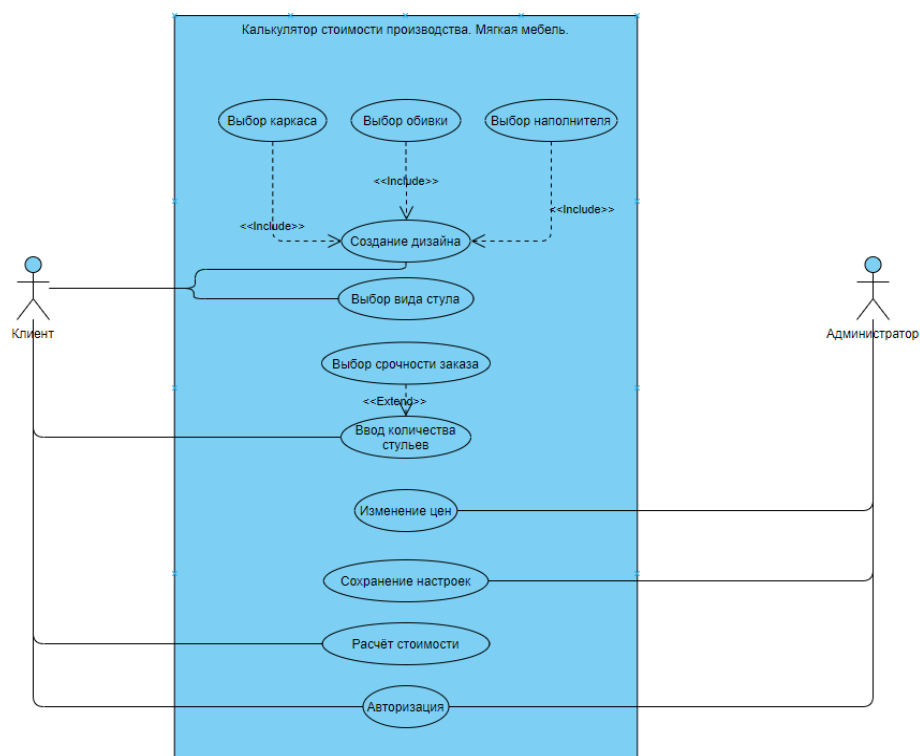


Рисунок 3 – Диаграмма вариантов использования.

Для понимания работы программы была построена диаграмма классов «Мебельного калькулятора» (рис.4).

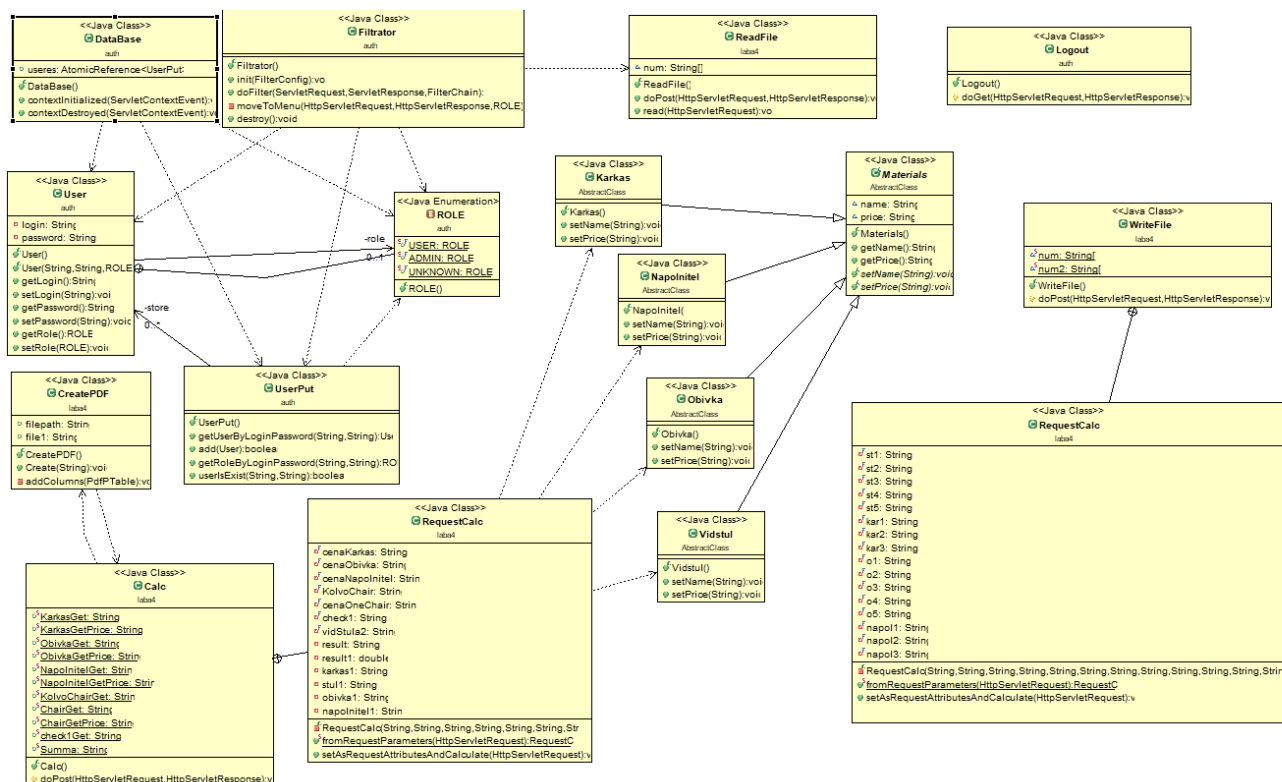


Рисунок 4 – Диаграмма классов.

Раздел 2. Техническое задание на создание программного продукта.

Смотрите Приложение 1.

Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux

Настройка операционной системы Windows 10:

Необходимо установить JDK, для этого необходимо скачать его с официального сайта (рис.6):

<https://www.oracle.com/ru/java/technologies/javase/javase-jdk8-downloads.html>











Linux x86 Compressed Archive	136.95 MB	 jdk-8u281-linux-i586.tar.gz
Linux x64 RPM Package	108.06 MB	 jdk-8u281-linux-x64.rpm
Linux x64 Compressed Archive	137.06 MB	 jdk-8u281-linux-x64.tar.gz
macOS x64	205.26 MB	 jdk-8u281-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.96 MB	 jdk-8u281-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.77 MB	 jdk-8u281-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.68 MB	 jdk-8u281-solaris-x64.tar.Z
Solaris x64	92.66 MB	 jdk-8u281-solaris-x64.tar.gz
Windows x86	154.69 MB	 jdk-8u281-windows-i586.exe
Windows x64	166.97 MB	 jdk-8u281-windows-x64.exe

Рисунок 6 – Версии JDK.

Для установки JDK следуем пунктам меню (рис.7-9).

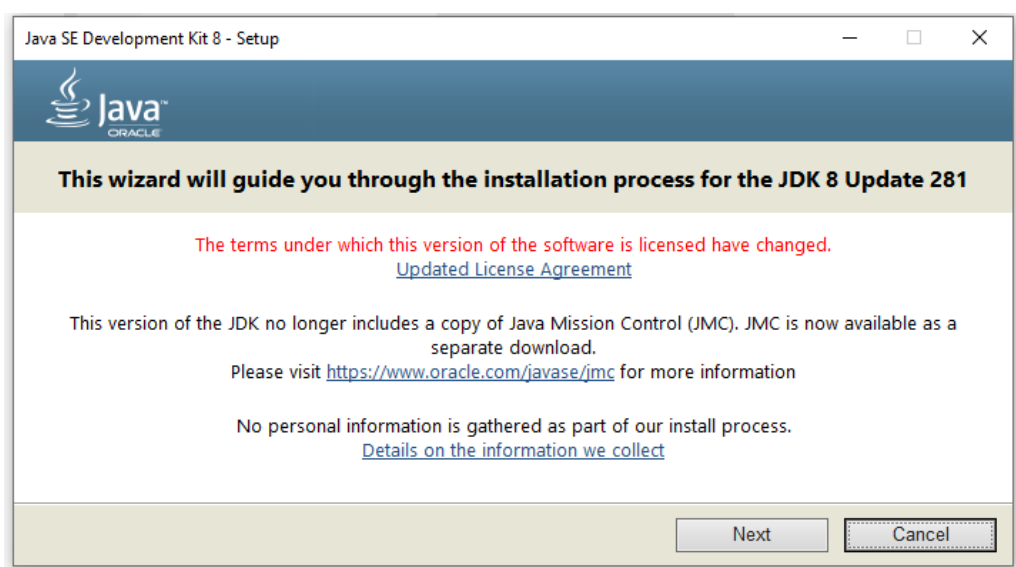


Рисунок 7 – Установка JDK, шаг 1.

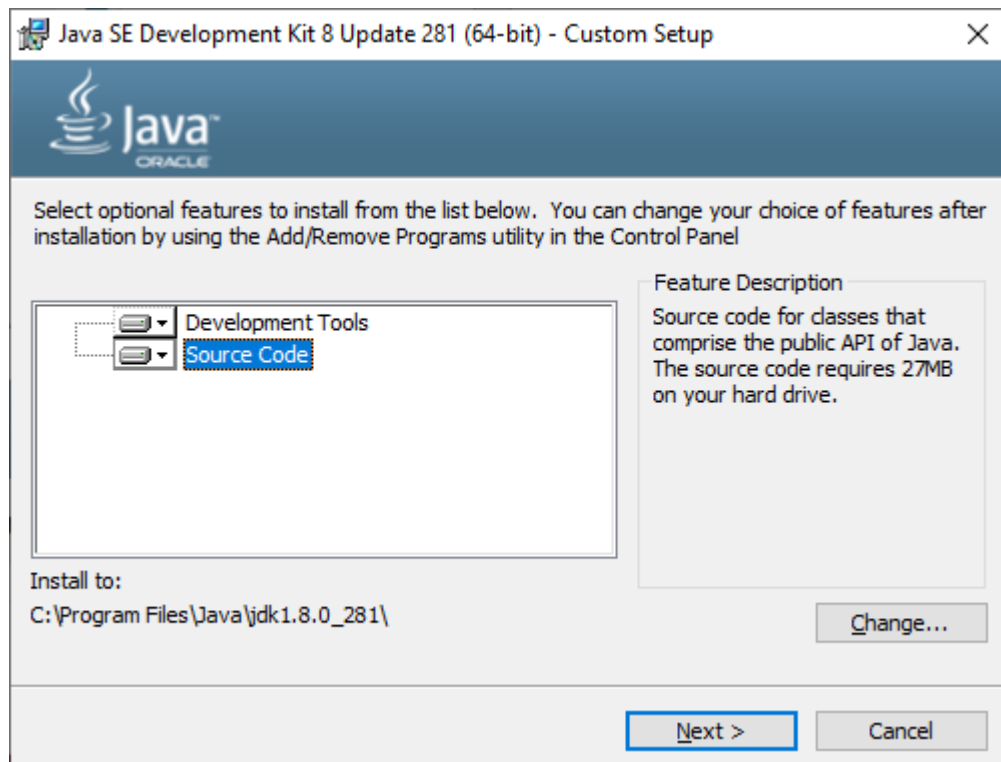


Рисунок 8 – Установка JDK, шаг 2.

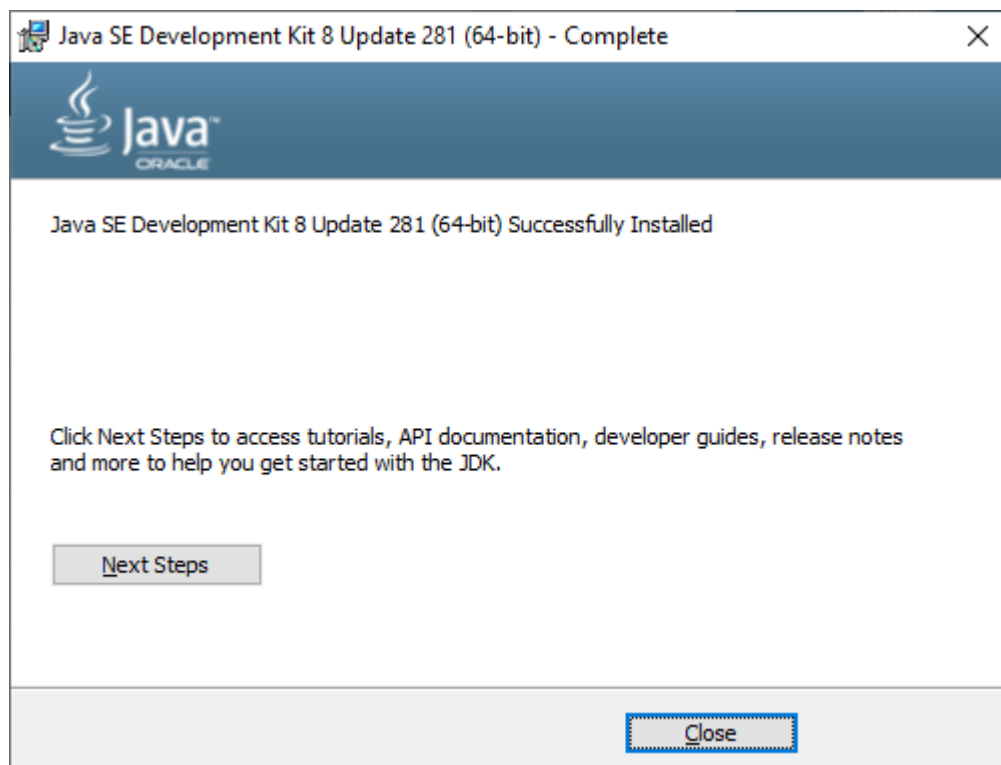


Рисунок 9 – Установка JDK, шаг 3.

Далее необходимо установить “Eclipse IDE for Enterprise Java and Web Developers” для этого переходим на официальный сайт: <https://www.eclipse.org/downloads/packages/>

Выбираем версию Windows x86_64 и скачиваем (рис.10).



Рисунок 10 – Скачивание Eclipse.

После скачивания распаковываем и запускаем. После запуска выбираем рабочий каталог и нажимаем “Launch” (рис.11).

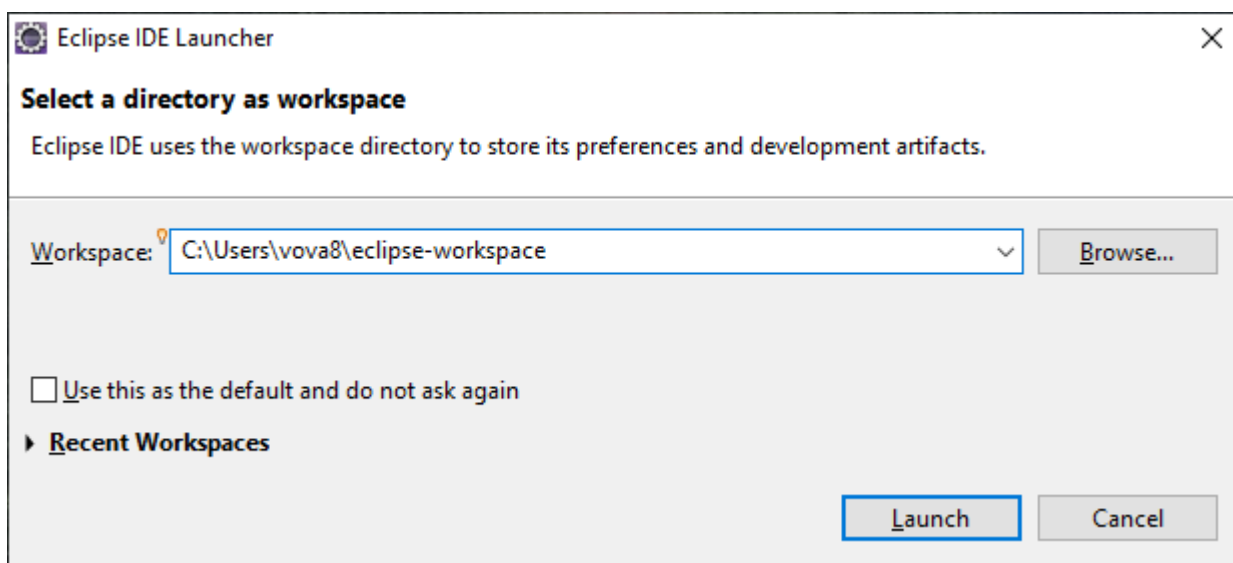


Рисунок 11 – Установка Eclipse.

Для использования git в eclipse необходимо открыть вкладку Window->Show View->Other (рис.12)

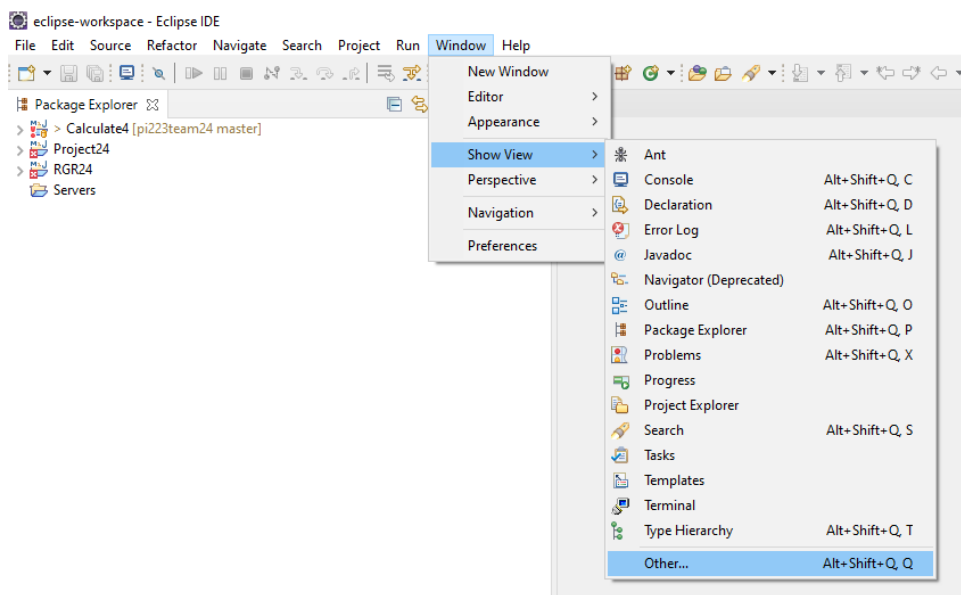


Рисунок 12 – Использование Git, шаг 1.

После этого выбрать Git->Git Repositories (рис.13).

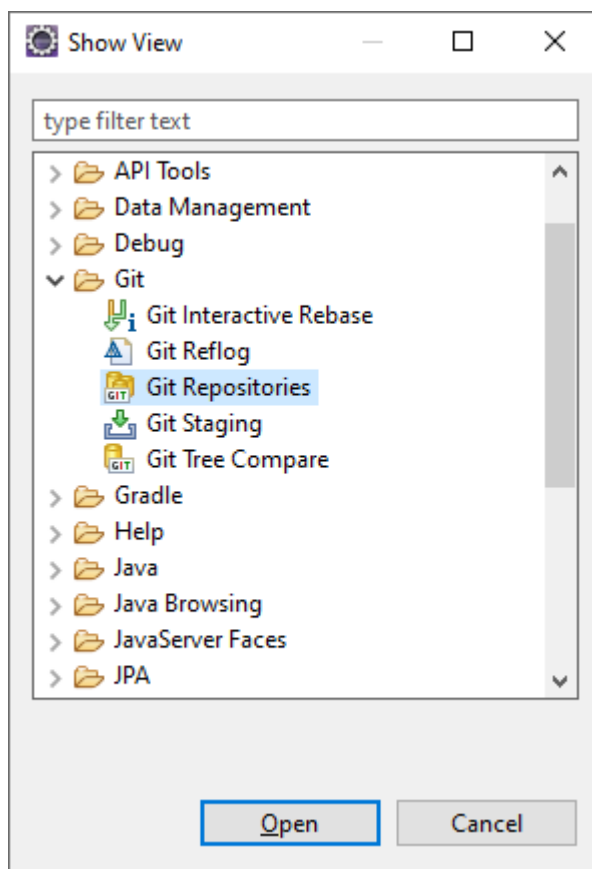


Рисунок 13 – Использование Git, шаг 2.

Для установки Maven необходимо скачать его с официального сайта (рис.14):

<https://maven.apache.org/download.cgi>

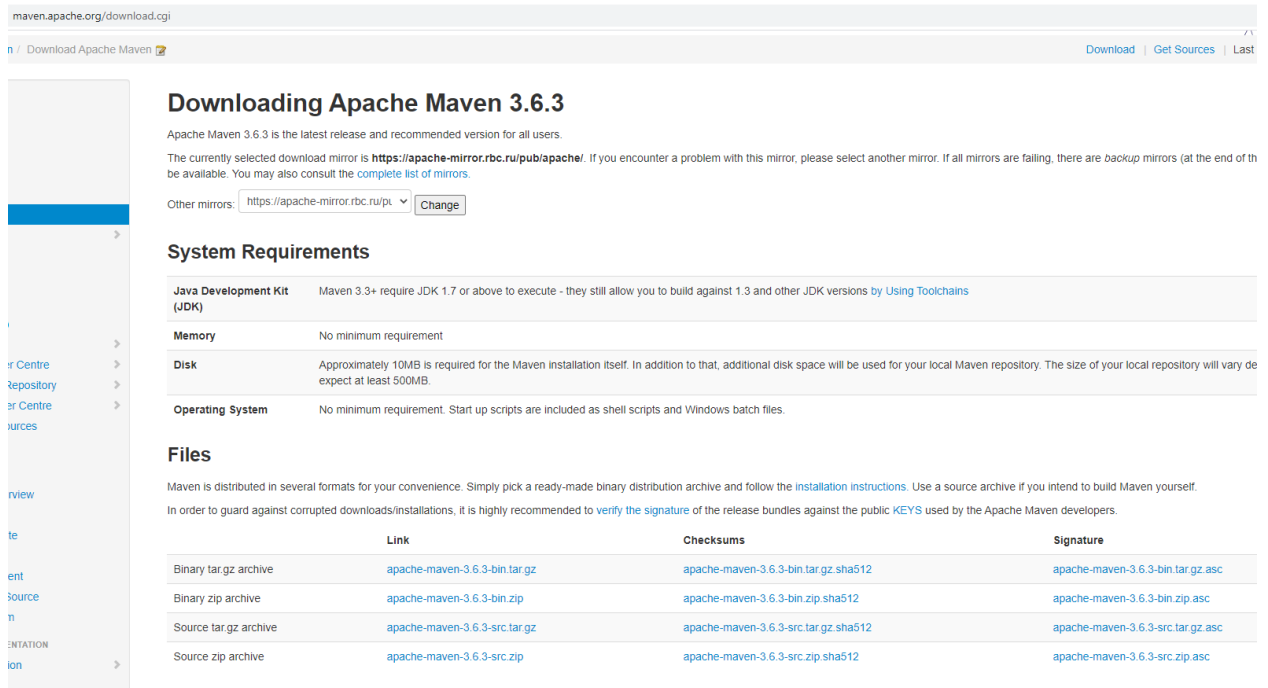


Рисунок 14 – Скачивание Maven.

Далее необходимо распаковать архив в любую директорию, к примеру “C:\Maven”. После этого создаем переменную M2_HOME со значением “C:\Maven”, создаем переменную M2 со значением «%M2_HOME%\bin», создаем переменную JAVA_HOME (если такая переменная отсутствует) с путем к JDK и изменяем переменную Path – дописать «%M2%». В итоге получится данный набор переменных (рис.15).

Переменные среды пользователя для vova8	
Переменная	Значение
JAVA_HOME	C:\Program Files\Java\jre1.8.0_281
M2	C:\Maven\bin
M2_HOME	C:\Maven
OneDrive	C:\Users\vova8\OneDrive
OneDriveConsumer	C:\Users\vova8\OneDrive
Path	"C:\Users\vova8\AppData\Local\Microsoft\WindowsApps;C:\Maven\bin";

Рисунок 15 – Переменные среды.

Для проверки работы Maven вводим в командной строке `mvn -version` (рис.16).

```
C:\Windows\system32>mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: C:\Maven\bin\..
Java version: 1.8.0_281, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jre1.8.0_281
Default locale: ru_RU, platform encoding: Cp1251
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Рисунок 16 – Проверка наличия Maven.

Также необходимо добавить зависимость JUnit, для этого необходимо нажать ПКМ на проект Maven->Add dependency, в поля Group Id и Artifact Id заполнить junit, а в поле Version написать 4.12 (рис.17-18).

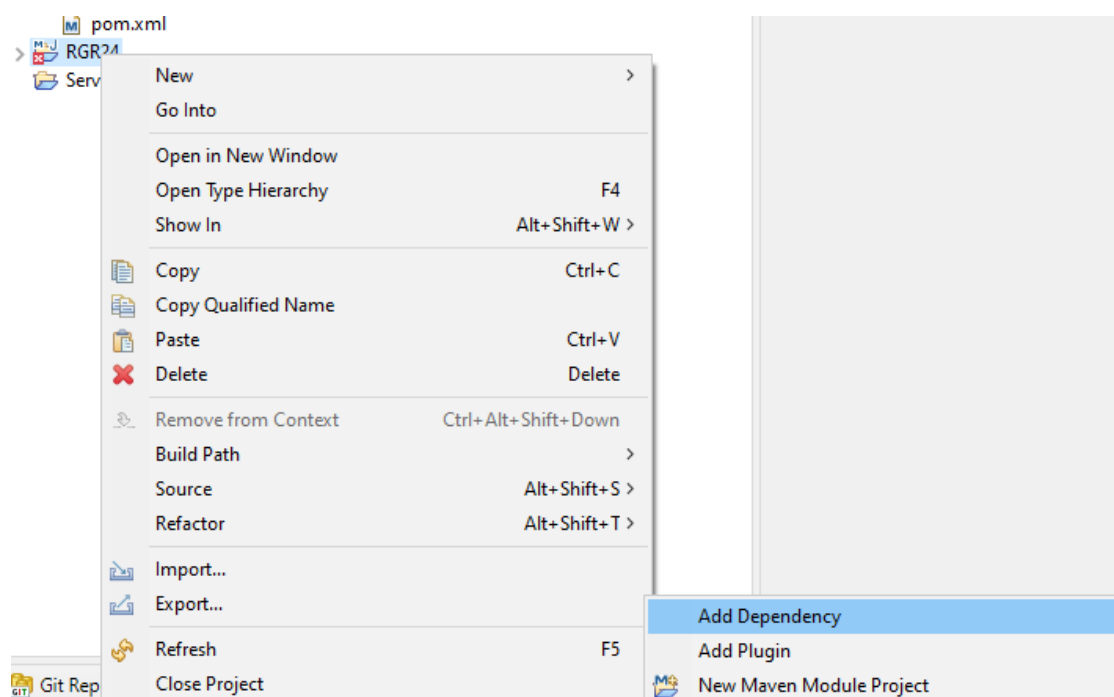


Рисунок 17 – Добавление JUnit, шаг 1.

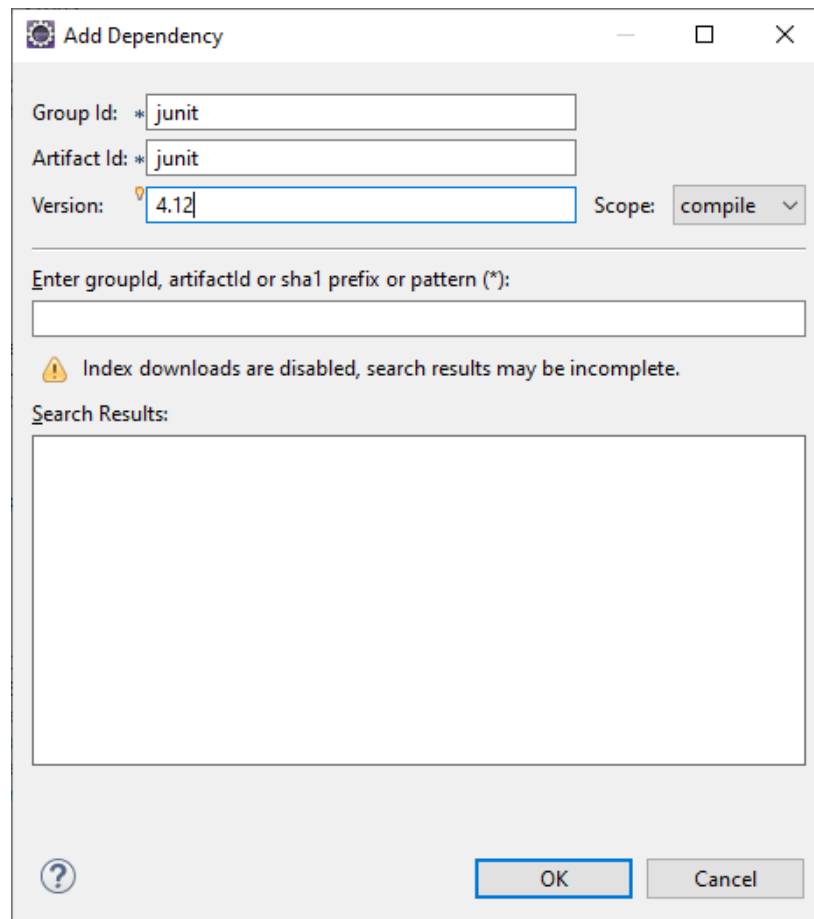


Рисунок 18 – Добавление JUnit, шаг 2

Настройка системы Linux – Ubuntu 20.04 и Debian 10.9

Поскольку Linux-подобные системы похожи в своей работе, настройка будет проводиться на двух похожих системах - **Ubuntu 20.04 и Debian 10.9**.

Так как в Linux-подобных системах работа в терминале преобладает над работой в графическом интерфейсе, установка всех программных продуктов и утилит будет проходить через командный интерпретатор – `bash`.

Первым делом необходимо обновить пакеты системы и ПО, сначала проверяется наличие обновлений (`sudo apt update`), затем идет обновление пакетов (`sudo apt upgrade`) (рис.19-20).

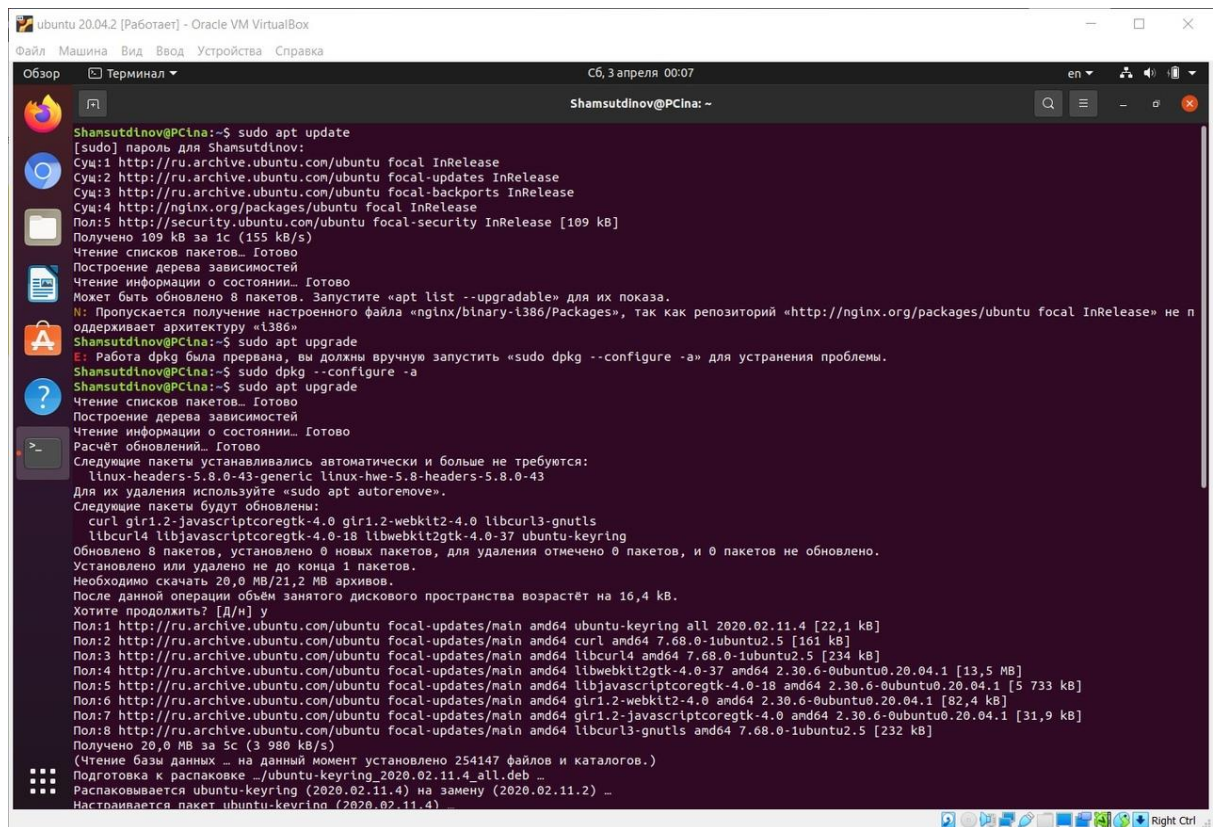


Рисунок 19 – Обновление Ubuntu.

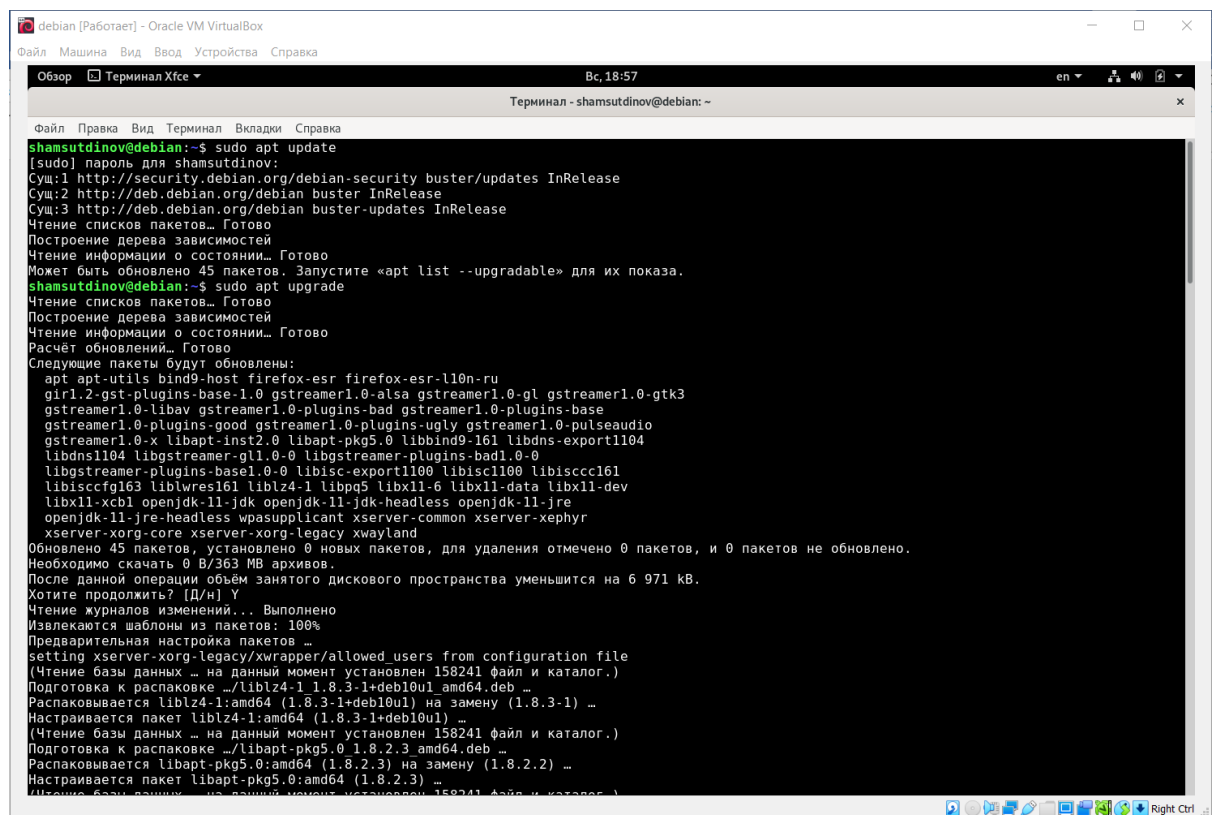
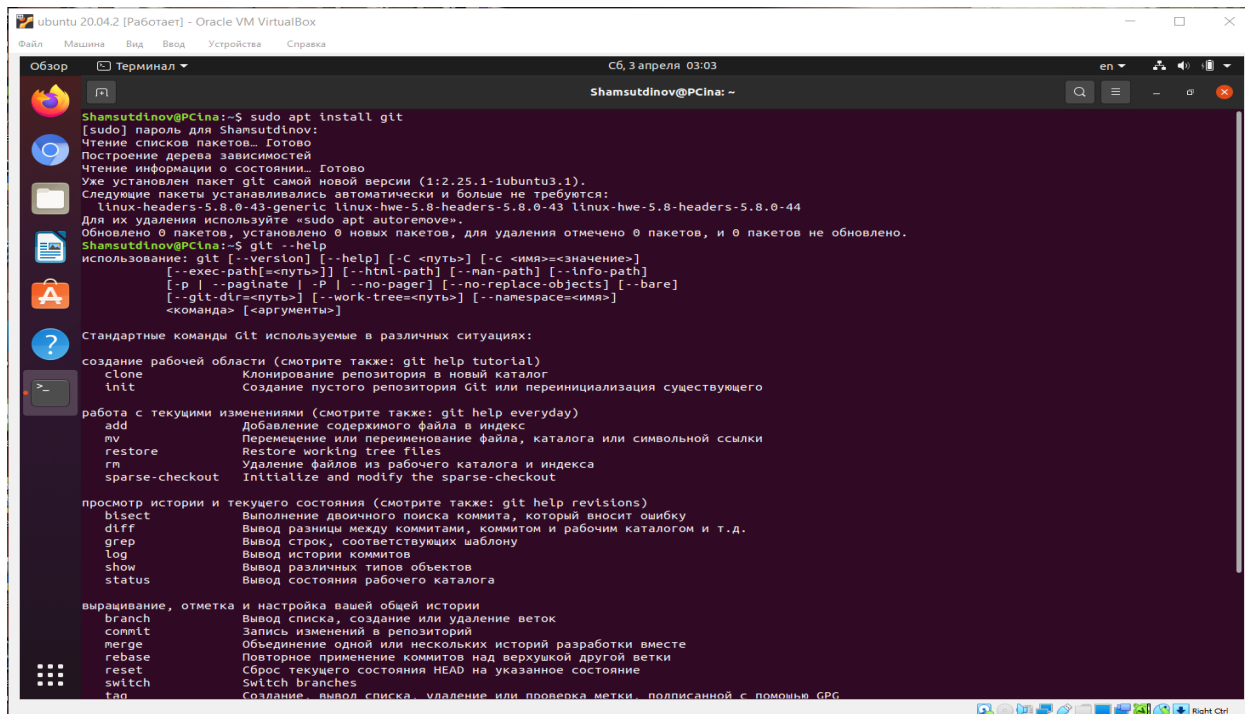


Рисунок 20 – Обновление Debian.

В разработке участвует система контроля версий Git, который устанавливается и настраивается через терминал(`sudo apt install git`), для более подробной информации о данной СКВ используется команда(`git <команда> --help`)

Установка клиента Git через терминал



```
ubuntu 20.04.2 [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка

Обзор Терминал C6, 3 апреля 03:03 en Shamsutdinov@PClna: ~

Shamsutdinov@PClna:~$ sudo apt install git
[sudo] пароль для Shamsutdinov:
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Уже установлен пакет git самой новой версии (1:2.25.1-1ubuntu3.1).
Следующие пакеты устанавливались автоматически и больше не требуются:
  linux-headers-5.8.0-43-generic linux-hwe-5.8-headers-5.8.0-43 linux-hwe-5.8-headers-5.8.0-44
Для их удаления используйте «sudo apt autoremove».
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
Shamsutdinov@PClna:~$ git --help
использование: git [--version] [--help] [-C <путь>] [-c <имя>=<значение>]
               [--exec-path[=<путь>]] [--html-path] [--man-path] [--info-path]
               [-p | --paginate] [-P | --no-pager] [--no-replace-objects] [--bare]
               [--git-dir=<путь>] [--work-tree=<путь>] [--namespace=<имя>]
               <команда> [<аргументы>]

Стандартные команды Git используемые в различных ситуациях:

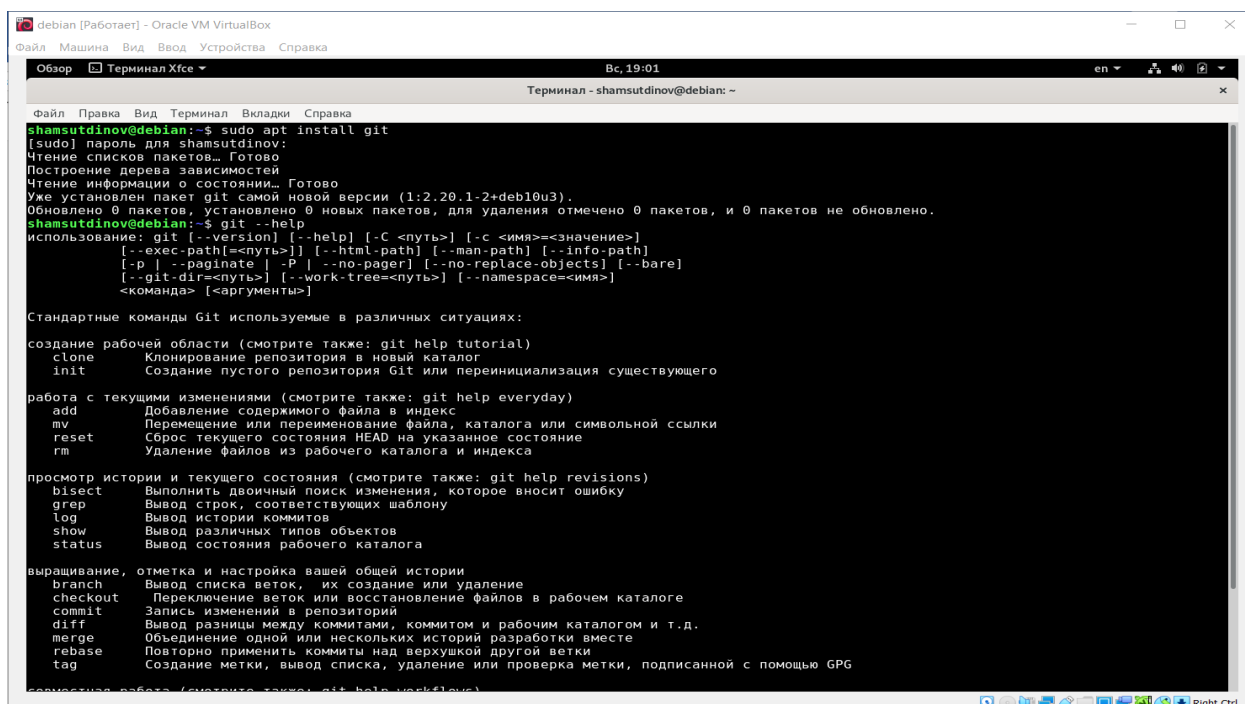
создание рабочей области (смотрите также: git help tutorial)
  clone      Клонирование репозитория в новый каталог
  init       Создание пустого репозитория Git или переинициализация существующего

работа с текущими изменениями (смотрите также: git help everyday)
  add        Добавление содержимого файла в индекс
  mv         Перемещение или переименование файла, каталога или символической ссылки
  restore    Restore working tree files
  rm         Удаление файлов из рабочего каталога и индекса
  sparse-checkout  Initialize and modify the sparse-checkout

просмотр истории и текущего состояния (смотрите также: git help revisions)
  bisect     Выполнение двоичного поиска коммита, который вносит ошибку
  diff       Вывод разницы между коммитами, коммитом и рабочим каталогом и т.д.
  grep       Вывод строк, соответствующих шаблону
  log        Вывод истории коммитов
  show       Вывод различных типов объектов
  status     Вывод состояния рабочего каталога

выращивание, отметка и настройка вашей общей истории
  branch     Вывод списка веток, их создание или удаление веток
  commit     Запись изменений в репозиторий
  merge      Объединение одной или нескольких историй разработки вместе
  rebase     Повторное применение коммитов над верхушкой другой ветки
  reset      Сброс текущего состояния HEAD на указанное состояние
  switch     Switch branches
  tag        Создание, вывод списка, удаление или проверка метки, подписанной с помощью GPG
```

Рисунок 21 – Установка на Ubuntu.



```
debian [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка

Обзор Терминал Xfce Bc, 19:01 en Терминал - shamsutdinov@debian: ~

Файл Правка Вид Терминал Вкладки Справка

shamsutdinov@debian:~$ sudo apt install git
[sudo] пароль для shamsutdinov:
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Уже установлен пакет git самой новой версии (1:2.20.1-2+deb10u3).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
shamsutdinov@debian:~$ git --help
использование: git [--version] [--help] [-C <путь>] [-c <имя>=<значение>]
               [--exec-path[=<путь>]] [--html-path] [--man-path] [--info-path]
               [-p | --paginate] [-P | --no-pager] [--no-replace-objects] [--bare]
               [--git-dir=<путь>] [--work-tree=<путь>] [--namespace=<имя>]
               <команда> [<аргументы>]

Стандартные команды Git используемые в различных ситуациях:

создание рабочей области (смотрите также: git help tutorial)
  clone      Клонирование репозитория в новый каталог
  init       Создание пустого репозитория Git или переинициализация существующего

работа с текущими изменениями (смотрите также: git help everyday)
  add        Добавление содержимого файла в индекс
  mv         Перемещение или переименование файла, каталога или символической ссылки
  reset      Сброс текущего состояния HEAD на указанное состояние
  rm         Удаление файлов из рабочего каталога и индекса

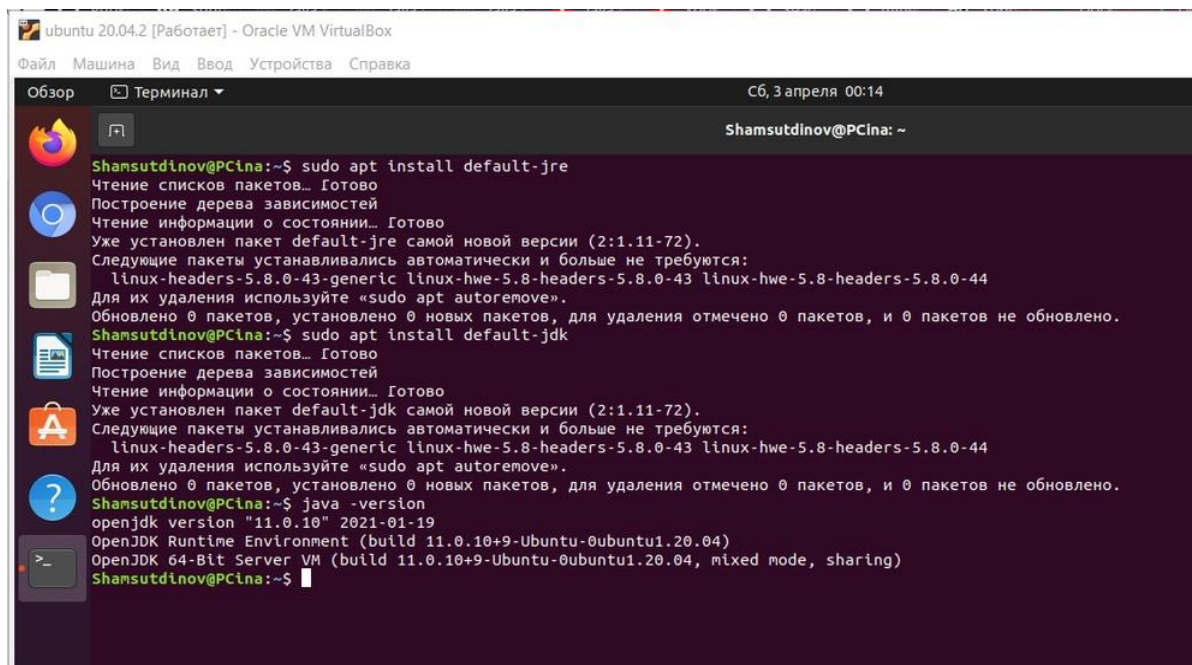
просмотр истории и текущего состояния (смотрите также: git help revisions)
  bisect     Выполнить двоичный поиск изменения, которое вносит ошибку
  grep       Вывод строк, соответствующих шаблону
  log        Вывод истории коммитов
  show       Вывод различных типов объектов
  status     Вывод состояния рабочего каталога

выращивание, отметка и настройка вашей общей истории
  branch     Вывод списка веток, их создание или удаление
  checkout   Переключение веток или восстановление файлов в рабочем каталоге
  commit     Запись изменений в репозиторий
  diff       Вывод разницы между коммитами, коммитом и рабочим каталогом и т.д.
  merge      Объединение одной или нескольких историй разработки вместе
  rebase     Повторно применить коммиты над верхушкой другой ветки
  tag        Создание метки, вывод списка, удаление или проверка метки, подписанной с помощью GPG
```

Рисунок 22 – Установка на Debian.

Для разработки ПО с помощью языка Java используется пакет инструментов JDK (Java Development Kit), JRE (Java Runtime Environment) представляет собой пакет инструментов для запуска Java-кода. Для установки данных пакетов используются команды `sudo apt install default-jdk` и `sudo apt install default-jre` соответственно.

Установка утилит для работы с java (рис.16).

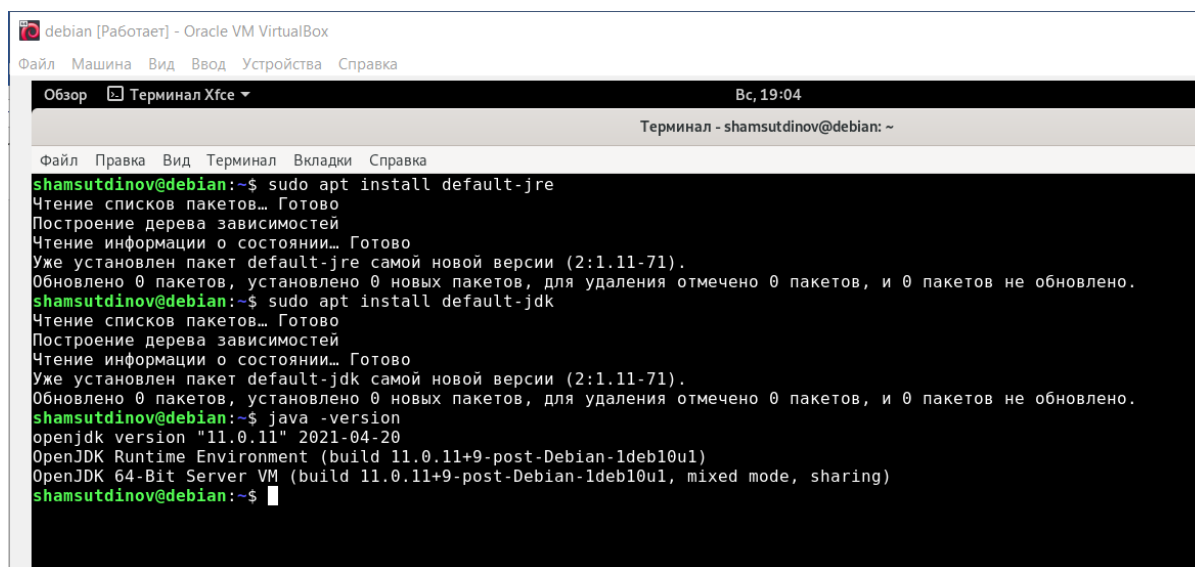


```
ubuntu 20.04.2 [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка

Обзор  Терминал  Сб, 3 апреля 00:14
Shamsutdinov@PCina: ~

Shamsutdinov@PCina:~$ sudo apt install default-jre
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Уже установлен пакет default-jre самой новой версии (2:1.11-72).
Следующие пакеты устанавливались автоматически и больше не требуются:
  linux-headers-5.8.0-43-generic linux-hwe-5.8-headers-5.8.0-43 linux-hwe-5.8-headers-5.8.0-44
Для их удаления используйте «sudo apt autoremove».
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
Shamsutdinov@PCina:~$ sudo apt install default-jdk
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Уже установлен пакет default-jdk самой новой версии (2:1.11-72).
Следующие пакеты устанавливались автоматически и больше не требуются:
  linux-headers-5.8.0-43-generic linux-hwe-5.8-headers-5.8.0-43 linux-hwe-5.8-headers-5.8.0-44
Для их удаления используйте «sudo apt autoremove».
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
Shamsutdinov@PCina:~$ java -version
openjdk version "11.0.10" 2021-01-19
OpenJDK Runtime Environment (build 11.0.10+9-Ubuntu-0ubuntu1.20.04)
OpenJDK 64-Bit Server VM (build 11.0.10+9-Ubuntu-0ubuntu1.20.04, mixed mode, sharing)
Shamsutdinov@PCina:~$
```

Рисунок 23 – Установка JRE и JDK на Ubuntu.



```
debian [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка

Обзор  Терминал  Xfce  Вс, 19:04
Терминал - shamsutdinov@debian: ~

Файл  Правка  Вид  Терминал  Вкладки  Справка

shamsutdinov@debian:~$ sudo apt install default-jre
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Уже установлен пакет default-jre самой новой версии (2:1.11-71).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
shamsutdinov@debian:~$ sudo apt install default-jdk
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Уже установлен пакет default-jdk самой новой версии (2:1.11-71).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
shamsutdinov@debian:~$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-post-Debian-1deb10u1)
OpenJDK 64-Bit Server VM (build 11.0.11+9-post-Debian-1deb10u1, mixed mode, sharing)
shamsutdinov@debian:~$
```

Рисунок 24 – Установка JRE и JDK на Debian.

Раздел 4. Настройка среды разработки для подключения к системе контроля версий.

Настройка Eclipse IDE для подключения к системе контроля версий идентична. Она будет выглядеть следующим образом:

1. Нажимаем в Eclipse последовательность кнопок «Help» – «Install New Software». Откроется окно, представленное в соответствии с рисунком 25, где в поле «Work with» необходимо ввести <https://download.eclipse.org/releases/juno>, далее выбрать модули «Eclipse Egit» и «Eclipse Egit Mylyn Github Feature».

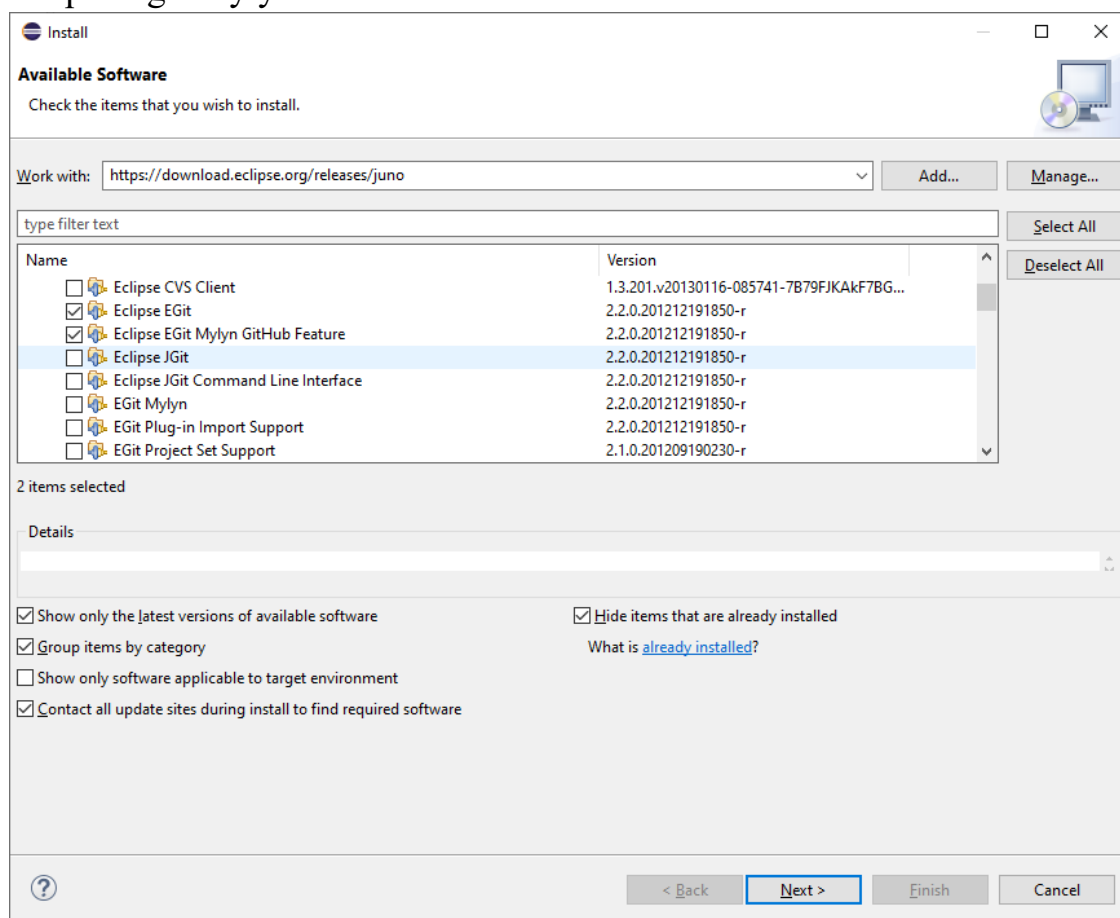


Рисунок 25 – Установка дополнений.

После установки необходимо перезагрузить Eclipse для дальнейшей работы дополнений.

2. Для того, чтобы использовать Git, нужно нажать на кнопку «Open Perspective», находящуюся в правом верхнем углу, и в появившемся окне выбрать Git (рис.26).

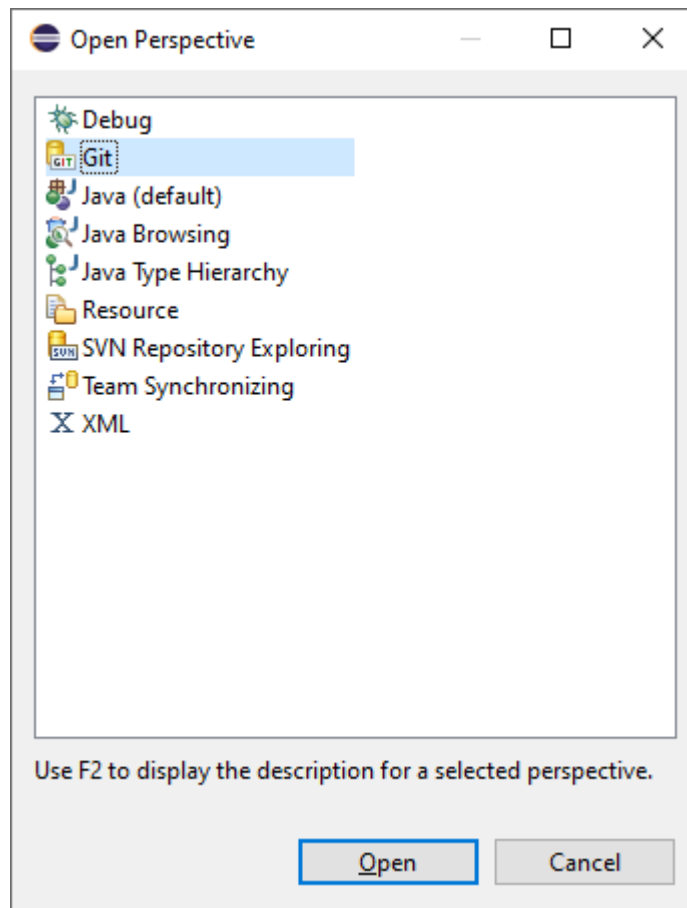


Рисунок 26 – Выбор перспективы.

После проделанных действий перед нами появляется страница для работы с Git.

3. Для клонирования репозитория из Git следует нажать на «Clone a Git repository». В поле «URL» нужно вставить ссылку на нужный репозиторий. В нашем случае вставляем ссылку на репозиторий с проектом курсовой работы (<https://github.com/PupkovEgor/KR24>) Оставшиеся поля в разделе «Location» заполняются автоматически. Далее следует ввести логин и пароль от аккаунта Github для авторизации (рис.27).

Clone Git Repository

Source Git Repository

Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

☒ Store in Secure Store

Рисунок 27 – Клонирование репозитория.

4. Для импорта проекта выбираем «Working tree», нажимаем ПКМ, далее «Import Project...», где в появившемся окне указываем путь к каталогу, в котором будет располагаться файлы из репозитория.
5. Все доступные функции будут располагаться в разделе «Team» (рис.28).

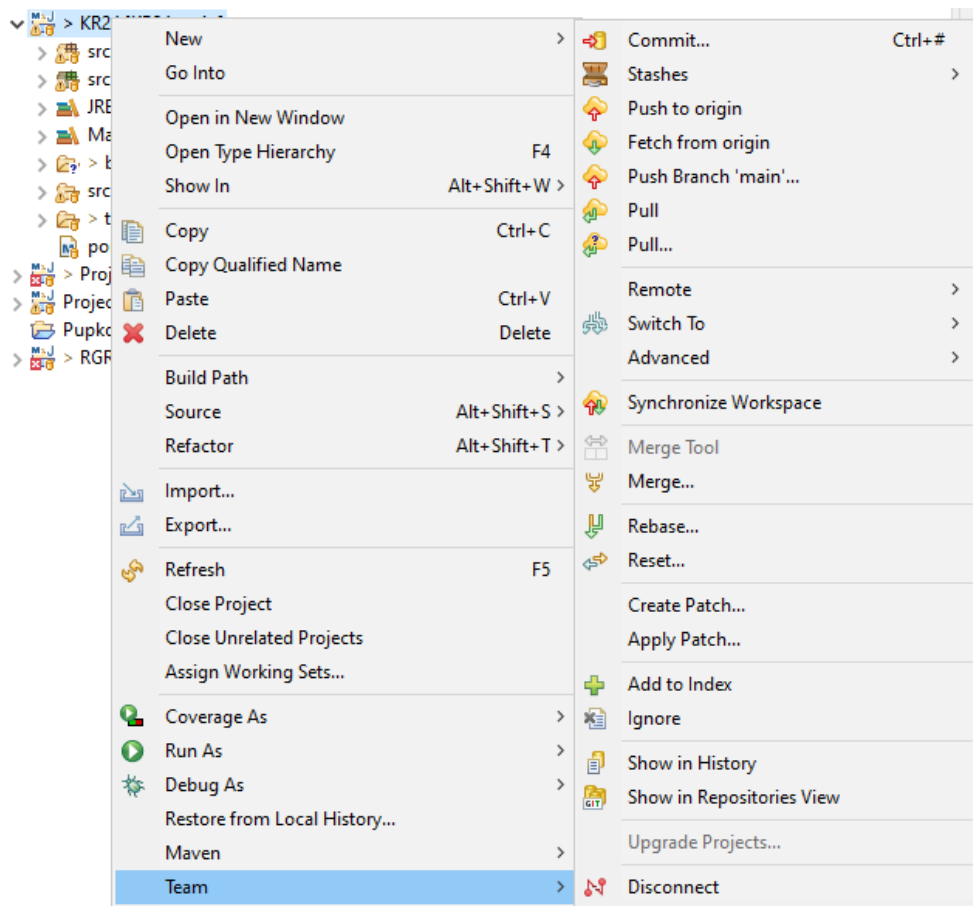


Рисунок 28 – Функции.

Раздел 5. Реализация исходного кода по зонам ответственности.

Работа по созданию Web-приложения велась по следующим зонам ответственности:

Таблица 2 – Зоны ответственности

№	ФИО разработчика/модератора	Зона ответственности
1	Пупков Егор Андреевич	Создание PDF-файла: классы CreatePDF, Calc.
2	Шамсутдинов Руслан Маратович	Создание формы авторизации: login.jsp Реализация авторизации: классы DataBase, Filtrator, Logout, User, UserPut.
3	Денисов Никита Андреевич	Создание графического интерфейса: формы admin_menu, Form, Results, Spravka. Создание классов Karkas, Napolnitel, Obivka, Vidstul. Создание абстрактного класса Materials.
4	Леонтьев Владимир Алексеевич	Реализация вычислений: класс Calc. Реализация формы администратора: классы ReadFile, WriteFile. Реализация проверки авторизации: форма NotAuth.

Ссылка на репозиторий с приложением: <https://github.com/PupkovEgor/kr24>

Раздел 6. Сборка и тестирование программного продукта

Тестирование программного продукта

В данном разделе размещается таблица с описанием UNIT-тестов и ответственных за их реализацию (табл.3).

Таблица 3 – Описание UNIT-тестов.

№	ФИО разработчика/ модератора	Описание UNIT- теста	№ приложения
1	Пупков Егор Андреевич	Проверяет создание pdf-файла. Класс Create PDF.	ПРИЛОЖЕНИЕ 4
2	Шамсутдинов Руслан Маратович	Наличие данных для авторизации. Класс DataBase.	ПРИЛОЖЕНИЕ 5
3	Денисов Никита Андреевич	Проверка работы абстрактного класса. Класс Materials.	ПРИЛОЖЕНИЕ 6
4	Леонтьев Владимир Алексеевич	Проверяет создание файла и запись значений в этот файл. Класс WriteFile.	ПРИЛОЖЕНИЕ 7

Сборка программного продукта.

Сборка программного продукта осуществляется при помощи Maven.

Как видно на рисунке 29, проект Calculate4 включает в себя каталоги artifacts, src/main/java, src/main/webapp, src/test и target, а также файлы README.md и pom.xml. В свою очередь, каталог webapp включает в себя подкаталог WEB-INF, а также файлы admin_menu.jsp, Form.jsp, login.jsp, Spravka.jsp и Results.jsp. Подкаталог WEB-INF включает в себя файл web.xml.

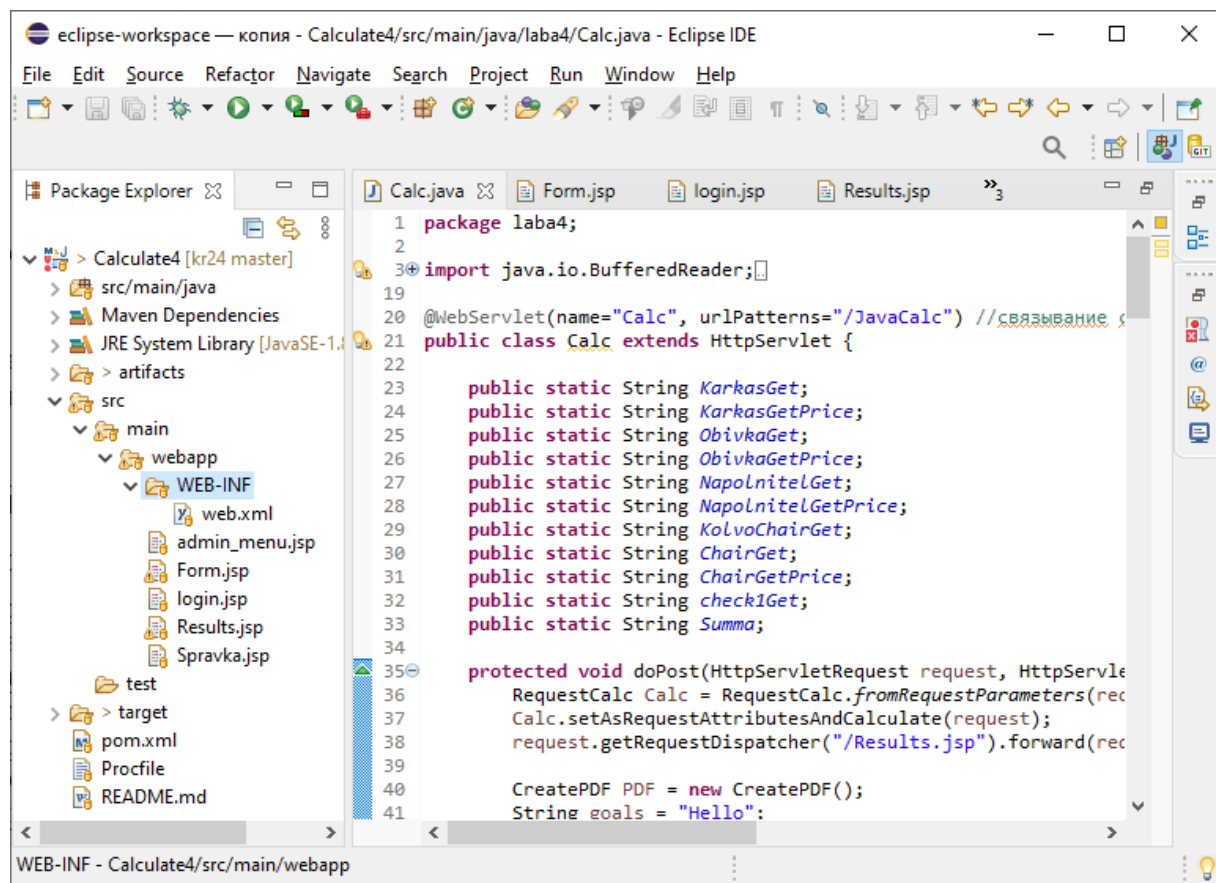


Рисунок 29 – Структура проекта

В каталоге artifacts находится собранный war-файл. В каталоге src/main/java расположены все классы приложения. В каталоге src/test находятся UNIT-тесты. В каталоге target.

Файлы admin_menu.jsp, Form.jsp, login.jsp, Spravka.jsp и Results.jsp являются графическим интерфейсом web-приложения:

- admin_menu.jsp – панель администратора.
- Form.jsp – основная форма калькулятора.
- login.jsp – форма авторизации.
- Spravka.jsp – форма, содержащая информацию о разработчиках.
- Results.jsp – форма с результатами работы калькулятора.

Файл web.xml определяет соответствие между путями URL и сервлетами, которые эти URL будут обрабатывать.

Файл pom.xml – это XML-файл, который содержит информацию о деталях проекта, и конфигурации, используемые для создания проекта Maven.

Файл README.md содержит ссылку на проект в Travis CI.

Чтобы собрать проект, нужно нажать Calculate4(ПКМ)→Run As→Maven build (рис.30).

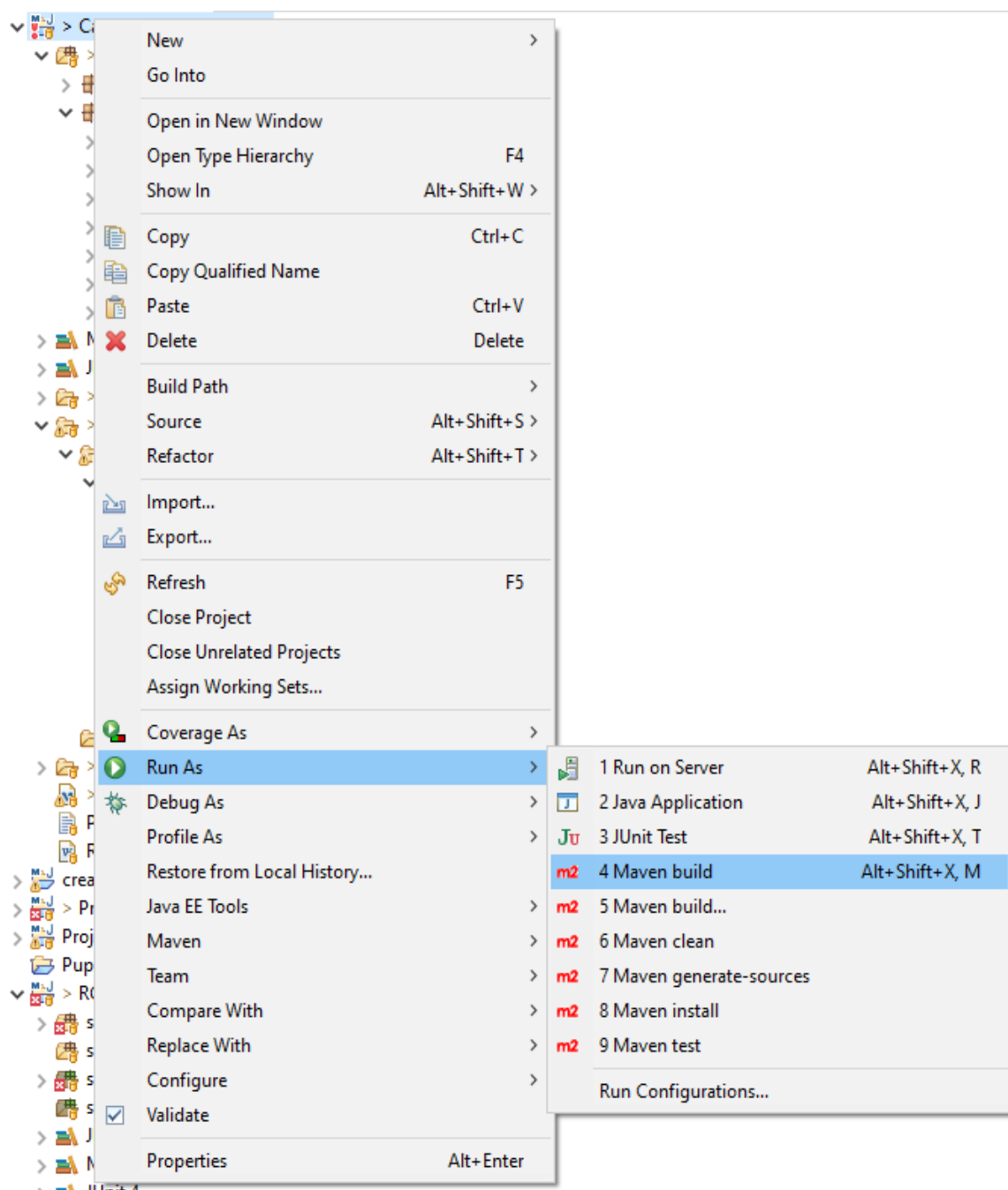


Рисунок 30 – Сборка проекта (1).

При первой сборке появится окно, в котором в строке Goals нужно указать package (рис.31).

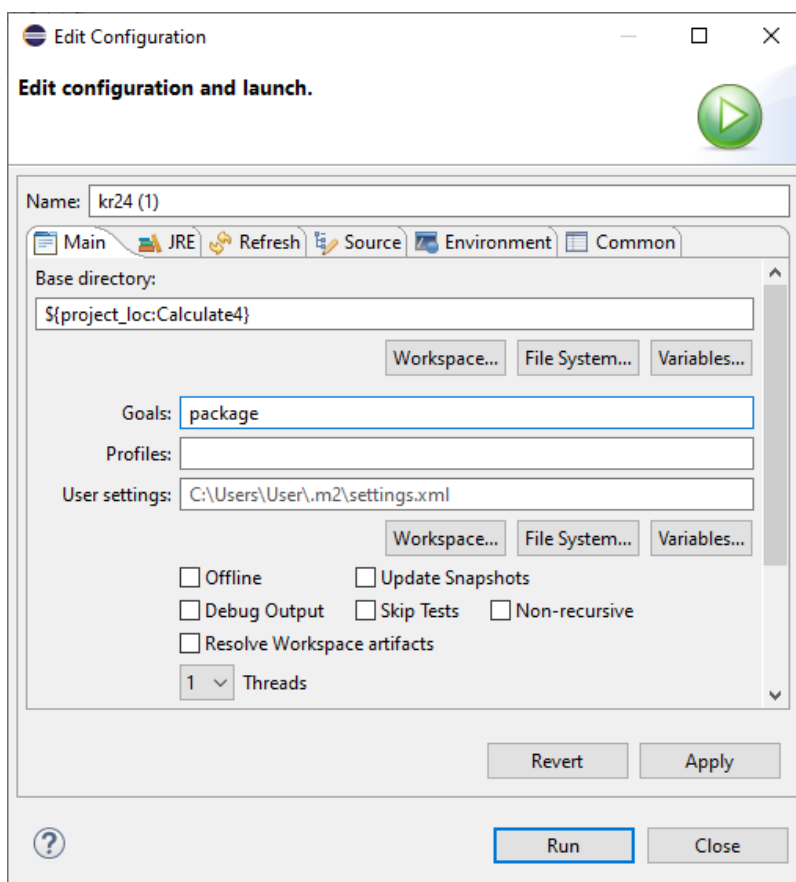


Рисунок 31 – Сборка проекта (2).

Через некоторое время проект соберётся в war-файл в каталоге artifacts.

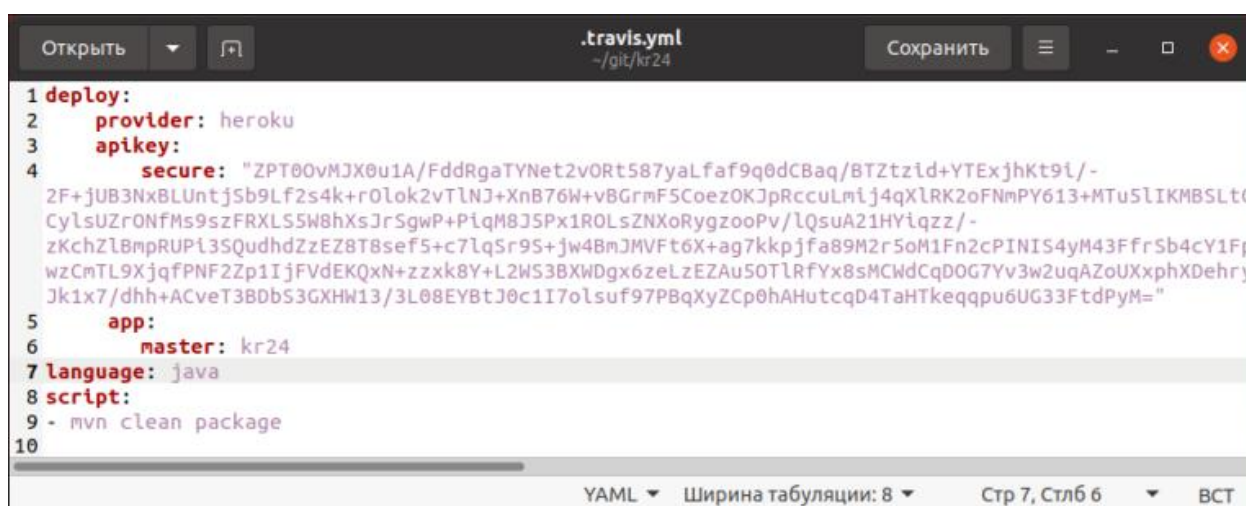
7. Настройка программной среды для развертывания и запуска программного продукта

Непрерывная интеграция (CI, англ. Continuous Integration) – это практика разработки программного обеспечения (ПО), которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения интеграционных проблем.

Для непрерывной интеграции был выбран Travis CI.

В качестве платформы непрерывной интеграции Travis CI поддерживает процесс разработки, автоматически тестируя программный код, обеспечивая обратную связь. Также Travis CI может автоматизировать другие процессы разработки, управляя развертыванием и уведомлениями.

Настройка системы непрерывной интеграции (рис.32):



```
1 deploy:
2   provider: heroku
3   apikey:
4     secure: "ZPT00vMJX0u1A/FddRgaTYNet2v0Rt587yaLfaf9q0dCBaq/BTZtZid+YTeXjhKt9i/-
5     2F+jUB3NxBLUntjSb9Lf2s4k+r0lok2vTLNJ+XnB76W+vbGrmf5Coez0KJpRccuLmij4qXlRK2oFNmPY613+MTu5lIKMBSLtG
6     CylsUZrONfMs9szFRXL55W8hXsJrSgwP+PiQM8J5Px1ROLsZNXoRygzooPv/lQsuA21HYiqzz/-
7     zKchZlBmpRUPi35QudhdZzEZ8T8sef5+c7lqSr9S+jw4BmJMVft6X+ag7kkpjfa89M2r5oM1Fn2cPINIS4yM43FfrSb4cY1Fp
8     wzCmTL9XjqfPNF2Zp1IjFVdEKQxN+zzxk8Y+L2W53BXWdGx6zeLzEZAu50TlrfYx8sMCWdCqDOG7Yv3w2uqAZoUXxphXDehry
9     Jk1x7/dhh+ACveT3BDbS3GXHW13/3L08EYBtJ0c1I7olsuf97PBqXyZCp0hANutcqD4TaHTkeqqu6UG33FtdPyM="
10   app:
11     master: kr24
12   language: java
13   script:
14     - mvn clean package
```

Рисунок 32 – Конфигурация .travis.yml

Для развёртывания и запуска программного продукта была выбрана платформа Heroku.

Heroku – это облачная платформа, основанная на управляемой контейнерной системе, с интегрированными службами передачи данных и мощной экосистемой для развертывания и запуска современных приложений.

Приложение автоматически разворачивается после каждого обновления репозитория по ссылке: <https://kr24.herokuapp.com/> (рис.33).

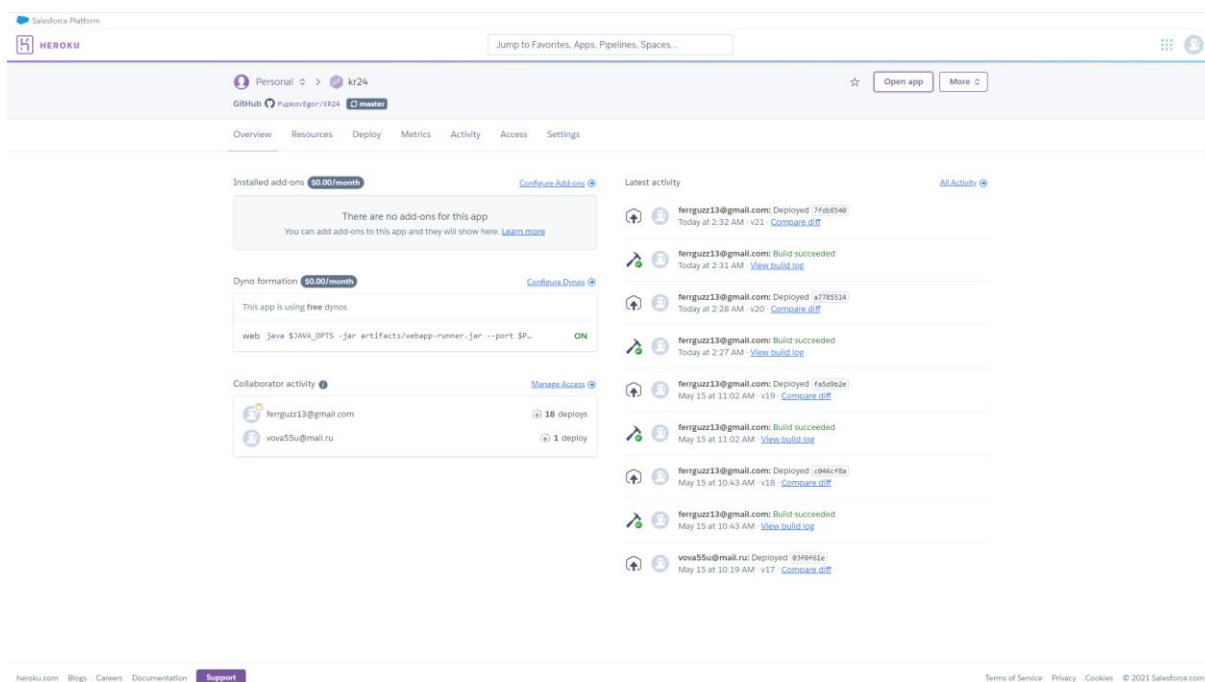


Рисунок 33 – Приложение, развернутое на Heroку.

Описание всех файлов работающей программы находится в Разделе 6.

Раздел 8. Руководство пользователя программного продукта.

Смотрите Приложение 2.

ФГБОУ ВО УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

СОГЛАСОВАНО

УТВЕРЖДАЮ

Доцент кафедры АСУ
ФГБОУ УГАТУ

Студент группы ПИ-223
ФИРТ ФГБОУ УГАТУ,
модератор

Личная Расшифровка
подпись подписи
Дата:

Личная Расшифровка
подпись подписи
Дата:

Калькулятор стоимости производства. Мягкая мебель.

Техническое задание

ЛИСТ УТВЕРЖДЕНИЯ

1304.300024.000 ТЗ

(Электронный)

Листов 5

СОГЛАСОВАНО

Представитель команды
разработчиков

Доцент кафедры АСУ
ФГБОУ УГАТУ

Студент группы ПИ-223
ФИРТ ФГБОУ УГАТУ,
модератор

Личная Расшифровка
подпись подписи
Дата:

Личная Расшифровка
подпись подписи
Дата:

Инб. М. подп.	Подп. и дата
Взам инб. М.	
Инб. М. подп.	Подп. и дата
Инб. М. подп.	

Утвержден
1304.300024.000 ПЗ

Калькулятор стоимости производства. Мягкая мебель.

Техническое задание

1304.300024.000 ТЗ

(Электронный)

Листов 5

Инд. N подл.	Подп. и дата	Взам. инд. N	Инд. N дубл.	Подп. и дата

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	
1. ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ.....	
2. НАЗНАЧЕНИЕ ДЛЯ РАЗРАБОТКИ.....	
2.1. Функциональное назначение.....	
2.2. Эксплуатационное назначение	
3. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ	
3.1. Требования к функциональным характеристикам	
3.2. Требования к надёжности	
3.3. Условия эксплуатации	
3.4. Требования к составу и параметрам технических средств	
3.5. Требования к информационной и программной совместимости	
4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ	
5. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ	
5.1. Ориентированная экономическая эффективность.	
5.2. Предполагаемая годовая потребность	
5.3. Экономические преимущества разработки	
6. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ	
7. ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ	
7.1. Виды испытаний	
7.2. Порядок приёмки	

ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку программы «Мебельный калькулятор», предназначенной для подсчёта стоимости изготавливаемой продукции в соответствии установленным параметрам. Предполагается, что использовать данную программу будут клиенты и администратор.

Автоматизированная система расчёта стоимости позволит улучшить процесс создания заказа для клиентов и процесс обработки заказа для производителя.

Кроме того, администратор сможет с лёгкостью изменить параметры для актуализации цен и прочих характеристик.

1. ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ.

Основанием для разработки данного ПО послужило задание на курсовую работу.

2. НАЗНАЧЕНИЕ РАЗРАБОТКИ.

Программа будет использоваться для приёма заказа на производство различных видов стульев.

2.1. Функциональное назначение.

Для клиента программа предоставляет возможность заполнить поля нужными характеристиками и получить полную стоимость готовой продукции.

Для администратора программа позволяет изменить параметры в математических вычислениях или выпадающих списках и изменить информацию на графическом интерфейсе ПО.

2.2. Эксплуатационное назначение.

Программа должна эксплуатироваться на сайте производителя мебели. Для клиента она представляется в виде графического интерфейса с полями для ввода параметров стульев. Для администратора программа представляется в виде графического интерфейса с полями для изменения значений переменных.

3. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ.

3.1. Требования к функциональным характеристикам

Программа должна обеспечивать возможность выполнения следующих функций:

- авторизация пользователя (клиент или администратор);
- выбор вида стула (игровое кресло, офисное кресло и т.д.);
- выбор материала каркаса (металл, дерево пластик);
- выбор материала обивки (ткань, замша и т.д.);
- выбор материала наполнителя (поролон, синтепон, хлопок);
- ввод количества стульев;
- выбор пункта «Срочный заказ»;
- расчёт стоимости (кнопка «Расчитать»);
- вывод информации о разработчиках (кнопка «Справка»);
- выход из программы (кнопка «Выход»).

3.2. Требования к надёжности.

Требования к обеспечению надежного функционирования программы:

- предусмотреть контроль вводимой информации;
- предусмотреть неизменяемые поля;

3.3. Условия эксплуатации.

Компьютер предназначен для работы в закрытом отапливаемом помещении при следующих условиях окружающей среды:

- температура окружающего воздуха от +10°C до +35°C ;
- атмосферное давление от 630 до 800 мм ртутного столба ;
- относительная влажность воздуха не более 80% ;
- запыленность воздуха не более 0,75 мг/м³ ;
- кроме этого, в воздухе не должно быть паров агрессивных жидкостей и веществ, вызывающих коррозию.

3.4. Требования к составу и параметрам технических средств

Система должна работать на IBM-совместимом персональном компьютере, выступающим в роле сервера.

Конфигурация:

- тип процессора – AMD и выше;
- объем ОЗУ – 1024 Мб и более;
- HDD – 40 Гб и более;
- ОС - Windows 2000 Server или Ubuntu Server;

3.5. Требования к информационной и программной совместимости.

Система должна работать под управлением семейства операционных систем Windows и Linux.

На системе должна быть установлена Java Virtual Machine.

Для создания ПО использовать: Git, GitHub, Maven.

В качестве среды разработки программы необходимо использовать Eclipse IDE.V

4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ.

Программная документация должна состоять из следующих документов:

1. Техническое задание.
2. Общие положения.
3. Термины и определения.
4. Основные надписи.
5. Описание программы.
6. Руководство пользователя.
7. Руководство программиста.
8. Описание языка.

5. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ.

5.1. Ориентированная экономическая эффективность.

Ориентированная экономическая эффективность не рассчитывается.

5.2. Предполагаемая годовая потребность.

Предполагаемое число использования программы – 0-50 раз в неделю в течении года.

5.3. Экономические преимущества разработки.

Экономические преимущества разработки не рассчитываются.

6. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ.

Стадии и этапы разработки, а также трудоёмкость выполнения и сроки представления описаны в таблице 1.

Таблица 1 – Стадии и этапы разработки.

Наименование этапа работ	Трудоёмкость выполнения, час	Процент к общей трудоёмкости выполнения	Срок предъявления консультанту
Получение и согласование задания	1,7	1,7%	27 неделя
Раздел 1. Описание предметной области	20	20%	29 неделя
Раздел 2. Техническое задание на создание программного продукта	10	10%	30 неделя
Раздел 3. Настройка среды разработки для операционных систем семейств Windows и Linux	10	10%	31 неделя
Раздел 4. Настройка среды разработки для подключения к системе контроля версий	7	7%	32 неделя
Раздел 5. Реализация исходного кода по зонам ответственности	23	23%	34 неделя
Раздел 6. Сборка и тестирование программного продукта	8	8%	35 неделя
Раздел 7. Настройка программной среды для развертывания и запуска программного продукта	10	10%	36 неделя
Раздел 8. Руководство пользователя программного продукта	10	10%	37 неделя
Защита	0,3	0,3%	38 неделя

7. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ.

7.1. Виды испытаний.

Процедура защиты курсовой работы предполагает следующие этапы:

1. Настройка среды Eclipse в нескольких операционных системах разных семейств.
2. Клонирование репозитория GitHub, извлечение рабочей копии и выполнение основных команд.
3. Работа с сервисом Travis CI.
4. Выполнить развертывание и запуск программного продукта.
5. Знание своей зоны ответственности.

7.2. Порядок приёмки.

Прием программы будет утвержден Казанцевым А.В. при корректной работе программы с различными входными данными.

**ФГБОУ ВО УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

СОГЛАСОВАНО

УТВЕРЖДАЮ

Доцент кафедры АСУ
ФГБОУ УГАТУ

Студент группы ПИ-223
ФИРТ ФГБОУ УГАТУ,
модератор

Личная Расшифровка
подпись подписи
Дата:

Личная Расшифровка
подпись подписи
Дата:

Калькулятор стоимости производства. Мягкая мебель.

Дополнение №1 к ТЗ на АС

ЛИСТ УТВЕРЖДЕНИЯ

1304.300024.000 ТЗ

(Электронный)

Листов 2

СОГЛАСОВАНО

Представитель команды
разработчиков

Доцент кафедры АСУ
ФГБОУ УГАТУ

Студент группы ПИ-223
ФИРТ ФГБОУ УГАТУ,
модератор

Личная Расшифровка
подпись подписи
Дата:

Личная Расшифровка
подпись подписи
Дата:

Подп. и дата	
Инф. и дата	
Взам. инф. и	
Подп. и дата	
Инф. и подп.	

Утвержден

1304.300024.000 ПЗ

Калькулятор стоимости производства. Мягкая мебель.

Дополнение №1

1304.300024.000 ТЗ

(Электронный)

Листов 2

Инд N подп	Подп и дата	Взам инд N	Инд N аудп	Подп и дата

1. Основание необходимости изменения технического задания.

Основанием для внесения изменений стало решение компании «Стул Стулыч» изменить панель администратора в приложении. Внесённые изменения будут описаны в разделе 2 настоящего дополнения.

2. Вносимые изменения.

Раздел «Назначение разработки». Заменить «Для администратора программа позволяет изменить параметры в математических вычислениях или выпадающих списках и изменить информацию на графическом интерфейсе ПО.» на «Для администратора программа позволяет изменить стоимость материалов и видов стульев и сохранить изменения.»

ФГБОУ ВО УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

СОГЛАСОВАНО

УТВЕРЖДАЮ

Доцент кафедры АСУ
ФГБОУ УГАТУ

Студент группы ПИ-223
ФИРТ ФГБОУ УГАТУ,
модератор

Личная Расшифровка
подпись подписи
Дата:

Личная Расшифровка
подпись подписи
Дата:

Калькулятор стоимости производства. Мягкая мебель.

Руководство пользователя

ЛИСТ УТВЕРЖДЕНИЯ

1304.300024.000 РП

(Электронный)

Листов 6

СОГЛАСОВАНО

Представитель команды
разработчиков

Доцент кафедры АСУ
ФГБОУ УГАТУ

Студент группы ПИ-223
ФИРТ ФГБОУ УГАТУ,
модератор

Личная Расшифровка
подпись подписи
Дата:

Личная Расшифровка
подпись подписи
Дата:

Инф. N подл	Подп. и дата
Инф. N инф. N	Инф. N инф. N
Инф. N инф. N	Инф. N инф. N
Инф. N инф. N	Инф. N инф. N
Инф. N инф. N	Инф. N инф. N

Утвержден

1304.300024.000 РП

Калькулятор стоимости производства. Мягкая мебель.

Руководство пользователя

1304.300024.000 РП

(Электронный)

Листов 6

Инф. N подл	Подп. и дата	Взам. инф. N	Инф. N дубл.	Подп. и дата

АННОТАЦИЯ

О руководстве пользователя

Настоящий документ содержит описание эксплуатации программного продукта "Калькулятор стоимости производства. Мягкая мебель", используемого для расчёта стоимости производства различных видов стульев.

В руководстве пользователя вы найдете следующую информацию:

- назначение программы
- функциональные возможности программы
- варианты работы с программой

СОДЕРЖАНИЕ

1. Назначение программы.....	5
2. Условия выполнения программы.....	5
3. Выполнение программы.....	5
4. Сообщения оператору.....	7
5. Лист регистрации изменений.....	9

1. Назначение программы

Программа «Калькулятор стоимости производства. Мягкая мебель.» предназначена для расчёта стоимости производства стульев.

Для клиента программа предоставляет возможность заполнить поля нужными характеристиками, получить полную стоимость готовой продукции и открыть или скачать PDF-файл с результатами, а также посмотреть данные о разработчиках.

Программа должна эксплуатироваться на сайте производителя мебели. Для клиента она представляется в виде графического интерфейса с полями для ввода параметров стульев и кнопками.

2. Условия выполнения программы

Воспользоваться данным программным продуктом можно при помощи ПК, смартфона или планшета, на котором установлен браузер (Yandex, Chrom, Mozilla и т.д.) и есть доступ к сети интернет.

3. Выполнение программы

Для выполнения программы нужно перейти на сайт, ввести данные для авторизации и нажать кнопку «Войти» (рис.1).

Вход в систему

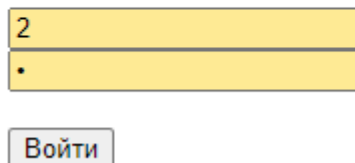


Рисунок 1 – Авторизация.

Затем нужно выбрать необходимые пункты из выпадающих списков, заполнить поле «Количество стульев», при необходимости выбрать пункт «Срочный заказ» и нажать на кнопку «Рассчитать» (рис.2), которая выводит результаты расчётов.

Калькулятор стоимости производства мягкой мебели

Вид стула:

Материал каркаса: Стоимость каркаса:

Материал обивки: Стоимость материала обивки:

Материал наполнителя: Стоимость наполнителя:

Количество стульев:

Стоимость работы за 1 стул:

Срочный заказ: ☒

[Logout](#)

Рисунок 2 – Форма заполнения данных.

Так же можно нажать на кнопку «Справка» и посмотреть данные о разработчиках.

Номера зачетных книжек указаны ниже:

Модератор: Пупков Егор Андреевич - 19130149

Первый участник: Шамсутдинов Руслан Маратович - 19130342

Второй участник: Денисов Никита Андреевич - 19130174

Третий участник: Леонтьев Владимир Алексеевич - 19130155

Рисунок 3 – Справка.

На форме с результатами можно открыть или скачать PDF-файл соответствующими кнопками, а также вернуться к форме с заполнением данных с помощью кнопки «Назад» (рис.4).

Итоговая стоимость производства мягкой мебели:

Ваши введенные данные:

Вид стула: Игровое кресло

Материал каркаса: Металл

Материал обивки: Искусственная кожа

Материал наполнителя: Поролон

Стоимость каркаса: 500

Стоимость обивки: 700

Стоимость наполнителя: 400

Количество стульев: 20

Стоимость работы за 1 стул: 7500

Срочный заказ: Да

Итоговая стоимость: 182000.0

[Open PDF-file](#) [Download PDF-file](#)

[Назад](#)

Рисунок 4 – Результат.

На страницах с заполнением данных и результатами присутствует кнопка «Logout» для перехода на форму авторизации.

4. Сообщения оператору

При не заполнении поля программа выдаст предупреждение «Вы пропустили это поле». При отсутствии выбора в выпадающем списке программа выдаст предупреждение «Выберите один из пунктов списка».

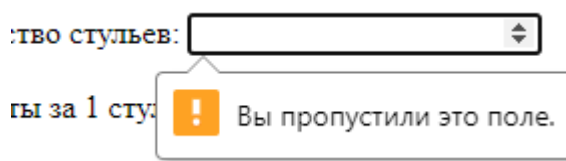


Рисунок 5 – Предупреждение «Вы пропустили это поле».

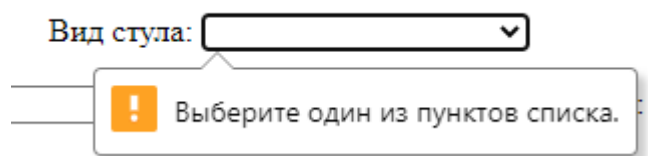


Рисунок 6 – Предупреждение «Выберите один из пунктов списка».

[illegible]

ФГБОУ ВО УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

СОГЛАСОВАНО

УТВЕРЖДАЮ

Доцент кафедры АСУ
ФГБОУ УГАТУ

Студент группы ПИ-223
ФИРТ ФГБОУ УГАТУ,
модератор

Личная Расшифровка
подпись подписи
Дата:

Личная Расшифровка
подпись подписи
Дата:

Калькулятор стоимости производства. Мягкая мебель.

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

1304.300024.000 ТП

(Электронный)

Листов 6

СОГЛАСОВАНО

Представитель команды
разработчиков

Доцент кафедры АСУ
ФГБОУ УГАТУ

Студент группы ПИ-223
ФИРТ ФГБОУ УГАТУ,
модератор

Личная Расшифровка
подпись подписи
Дата:

Личная Расшифровка
подпись подписи
Дата:

Подп. и дата	
Инф. и дата	
Взам. инф. и дата	
Подп. и дата	
Инф. и дата	

Утвержден

1304.300024.000 ТП

Калькулятор стоимости производства. Мягкая мебель.

Текст программы

1304.300024.000 ТП

(Электронный)

Листов 6

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

В данном документе представлен полный код программы «Мебельный калькулятор», предназначенный для расчёта стоимости производства стульев.

Текст программы представлен в виде записей на языках Java и HTML.

СОДЕРЖАНИЕ

АННОТАЦИЯ.....	2
Текст программы.....	4
Лист регистрации изменений.....	25

Текст программы

Программный код класса Karkas:

```
package AbstractClass;

public class Karkas extends Materials {
    @Override
    public void setName(String name) {
        this.name = name;
    }
    @Override
    public void setPrice(String price) {
        this.price = price;
    }
}
```

Программный код абстрактного класса Materials:

```
package AbstractClass;

public abstract class Materials {
    String name;
    String price;

    public String getName() {
        return name;
    }
    public String getPrice() {
        return price;
    }

    public abstract void setName(String name);
    public abstract void setPrice(String price);
}
```

Программный код класса Napolnitel:

```
package AbstractClass;

public class Napolnitel extends Materials {
    @Override
    public void setName(String name) {
        this.name = name;
    }
    @Override
    public void setPrice(String price) {
        this.price = price;
    }
}
```

Программный код класса Obivka:

```
package AbstractClass;

public class Obivka extends Materials {
    @Override
    public void setName(String name) {
        this.name = name;
    }
    @Override
    public void setPrice(String price) {
        this.price = price;
    }
}
```

Программный код класса Vidstul:

```
package AbstractClass;

public class Vidstul extends Materials {
    @Override
    public void setName(String name) {
        this.name = name;
    }
    @Override
    public void setPrice(String price) {
        this.price = price;
    }
}
```

Программный код класса DataBase:

```
package auth;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;
import java.util.concurrent.atomic.AtomicReference;

import static auth.User.ROLE.ADMIN;
import static auth.User.ROLE.USER;

@WebListener
public class DataBase implements ServletContextListener {

    private AtomicReference<UserPut> useres;

    @Override
    public void contextInitialized(ServletContextEvent servletContextEvent) {

        useres = new AtomicReference<> (new UserPut());

        useres.get().add(new User("Ruslan", "chunchunmaru", ADMIN));
        useres.get().add(new User("Egor", "amidamaru", ADMIN));
        useres.get().add(new User("Vova", "12345qwerty", ADMIN));
        useres.get().add(new User("Nikita", "NonPlayerCharacter", ADMIN));
        useres.get().add(new User("1", "1", ADMIN));
        useres.get().add(new User("2", "2", USER));
    }
}
```



```

        final ServletContext servletContext =
            servletContextEvent.getServletContext();

        servletContext.setAttribute("useres", useres);
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        useres = null;
    }
}

```

Программный код класса Filtrator:

```

package auth;

import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import laba4.ReadFile;

import java.io.IOException;
import java.util.concurrent.atomic.AtomicReference;

import static java.util.Objects.nonNull;

//Acidification filter.
public class Filtrator implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
    }

    @Override
    public void doFilter(final HttpServletRequest request,
                        final HttpServletResponse response,
                        final FilterChain filterChain)

        throws IOException, ServletException {

        final HttpServletRequest req = (HttpServletRequest) request;
        final HttpServletResponse res = (HttpServletResponse) response;

        final String login = req.getParameter("login");
        final String password = req.getParameter("password");

        @SuppressWarnings("unchecked")
        final AtomicReference<UserPut> useres = (AtomicReference<UserPut>)
req.getServletContext().getAttribute("useres");

        final HttpSession session = req.getSession();

        //Logged user.
        if (nonNull(session) &&
            nonNull(session.getAttribute("login")) &&

```

```

        nonNull(session.getAttribute("password"))) {

            final User.ROLE role = (User.ROLE) session.getAttribute("role");

            moveToMenu(req, res, role);

        } else if (useres.get().userIsExist(login, password)) {

            final User.ROLE role = useres.get().getRoleByLoginPassword(login, password);

            req.getSession().setAttribute("password", password);
            req.getSession().setAttribute("login", login);
            req.getSession().setAttribute("role", role);

            moveToMenu(req, res, role);

        } else {

            moveToMenu(req, res, User.ROLE.UNKNOWN);

        }
    }

    private void moveToMenu(final HttpServletRequest req,
                            final HttpServletResponse res,
                            final User.ROLE role)
        throws ServletException, IOException {

        if (role.equals(User.ROLE.ADMIN)) {
            req.getSession().setAttribute("loginADMIN", "admin");
            req.getRequestDispatcher("/admin_menu.jsp").forward(req, res);

        } else if (role.equals(User.ROLE.USER)) {
            req.getSession().setAttribute("loginUSER", "user");
            ReadFile read = new ReadFile();
            read.doPost(req, res);

        } else {
            req.getRequestDispatcher("/login.jsp").forward(req, res);
        }
    }

    @Override
    public void destroy() {
    }

}

```

Программный код класса Logout:

```

package auth;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

```

```

import javax.servlet.http.HttpSession;

public class Logout extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {

        final HttpSession session = req.getSession();

        session.removeAttribute("password");
        session.removeAttribute("login");
        session.removeAttribute("role");
        session.removeAttribute("loginUSER");
        session.removeAttribute("loginADMIN");

        resp.sendRedirect(req.getContextPath() + "/");

    }
}

```

Программный код класса User:

```

package auth;

import auth.User.ROLE;

public class User {

    private String login;

    private String password;

    private ROLE role;

    public User() {
    }

    public User(String login, String password, ROLE role) {
        this.login = login;
        this.password = password;
        this.role = role;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}

```

```

    }

    public ROLE getRole() {
        return role;
    }

    public void setRole(ROLE role) {
        this.role = role;
    }

    public enum ROLE {
        USER, ADMIN, UNKNOWN
    }
}

```

Программный код класса UserPut:

```

package auth;

import java.util.ArrayList;
import java.util.List;

public class UserPut {

    private final List<User> store = new ArrayList<>();

    public User getUserByLoginPassword(final String login, final String password) {

        User result = new User();

        for (User user : store) {
            if (user.getLogin().equals(login) && user.getPassword().equals(password)) {
                result = user;
            }
        }

        return result;
    }

    public boolean add(final User user) {

        for (User u : store) {
            if (u.getLogin().equals(user.getLogin()) &&
u.getPassword().equals(user.getPassword())) {
                return false;
            }
        }

        return store.add(user);
    }

    public User.ROLE getRoleByLoginPassword(final String login, final String password) {
        User.ROLE result = User.ROLE.UNKNOWN;

        for (User user : store) {
            if (user.getLogin().equals(login) && user.getPassword().equals(password)) {
                result = user.getRole();
            }
        }
    }
}

```

```

        return result;
    }

    public boolean userIsExist(final String login, final String password) {

        boolean result = false;

        for (User user : store) {
            if (user.getLogin().equals(login) && user.getPassword().equals(password)) {
                result = true;
                break;
            }
        }

        return result;
    }
}

```

Программный код класса Calc:

```

package laba4;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import AbstractClass.Karkas;
import AbstractClass.Napolnitel;
import AbstractClass.Obivka;
import AbstractClass.Vidstul;

@WebServlet(name="Calc", urlPatterns={"/JavaCalc"}) //связывание сервлета с URL
public class Calc extends HttpServlet {

    public static String KarkasGet;
    public static String KarkasGetPrice;
    public static String ObivkaGet;
    public static String ObivkaGetPrice;
    public static String NapolnitelGet;
    public static String NapolnitelGetPrice;
    public static String KolvoChairGet;
    public static String ChairGet;
    public static String ChairGetPrice;
    public static String check1Get;
    public static String Summa;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        if (request.getSession().getAttribute("loginUSER") == "user") {
            RequestCalc Calc = RequestCalc.fromRequestParameters(request);
            Calc.setAsRequestAttributesAndCalculate(request);
            CreatePDF PDF = new CreatePDF();

```

```

        String goals = "Hello";
        PDF.Create(goals);
        request.getRequestDispatcher("/Results.jsp").forward(request, response);
    }
    else {
        request.getRequestDispatcher("/NotAuth.jsp").forward(request,
response);
    }
}
private static class RequestCalc {
    private final String cenaKarkas;
    private final String cenaObivka;
    private final String cenaNapolnitel;
    private final String KolvoChair;
    private final String cenaOneChair;
    private final String check1;
    private final String vidStula2;
    private String result;
    private double result1;
    private String karkas1;
    private String stul1;
    private String obivka1;
    private String napolnitel1;

    private RequestCalc (String priceKarkas,
        String priceObivka, String priceNapolnitel,String NumberChair,
String priceForOneChair,String check,String VidStula) {
        this.cenaKarkas = priceKarkas;
        this.cenaObivka = priceObivka;
        this.cenaNapolnitel = priceNapolnitel;
        this.KolvoChair = NumberChair;
        this.cenaOneChair = priceForOneChair;
        this.vidStula2 = VidStula;
        if (check == null) {
            check1 = "Нет";
        } else {
            check1 = "Да";
        }
    }

    public static RequestCalc fromRequestParameters(HttpServletRequest request) {
        return new RequestCalc(
            request.getParameter("priceKarkas"),
            request.getParameter("priceObivka"),
            request.getParameter("priceNapolnitel"),
            request.getParameter("NumberChair"),
            request.getParameter("priceForOneChair"),
            request.getParameter("check"),
            request.getParameter("VidStula"));
    }

    public void setAsRequestAttributesAndCalculate(HttpServletRequest request) {
        request.setAttribute("first_result", cenaKarkas);
        request.setAttribute("second_result", cenaObivka);
        request.setAttribute("third_result", cenaNapolnitel);
        request.setAttribute("four_result", KolvoChair);
        request.setAttribute("five_result", cenaOneChair);
        request.setAttribute("checked", check1);

        String num[] = new String[20];
        try {

```

```

        File file = new File("program1.txt");
        if(file.exists()){
            BufferedReader br = new BufferedReader(new
FileReader(file.getAbsolutePath()));
            for (int i = 0; i < 20; i++) {
                num[i] = br.readLine();
            }
        }
    }
    catch (Exception e) {
        System.err.println(e.getMessage());
    }

    Vidstul GameChair = new Vidstul(); GameChair.setPrice(num[0]);
GameChair.setName("Игровое кресло");
    Vidstul OfficeChair = new Vidstul(); OfficeChair.setPrice(num[1]);
OfficeChair.setName("Офисное кресло");
    Vidstul Pufik = new Vidstul(); Pufik.setPrice(num[2]);
Pufik.setName("Пуфик");
    Vidstul KachalkaChair = new Vidstul(); KachalkaChair.setPrice(num[3]);
KachalkaChair.setName("Кресло-качалка");
    Vidstul MyagkiyChair = new Vidstul(); MyagkiyChair.setPrice(num[4]);
MyagkiyChair.setName("Стул с мягкой спинкой");
    if (vidStula2.equals(GameChair.getPrice())) { stul1 =
GameChair.getName();}
    if (vidStula2.equals(OfficeChair.getPrice())) { stul1 =
OfficeChair.getName();}
    if (vidStula2.equals(Pufik.getPrice())) { stul1 = Pufik.getName();}
    if (vidStula2.equals(KachalkaChair.getPrice())) { stul1 =
KachalkaChair.getName();}
    if (vidStula2.equals(MyagkiyChair.getPrice())) { stul1 =
MyagkiyChair.getName();}

    Karkas Karkasmet = new Karkas(); Karkasmet.setPrice(num[5]);
Karkasmet.setName("Металл");
    Karkas Karkasderev = new Karkas(); Karkasderev.setPrice(num[6]);
Karkasderev.setName("Дерево");
    Karkas Karkasplastic = new Karkas(); Karkasplastic.setPrice(num[7]);
Karkasplastic.setName("Пластик");
    if (cenaKarkas.equals(Karkasmet.getPrice())) { karkas1 =
Karkasmet.getName();}
    if (cenaKarkas.equals(Karkasderev.getPrice())) { karkas1 =
Karkasderev.getName();}
    if (cenaKarkas.equals(Karkasplastic.getPrice())) { karkas1 =
Karkasplastic.getName();}

    Obivka IskKoja = new Obivka();IskKoja.setPrice(num[8]);
IskKoja.setName("Искусственная кожа");
    Obivka EcoKoja = new Obivka();EcoKoja.setPrice(num[9]);
EcoKoja.setName("Эко-кожа");
    Obivka Tkan = new Obivka();Tkan.setPrice(num[10]);
Tkan.setName("Ткань");
    Obivka Zamcha = new Obivka();Zamcha.setPrice(num[11]);
Zamcha.setName("Замша");
    Obivka Velvet = new Obivka();Velvet.setPrice(num[12]);
Velvet.setName("Вельвет");
    if (cenaObivka.equals(IskKoja.getPrice())) { obivka1 =
IskKoja.getName();}
    if (cenaObivka.equals(EcoKoja.getPrice())) { obivka1 =
EcoKoja.getName();}
    if (cenaObivka.equals(Tkan.getPrice())) { obivka1 = Tkan.getName();}

```

```

        if (cenaObivka.equals(Zamcha.getPrice())) { obivka1 =
Zamcha.getName();}
        if (cenaObivka.equals(Velvet.getPrice())) { obivka1 =
Velvet.getName();}

        Napolnitel Porolon = new Napolnitel();Porolon.setPrice(num[13]);
Porolon.setName("Поролон");
        Napolnitel Sintepon = new Napolnitel();Sintepon.setPrice(num[14]);
Sintepon.setName("Синтепон");
        Napolnitel Voylok = new Napolnitel();Voylok.setPrice(num[15]);
Voylok.setName("Войлок");
        if (cenaNapolnitel.equals(Porolon.getPrice())) { napolnitel1 =
Porolon.getName();}
        if (cenaNapolnitel.equals(Sintepon.getPrice())) { napolnitel1 =
Sintepon.getName();}
        if (cenaNapolnitel.equals(Voylok.getPrice())) { napolnitel1 =
Voylok.getName();}

        request.setAttribute("karkas_result", karkas1);
        request.setAttribute("vidstula_result", stul1);
        request.setAttribute("obivka_result", obivka1);
        request.setAttribute("napolnitel_result", napolnitel1);

        double Kark=0;
        double Obiv=0;
        double Napoln=0;
        double Kolvo=0;
        double PriceOne=0;
        double CenaObivki=0;
        try {
            Kark=Double.parseDouble(cenaKarkas);
            Obiv=Double.parseDouble(cenaObivka);
            Napoln=Double.parseDouble(cenaNapolnitel);
            Kolvo=Double.parseDouble(KolvoChair);
            PriceOne=Double.parseDouble(cenaOneChair);
        }
        catch (Exception e) {
            Kark = 0;
            Obiv = 0;
            Napoln = 0;
            Kolvo = 0;
            PriceOne = 0;
        }

        CenaObivki = Obiv+Napoln;
        result1=(Kark+CenaObivki+PriceOne)*Kolvo;
        result = "" + result1;
        request.setAttribute("result", result);

        KarkasGet=karkas1;
        KarkasGetPrice=cenaKarkas;
        ObivkaGetPrice=cenaObivka;
        ObivkaGet=obivka1;
        NapolnitelGet=napolnitel1;
        NapolnitelGetPrice=cenaNapolnitel;
        KolvoChairGet=KolvoChair;
        ChairGetPrice=cenaOneChair;
        ChairGet=stul1;
        check1Get=check1;
        Summa=result;
    }

```



```

    }
}

```

Программный код класса CreatePDF:

```

package laba4;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.URISyntaxException;
import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.Font;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.pdf.BaseFont;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;

public class CreatePDF {

    public String filepath;
    public String file1 ;

    public void Create(String numberpdf) throws IOException {
        Document document = new Document();
        try {
            File file = new
File(CreatePDF.class.getProtectionDomain().getCodeSource().getLocation().toURI());
            file1 = new File(file.getParent()).getParent();
            filepath = file1 + "/Check.pdf";
        } catch (URISyntaxException e2) {
            e2.printStackTrace();
        }
        try {
            PdfWriter.getInstance(document, new FileOutputStream(filepath));
        } catch (FileNotFoundException | DocumentException e) {
            e.printStackTrace();
        }

        document.open();

        BaseFont times = null;
        try {
            times = BaseFont.createFont(file1 + "/fonts/times.ttf",
"cp1251", BaseFont.EMBEDDED);
        } catch (DocumentException | IOException e) {
            e.printStackTrace();
        }

        String string_pdf = "Результат выполнения работы приложения";
        Paragraph paragraph = new Paragraph();
        paragraph.add(new Paragraph(string_pdf, new Font(times,14)));

        String string_pdf2 = "С заказом можно ознакомиться в таблице ниже:";
        paragraph.add(new Paragraph(string_pdf2, new Font(times,14)));
    }
}

```

```

    try {
        document.add(paragraph);
    } catch (DocumentException e1) {
        e1.printStackTrace();
    }

    paragraph.clear();
    String string_pdf3 = " ";
    paragraph.add(new Paragraph(string_pdf3, new Font(times,14)));

    try {
        document.add(paragraph);
    } catch (DocumentException e1) {
        e1.printStackTrace();
    }

    paragraph.clear();
    paragraph.add(new Paragraph(string_pdf3, new Font(times,14)));

    try {
        document.add(paragraph);
    } catch (DocumentException e1) {
        e1.printStackTrace();
    }

    PdfPTable table = new PdfPTable(3);
    addColumns(table);
    try {
        document.add(table);
    } catch (DocumentException e) {
        e.printStackTrace();
    }

    document.close();
}

private void addColumns(PdfPTable table) {
    BaseFont times = null;
    try {
        times = BaseFont.createFont(file1 + "/fonts/times.ttf", "cp1251",
BaseFont.EMBEDDED);
    } catch (DocumentException | IOException e) {
        e.printStackTrace();
    }

    String cell11 = "";
    String cell12 = "Вид";
    String cell13 = "Цена (руб)";
    String cell14 = "Каркас";
    String cell15 = Calc.KarkasGet;
    String cell16 = Calc.KarkasGetPrice;
    String cell17 = "Обивка";
    String cell18 = Calc.ObivkaGet;
    String cell19 = Calc.ObivkaGetPrice;

```

```

String cell110 = "Наполнитель";
String cell111 = Calc.NapolnitelGet;
String cell112 = Calc.NapolnitelGetPrice;
String cell113 = "Стул";
String cell114 = Calc.ChairGet;
String cell115 = Calc.ChairGetPrice;
String cell116 = "Количество стульев";
String cell117 = Calc.KolvoChairGet;
String cell118 = "";
String cell119 = "Срочный заказ?";
String cell120 = Calc.check1Get;
String cell121 = "";
String cell122 = "";
String cell123 = "ИТОГО:";
String cell124 = Calc.Summa;

table.addCell(new Phrase(cell11, new Font(times,14)));
table.addCell(new Phrase(cell12, new Font(times,14)));
table.addCell(new Phrase(cell13, new Font(times,14)));
table.addCell(new Phrase(cell14, new Font(times,14)));
table.addCell(new Phrase(cell15, new Font(times,14)));
table.addCell(new Phrase(cell16, new Font(times,14)));
table.addCell(new Phrase(cell17, new Font(times,14)));
table.addCell(new Phrase(cell18, new Font(times,14)));
table.addCell(new Phrase(cell19, new Font(times,14)));
table.addCell(new Phrase(cell110, new Font(times,14)));
table.addCell(new Phrase(cell111, new Font(times,14)));
table.addCell(new Phrase(cell112, new Font(times,14)));
table.addCell(new Phrase(cell113, new Font(times,14)));
table.addCell(new Phrase(cell114, new Font(times,14)));
table.addCell(new Phrase(cell115, new Font(times,14)));
table.addCell(new Phrase(cell116, new Font(times,14)));
table.addCell(new Phrase(cell117, new Font(times,14)));
table.addCell(new Phrase(cell118, new Font(times,14)));
table.addCell(new Phrase(cell119, new Font(times,14)));
table.addCell(new Phrase(cell120, new Font(times,14)));
table.addCell(new Phrase(cell121, new Font(times,14)));
table.addCell(new Phrase(cell122, new Font(times,14)));
table.addCell(new Phrase(cell123, new Font(times,14)));
table.addCell(new Phrase(cell124, new Font(times,14)));

    }
}

```

Программный код класса ReadFile:

```

package laba4;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;

```

```

import javax.servlet.http.HttpServletResponse;

@WebServlet(name="Read", urlPatterns="/Read")
public class ReadFile extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        read(request);
        request.getRequestDispatcher("/Form.jsp").forward(request, response);
    }
    String num[] = new String[20];
    public void read(HttpServletRequest request) {
        try {
            File file = new File("program1.txt");
            if(file.exists()){
                BufferedReader br = new BufferedReader(new
FileReader(file.getAbsolutePath()));
                for (int i = 0; i < 20; i++) {
                    num[i] = br.readLine();
                }
            }
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }

        try {
            request.setAttribute("vid1", num[0]);
            request.setAttribute("vid2", num[1]);
            request.setAttribute("vid3", num[2]);
            request.setAttribute("vid4", num[3]);
            request.setAttribute("vid5", num[4]);
            request.setAttribute("kark1", num[5]);
            request.setAttribute("kark2", num[6]);
            request.setAttribute("kark3", num[7]);
            request.setAttribute("obiv1", num[8]);
            request.setAttribute("obiv2", num[9]);
            request.setAttribute("obiv3", num[10]);
            request.setAttribute("obiv4", num[11]);
            request.setAttribute("obiv5", num[12]);
            request.setAttribute("napoln1", num[13]);
            request.setAttribute("napoln2", num[14]);
            request.setAttribute("napoln3", num[15]);
        }
        catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
}

```

Программный код класса WriteFile:

```

package laba4;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;

```

```

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import AbstractClass.Karkas;

@WebServlet(name="WriteFile", urlPatterns="/WriteFile")
public class WriteFile extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        if (request.getSession().getAttribute("loginADMIN") == "admin") {
            RequestCalc Calc = RequestCalc.fromRequestParameters(request);
            Calc.setAsRequestAttributesAndCalculate(request);
            response.sendRedirect(request.getContextPath() + "/admin_menu.jsp");
        }
        else {
            request.getRequestDispatcher("/NotAuth.jsp").forward(request,
response);
        }
    }
    static String num[] = new String[20];
    static String num2[] = new String[20];
    private static class RequestCalc {
        private final String st1;
        private final String st2;
        private final String st3;
        private final String st4;
        private final String st5;
        private final String kar1;
        private final String kar2;
        private final String kar3;
        private final String o1;
        private final String o2;
        private final String o3;
        private final String o4;
        private final String o5;
        private final String napol1;
        private final String napol2;
        private final String napol3;
        private RequestCalc (String v1,String v2,String v3,String v4,String v5,String
k1,String k2,String k3,
                        String ob1, String ob2, String ob3, String ob4, String ob5,
String nap1, String nap2, String nap3 ) {
            this.st1 = v1;
            this.st2 = v2;
            this.st3 = v3;
            this.st4 = v4;
            this.st5 = v5;
            this.kar1 = k1;
            this.kar2 = k2;
            this.kar3 = k3;
            this.o1 = ob1;
            this.o2 = ob2;
            this.o3 = ob3;
            this.o4 = ob4;
            this.o5 = ob5;
            this.napol1 = nap1;
            this.napol2 = nap2;
            this.napol3 = nap3;
        }
        public static RequestCalc fromRequestParameters(HttpServletRequest request) {

```

```

        return new RequestCalc(
            request.getParameter("v1"),
            request.getParameter("v2"),
            request.getParameter("v3"),
            request.getParameter("v4"),
            request.getParameter("v5"),
            request.getParameter("k1"),
            request.getParameter("k2"),
            request.getParameter("k3"),
            request.getParameter("ob1"),
            request.getParameter("ob2"),
            request.getParameter("ob3"),
            request.getParameter("ob4"),
            request.getParameter("ob5"),
            request.getParameter("nap1"),
            request.getParameter("nap2"),
            request.getParameter("nap3"));
    }

    public void setAsRequestAttributesAndCalculate(HttpServletRequest request) {
        try {
            File file = new File("program1.txt");
            if(file.exists()){
                BufferedReader br = new BufferedReader(new
FileReader(file.getAbsolutePath()));

                for (int i = 0; i < 20; i++) {
                    num[i] = br.readLine();
                }
                num2[0] = st1;
                num2[1] = st2;
                num2[2] = st3;
                num2[3] = st4;
                num2[4] = st5;
                num2[5] = kar1;
                num2[6] = kar2;
                num2[7] = kar3;
                num2[8] = o1;
                num2[9] = o2;
                num2[10] = o3;
                num2[11] = o4;
                num2[12] = o5;
                num2[13] = napol1;
                num2[14] = napol2;
                num2[15] = napol3;
                for (int i = 0; i < 20; i++) {
                    if (num2[i] == "") {
                        num2[i] = num[i];
                    }
                }
            }
            PrintWriter zapis = new PrintWriter(file.getAbsolutePath());
            try {
                zapis.write(num2[0] + "\n" + num2[1] + "\n" + num2[2] + "\n" + num2[3] + "\n" + num2[4] +
"\n" + num2[5] + "\n" + num2[6] + "\n" + num2[7] + "\n" + num2[8] + "\n" + num2[9] + "\n"
+num2[10] + "\n" + num2[11] + "\n" + num2[12] + "\n" + num2[13] + "\n" + num2[14] + "\n" +
num2[15]);
            }
            finally {
                zapis.close();
            }
        } else {
            file.createNewFile();
            PrintWriter zapis = new PrintWriter(file.getAbsolutePath());

```

```

        try {
            zapis.write(st1 + "\n" + st2 + "\n" + st3 + "\n" + st4 + "\n" + st5 + "\n" + kar1 + "\n" +
            kar2 + "\n" + kar3 + "\n" + o1 + "\n" + o2 + "\n" + o3 + "\n" + o4 + "\n" + o5 + "\n" +
            napol1 + "\n" + napol2 + "\n" + napol3);
        }
        finally {
            zapis.close();
        }
    }
}
catch (Exception e) {
    System.err.println(e.getMessage());
}
}
}
}

```

Программный код формы admin_menu:

```

<html>
<head>
    <title>ADMIN</title>
</head>
<body>
<form action="${pageContext.request.contextPath}/WriteFile" method="post">
<h1 style="position:relative;text-align:center">Панель администратора</h1>
<h3>Стоимость вида стула:</h3>
    <p>Игровой стул:
    <input name="v1" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Офисное кресло:
    <input name="v2" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Пуфик:
    <input name="v3" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Кресло-качалка:
    <input name="v4" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Стул с мягкой спинкой:
    <input name="v5" type="number" min="1" max="10000" name="NumberChair"></p>
<h3>Стоимость материала каркаса:</h3>
    <p>Металл:
    <input name="k1" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Дерево:
    <input name="k2" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Пластик:
    <input name="k3" type="number" min="1" max="10000" name="NumberChair"></p>
<h3>Стоимость материала обивки:</h3>
    <p>Искусственная кожа:
    <input name="ob1" type="number" min="1" max="10000" name="NumberChair" ></p>
    <p>Эко-кожа:
    <input name="ob2" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Ткань:
    <input name="ob3" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Замша:
    <input name="ob4" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Вельвет:
    <input name="ob5" type="number" min="1" max="10000" name="NumberChair"></p>
<h3>Стоимость материала наполнителя:</h3>
    <p>Поролон:
    <input name="nap1" type="number" min="1" max="10000" name="NumberChair"></p>
    <p>Синтепон:
    <input name="nap2" type="number" min="1" max="10000" name="NumberChair"></p>

```

```

<p>Войлок:
<input name="nap3" type="number" min="1" max="10000" name="NumberChair"></p>
<input style="position:relative;left: 45%" type="submit" value="Изменить значения">
</form>
<a href="<c:url value='/logout' />" style="position:relative;left: 45%">Logout</a>
</body>
</html>

```

Программный код формы Form:

```

<html>
<head>
<script>
function myFunction1(e) {
    document.getElementById("priceForOneChair").value = e.target.value
}
function myFunction2(e) {
    document.getElementById("priceKarkas").value = e.target.value
}
function myFunction3(e) {
    document.getElementById("priceObivka").value = e.target.value
}
function myFunction4(e) {
    document.getElementById("priceNapolnitel").value = e.target.value
}
function myFunction5(e) {
    var x = document.getElementById("kk").value;
    var x1 = document.getElementById("priceForOneChair").value = x*1.5;
    var cb = document.getElementById("checkbox_check");
    if (cb.checked) {
        document.getElementById("priceForOneChair").value = x*1.5;
    } else {
        document.getElementById("priceForOneChair").value = x1/1.5;
    }
}
</script>
<meta charset="UTF-8">
<title>Калькулятор стоимости производства мягкой мебели</title>
</head>
<body>
<h1 style="position:relative;text-align:center">Калькулятор стоимости производства мягкой мебели</h1>
<form action="{pageContext.request.contextPath}/JavaCalc" method="post">
<label style="position:relative;left: 45%">Вид стула:</label>
    <select id="kk" name="VidStula" required onchange="myFunction1(event);"
style="position:relative;left: 45%">
        <option selected disabled =""></option>
        <option value="{ vid1 }">Игровое кресло</option>
        <option value="{ vid2 }">Офисное кресло</option>
        <option value="{ vid3 }">Пуфик</option>
        <option value="{ vid4 }">Кресло-качалка</option>
        <option value="{ vid5 }">Стул с мягкой спинкой</option>
    </select>
<p><label style="position:relative;left: 34%;">Материал каркаса:</label>
    <select name="priceKarkas" required onchange="myFunction2(event);"
style="position:relative;left: 34%;width:8%">
        <option selected disabled =""></option>
        <option value="{ kark1 }">Металл</option>
        <option value="{ kark2 }">Дерево</option>
        <option value="{ kark3 }">Пластик</option>

```



```

</select>
<label style="position:relative;left: 38.5%">Стоимость каркаса:</label>
  <input id="priceKarkas" readonly size="15" style="position:relative;left:
38.5%">
</p>
  <p><label style="position:relative;left: 34%">Материал обивки:</label>
    <select name="priceObivka" required onchange="myFunction3(event);"
style="position:relative;left: 34%;width:8%" >
      <option selected disabled =""></option>
      <option value="{ obiv1 }">Искусственная кожа</option>
      <option value="{ obiv2 }">Эко-кожа</option>
      <option value="{ obiv3 }">Ткань</option>
      <option value="{ obiv4 }">Замша</option>
      <option value="{ obiv5 }">Вельвет</option>
    </select>
    <label style="position:relative;left: 35%">Стоимость материала обивки:</label>
      <input id="priceObivka" readonly size="15"
style="position:relative;left:35%">
  </p>
  <p><label style="position:relative;left: 34%">Материал наполнителя:</label>
    <select name="priceNapolnitel" required onchange="myFunction4(event);"
style="position:relative;left: 34%;width:6%">
      <option selected disabled =""></option>
      <option value="{ napoln1 }">Поролон</option>
      <option value="{ napoln2 }">Синтепон</option>
      <option value="{ napoln3 }">Войлок</option>
    </select>
    <label style="position:relative;left: 36.9%">Стоимость наполнителя:</label>
      <input id="priceNapolnitel" readonly size="15"
style="position:relative;left: 36.9%">
  </p>
  <p> <label style="position:relative;left: 40%">Количество стульев:</label>
    <input type="number" min="1" name="NumberChair" required
style="position:relative;left: 40%"> </p>
  <p> <label style="position:relative;left: 37%">Стоимость работы за 1 стул:</label>
    <input id="priceForOneChair" name="priceForOneChair" readonly
style="position:relative;left: 37%"> </p>
  <p><label style="position:relative;left: 41.5%">Срочный заказ:</label>
    <input type="checkbox" onclick="myFunction5(event);" id="checkbox_check"
name="check" style="position:relative;left: 41.5%"> </p>
    <input style="position:relative;left: 45%" type="submit" value=" Рассчитать ">
  </form>
  <form action="{pageContext.request.contextPath}/Spravka.jsp" method="post">
    <input style="position:relative;left: 45%" type="submit" value=" Справка ">
  </form>
  <a href="{c:url value='/Logout' /}" style="position:relative;left: 45%">Logout</a>
</body>
</html>

```

Программный код формы login:

```

<html>
<head>
  <title>Login</title>

</head>
<body>

  <div class="form">

```

```

<h1>Вход в систему</h1><br>
<form method="post" action="">

    <input type="text" required placeholder="Login" name="Login"><br>
    <input type="password" required placeholder="password"
name="password"><br><br>
    <input class="button" type="submit" value="Войти">

</form>
</div>
</body>
</html>

```

Программный код формы NotAuth:

```

<html>
<head>
<meta charset="UTF-8">
<title>Не авторизован</title>
</head>
<body>
    <p>Пожалуйста пройдите авторизацию</p>
    <a href="<c:url value='/Logout' />">Авторизация</a>
</body>
</html>

```

Программный код формы Results:

```

<html>
<head>
<meta charset="UTF-8">
<title>Стоимость производства мягкой мебели:</title>
</head>
<body>
<h1>Итоговая стоимость производства мягкой мебели:</h1>
<h2>Ваши введенные данные:</h2>
<p><strong>Вид стула:</strong> ${vidstula_result}</p>
<p><strong>Материал каркаса:</strong> ${karkas_result}</p>
<p><strong>Материал обивки:</strong> ${obivka_result}</p>
<p><strong>Материал наполнителя:</strong> ${napolnitel_result}</p>
<p><strong>Стоимость каркаса:</strong> ${first_result}</p>
<p><strong>Стоимость обивки:</strong> ${second_result}</p>
<p><strong>Стоимость наполнителя:</strong> ${third_result}</p>
<p><strong>Количество стульев:</strong> ${four_result}</p>
<p><strong>Стоимость работы за 1 стул:</strong> ${five_result}</p>
<p><strong>Срочный заказ:</strong> ${checked}</p>
<p><strong>Итоговая стоимость:</strong> ${result}</p>
<a href="Check.pdf"> Open PDF-file</a>
<a href="Check.pdf" download> Download PDF-file</a>
<form action="${pageContext.request.contextPath}/Read" method="post" >
    <input type="submit" name="sign" value="Назад">
</form>
<a href="<c:url value='/Logout' />" style="position:relative;left: 45%">Logout</a>
</body>
</html>

```

Программный код формы Spravka:

```
<html>
<head>
<meta charset="UTF-8">
<title>Справка:</title>
</head>
<body>
<h1>Номера зачетных книжек указаны ниже:</h1>
<p><strong>Модератор: Пупков Егор Андреевич - 19130149</strong></p>
<p><strong>Первый участник: Шамсутдинов Руслан Маратович - 19130342</strong></p>
<p><strong>Второй участник: Денисов Никита Андреевич - 19130174</strong></p>
<p><strong>Третий участник: Леонтьев Владимир Алексеевич - 19130155</strong></p>
<form action="{pageContext.request.contextPath}/Read" method="post">
    <input type="submit" name="sign" value="Назад">
</form>
</body>
</html>
```

[illegible]

TestPDF

```
package testPDF;

import static org.junit.Assert.*;
import org.junit.Test;
import com.itextpdf.text.pdf.PdfPTable;
import laba4.CreatePDF;

public class TestPDF {
    @Test
    public void test() {
        CreatePDF k = new CreatePDF();
        PdfPTable table = new PdfPTable(3);
        k.addColumn(table);
        assertEquals("Капка", k.getCellGet());
    }
}
```

DataTest

```
package auth;

import static org.junit.Assert.*;

import org.junit.Test;

public class DataTest {

    @Test
    public void test() {
        DataBase bus = new DataBase();
        assertNull(bus.useres);
    }

}
```

TestAbstr

```
package auth;

import static org.junit.Assert.*;

import org.junit.Test;

import AbstractClass.Karkas;
public class TestAbstr {

    @Test //Тест абстрактного класса
    public void test() {
        Karkas kark= new Karkas();
        String N= "name";
        kark.setName("name");
        String N1= kark.getName();

        assertEquals(N, N1); //Проверка на нулевое значение метода
    }
}
```

TestWrite

```

package testWrite;

import static org.junit.Assert.*;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.PrintWriter;

import org.junit.Test;

public class TestWrite {

    @Test
    public void test() {
        String num[] = new String[20];
        File file = new File("program1.txt");
        try {
            if(file.exists()){
                PrintWriter zapis = new PrintWriter(file.getAbsolutePath());
                try {
                    zapis.write("1" + "\n" + "2" + "\n");
                }
                finally {
                    zapis.close();
                }
            } else {
                file.createNewFile();
                PrintWriter zapis = new PrintWriter(file.getAbsolutePath());
                try {
                    zapis.write("1" + "\n" + "2" + "\n");
                }
                finally {
                    zapis.close();
                }
            }
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }

        try {
            BufferedReader br = new BufferedReader(new
            FileReader(file.getAbsolutePath()));
            for (int i = 0; i < 20; i++) {
                num[i] = br.readLine();
            }
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }

        assertTrue(file.exists());
        assertEquals("1", num[0]);
        assertEquals("2", num[1]);
    }
}

```


СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Каталог ГОСТ 2021. URL: <https://internet-law.ru/gosts/>
2. Wikipedia – Мягкая мебель.
URL: https://ru.wikipedia.org/wiki/Мягкая_мебель
3. Организация производства мягкой мебели.
URL: <https://mmkc.su/novichkam/proizvodstvo-myagkoi-mebeli/>
4. Производство мягкой мебели как бизнес.
URL: <https://moybiznes.org/proizvodstvo-myagkoy-mebeli/>
5. Wikipedia – Debian. URL: <https://ru.wikipedia.org/wiki/Debian>
6. Wikipedia – Travis CI. URL: https://ru.wikipedia.org/wiki/Travis_CI
7. Wikipedia – Непрерывная интеграция.
URL: https://ru.wikipedia.org/wiki/Непрерывная_интеграция