

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»

Выполнил:  
Ст. гр. ПИ-221  
Белоусов А.Н.  
Боголюбов М.В.  
Михайлов В.А.  
Тахаев А.Г.

Проверил:  
Преподаватель  
Казанцев А.В.

Уфа – 2021

# Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования  
«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

## ЗАДАНИЕ

на курсовую работу по дисциплине «Информационные системы»

Студент	<u>Белоусов А.Н.</u> Фамилия И.О.	Группа	<u>ПИ-221</u> номер группы	Консультант	<u>Казанцев А.В.</u> Фамилия И.О.
Студент	<u>Боголюбов М.В.</u> Фамилия И.О.	Группа	<u>ПИ-221</u> номер группы	Консультант	<u>Казанцев А.В.</u> Фамилия И.О.
Студент	<u>Михайлов В.А.</u> Фамилия И.О.	Группа	<u>ПИ-221</u> номер группы	Консультант	<u>Казанцев А.В.</u> Фамилия И.О.
Студент	<u>Тахаев А.Г.</u> Фамилия И.О.	Группа	<u>ПИ-221</u> номер группы	Консультант	<u>Казанцев А.В.</u> Фамилия И.О.

1. Тема курсового проекта: Разработка кроссплатформенного программного продукта на языке JAVA с использованием системы контроля версий.  
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 выносятся 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.

Студент \_\_\_\_\_ Белоусов А.Н.

Студент \_\_\_\_\_ Боголюбов М.В.

Студент \_\_\_\_\_ Михайлов В.А.

Студент \_\_\_\_\_ Тахаев А.Г.

## Содержание

Раздел 1.	Описание предметной области .....	5
1.1	Определение бизнес-процесса «отделочные работы» .....	5
1.2	Описание процесса «ремонт» .....	5
1.3	Регламентирующие документы для бизнес-процесса «отделочные работы» .....	7
1.4	Описание предприятия .....	7
1.5	Описание структуры и взаимодействие пользователей с программным продуктом .....	10
1.6	Математическая модель работы ПП .....	12
Раздел 2.	Техническое задание .....	15
Раздел 3.	Настройка среды разработки для операционных систем семейств Windows и Linux .....	16
3.1	Настройка среды разработки для дистрибутива Ubuntu 20.04 .....	16
3.2	Настройка среды разработки для Windows 10 .....	19
3.3	Настройка среды разработки для дистрибутива OpenSuse 15.2 .....	21
3.4	Создание проекта Maven и работа с JUnit .....	24
Раздел 4.	Настройка среды разработки для подключения к системе контроля версий .....	26
4.1	Настройка среды разработки Eclipse для работы с GitHub .....	26
4.2	Клонирование репозитория с GitHub в Eclipse .....	27
Раздел 5.	Реализация исходного кода по зонам ответственности .....	30
Раздел 6.	Сборка и тестирование программного продукта .....	32
Раздел 7.	Настройка программной среды для развертывания и запуска программного продукта .....	36
Раздел 8.	Руководство пользователя .....	39
	Заключение .....	40
	Приложения .....	41
	Список литературы .....	45

## **1 Описание предметной области**

Тематика в данной курсовой работе заключается в разработке калькулятора ремонта. Калькулятор специализируется на отделочных работах. Назначение данного калькулятора - расчет затрат за услуги с возможностью настроек параметров.

Отделочные работы — это комплекс строительных работ, связанных с наружной и внутренней отделкой зданий и сооружений с целью повышения их эксплуатационных и эстетических качеств.

### **1.1 Определение бизнес-процесса «отделочные работы»**

Бизнес-процесс «отделочные работы» используется в сфере услуг. Услуга, при данном бизнес процессе, определяется как - деятельность по производству продукта (материального или нематериального), осуществляемая по заказу клиента (потребителя), совместно с клиентом и за клиента, с передачей продукта клиенту с целью обмена. Данная услуга включает в себя:

- а) проектирование продукта и процесса его создания (согласование заказа);
- б) оценку(приемку) продукта.

Более точное определение - данный бизнес-процесс ориентируется на предоставление услуг: связанных с водоснабжением, отоплением, вентиляцией, кондиционированием и так далее. Услуги распространяются для применения на такие строения как: офисные помещения, квартиры, коттеджи, частные дома и различные дачные строения.

### **1.2 Описание процесса «ремонт»**

#### **1.2.1 Определение ремонта**

Ремонт - процесс изменения, восстановления, улучшения, доведения до первоначальных характеристик зданий и сооружений. Ремонт, согласно общепринятым нормам, имеет несколько видов:

- а) Капитальный ремонт — комплекс значительных работ по улучшению состояния зданий и сооружений;

б) Косметический ремонт — вид ремонта жилого помещения является собой упрощенную версию капитального ремонта и зачастую используется для обновления дизайна и/или устранения визуальных дефектов интерьера;

с) Евроремонт — отделка интерьера с применением современных технологий и материалов;

д) Ремонт коммуникаций — работы по замене сантехнических труб, электропроводки.

### 1.2.2 Срок и этапы ремонта

Для составления срока ремонта, определяют и анализируют основные факторы, которые влияют на длительность проведения ремонта. Основные факторы:

- а) Площадь строения;
- б) Вид, тип и класс ремонта;
- с) Навыки рабочей группы;
- д) Уникальные и творческие работы (дизайн проект);
- е) Исходное состояние строения.

Исходное состояние строения имеет 5 категорий:

- а) Исправное состояние;
- б) Работоспособное состояние;
- с) Ограниченно работоспособное состояние;
- д) Недопустимое состояние;
- е) Аварийное состояние.

Срок ремонта указывается в договорах и имеет 2 типа приоритетный и не приоритетный. За несоблюдение срока — штраф.

### **1.3 Регламентирующие документы для бизнес-процесса «отделочные работы»**

При ремонте строений производятся технически сложные комплексные мероприятия, в которые входят черновая и финишная отделка, расчет и монтаж коммуникаций и электросетей. Не зависимо от объема работа и действий работника, данная деятельность регламентируется строительными стандартами и нормами. Соответствующие стандарты:

а) СП 71.13330.2017 Изоляционные и отделочные покрытия. Актуализированная редакция СНиП 3.04.01-87 (с Изменением N 1);

б) СП 76.13330.2016 Электротехнические устройства. Актуализированная редакция СНиП 3.05.06-85;

в) СП 73.13330.2016 Внутренние санитарно-технические системы зданий. СНиП 3.05.01-85 (с Изменением N 1);

г) закон РФ № 384-ФЗ от 30.12.2009 г. (с изменениями на 2 июля 2013 года) «Технический регламент о безопасности зданий и сооружений»;

д) СП 29.13330.2011 Полы. Актуализированная редакция СНиП 2.03.13-88 (с Изменением N 1);

е) СП 60.13330.2016 Отопление, вентиляция и кондиционирование воздуха. Актуализированная редакция СНиП 41-01-2003 (с Изменением N 1);

ж) ФЗ о техническом регулировании (с изменениями на 28 ноября 2018 года);

з) ФЗ технический регламент о безопасности зданий и сооружений (с изменениями на 2 июля 2013 года).

### **1.4 Описание предприятия**

#### **1.4.1 Общее описание предприятия**

Предприятие ООО «Ремонтов» специализируется на отделочных работах, услуги которых указаны в таблице 1 с распределением на типы ремонта. При капитальном ремонте, на услуги из обычного ремонта действует скидка.

Предприятие предоставляет клиентам выбор класса ремонта. Всего имеется 3 класса:

- а) Эконом-класс;
- б) Комфорт-класс;
- с) Бизнес-класс.

Чем выше класс ремонта, тем более сложные материалы и виды отделки будут использоваться при проведении работ.

Таблица 1 – Услуги

Название услуги	Тип ремонта
Подготовка стен, потолков под финиш	Обычный ремонт
Оклейка обоями	
Покраска стен потолков	
Шумоизоляция стен	
Эскизный проект перепланировки	Капитальный ремонт
Штукатурка стен, потолков	
Очистка поверхностей от старых покрытий	
Шпатлевка стен потолков	
Нанесение декоративных покрытий	
Облицовка плиткой стен, потолков	
Укладка ламината	
Оклейка панелями потолок	

#### 1.4.2 Классификация предприятия

Классификация предприятия:

- а) по форме собственности — частное;
- б) по отраслевой принадлежности — строительство;



- с) по организационно-правовой форме — общество с ограниченной ответственностью;
- д) по размеру предприятия — мелкое;
- е) по используемым ресурсам — трудоемкое;
- ф) по цели деятельности предприятия — коммерческое.

### 1.4.3 Описание процессов и объектов предприятия в графическом виде

Мнемосхема отражает состояние системы, процессов и устройств. Мнемосхема облегчает понимание сущности процесса и назначение различных служб и объектов.

Описание мнемосхемы для бизнес-процесса «отделочные работы» (см. рис. 1).

Для получения расчета услуг покупатель производит запрос на сайт предприятия. В данном сайте покупатель выбирает услуги и настраивает параметры. После запускает выполнение расчета, с последующим выводом информации о стоимости ремонта и услуг. Начальство заключает договор с поставщиком, который доставляет стройматериалы и отдает указания связующему звену. Связующее звено предоставляет начальству отчет по проделанной работе и отправляет задания для рабочего коллектива. Рабочий коллектив выполняет поставленное задание. Интеллектуальные работники рабочего коллектива производят отладку калькулятора.

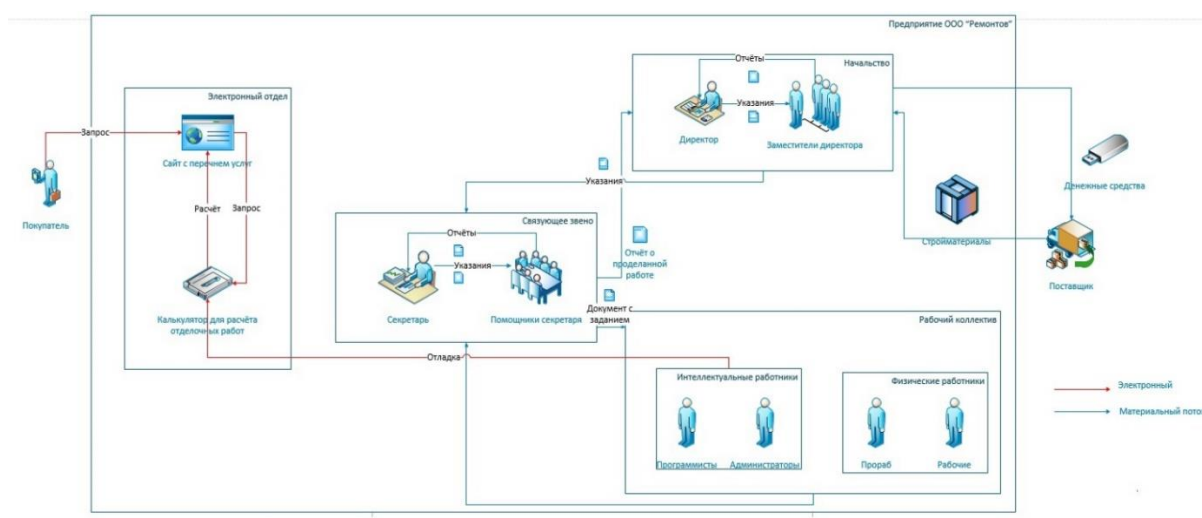


Рисунок 1 – Мнемосхема

## 1.5 Описание структуры и взаимодействие пользователей с программным продуктом

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

Описание диаграммы классов для калькулятора (см. рис. 2):

Диаграмма имеет следующие классы:

- 1) Inter1 (интерфейс);
- 2) Inter2 (интерфейс);
- 3) CalcPDF (сервлет);
- 4) RequestCalc;
- 5) Calc (сервлет);
- 6) Level (Абстрактный класс);
- 7) Level2 (Производный класс).

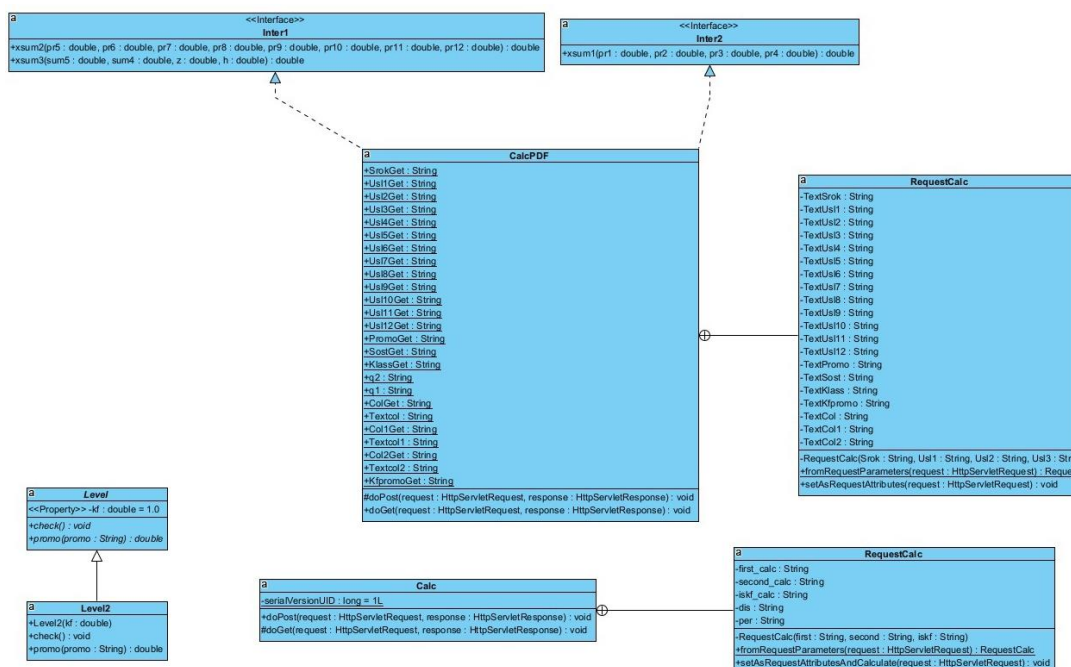


Рисунок 2 – Диаграмма классов

Диаграмма вариантов использования определяет последовательность действий, которая должна быть выполнена разрабатываемой системой при взаимодействии ее с соответствующим действующим лицом.

Описание диаграммы вариантов использования при взаимодействии пользователя с калькулятором (см. рис. 3):

Последовательность действий пользователя имеет несколько вариантов:

- 1) Пользователь авторизуется, как обычный пользователь, и использует калькулятор, в котором есть вывод результата, загрузка PDF и прослушивание музыки;
- 2) Пользователь авторизуется, как администратор, и может настраивать калькулятор (переопределять коэффициенты для расчета услуг) и менять цвет формы.

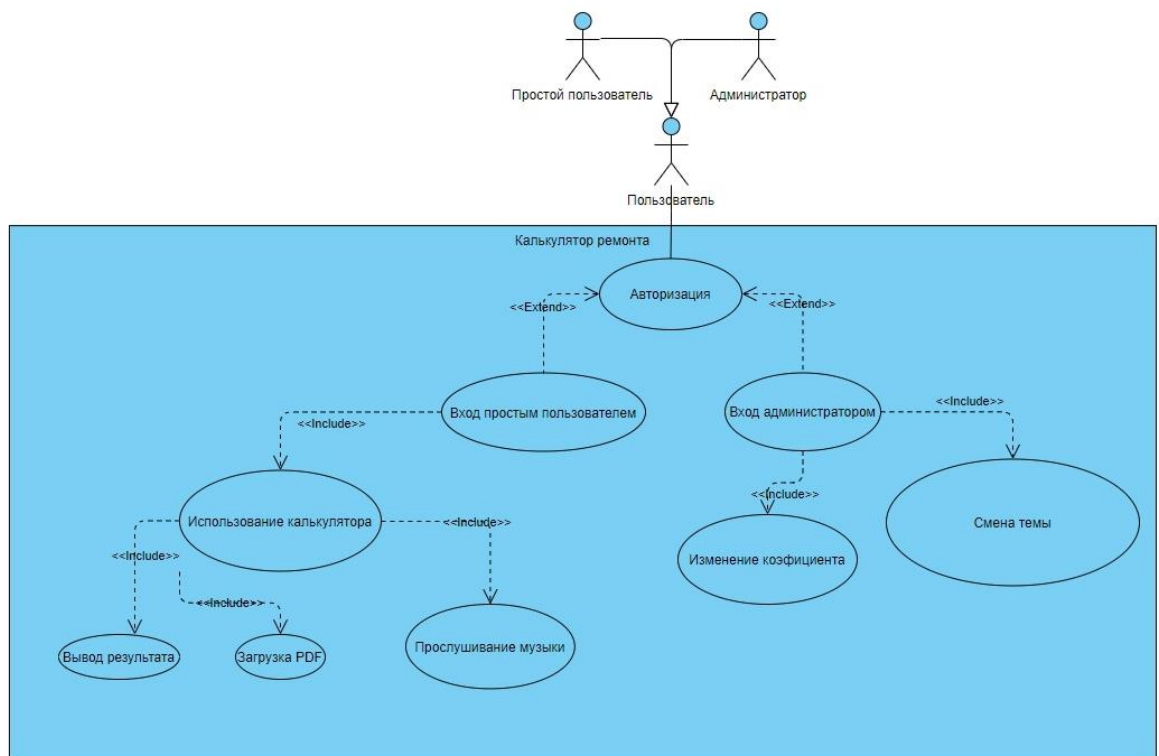


Рисунок 3 – Диаграмма вариантов использования

## 1.6 Математическая модель работы ПП

Математическая модель работы программы «Ремонтный калькулятор».

Стоимость ремонта рассчитывается по формуле (1):

$$C_{gen} = C_{com} \cdot p \cdot g - C_{com} \cdot r \quad (1)$$

Где:

$C_{gen}$  – окончательная стоимость ремонта, руб.;

$C_{com}$  – общая сумма стоимости услуг, руб.;

$p$  – коэффициент приоритетности ремонта;

$g$  – коэффициент от исходного состояния строения;

$r$  – коэффициент скидки от промокода.

Общая сумма стоимости услуг рассчитывается по формуле (2):

$$C_{com} = (\sum d_j + \sum t_i - \sum t_i \cdot z) \cdot h \quad (2)$$

Где:

$d_j$  – стоимость услуг от капитального типа ремонта;

$t_i$  – стоимость услуг от обычного типа ремонта;

$z$  – коэффициент скидки от наличия услуг от капитального типа ремонта;

$h$  – коэффициент класс ремонта.

### Пример 1.

Клиент желает сделать экономный ремонт (кф. 1) в квартире: сделать шумоизоляцию стен ( $16\text{м}^2$ , 1000 руб/м<sup>2</sup>), оклейку стен обоями ( $22\text{м}^2$ , 150 руб/м<sup>2</sup>) и покраска потолков ( $20\text{м}^2$ , 120 руб/м<sup>2</sup>). Квартира в исправном состоянии (кф.1), срок без приоритета (кф. 1), промокода нет (кф. 0).

#### Расчет

Найдем общую сумму стоимости услуг по формуле (2):

$$C_{com} = (0 + (16000 + 3300 + 2400) - 0) \cdot 1 = 21700$$

Найдем стоимость ремонта по формуле (1):

$$C_{gen} = 21700 \cdot 1 \cdot 1 - 21700 \cdot 0 = 21700$$

Итого стоимость ремонта составит 21 700 рублей.

### Пример 2.

Клиент желает сделать ремонт класса — комфорт (кф. 1,2) в квартире, ему необходимо сделать шумоизоляцию стен ( $20\text{м}^2$ , 1000 руб/м<sup>2</sup>), оклейку стен обоями ( $24\text{м}^2$ , 150 руб/м<sup>2</sup>) и шпаклевку потолков ( $18\text{м}^2$ , 400 руб/м<sup>2</sup>). Квартира в ограниченно работоспособном состоянии (1,1), срок приоритетный (кф. 1,1), промокод (кф. 0,02) есть.

#### Расчет

Найдем общую сумму стоимости услуг по формуле (2):

$$C_{com} = (7200 + 23600 - 23600 \cdot 0,02) \cdot 1,2 = 36393,6$$

Найдем стоимость ремонта по формуле (1):

$$C_{gen} = 36393,6 \cdot 1,1 \cdot 1,1 - 36393,6 \cdot 0,05 = 42216,576$$

Итого стоимость ремонта составит 42 216,576 рублей.

## **2 Техническое задание**

См. отдельный документ техническое задание.

### 3 Настройка среды разработки для операционных систем семейств Windows и Linux

Разработка любого программного продукта требует настройку среды разработки и операционной системы. Для нашего программного продукта требуется настройка среды разработки Eclipse в двух дистрибутивах семейства Linux (Ubuntu 20.04 и OpenSUSE Leap 15.2) и одного дистрибутива Windows (Windows 10).

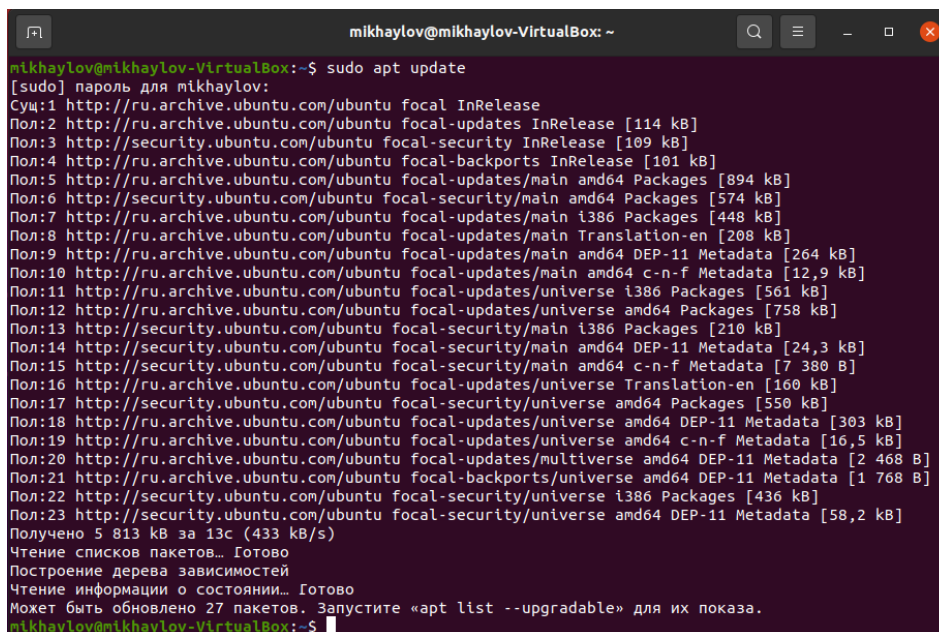
Порядок настройки среды разработки в дистрибутивах должен включать в себя:

- 1) Установка среды разработки Eclipse;
- 2) Установка Git;
- 3) Установка Maven.

#### 3.1 Настройка среды разработки для дистрибутива Ubuntu 20.04

Для настройки среды разработки в дистрибутиве Ubuntu20.04 из семейства Linux необходимо следующее:

- 1) Проверить доступные обновления. Выполняется командой `sudo apt update` в командном интерпретаторе (см. рис. 4).



```
mikhaylov@mikhaylov-VirtualBox: ~  
[sudo] пароль для mikhaylov:  
Сущ:1 http://ru.archive.ubuntu.com/ubuntu focal InRelease  
Пол:2 http://ru.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]  
Пол:3 http://security.ubuntu.com/ubuntu focal-security InRelease [109 kB]  
Пол:4 http://ru.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]  
Пол:5 http://ru.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [894 kB]  
Пол:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [574 kB]  
Пол:7 http://ru.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [448 kB]  
Пол:8 http://ru.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [208 kB]  
Пол:9 http://ru.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [264 kB]  
Пол:10 http://ru.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [12,9 kB]  
Пол:11 http://ru.archive.ubuntu.com/ubuntu focal-updates/universe i386 Packages [561 kB]  
Пол:12 http://ru.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [758 kB]  
Пол:13 http://security.ubuntu.com/ubuntu focal-security/main i386 Packages [210 kB]  
Пол:14 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [24,3 kB]  
Пол:15 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [7 380 B]  
Пол:16 http://ru.archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [160 kB]  
Пол:17 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [550 kB]  
Пол:18 http://ru.archive.ubuntu.com/ubuntu focal-updates/universe amd64 DEP-11 Metadata [303 kB]  
Пол:19 http://ru.archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [16,5 kB]  
Пол:20 http://ru.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 DEP-11 Metadata [2 468 B]  
Пол:21 http://ru.archive.ubuntu.com/ubuntu focal-backports/universe amd64 DEP-11 Metadata [1 768 B]  
Пол:22 http://security.ubuntu.com/ubuntu focal-security/universe i386 Packages [436 kB]  
Пол:23 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [58,2 kB]  
Получено 5 813 kB за 13с (433 kB/s)  
Чтение списков пакетов... Готово  
Построение дерева зависимостей  
Чтение информации о состоянии... Готово  
Может быть обновлено 27 пакетов. Запустите «apt list --upgradable» для их показа.  
mikhaylov@mikhaylov-VirtualBox:~$
```

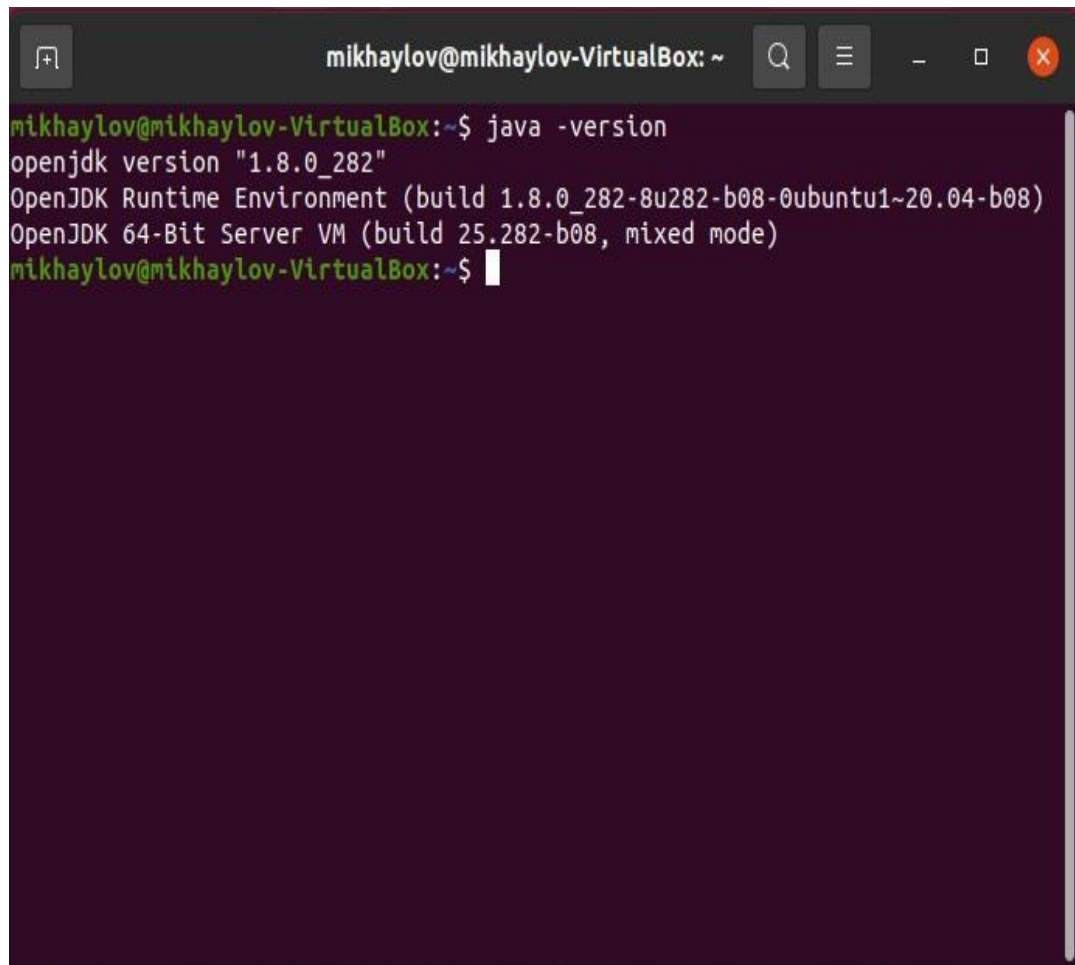
Рисунок 4 – Результат выполнения команды `sudo apt update`



2) Запустить процесс обновления. Выполняется командой `sudo apt update` в командном интерпретаторе.

3) Установить JDK (OpenJDK 8). Выполняется командой `sudo apt install openjdk-8-jdk` в командном интерпретаторе.

4) После завершения установки необходимо проверить правильность установки JDK выполнив команду `java -version`. В итоге в терминале должна быть выведена информация об установленной версии JDK (см. рис. 5).

A screenshot of a terminal window titled 'mikhaylov@mikhaylov-VirtualBox: ~'. The terminal shows the command 'java -version' being executed. The output is as follows:

```
mikhaylov@mikhaylov-VirtualBox:~$ java -version
openjdk version "1.8.0_282"
OpenJDK Runtime Environment (build 1.8.0_282-8u282-b08-0ubuntu1~20.04-b08)
OpenJDK 64-Bit Server VM (build 25.282-b08, mixed mode)
mikhaylov@mikhaylov-VirtualBox:~$
```

Рисунок 5 – Информация JDK

5) Загрузить Eclipse IDE for Enterprise Java Developers с официальной сайта <https://www.eclipse.org/> и установить.

6) После установки следует распаковать дистрибутив в домашнюю папку пользователя и выполнить файл `eclipse` из каталога `eclipse-installer`. На рисунке 6 показан загруженный дистрибутив, который находится в архиве.

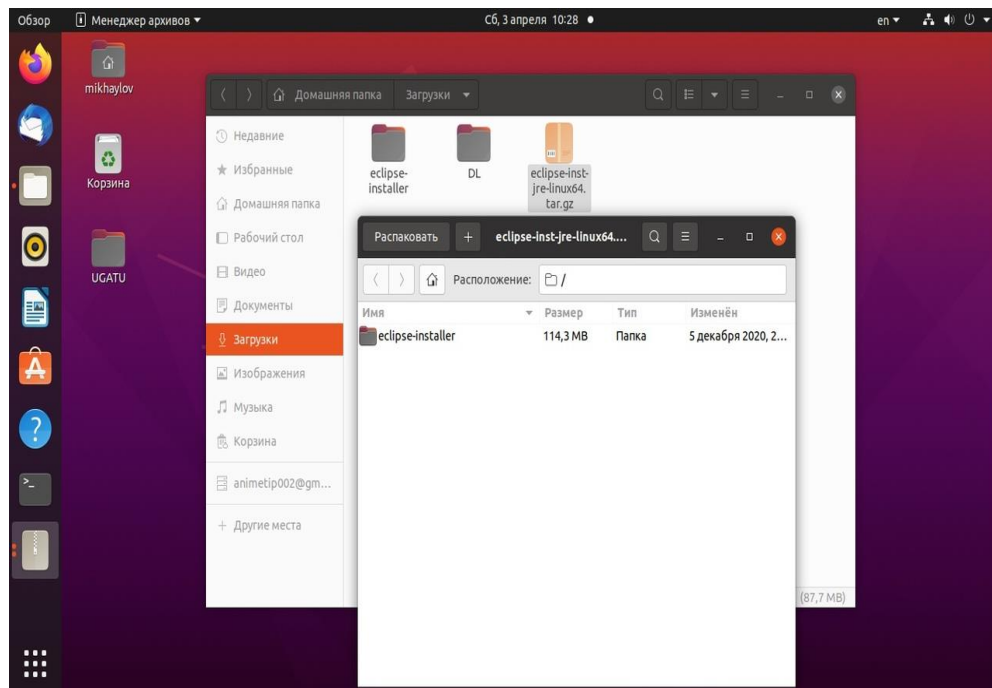


Рисунок 6 – Загруженный дистрибутив

7) Установить Git. В новых версиях Eclipse, Git уже встроен в состав среды разработки и нужно только добавить Git перспективу в панель перспектив. Для добавления Git перспективу на панель перспектив, необходимо перейти к “Window -> Perspective-> Open Perspective-> Other ...”, выбрать Git и нажать кнопку Open.

8) Установить Maven. В новых версиях Eclipse, Maven уже встроен в состав среды разработки (см. рис. 7).

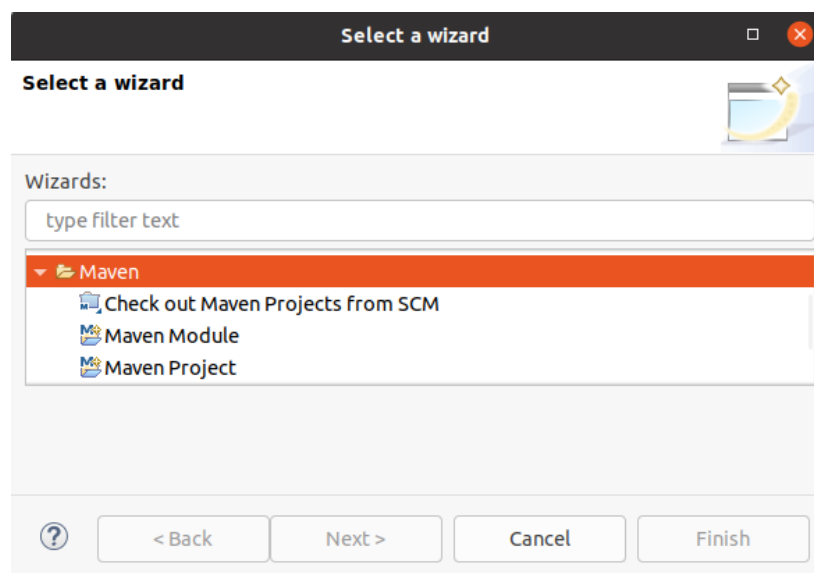


Рисунок 7 – Окно для создания проекта в Eclipse

### 3.2 Настройка среды разработки для Windows 10

Для настройки среды разработки в дистрибутиве Windows 10 из семейства Windows необходимо следующее:

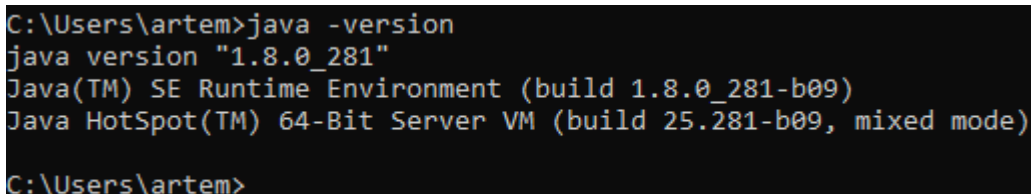
1) Скачать JDK с официального сайта Oracle <https://www.oracle.com/ru/java/technologies/javase-downloads.html>.

2) Сохранить файл на компьютер и запустить. После появления окна установщика последовательно проходим все стадии установки JDK.

3) Настроить системную переменную JAVA\_HOME, которая содержит путь к папке, где установлена Java. Это необходимо для того, чтобы другие программы могли использовать Java.

4) В системную переменную Path добавить путь к папке bin в папке с JDK. Для этого нужно изменить системную переменную Path, в конце существующей строки нужно добавить точку с запятой, после нее добавить следующее: %JAVA\_HOME%\bin.

5) Проверить правильность вышеперечисленных действий. Для этого выполнить в командном окне команду java -version (см. рис. 8).



```
C:\Users\artem>java -version
java version "1.8.0_281"
Java(TM) SE Runtime Environment (build 1.8.0_281-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.281-b09, mixed mode)

C:\Users\artem>
```

Рисунок 8 - Версия JDK

6) Загрузить Eclipse IDE for Enterprise Java Developers с официального сайта <https://www.eclipse.org/> и установить.

7) После установки следует распаковать дистрибутив Eclipse на любой локальный диск (например, диск C:) и выполнить файл eclipse из каталога eclipse. На рисунке 9 показан загруженный дистрибутив, который находится в архиве.

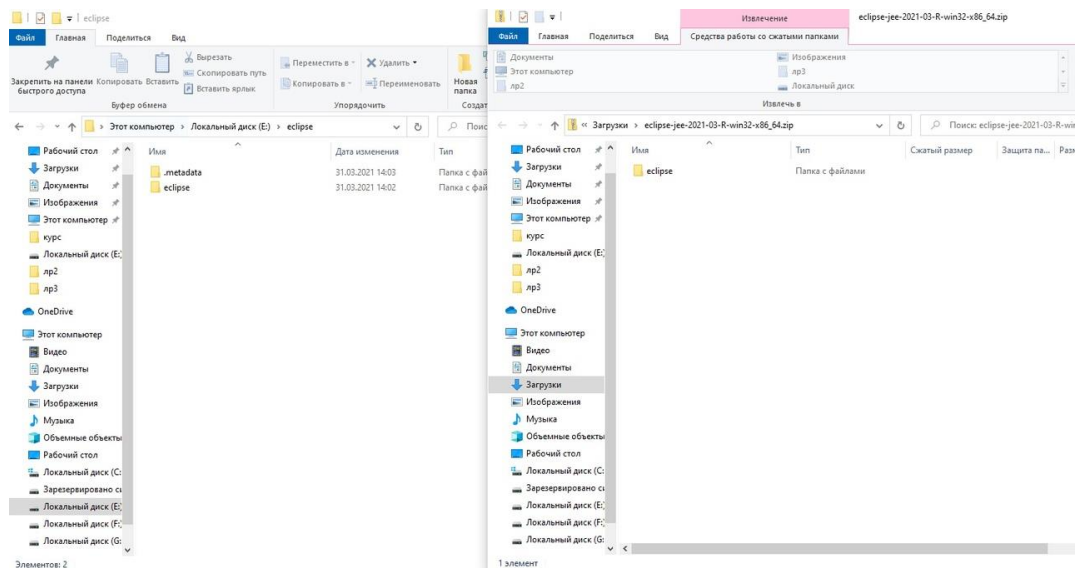


Рисунок 9 - Загруженный дистрибутив

8) Установить Git. В новых версиях Eclipse, Git уже встроен в состав среды разработки и нужно только добавить Git перспективу в панель перспектив. Для добавления Git перспективу на панель перспектив, необходимо перейти к “Window -> Perspective-> Open Perspective-> Other ...”, выбрать Git и нажать кнопку Open.

9) Установить Maven. В новых версиях Eclipse, Maven уже встроен в состав среды разработки (см. рис. 10).

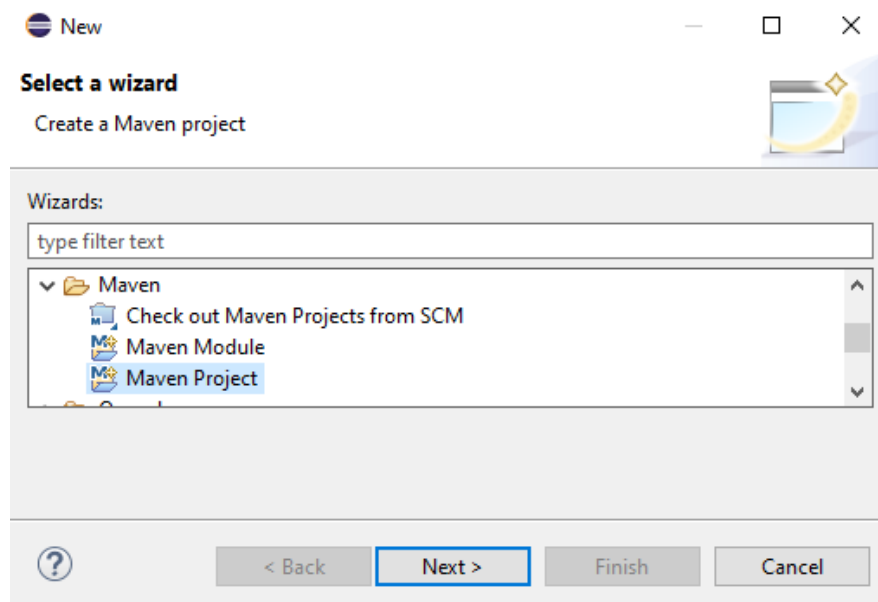


Рисунок 10 - Окно для создания проекта в Eclipse

### 3.3 Настройка среды разработки для дистрибутива OpenSuse 15.2

Для настройки среды разработки в дистрибутиве OpenSuse 15.2 из семейства Linux необходимо следующее:

1) Проверить доступные обновления. Выполняется командой `sudo zypper update` в командном интерпретаторе (см. рис. 11).

```
belousov@localhost:~> sudo zypper update
[sudo] пароль для root:
Получение метаданных репозитория "Основной репозиторий обновлений" .....[готово]
Сбор кэша репозитория "Основной репозиторий обновлений" .....[готово]
Загрузка данных о репозиториях...
Чтение установленных пакетов...

Будет обновлен 1 пакет:
  yast2-bootloader

1 пакет для обновления.
Общий размер загрузки: 104,4 KiB. Уже кэшировано: 0 B. После этой операции будет
использовано дополнительно 789,0 B.
Продолжить? [y/n/v/...? выводит все параметры] (y): y
Получение пакет yast2-bootloader-4.2.28-lp152.2.9.1.x86_64
(1/1), 104,4 KiB (237,9 KiB после распаковки)
Получение разности: ./x86_64/yast2-bootloader-4.2.27_4.2.28-lp152.2.6.1_lp152.2.
9.1.x86_64.drpm, 43,8 KiB
Получение: yast2-bootloader-4.2.27_4.2.28-lp152.2.6.1_lp152.2.9.1.x86_64[готово]
Применение разности: ./yast2-bootloader-4.2.27_4.2.28-lp152.2.6.1_lp152.2.9.1.x86_64[готово]
```

Рисунок 11 – Результат выполнения команды `sudo zypper update`

2) Запустить процесс обновления. Выполняется командой `sudo zypper update` в командном интерпретаторе.

3) Установить JDK (OpenJDK 8). Выполняется командой `sudo zypper in java-1_8_0-openjdk-devel` в командном интерпретаторе (см. рис. 12).

```
belousov@localhost.localdomain:~
(4/4), 17,1 MiB ( 21,4 MiB после распаковки)
Получение: java-1_8_0-openjdk-devel-1.8.0.282-lp152.2.9.1.x86_64[готово (4,8 MiB/s)]

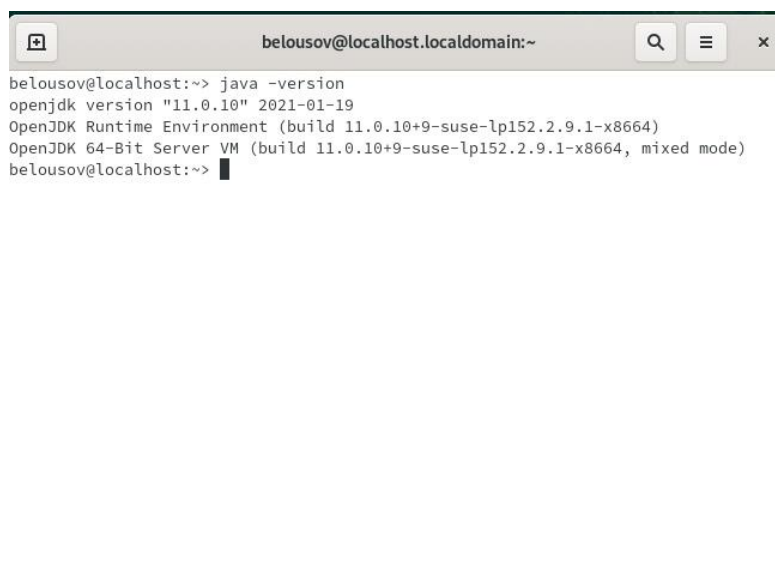
Проверка на конфликты файлов: .....[готово]
(1/4) Установка: lksctp-tools-1.0.16-lp152.3.5.x86_64 .....[готово]
(2/4) Установка: java-1_8_0-openjdk-headless-1.8.0.282-lp152.2.9.1.x86_64[готово]
Дополнительный вывод rpm:
update-alternatives: using /usr/lib64/jvm/jre-1.8.0-openjdk to provide /usr/lib6
4/jvm/jre-1.8.0 (jre_1.8.0) in auto mode

(3/4) Установка: java-1_8_0-openjdk-1.8.0.282-lp152.2.9.1.x86_64 .....[готово]
(4/4) Установка: java-1_8_0-openjdk-devel-1.8.0.282-lp152.2.9.1.x86_64 .....[готово]
Дополнительный вывод rpm:
update-alternatives: using /usr/lib64/jvm/java-1.8.0-openjdk/bin/javac to provid
e /usr/bin/javac (javac) in auto mode
update-alternatives: using /usr/lib64/jvm/java-1.8.0-openjdk to provide /usr/lib
64/jvm/java-openjdk (java_sdk_openjdk) in auto mode
update-alternatives: using /usr/lib64/jvm/java-1.8.0-openjdk to provide /usr/lib
64/jvm/java-1.8.0 (java_sdk_1.8.0) in auto mode

Выполнение скриптов %posttrans .....[готово]
belousov@localhost:~>
```

Рисунок 12 – Результат выполнения команды `sudo zypper in java-1_8_0-openjdk-devel`

4) После завершения установки необходимо проверить правильность установки JDK набрав команду `java -version`. В итоге в терминале должна быть выведена информация об установленной версии JDK (см. рис. 13).



```
belousov@localhost:~> java -version
openjdk version "11.0.10" 2021-01-19
OpenJDK Runtime Environment (build 11.0.10+9-suse-lp152.2.9.1-x86_64)
OpenJDK 64-Bit Server VM (build 11.0.10+9-suse-lp152.2.9.1-x86_64, mixed mode)
belousov@localhost:~>
```

Рисунок 13 – Информация JDK

4) Загрузить Eclipse IDE for Enterprise Java Developers с официальной сайта <https://www.eclipse.org/> и установить.

5) После установки следует распаковать дистрибутив в домашнюю папку пользователя и выполнить файл eclipse из каталога eclipse-installer. На рисунке 14 показан загруженный дистрибутив, который находится в архиве.

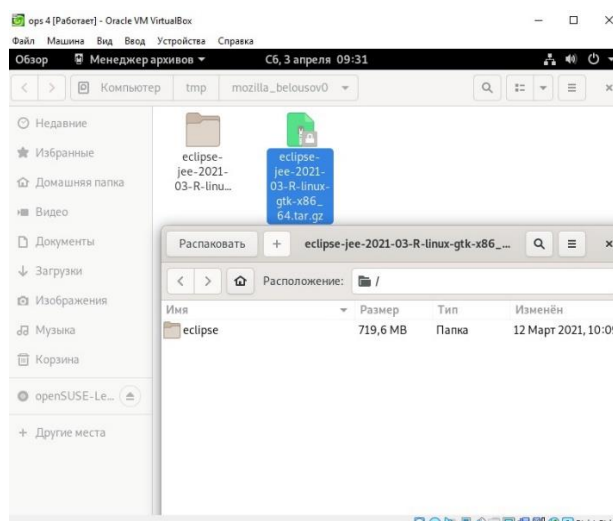


Рисунок 14 - Загруженный дистрибутив

6) Установить Git. В новых версиях Eclipse, Git уже встроен в состав среды разработки и нужно только добавить Git перспективу в панель перспектив. Для добавления Git перспективу на панель перспектив, необходимо перейти к “Window -> Perspective-> Open Perspective-> Other ...”, выбрать Git и нажать кнопку Open (см. рис. 15).

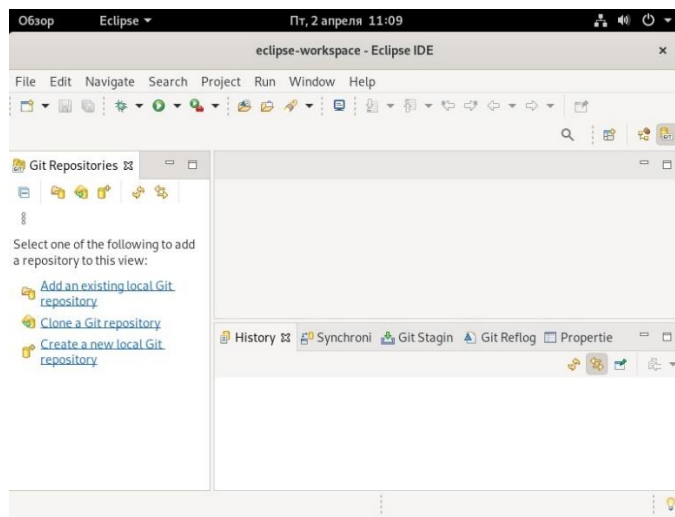


Рисунок 15 – Окно Git Repositories

7) Установить Maven. В новых версиях Eclipse, Maven уже встроен в состав среды разработки (см. рис. 16).

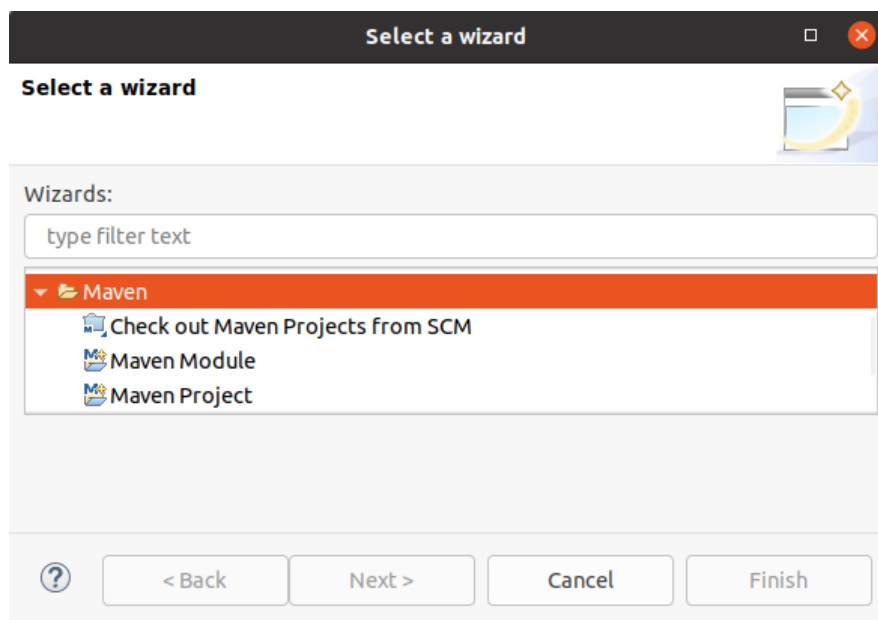


Рисунок 16 – Окно для создания проекта в Eclipse

### 3.4 Создание проекта Maven и работа с JUnit

Чтобы создать проект Maven необходимо:

- 1) Перейти к “File -> New-> Other”, развернуть каталог “Maven” и выбрать “Maven Project” (см. рис. 17).

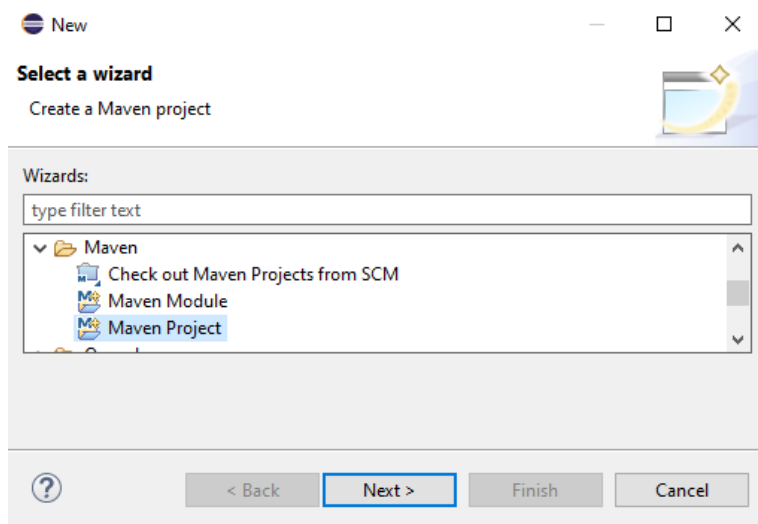


Рисунок 17 - Окно для создания проекта в Eclipse

- 2) Для пропуска выбора архетипа (см. рис. 18) выбирать “Create a simple project”, в случае если нужно выбирать архетип, то галочка не ставится.

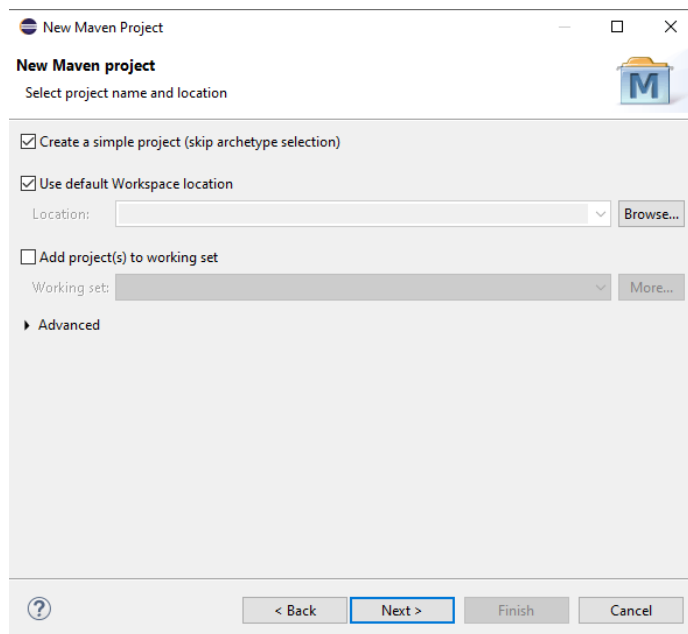


Рисунок 18 – Создание Maven проект в Eclipse



Для работы с JUnit необходимо добавить зависимость JUnit в Maven проект (см. рис. 19).

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.2</version>
</dependency>
```

Рисунок 19 – Зависимость JUnit 3

## 4 Настройка среды разработки для подключения к системе контроля версий

Среда разработки Eclipse универсальна для операционных систем семейства Windows и Linux. Поэтому порядок действий для настройки среды разработки Eclipse для работы с GitHub не зависит от операционной системы.

Ссылка на репозиторий GitHub:

<https://github.com/MikhaylovVladislav/Team7CW.git>

### 4.1 Настройка среды разработки Eclipse для работы с GitHub

Для настройки среды разработки для работы с GitHub необходимо следующее:

- 1) Установить Git. В новых версиях Eclipse, Git уже встроен в состав среды разработки;
- 2) Добавить Git перспективу в панель перспектив. Для добавления Git перспективу на панель перспектив, необходимо перейти к “Window -> Perspective-> Open Perspective-> Other ...”, выбрать Git и нажать кнопку Open (см. рис. 20).

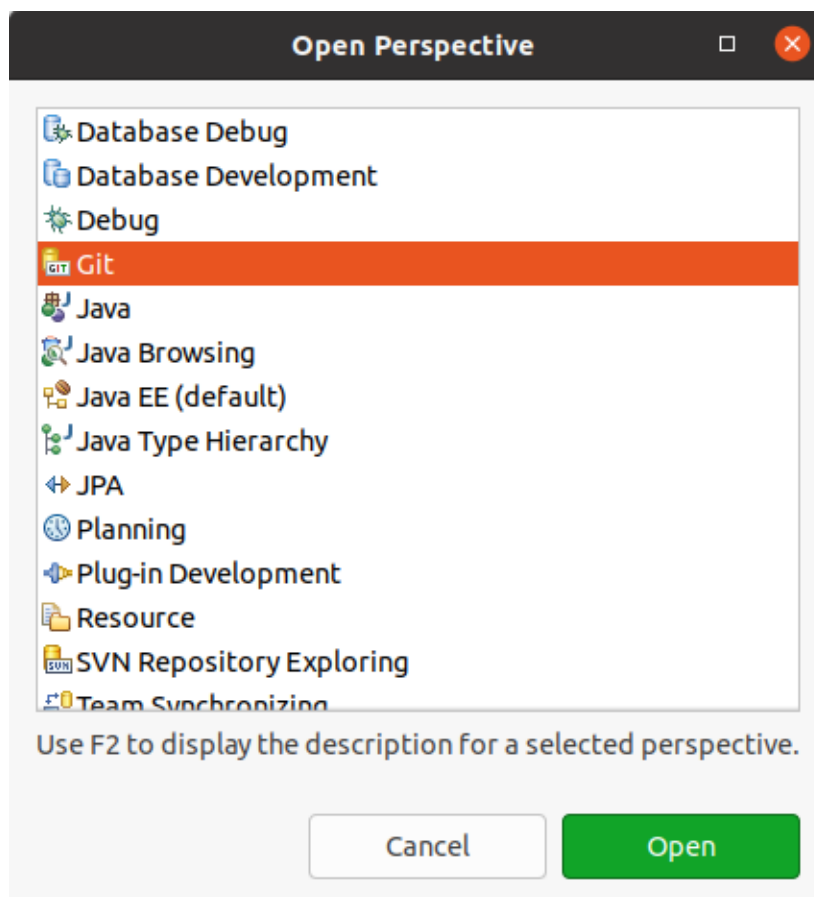


Рисунок 20 – Список перспектив

## 4.2 Клонирование репозитория с GitHub в Eclipse

Для клонирования репозитория с GitHub необходимо следующее:

- 1) Открыть в Eclipse перспективу Git (см. рис. 21);

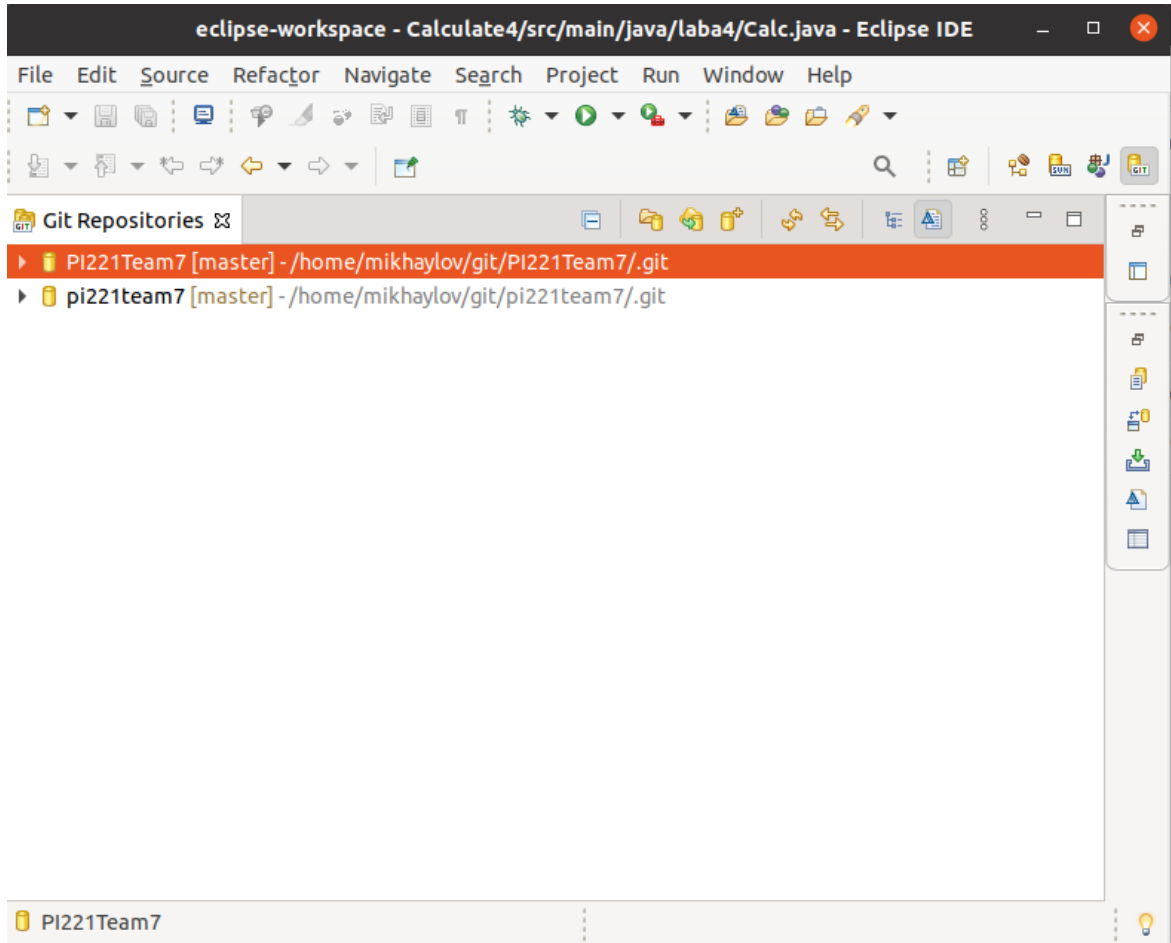


Рисунок 21 – Перспектива Git

- 2) Вызвать мастер установки с помощью “File > Import... > Git > Projects from Git > Next > Clone URI > Next” или с помощью кнопки “Clone a Git Repository ...” на панели инструментов в перспективе Git.

- 3) На 1 странице мастера установки ввести расположение удаленного репозитория и, если репозиторий имеет закрытый доступ, ввести имя пользователя и пароль с аккаунта GitHub (см. рис. 22)

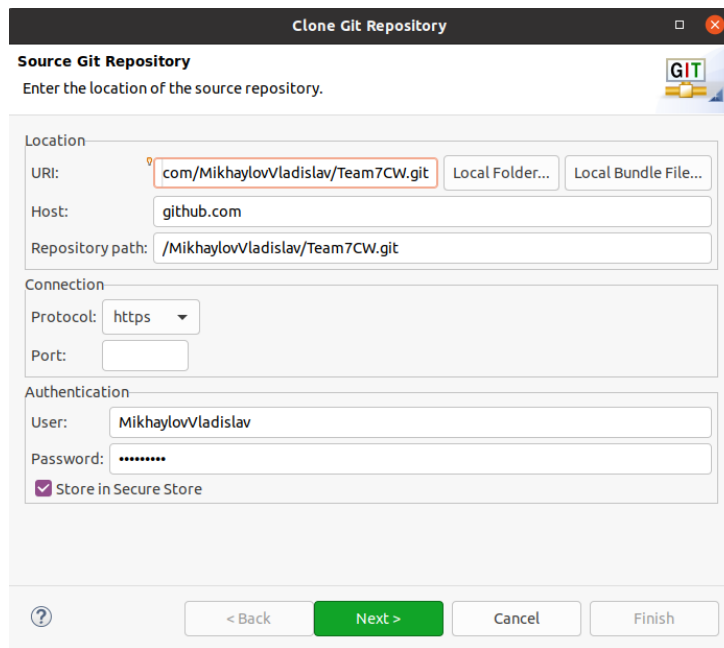


Рисунок 22 – Окно клонирования репозитория Git

4) Выбрать ветку, которую нужно клонировать из удаленного репозитория (см. рис. 23).

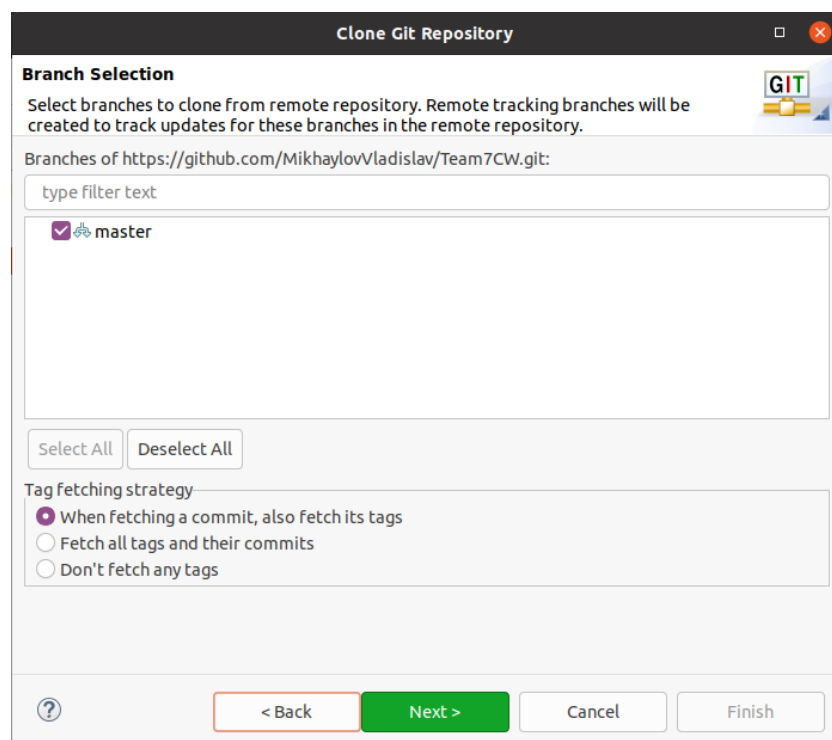


Рисунок 23 - Окно клонирования репозитория Git

5) Определить место хранения репозитория в локальной файловой системе (см. рис. 24).

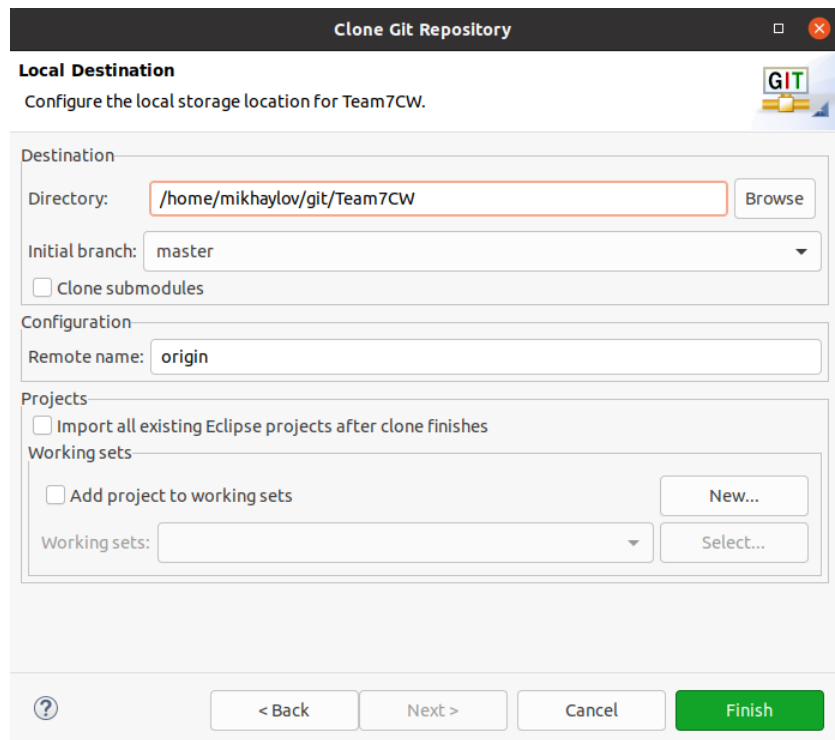


Рисунок 24 - Окно клонирования репозитория Git

6) Для импорта проекта (получение рабочей копии) необходимо далее перейти к «Working Tree», вызвать контекстное меню и выбрать «Import projects...» (см. рис. 25)

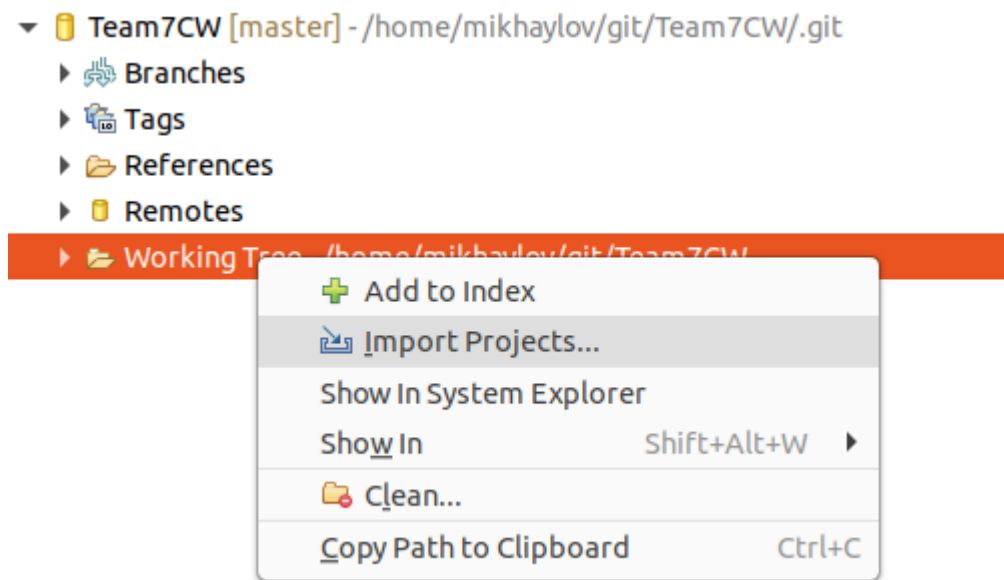


Рисунок 25 – Импорт проекта

7) Для работы с рабочей копией вызывается контекстное меню проекта и далее выбирается вкладка Team (см. рис. 26)

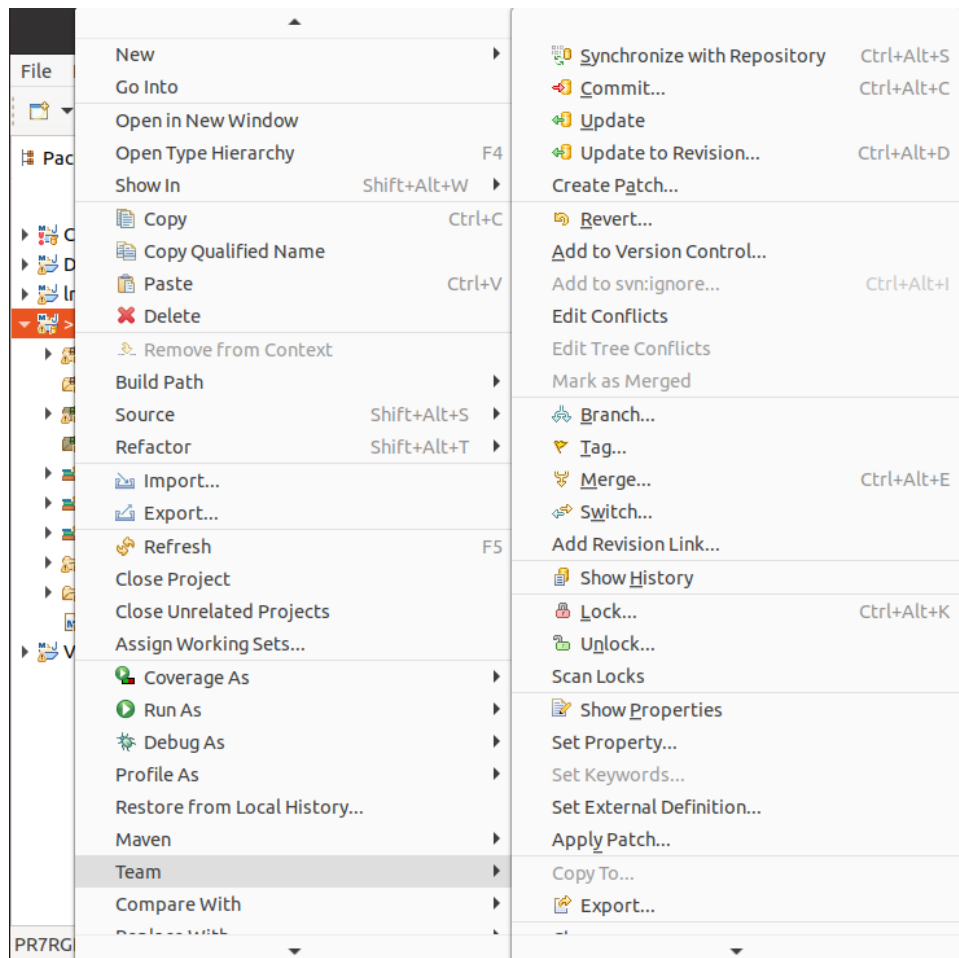


Рисунок 26 – Вкладка Team

## 5 Реализация исходного кода по зонам ответственности

Разрабатываемый калькулятор ремонта реализован в виде веб-приложения. Данный программный продукт реализует авторизацию, которая имеет роль администратора и обычного пользователя, выгрузку PDF файла, в котором хранятся данные о стоимости ремонта, загрузку конфигурационного файла, доступную только администратору, для веб-приложения и расчет необходимых услуг.

В таблице 2 приведено описание зон ответственности для разработчиков и модераторов.

Таблица 2 – Зона ответственности

№	ФИО разработчика/модератора	Зона ответственности
1	Белоусов Артем Николаевич	Интерфейсы (Inter1, Inter2), авторизация (Calc).
2	Боголюбов Максим Витальевич	Переопределение методов и производные классы (Level2), панель администратора (CalcPDF, FaceCalc).
3	Михайлов Владислав Александрович	Интерфейс веб-приложения (Auth.jsp, FaceCalc.jsp, Results.jsp).
4	Тахаев Арсений Гуннарович	Абстрактный класс (Level) и методы, формирование PDF файла (Results) и каскадная таблицы стилей (style, style1, style1).

Текст программы см. в отдельном документе “текст программы”.

Ссылка на репозиторий GitHub:

<https://github.com/MikhaylovVladislav/Team7CW.git>

В таблице 3 приведены пароли и логины для 4 учетных записей.

Таблица 3 – Учетные записи

Логин	Пароль	Роль
1234	4321	Администратор
1	1	Пользователь
2	2	Пользователь
3	3	Пользователь

## 6 Сборка и тестирование программного продукта

В таблице 4 приведено описание Unit-тестов и ответственных за их реализацию.

Таблица 4 – Описание Unit-тестов

№	ФИО разработчика/модератора	Описание Unit-тестов	№ приложения
1	Белоусов Артем Николаевич	Проверка на Null метода xsum2 из интерфейса Inter1	Приложение А
2	Боголюбов Максим Витальевич	Проверка заданного коэффициента в классе Level2	Приложение Б
3	Михайлов Владислав Александрович	Проверка на Null метода xsum5 из интерфейса Inter2	Приложение В
4	Тахаев Арсений Гуннарович	Проверка на Null метода xsum3из интерфейса Inter1	Приложение Г

Maven-проекты определяются как XML-файлы с названием pom.xml. Помимо всего прочего, этот файл определяет имя проекта, версию, а также зависимости от сторонних библиотек.

pom.xml из проекта по курсовой работе показан в приложении Д. Он включает следующие детали конфигурации проекта:

- modelVersion - версия POM-модели;
- groupId - группа или организация, к которой принадлежит проект. Чаще всего выражается в виде перевернутого наоборот доменного имени;
- artifactId - имя, которое будет передано библиотеке экземпляра(artifact) проекта (к примеру, имя его JAR или WAR файла);
- version - версия, с которой будет собран проект;
- packaging - как проект должен быть упакован. По умолчанию, с "jar" упаковывается в JAR-файл, "war" - WAR-файл;
- name – название проекта;
- description – описание проекта;
- url – сайт проекта;
- dependencies – зависимости проекта;



- `pluginManagement/` определяет настройки для плагинов, которые будут наследоваться модулями в вашей сборке. Это отлично подходит для случаев, когда у вас есть родительский файл `pom`;
- `plugins` это фактический вызов плагина. Это может или не может быть унаследовано от `<pluginManagement/>`;
- `build` - содержит информацию по самой сборке: где находятся исходные файлы, где находятся ресурсы, какие плагины используются.

Как видно на рисунке 27 проект `Calculate4` включает в себя каталоги `artifacts`, `src/main/java`, `src/main/webapp`, `src/test/java` и `target`, а также файлы `README.md`, `procfile` и `pom.xml`. В свою очередь, каталог `webapp` включает в себя подкаталоги `fonts`, `lib` и `WEB-INF`, а также файлы `Check.pdf`, `Auth.jsp`, `FaceCalc.jsp`, `Form.jsp` и `Results.jsp`, `web.xml`. Назначение этих каталогов и файлов:

- `artifacts` – это сборка активов проекта, которые был собран для тестирования, развертывания или распространения программного решения или его части;
- `src/main/java` – это корневой каталог для вашего исходного кода, который используется для реального производства;
- `src/test/java` - это корневой каталог для вашего кода, который не используется для реального производства;
- `src/main/webapp` – тут располагается всё, что связано с веб-контентом. `jsp`-страницы, `web.xml`, тайлсы, картинки, `css`-файлы, `jQuery`-скрипты и так далее.
- `target` – хранит готовые артефакты;
- `WEB-INF` - содержит все вещи, связанные с приложением, которые не находятся в корневом каталоге документа приложения;
- `fonts` – содержит шрифты, необходимые для приложения;
- `lib` - содержит все файлы `JAR`, необходимые для приложения;
- `Check.pdf` – хранит информацию о результате калькулятора ремонта;
- `file.jsp` - это не стандартная `html`-страница;
- `web.xml` - хранит информацию о конфигурации приложения;
- `README.md` - (справка) файл в котором описываются те или иные аспекты приложения;
- `procfile` - механизм для объявления команд, которые запускают динамо приложения на платформе `Heroku`.

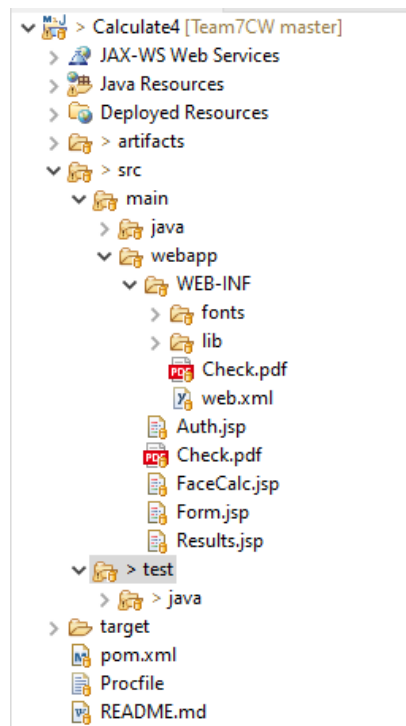


Рисунок 27 – Структура проекта

Чтобы выполнить сборку проекта необходимо:

- 1) Вызвать контекстное меню нажатием ПКМ по иконке проекта
- 2) Раскрыть пункт Run As (см. рис. 28)

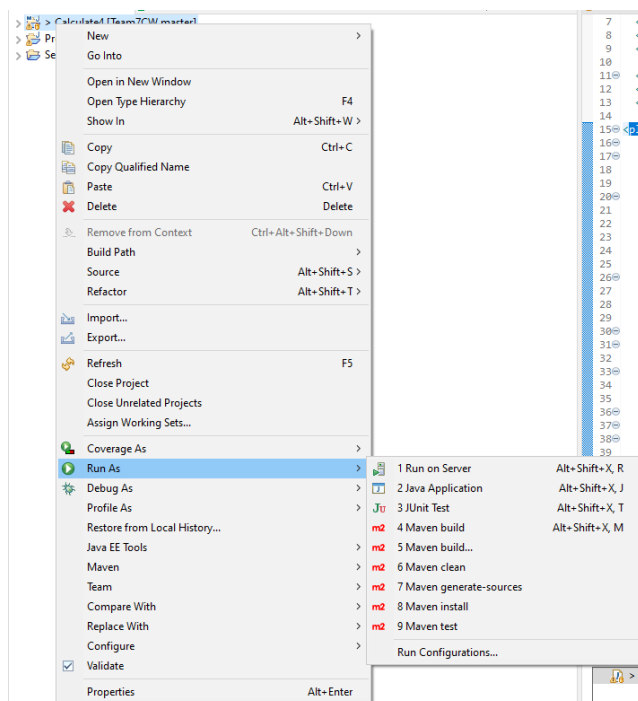


Рисунок 28 – Контекстное меню

3) Выбрать Maven build

4) Ввести в поле Goals: package (см. рис. 29)

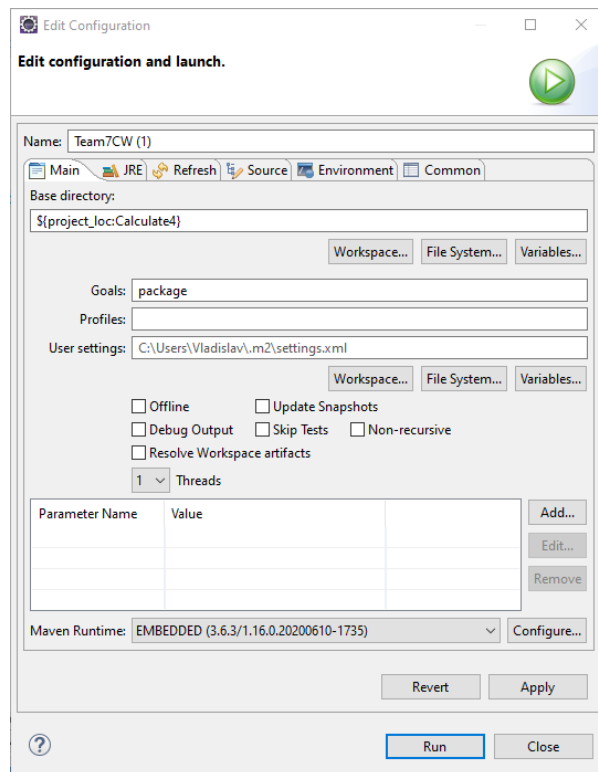


Рисунок 29 – Окно конфигурации сборки проекта

5) Нажать кнопку Run

6) Убедиться, что сборка прошла успешно, просмотрев консоль (см. рис. 30).

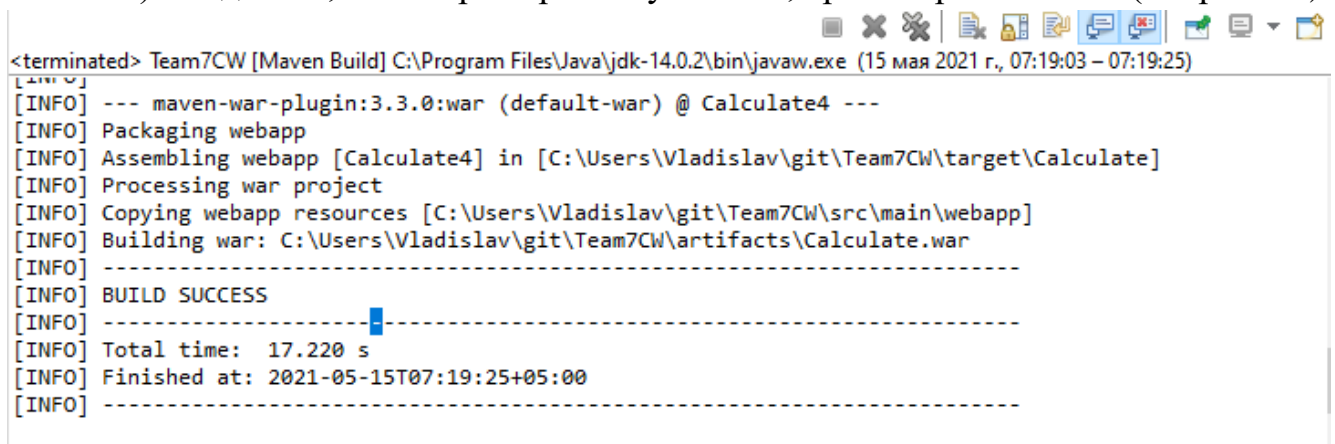


Рисунок 30 – Сообщение об успешной сборке в консоли

## 7 Настройка программной среды для развертывания и запуска программного продукта

Непрерывная интеграция — это практика регулярной интеграции изменений кода в главную ветку репозитория и проведения тестирования в отношении изменений.

Для сборки и тестирования ремонтного калькулятора используется распределенный веб сервис Travis ci.

Для работы с непрерывной сборкой и тестированием, необходимо настроить travis ci:

- 1) Добавить репозиторий с GitHub в travis ci;
- 2) Создать конфигурационный файл .travis.yml.

Файл .travis.yml описывает процесс сборки. Сборка в Travis CI - это последовательность этапов. Каждый этап состоит из параллельно выполняемых заданий. Конфигурационный файл .travis.yml находится в корне проекта и имеет содержание показанное на рисунке 31. Назначение каждого этапа:

- 1) before\_script - выполнение команд перед этапом запуска сборки;
- 2) script – запуск скрипта сборки;
- 3) after\_script - выполнение команд после этапа запуска сборки;
- 4) after\_success - выполнение команд, когда сборка завершается успешно;
- 5) after\_failure - выполнение команд, когда сборка завершается неудачно;
- 6) deploy – развертывание.

```
1 language: java
2 before_script:
3   - echo "Starting build"
4 script:
5   - mvn clean package
6 after_script:
7   - echo "Script finished"
8 after_success:
9   - echo "Build ready"
10  - ls -l $TRAVIS_BUILD_DIR/target
11 after_failure:
12   - echo "Build failure"
13 deploy:
14   skip_cleanup: true
```

Рисунок 31 – Конфигурационный файл .travis.yml

В курсовой работе подготовленная среда является либо платформа Heroku. Чтобы развернуть на Heroku, необходимо создать приложение и внутри этого приложения в разделе “deploy” подключить репозиторий GitHub, а для автоматического развертывания необходимо поставить галочку в поле “Wait for CI to pass before deploy” и нажать на кнопку Enable Automatic Deploys (см. рис. 32).

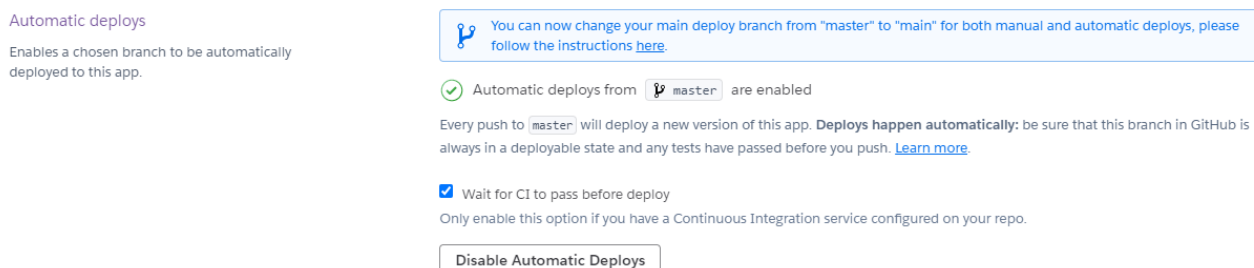


Рисунок 32 – Раздел deploy

Разрабатываемый программный продукт является кроссплатформенным. Веб-приложения работает на операционной системе Linux(дистрибутив Ubuntu и OpenSuse и Knoppix) (см. рис. 33) и Windows(Windows 10)(см. рис. 34 ), а также поддерживается в браузерах (Google Chrome, Яндекс и Mazilla).

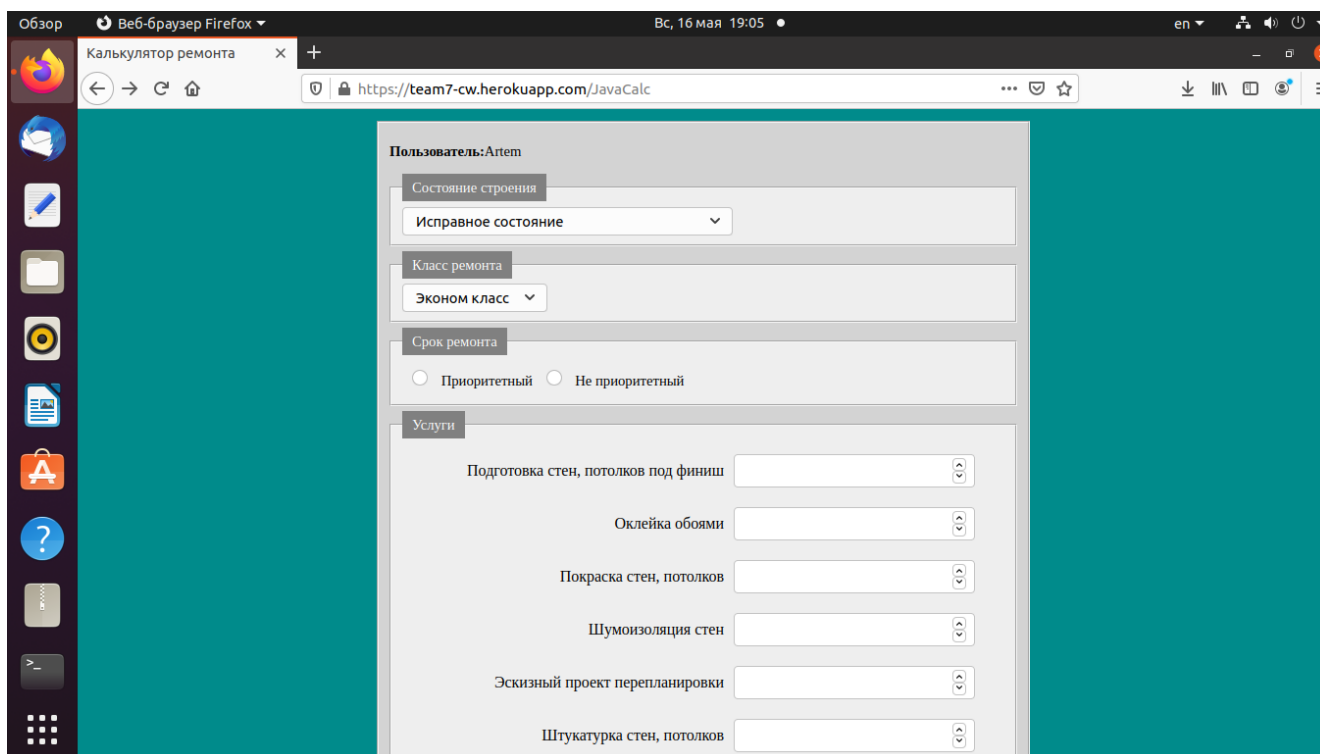


Рисунок 33 – Ubuntu 20.04 (Mazilla)

Калькулятор ремонта

team7-cw.herokuapp.com/javaCalc

Пользователь: Arsenii

Состояние строения  
Исправное состояние

Класс ремонта  
Эконом класс

Срок ремонта  
☐ Приоритетный ☐ Не приоритетный

Услуги

Подготовка стен, потолков под финиш	
Оклеивка обоев	
Покраска стен, потолков	
Шумоизоляция стен	
Эскизный проект перепланировки	
Штукатурка стен, потолков	
Очистка поверхностей от старых покрытий	
Шпатлевка стен, потолков	
Нанесение декоративных покрытий	
Облицовка плиткой стен	
Укладка ламината	
Оклеивка панелями потолков	

Промокли

Введите промокли

Рисунок 34 – Windows 10 (Google Chrome)

## **8 Руководство пользователя программного продукта**

См. отдельный документ “руководство пользователя(оператора)”.

## **Заключение**

В данной курсовой работе была рассмотрена предметная область и создано веб-приложение позволяющее рассчитать стоимость выполнения строительных работ по введенным параметрам. Была рассмотрена математическая модель программного обеспечения, а также проведены пробные расчеты



## Приложение А

```
package labtest;

import static org.junit.Assert.*;
import org.junit.Assert;
import org.junit.Test;
import laba4.Inter1;

public class BelousovTest implements Inter1 {
    double expected = xsum2(0,0,0,0,0,0,0,0);
    @Test
    public void testToDouble() throws Exception {

        Assert.assertNotNull(expected);
    }
}
```

## Приложение Б

```
package labtest;

import static org.junit.Assert.*;
import org.junit.Assert;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import laba4.Level2;

public class BogolyubovTest{

    Level2 tes = new Level2(2);
    double k = tes.getKf();

    @Test
    public void testToDouble() {
        Assert.assertEquals(k,3,1);
    }
}
```

## Приложение В

```
package labtest;

import static org.junit.Assert.*;
import org.junit.Test;
import laba4.Inter2;

public class TestCalc implements Inter2{

    double expected = xsum1(0,0,0,0);

    @Test
    public void testToDouble() throws Exception {

        assertNotNull(expected);

    }
}
```

## Приложение Г

```
package labtest;

import static org.junit.Assert.*;
import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;
import laba4.Inter1;

public class TakhaevTest implements Inter1{

    double exp = xsum3(0,0);

    @Test
    public void testTakhaev() throws Exception {
        Assert.assertNotNull(exp);

    }
}
```

## Приложение Д

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>Calculate</groupId>
  <artifactId>Calculate4</artifactId>
  <version>1</version>
  <packaging>war</packaging>
  <name>Laba4</name>
  <description>Test for laba4</description>
  <url>http://maven.apache.org</url>

  <build>
    <finalName>Calculate</finalName>
    <defaultGoal>package</defaultGoal>

    <pluginManagement>
      <plugins>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.3</version>
          <configuration>
            <source>1.8</source>
            <target>1.8</target>
          </configuration>
        </plugin>

        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-dependency-plugin</artifactId>
          <version>3.1.2</version>
          <executions>
            <execution>
              <phase>package</phase>
              <goals>
                <goal>copy</goal>
              </goals>
              <configuration>
                <artifactItems>
                  <artifactItem>
                    <groupId>com.heroku</groupId>
                    <artifactId>webapp-runner</artifactId>
                    <version>9.0.31.0</version>
                    <destFileName>webapp-runner.jar</destFileName>
                  </artifactItem>
                </artifactItems>
                <outputDirectory>artifacts</outputDirectory>
              </configuration>
            </execution>
          </executions>
        </plugin>

        <plugin>
```

## Продолжение приложения Д

```
<artifactId>maven-war-plugin</artifactId>
  <version>3.3.0</version>
  <configuration>

  <goal>war:inplace</goal>
    <outputDirectory>artifacts</outputDirectory>
  </configuration>
</plugin>

  </plugins>
</pluginManagement>
</build>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
    </dependency>
    <dependency>
      <groupId>javax.el</groupId>
      <artifactId>el-api</artifactId>
      <version>2.2</version>
    </dependency>

    <dependency>
      <groupId>com.github.jsimone</groupId>
      <artifactId>webapp-runner</artifactId>
      <version>9.0.27.1</version>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>com.itextpdf</groupId>
      <artifactId>itextpdf</artifactId>
      <version>5.5.13</version>
    </dependency>

    <dependency>
      <groupId>commons-fileupload</groupId>
      <artifactId>commons-fileupload</artifactId>
      <version>1.4</version>
    </dependency>

  </dependencies>
</project>
```

## Список литературы

1. Электронный фонд правовой и нормативно-технической документации, 2020. URL: <http://docs.cntd.ru/> (дата обращения: 21.05.2021).
2. КонсультантПлюс, 2021. URL: <http://www.consultant.ru/> (дата обращения: 21.05.2021).
3. Tproger, 2021. URL <https://tproger.ru/translations/building-a-web-app-with-java-servlets/> (дата обращения: 21.05.2021).
4. betacode, 2021. URL <https://betacode.net/10169/java-servlet/> (дата обращения: 21.05.2021).
5. Строительный портал Fixmaster, 2021. URL: <https://fixmaster74.ru/remont/vidy-remontno-otdelochnyh-rabot-urok-tehnologii.html> (дата обращения: 21.05.2021).