

*В период лабораторной подготовки вам необходимо проделать все действия, приведенные в примере. Итоговые результаты необходимо представить в формате отчетной презентации PowerPoint, с визуализацией каждого этапа выполненной работы.*

## Оглавление

Этап 1.....	3
Установка JDK.....	3
Установка Eclipse.....	6
Установка Apache NetBeans .....	12
Этап 2.....	15
Запуск Eclipse и создание проекта .....	15
Создание класса .....	17
Вывод надписи в консоль .....	19
Вывод слов в столбец.....	20
Вывод возраста в одной строке .....	21
Этап 3.....	22
Создание проекта в NetBeans .....	22
Создание формы .....	24
Создание элементов формы.....	26
Этап 4.....	33
Создание мини-калькулятора.....	33
Этап 5.....	36
Установка символов табуляции в качестве отступов .....	36
Переименование элементов проекта .....	39
Создание нового пакета .....	42
Перемещение класса из одного пакета в другой .....	43
Экспорт проекта в архив zip .....	45
Удаление проекта .....	47
Импорт проекта из архива zip .....	48
Этап 6.....	51
Часть 1. Создание овала.....	51
Часть 2. Работа с изображениями .....	55
Часть 3. Анимация.....	60
Готовая программа .....	62
Готовый код программы .....	63
Кейс-задание.....	65
GitHub .....	66

## Этап 1.

### Установка JDK

Для того, чтобы установить необходимое программное обеспечение, необходимо скачать, разархивировать необходимые файлы, показанные на рисунке 1, и запустить установку.

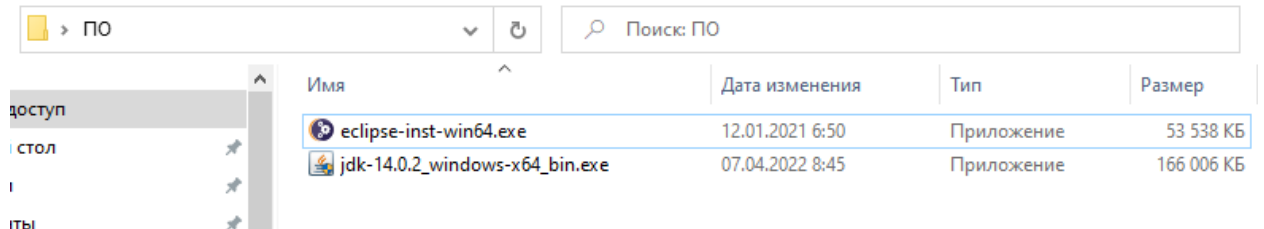


Рисунок. 1. Расположение установщиков

Первым необходимо установить Java Development Kit. Двойным кликом по установщику, после появится диалоговое окно с приветствием, как показано на рисунке 2. После чего необходимо нажать на Next.

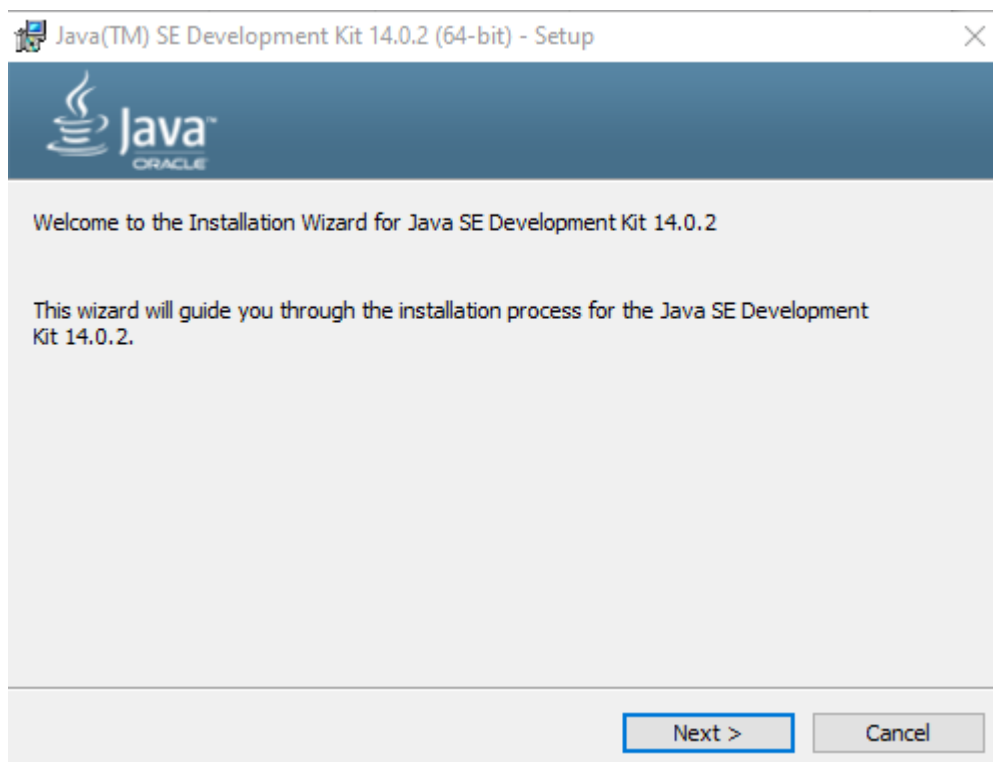


Рисунок. 2. Приветствие

В следующем окне, показанном на рисунке 3, необходимо выбрать путь для загрузки программ. В данном случае нужно не вносить изменения и нажать кнопку Next.

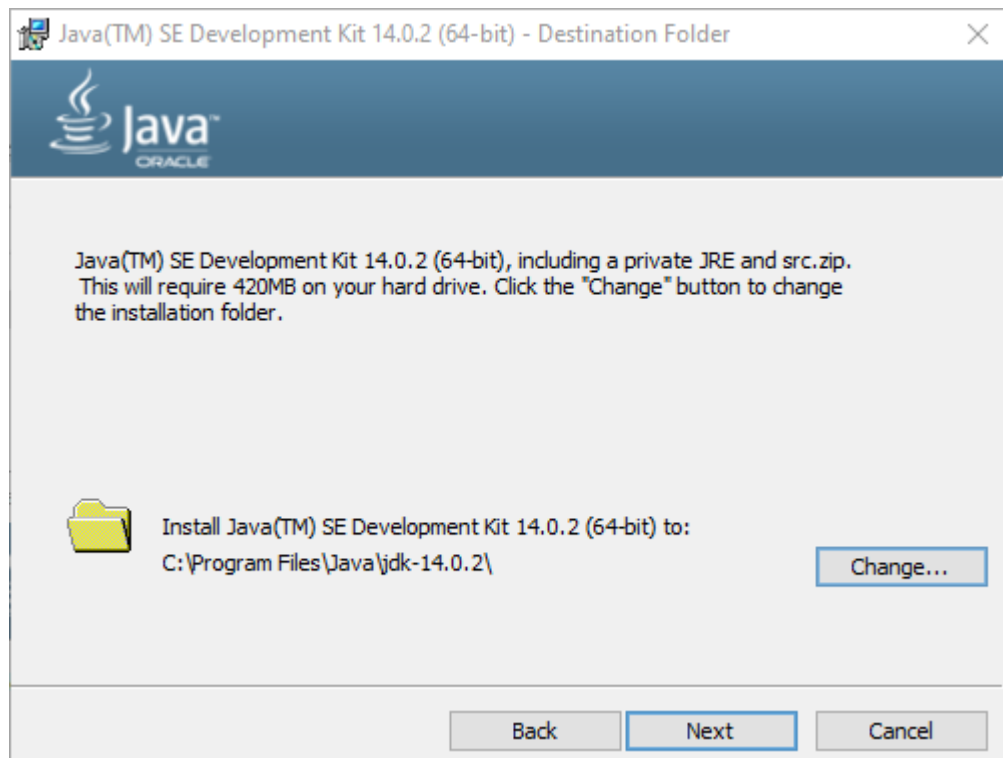


Рисунок. 3. Путь установки

После чего начнется установка программы, как показано на рисунке 4.

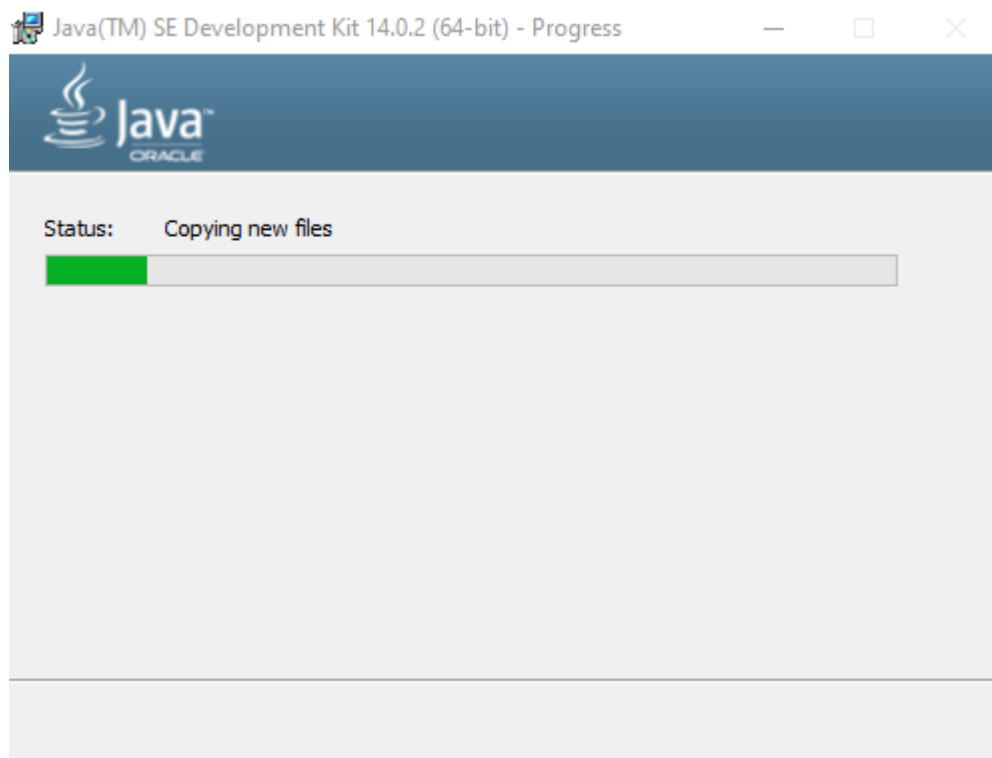


Рисунок. 4. Установка программы

После этого появится первое с сообщением об успешной установке программы, как показано на рисунке 5.

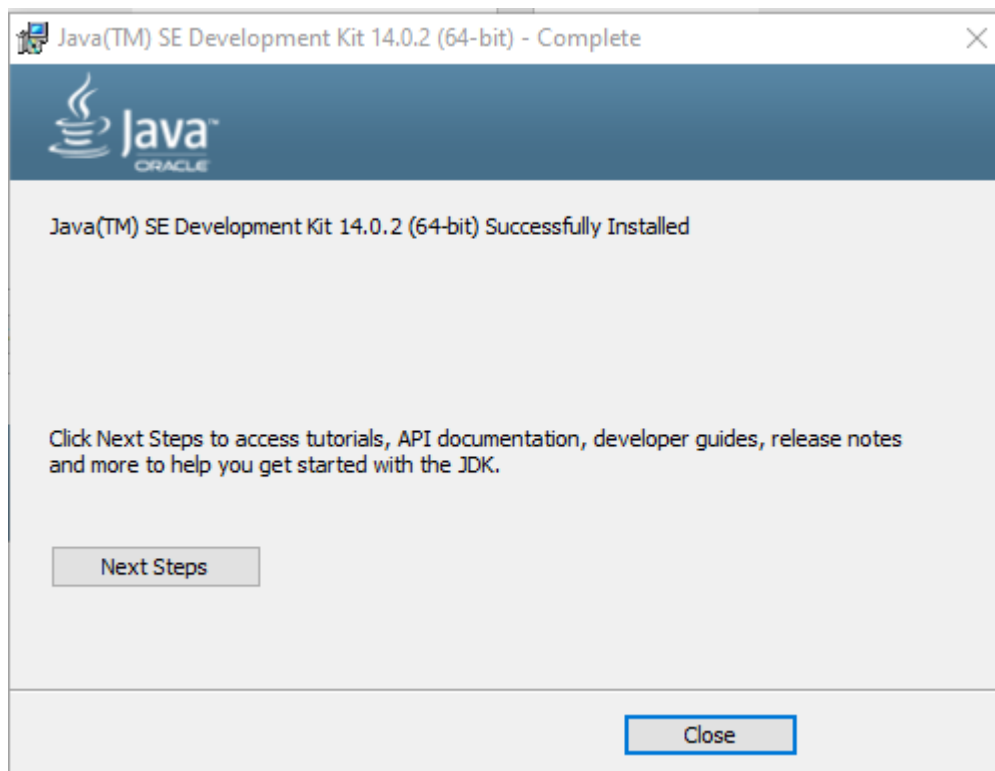


Рисунок. 5. Завершение установки

При необходимости можно ознакомиться с документацией, которая откроется после нажатия на кнопку Next Steps, как показано на рисунке 6.

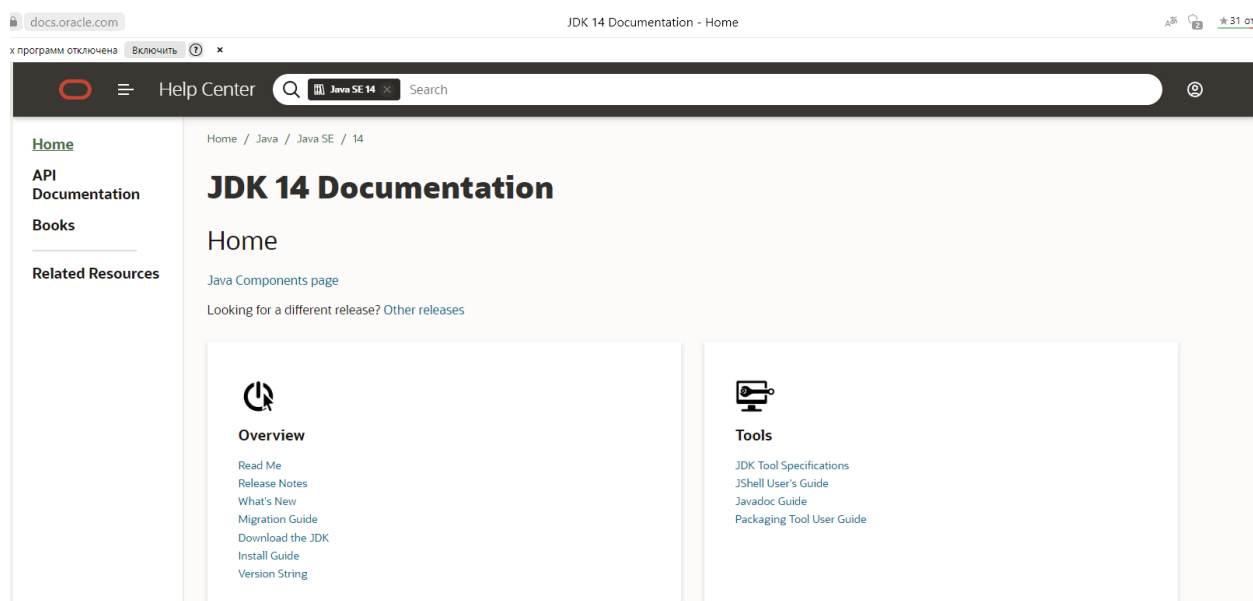


Рисунок. 6. Документация

## Установка Eclipse

Необходимо два раза кликнуть на файл-установщик, после чего появится следующее изображение, показанное на рисунке 7.



Рисунок. 7. Запуск установщика

Далее требуется выбрать нужный тип среды разработчика. Необходимо выбрать самый первый, как показано на рисунке 8.

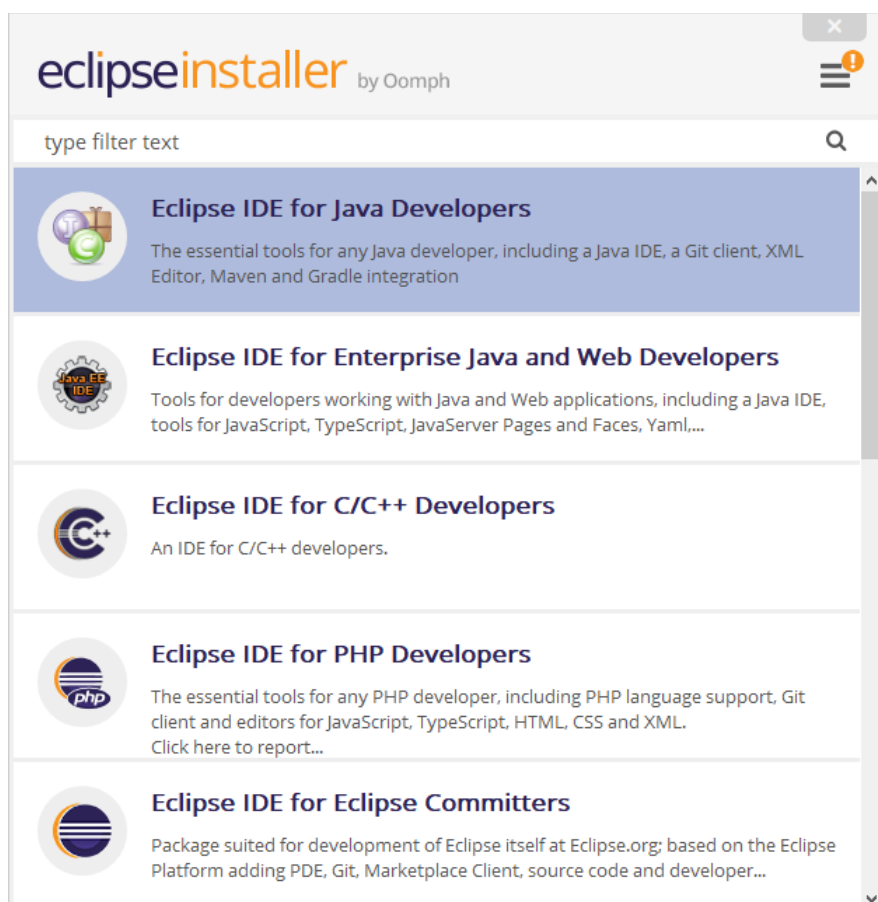


Рисунок. 8. Выбор ПО

Затем предлагается выбрать нужные опции, после чего нужно нажать на Install, как показано на рисунке 9.

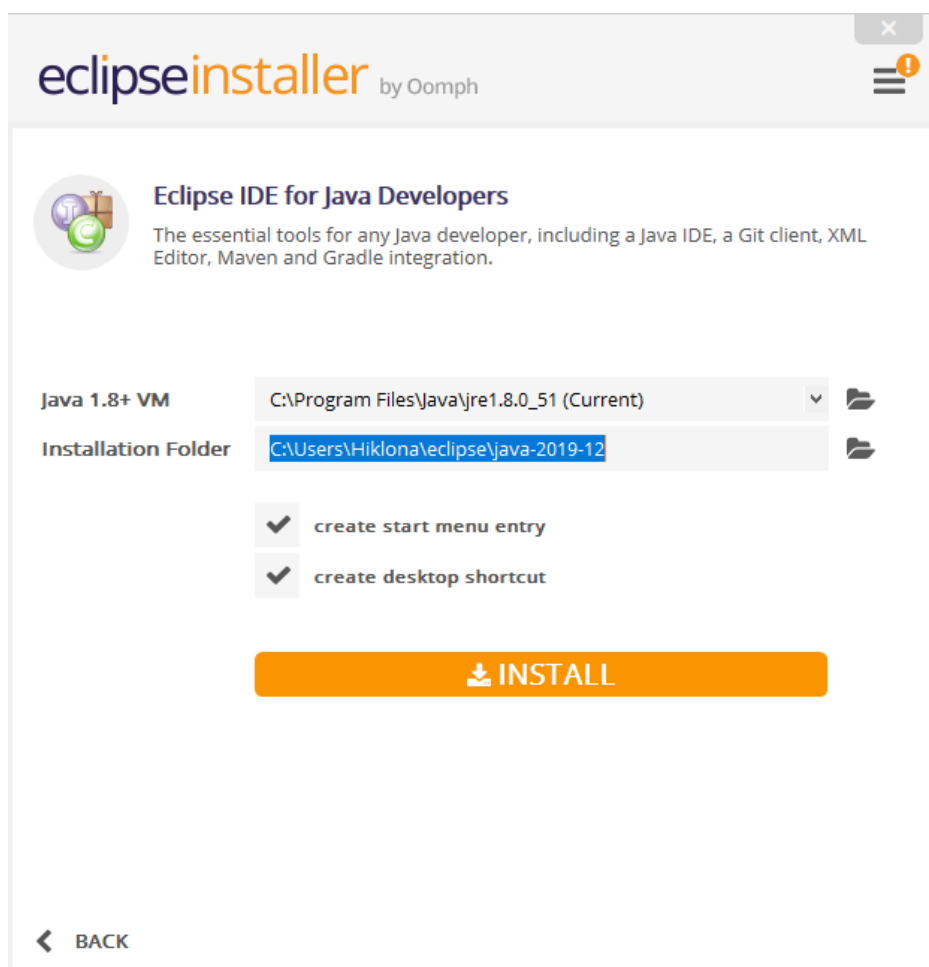


Рисунок. 9. Выбор расположения

После этого будет открыто дополнительное диалоговое окно с лицензионным соглашением, как показано на рисунке 10.

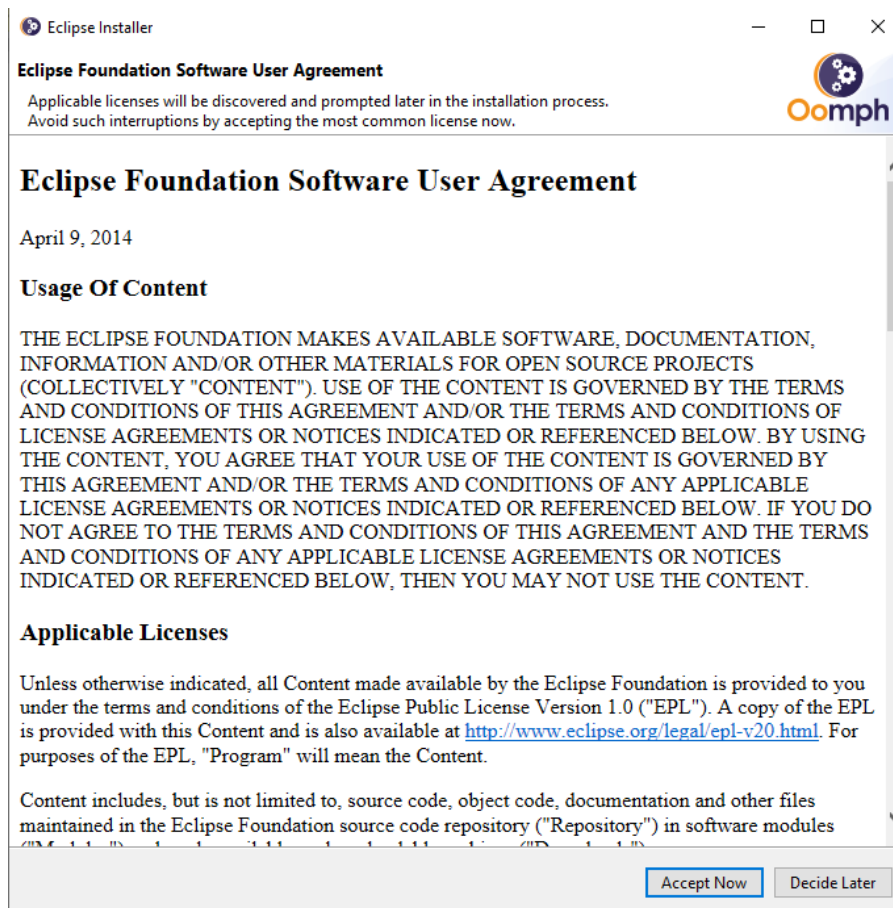


Рисунок. 10. Согласие

Далее последует непосредственная установка программы, как показано на рисунке 11.



Рисунок. 11. Установка

После установка будет завершена, как показано на рисунке 12. После этого можно нажать Launch для запуска программы сразу же.



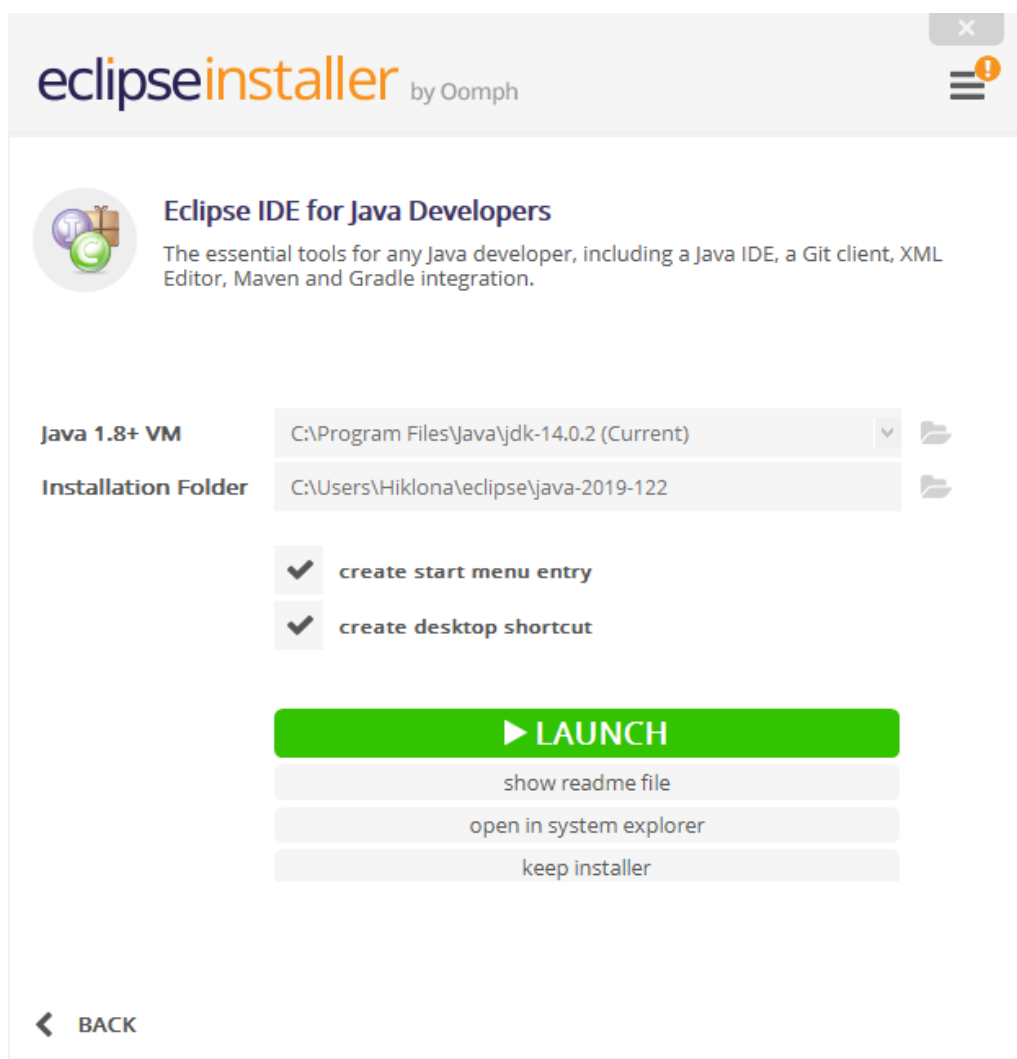


Рисунок. 12. Завершение установки

Если выбрать Launch, будет показан следующее изображение, как показано рисунке

13.



Рисунок. 13. Запуск программы

Далее будет открыто диалоговое окно, предлагающее выбрать рабочую область, как показано рисунке 14. Можно всё оставить без изменений и нажать Launch.

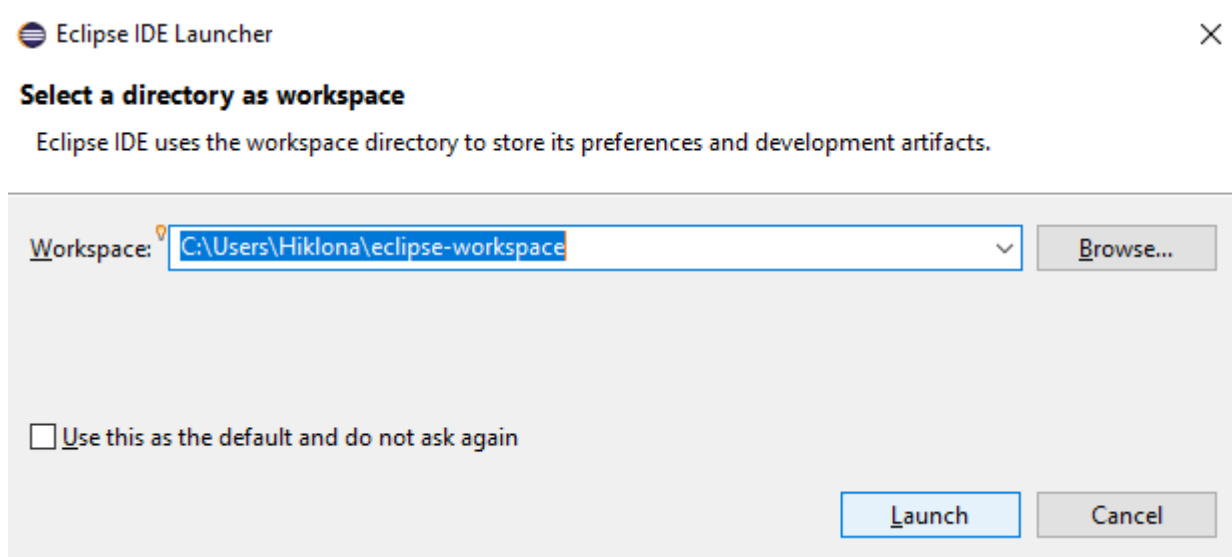


Рисунок. 14. Выбор директории

Далее снова будет показан баннер, показываемый при запуске программы, как показано на рисунке 15.



Рисунок. 15. Начало

После этого программа будет запущена, как показано на рисунке 20.

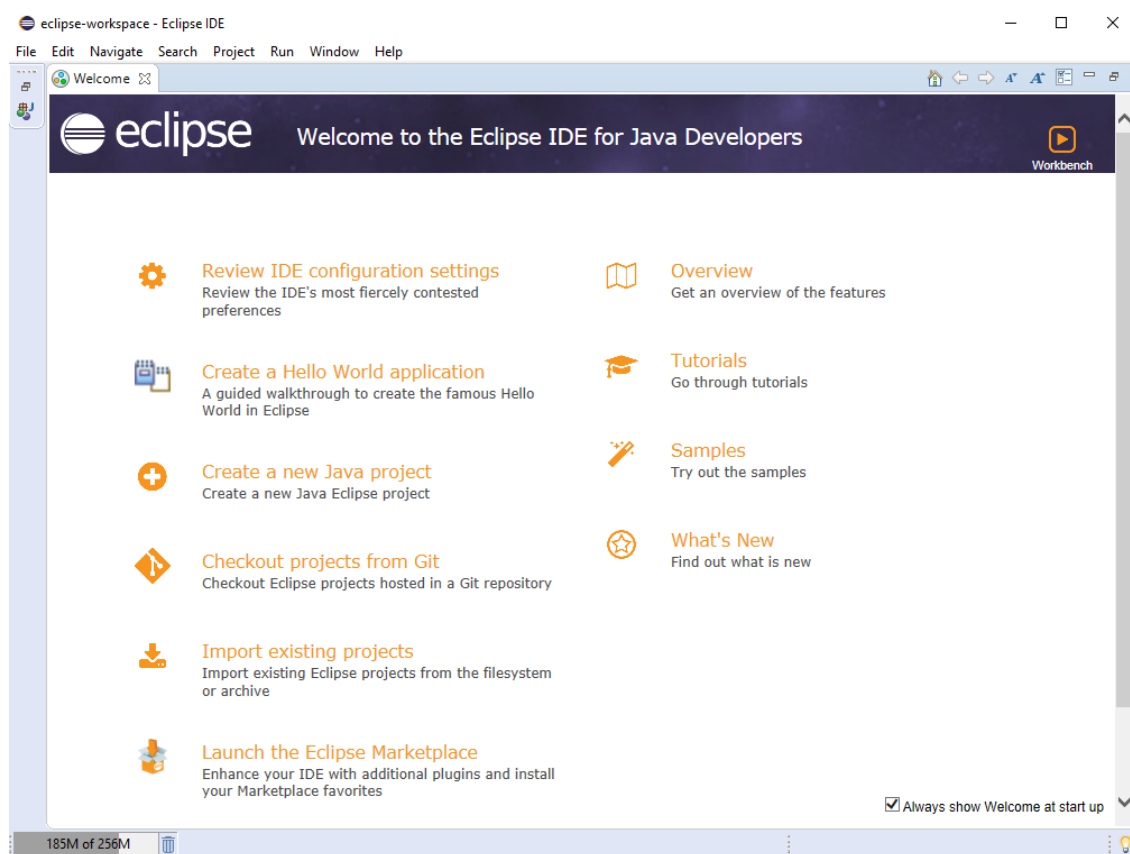


Рисунок. 16. Установка программы

## Установка Apache NetBeans

Открываем установщик Apache, после чего появляется окно с приветствием, для продолжения необходимо нажать Next, как показано на рисунке 17.

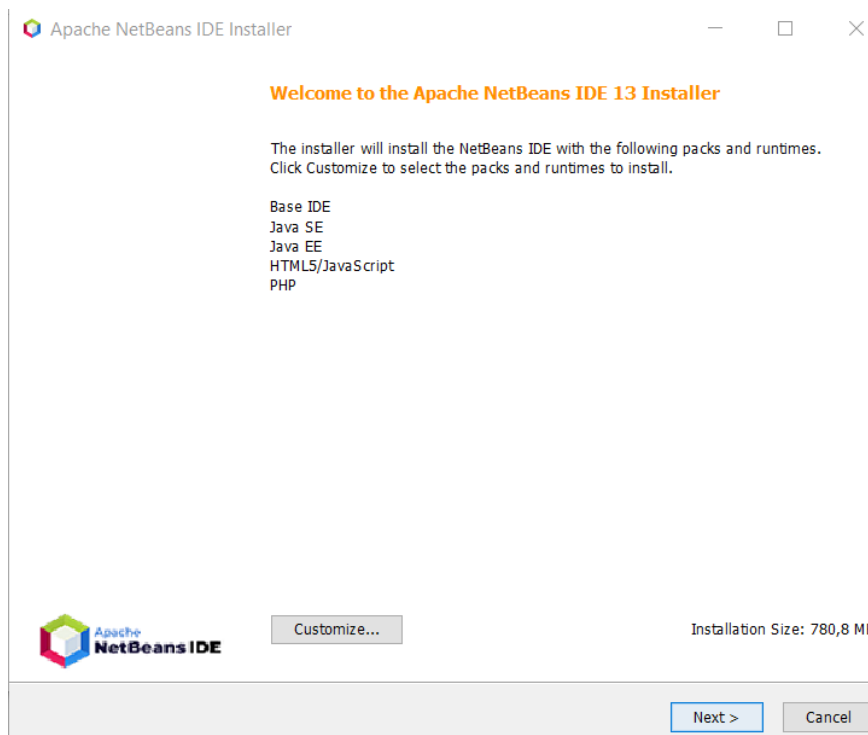


Рисунок. 17. Открытие установщика

После подтверждаем лицензионное соглашение и нажимаем Next, как показано на рисунке 18.

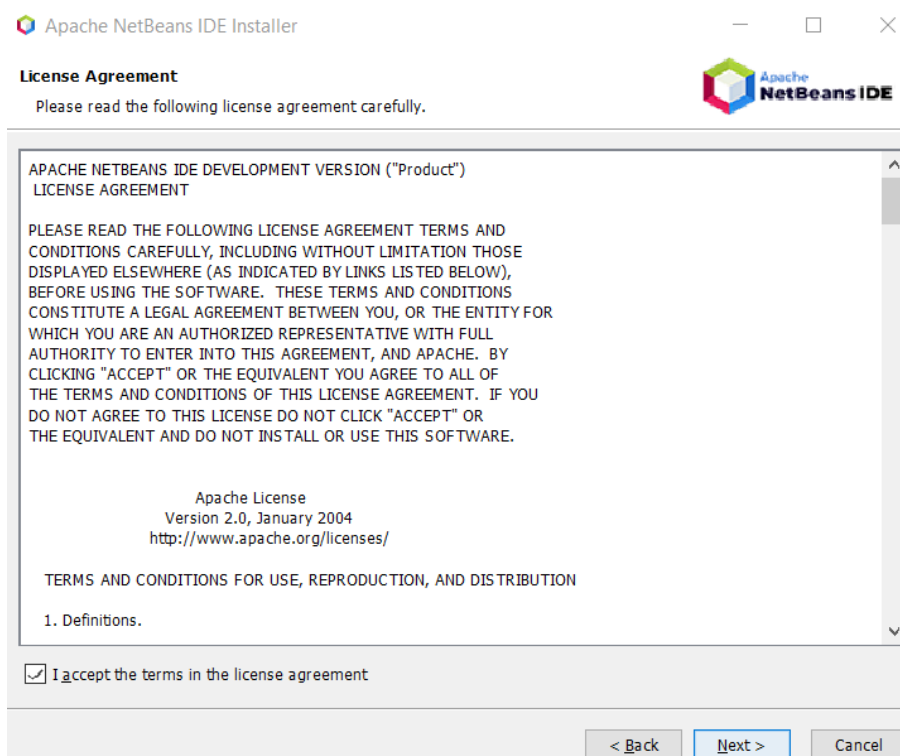


Рисунок. 18. Соглашение с лицензией

Далее необходимо выбрать путь, куда будет установлен Apache. Оставляем без изменений. Нажимаем Next, как показано на рисунке 19.

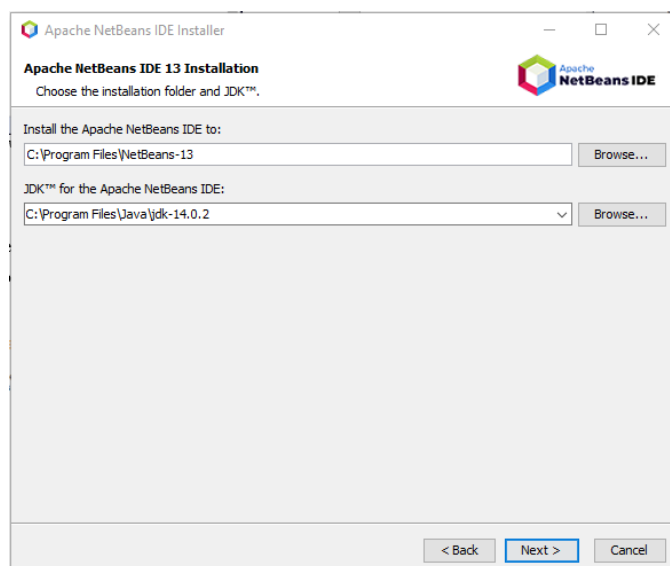


Рисунок. 19. Выбор пути

Для того, чтобы начать установку, нужно нажать кнопку установить.

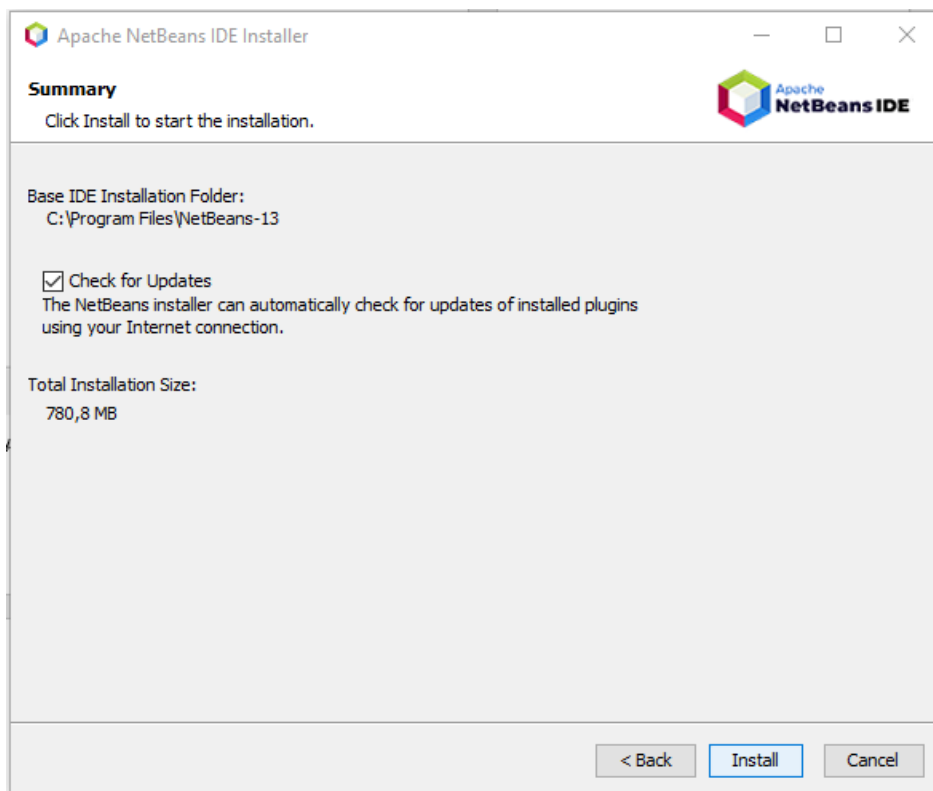


Рисунок. 20. Начало установки

После выполненных действий появится окно загрузки, показанное на рисунке 21.

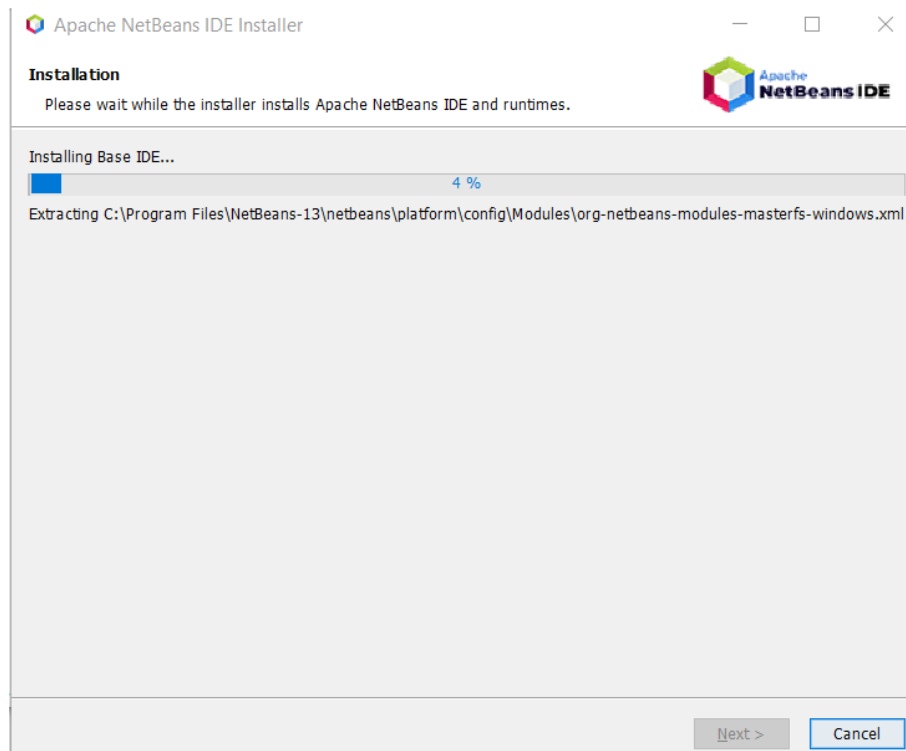


Рисунок. 21. Установка

Установка прошла успешно, нажимаем Finish.

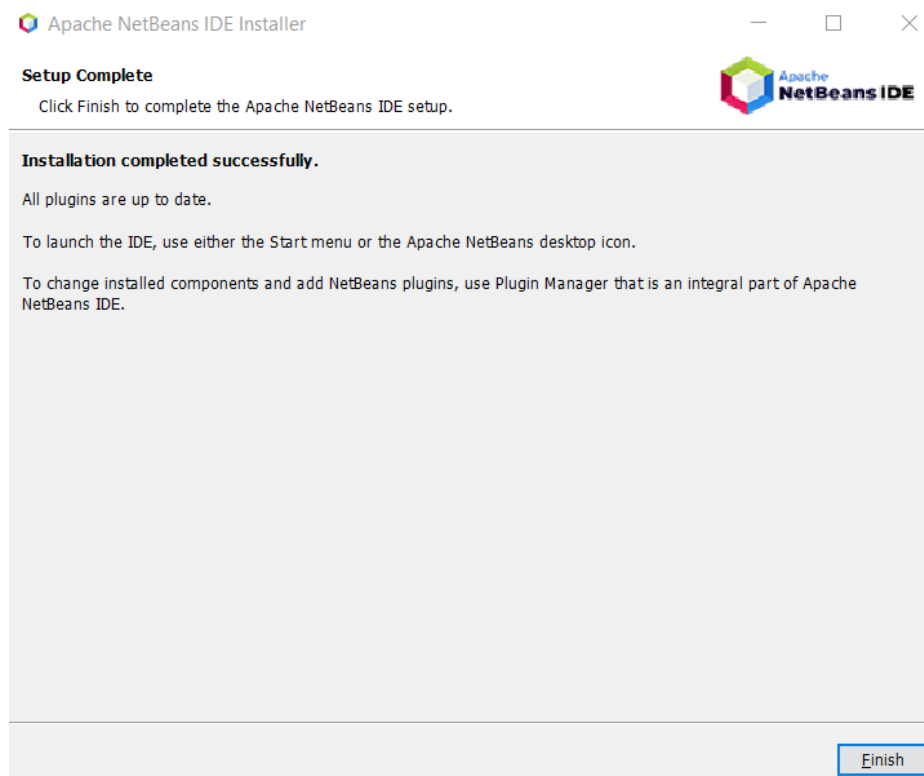


Рисунок 1. Установка завершена.

## Этап 2.

### Запуск Eclipse и создание проекта

Для начала необходимо запустить Eclipse, после необходимо создать проект, нажав на File, затем New, Java Project, как изображено на рисунке 22.

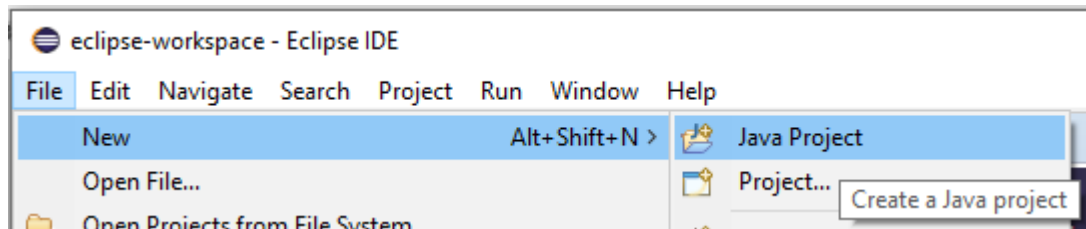


Рисунок. 22. Создание проекта

Далее появится диалоговое окно, в котором требуется выбрать нужные настройки, как показано на рисунке 23. В данном случае необходимо ввести имя проекта, выбрать JRE (JavaSE-13).

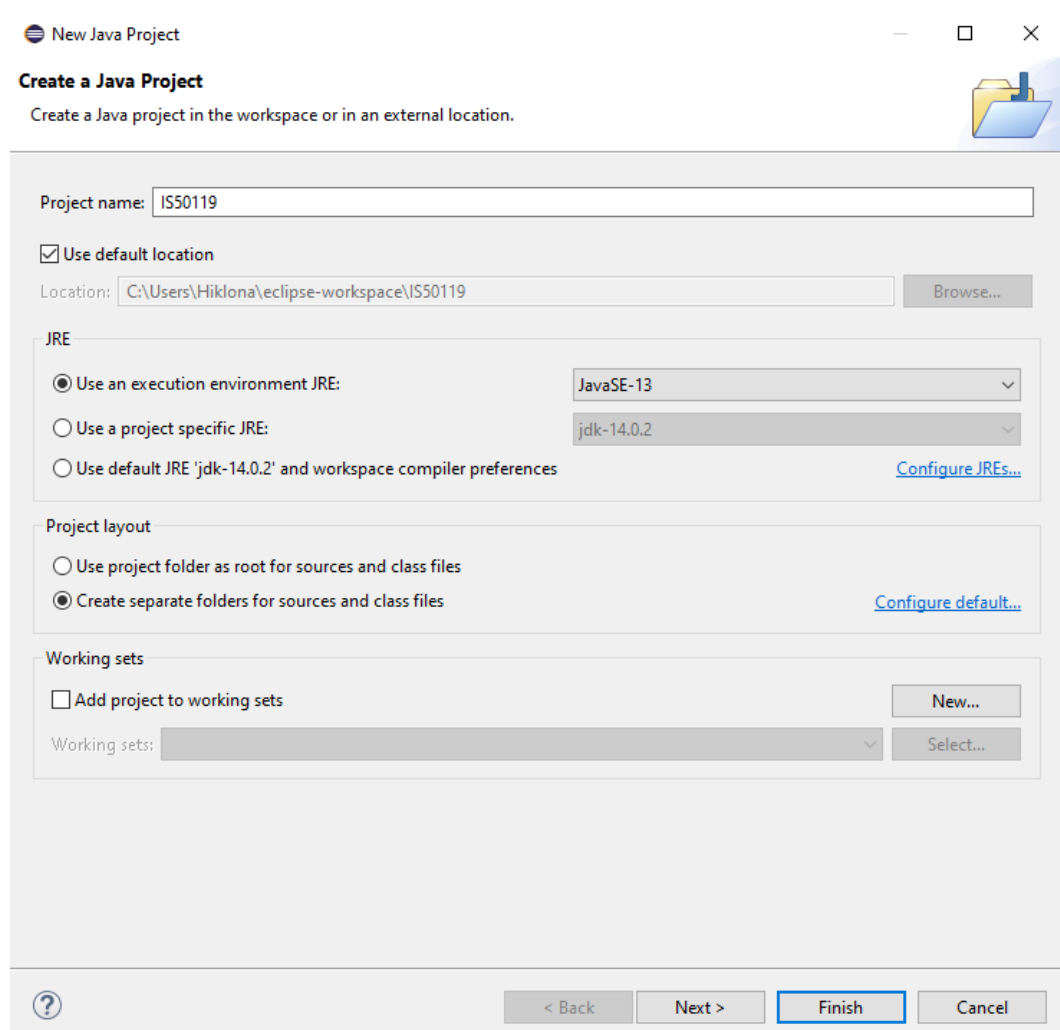


Рисунок. 23. Необходимые настройки

После этого появится созданный проект, как показано на рисунке 24..

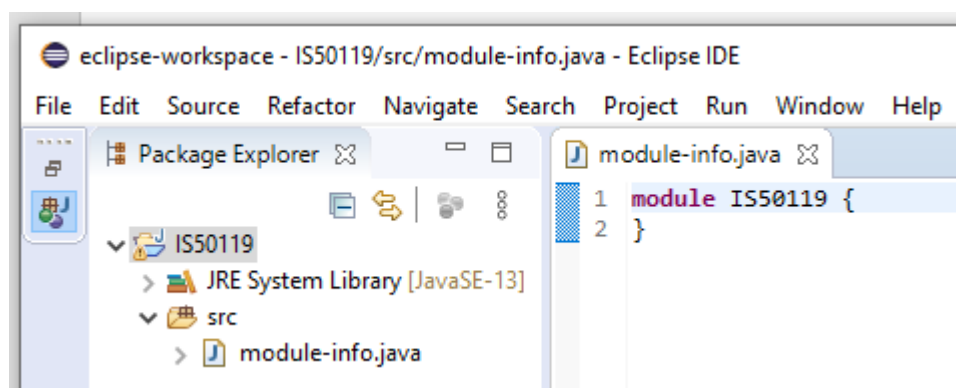


Рисунок. 24. Навигатор



## Создание класса

Далее необходимо нажать правой кнопкой мыши на src, затем выбрать New, после Class, как показано на рисунке 25.

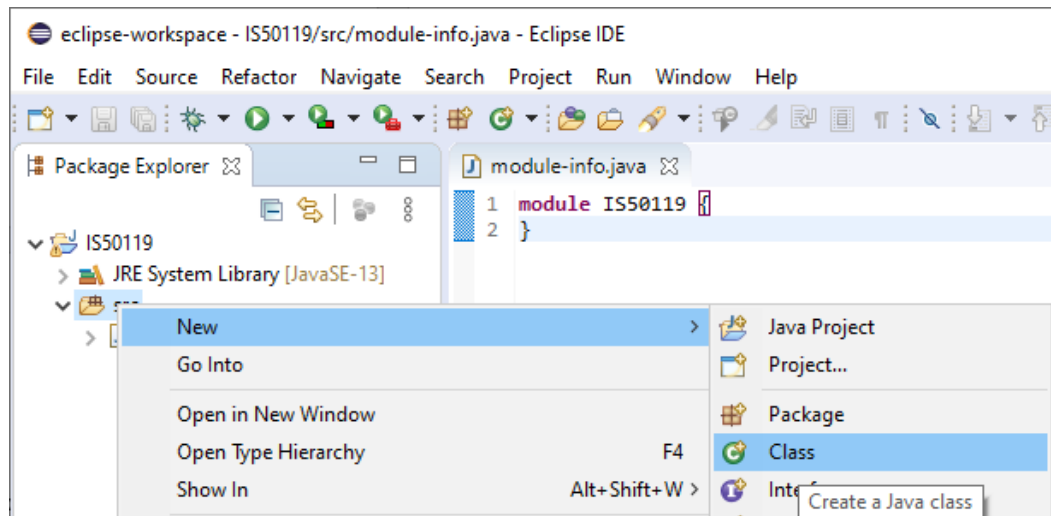


Рисунок. 25. Создание класса

После этого будет открыться диалоговое окно с выбором параметров, где необходимо ввести имя класса и установить галочку (public static void main), как показано на рисунке 26.

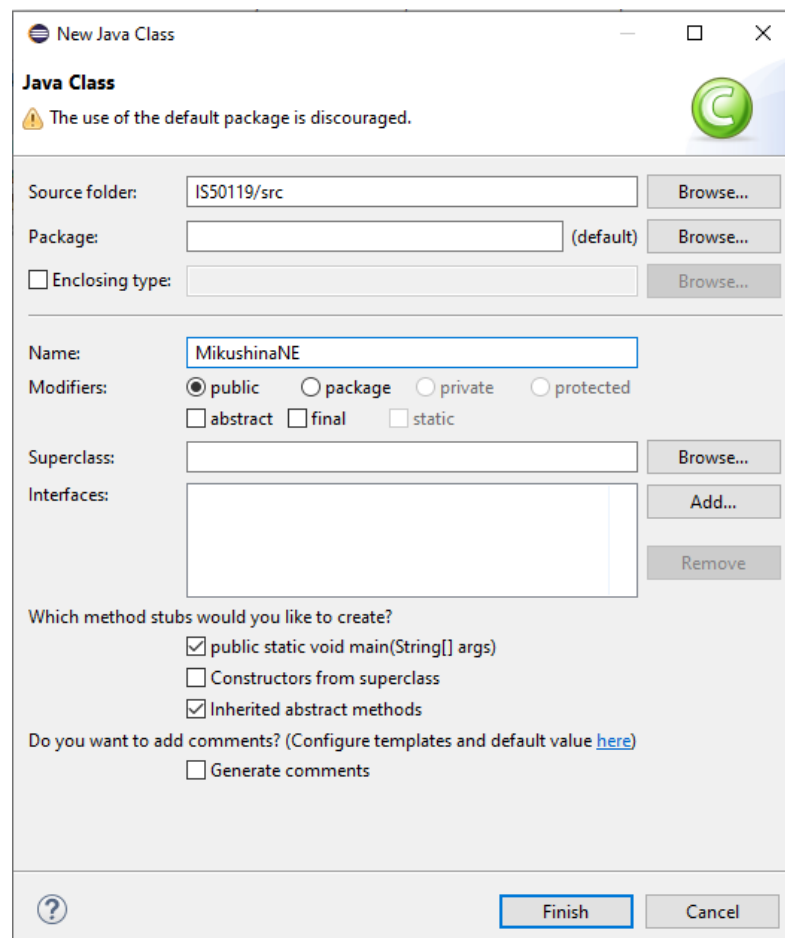


Рисунок. 26. Настройки класса

После этого класс появится в проекте, как показано на рисунке 27.

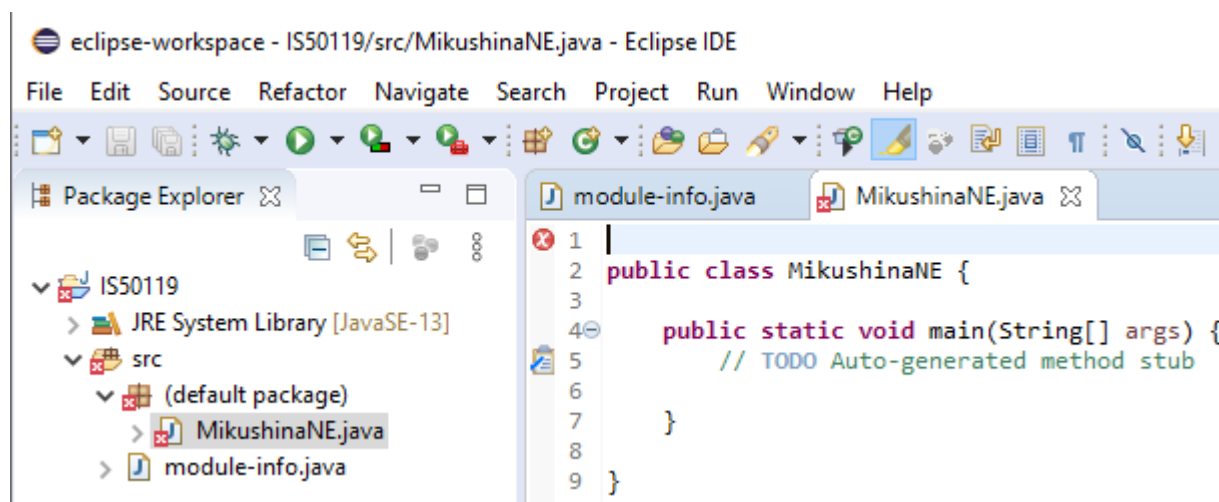
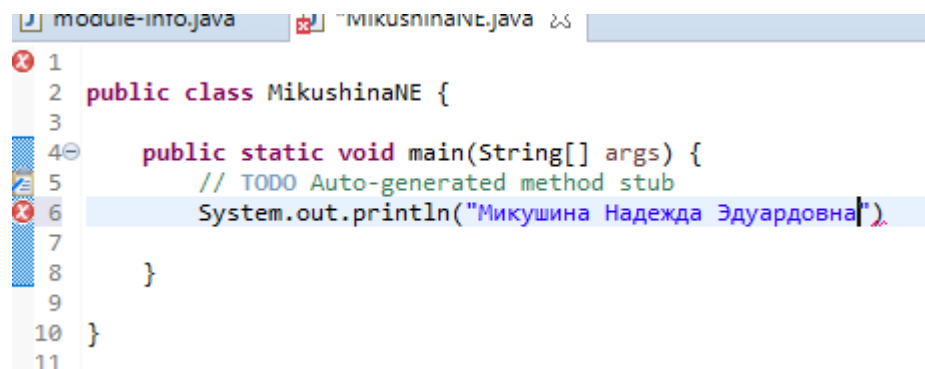


Рисунок. 27. Созданный класс

## Вывод надписи в консоль

Для того, чтобы вывести надпись в консоль необходимо использовать конструкцию `system.out.println()`, как показано на рисунке 28.



```
1 public class MikushinaNE {
2
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         System.out.println("Микушина Надежда Эдуардовна");
7     }
8 }
9
10
11
```

Рисунок. 28. Код первого задания

Для запуска проекта необходимо нажать на кнопку запуска в левом верхнем углу, как показано на рисунке 29.

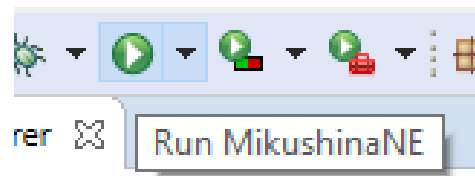


Рисунок. 29. Запуск

После запуска программы в консоли появится результат работы.

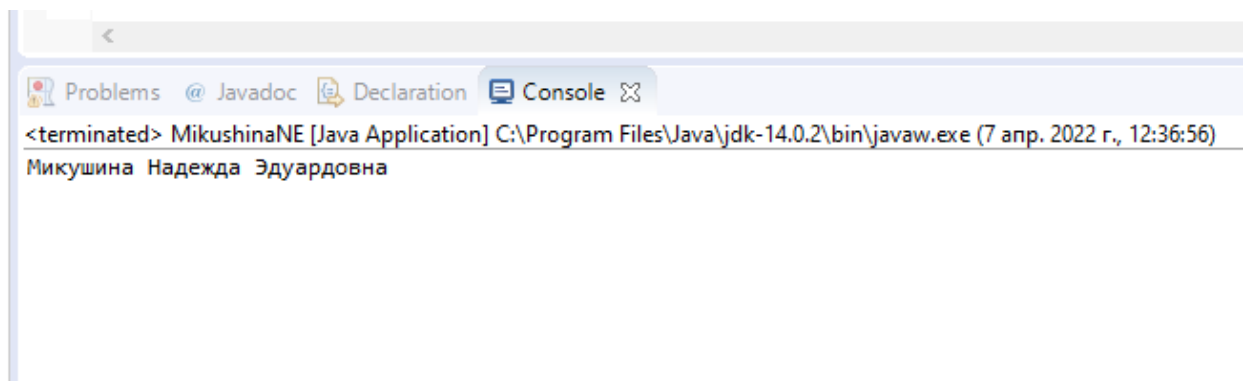
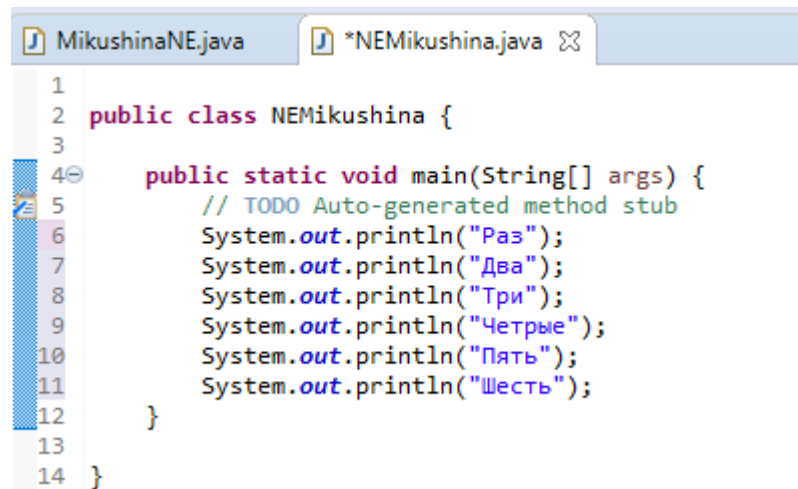


Рисунок. 30. Результат первого задания

## Вывод слов в столбец

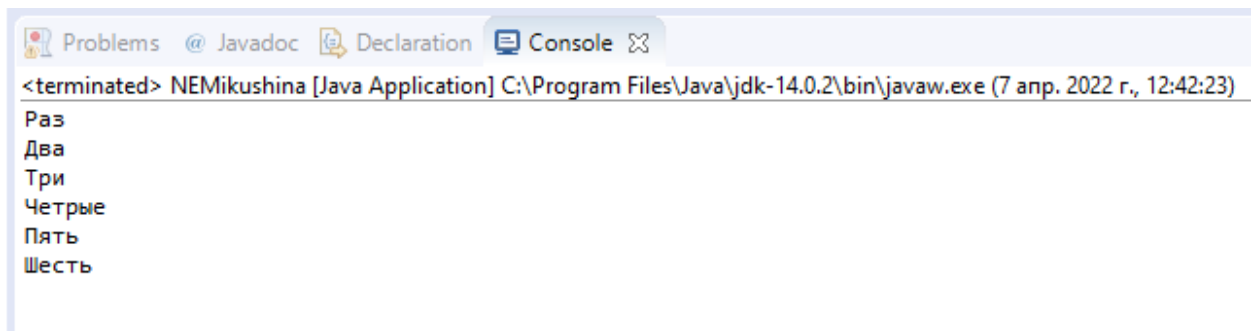
Для того, чтобы построчно вывести слова, необходимо использовать команду `system.out.println()`. Код задания показан на рисунке 31.



```
1  
2 public class NEMikushina {  
3  
4     public static void main(String[] args) {  
5         // TODO Auto-generated method stub  
6         System.out.println("Раз");  
7         System.out.println("Два");  
8         System.out.println("Три");  
9         System.out.println("Четыре");  
10        System.out.println("Пять");  
11        System.out.println("Шесть");  
12    }  
13  
14 }
```

Рисунок. 31. Код второго задания

На рисунке 32 показан результат выполнения второго задания.

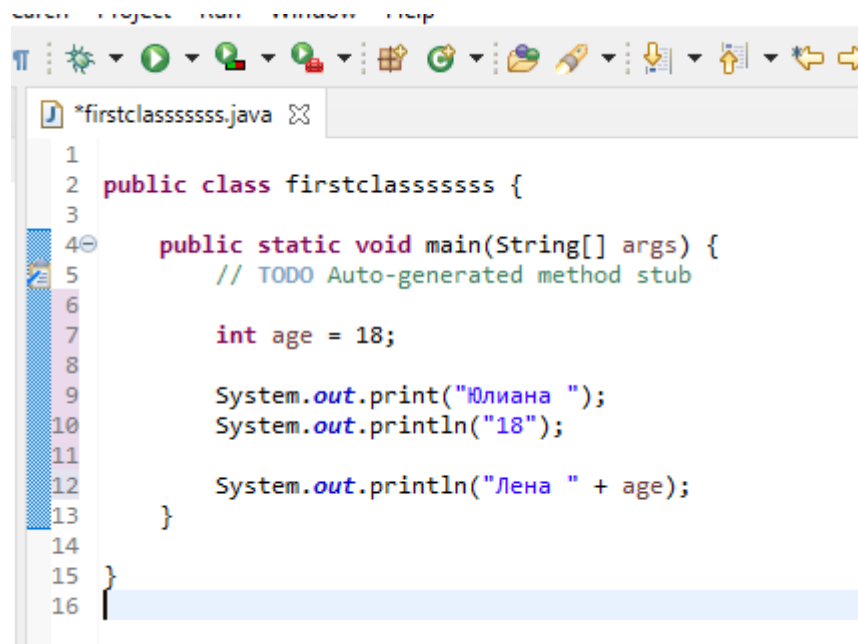


```
<terminated> NEMikushina [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (7 апр. 2022 г., 12:42:23)  
Раз  
Два  
Три  
Четыре  
Пять  
Шесть
```

Рисунок. 32. Результат второго задания

## Вывод возраста в одной строке

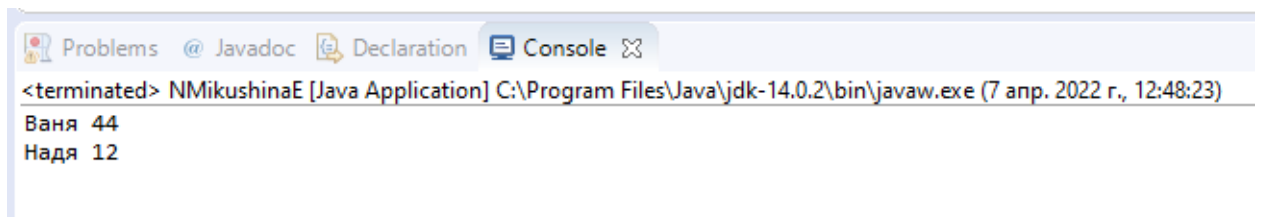
Для выполнения задания необходимо обозначить новую переменную, в которой будет храниться возраст. Ниже показан код выполнения задания двумя способами.



```
1
2 public class firstclasssssss {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7         int age = 18;
8
9         System.out.print("Юлиана ");
10        System.out.println("18");
11
12        System.out.println("Лена " + age);
13    }
14
15 }
16
```

Рисунок. 33. Код третьего задания

Результат выполнения работы показан на рисунке 34.



```
<terminated> NMikushinaE [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (7 апр. 2022 г., 12:48:23)
Ваня 44
Надя 12
```

Рисунок. 34. Результат третьего задания

### Этап 3.

#### Создание проекта в NetBeans

Перед началом работы необходимо создать новый проект, как показано на рисунке 35.

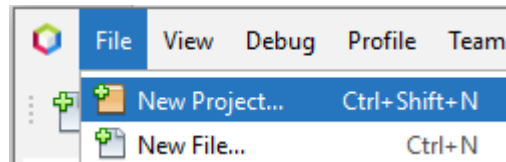


Рисунок. 35. Создание проекта

После откроется диалоговое окно, в котором необходимо выбрать категорию и тип проекта так, как показано на рисунке 36.

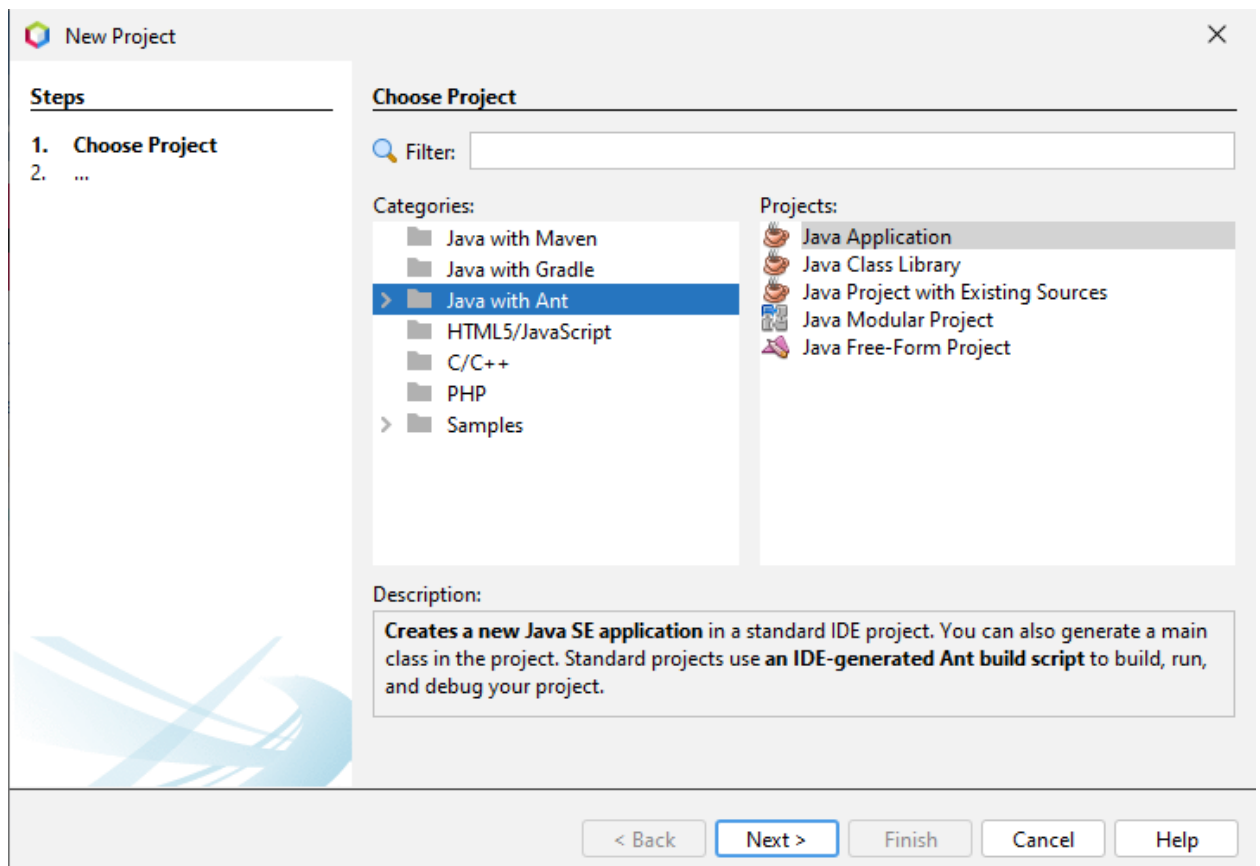


Рисунок. 36. Выбор типа проекта

Далее нужно выбрать имя проекта и путь, а так же убрать галочку, как показано на рисунке 37.

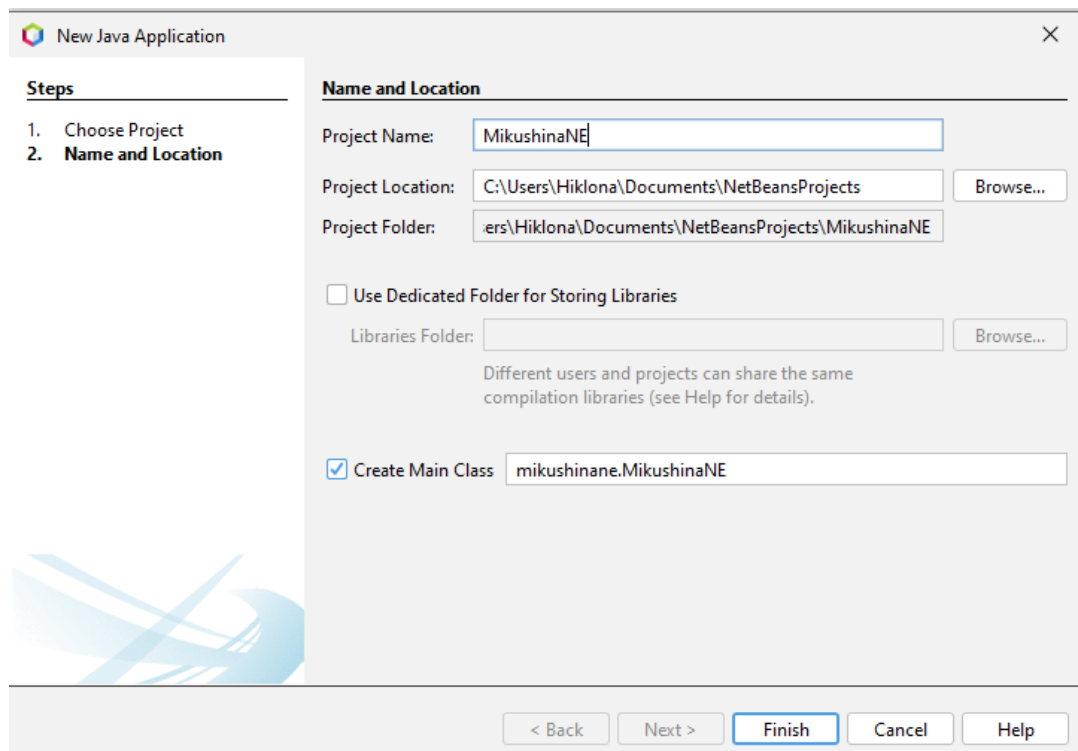


Рисунок. 37. Имя и расположения проекта

Созданный проект показан на рисунке 38.

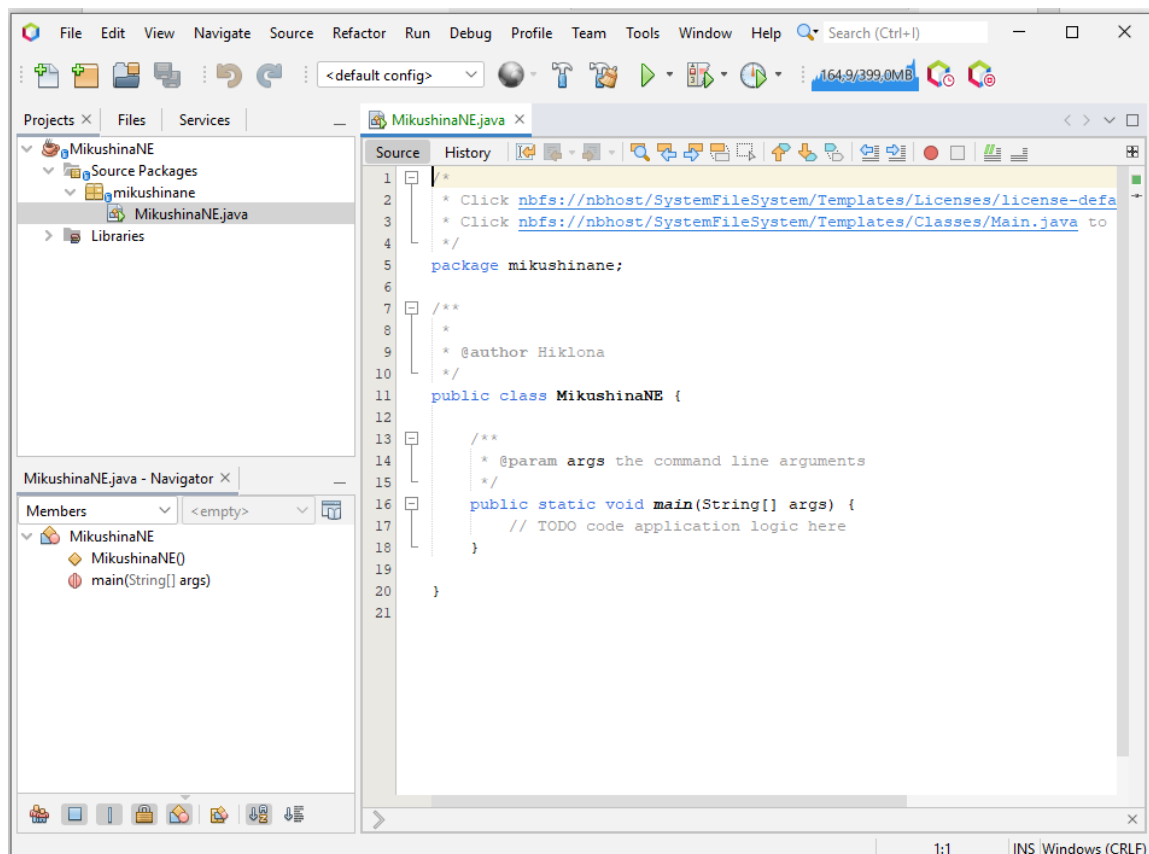


Рисунок. 38. Созданный проект

## Создание формы

Далее необходимо создать форму, нажав правой кнопкой мыши на название проекта, как показано на рисунке 39.

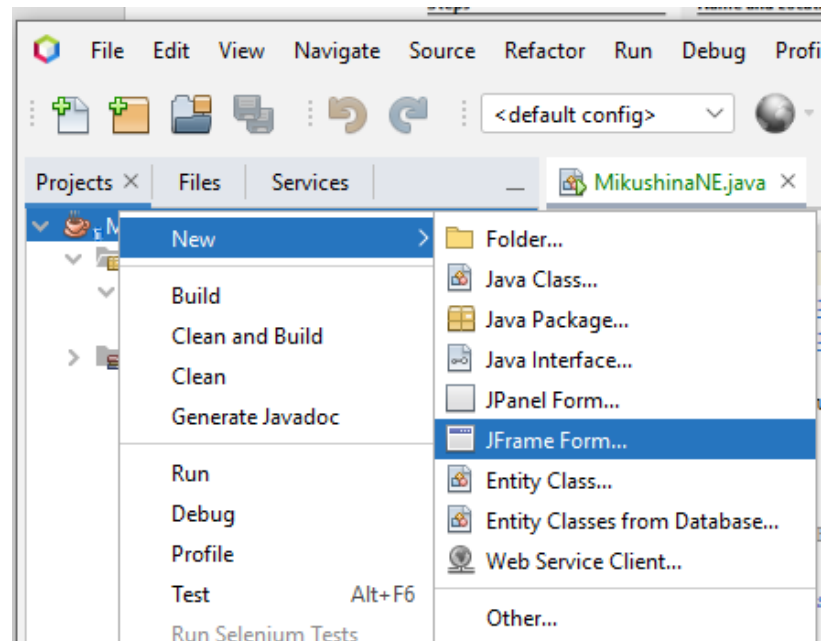


Рисунок. 39. Создание формы

В диалоговом окне требуется выбрать имя и расположение формы.

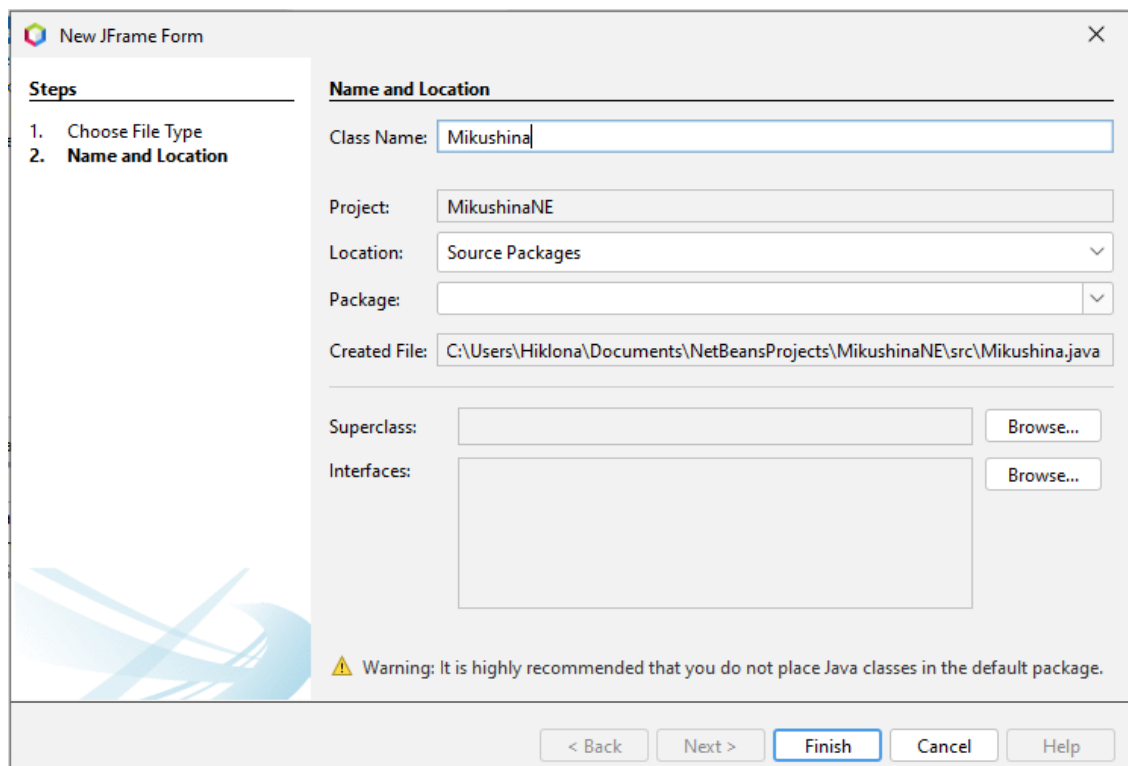


Рисунок. 40. Параметры формы

После чего создается форма, как показано на рисунке 41.



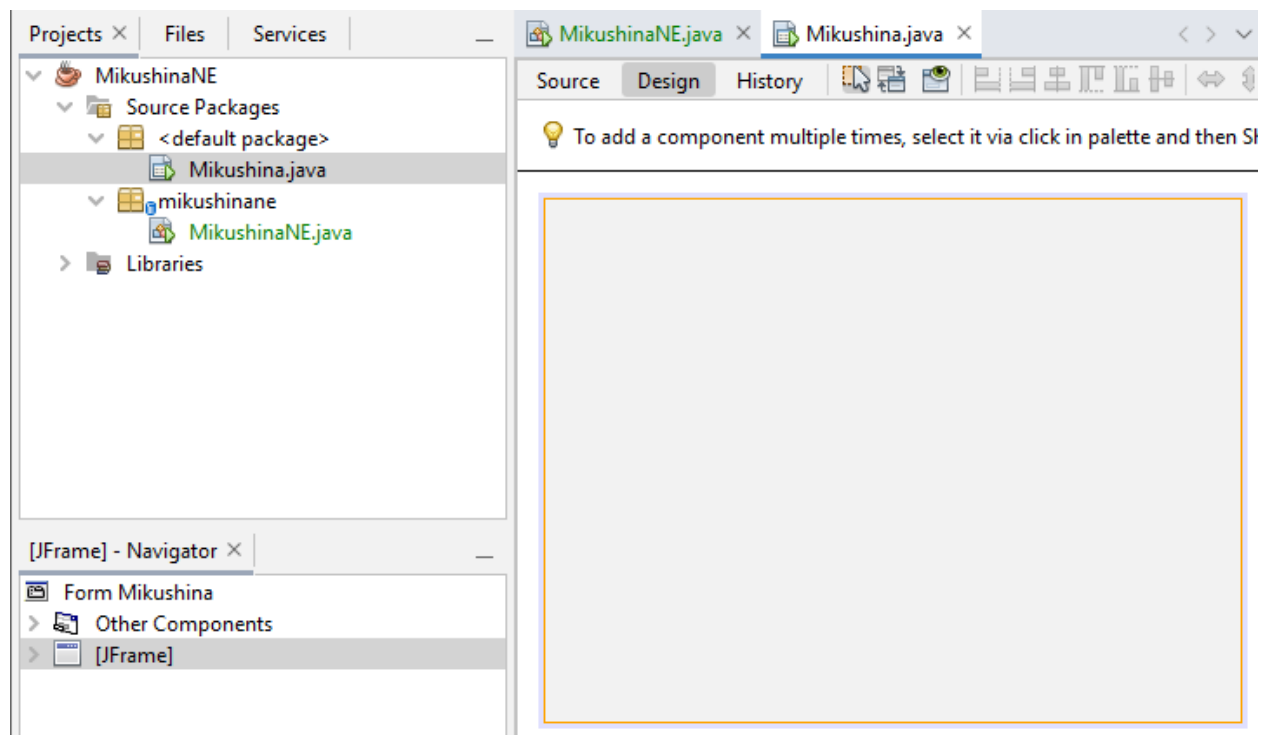


Рисунок. 41. Созданная форма

## Создание элементов формы

Далее нужно перетащить кнопку на форму, как показано на рисунке 42.

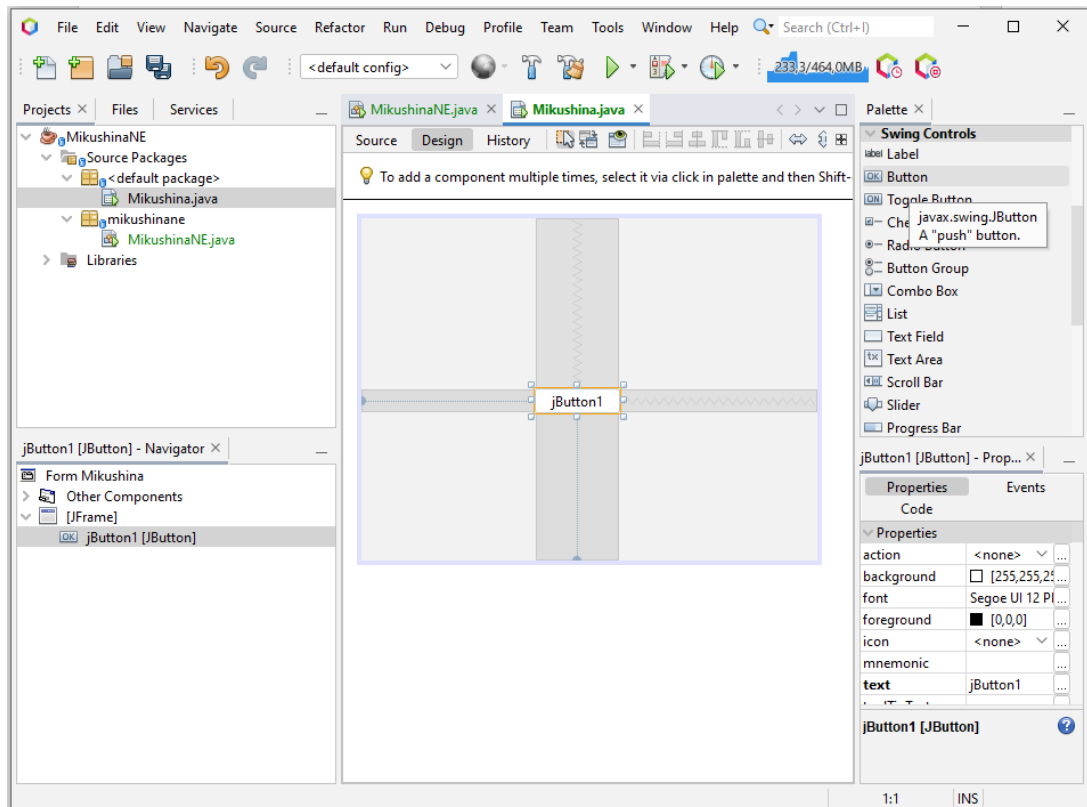


Рисунок. 42. Добавление кнопки

Для того, чтобы изменить текст кнопки, необходимо поменять значение в свойствах элемента, как показано на рисунке 43.

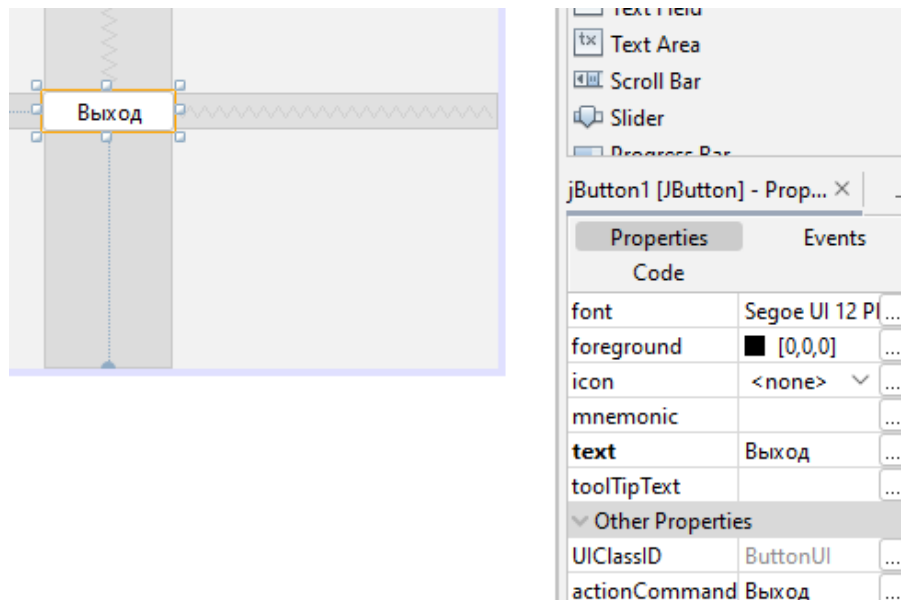


Рисунок. 43. Текст кнопки

В обработке событий требуется создать код, показанный на рисунке 44.

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

```

Рисунок. 44. Обработчик события

Готовый проект показан на рисунке 45.

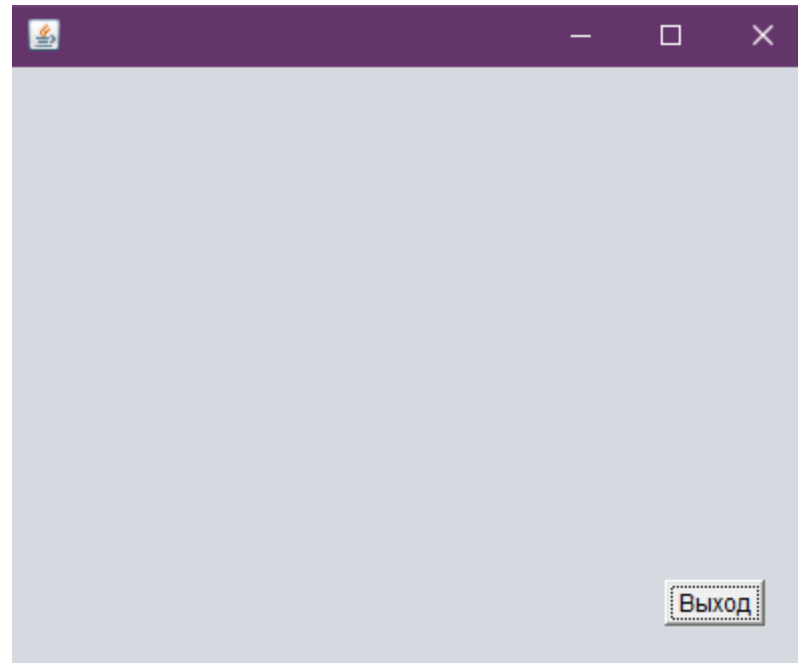


Рисунок. 45. Запущенная программа

После закрытия программы в консоли будет показано, что сборка завершена.

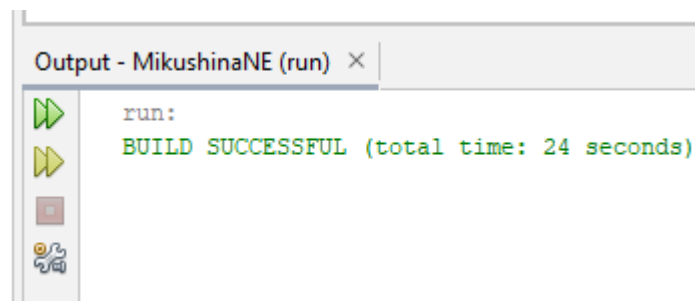


Рисунок. 46. Сборка завершена

Далее нужно добавить еще элементы на форму, а так же изменить в них текст, как показано на рисунке 47.

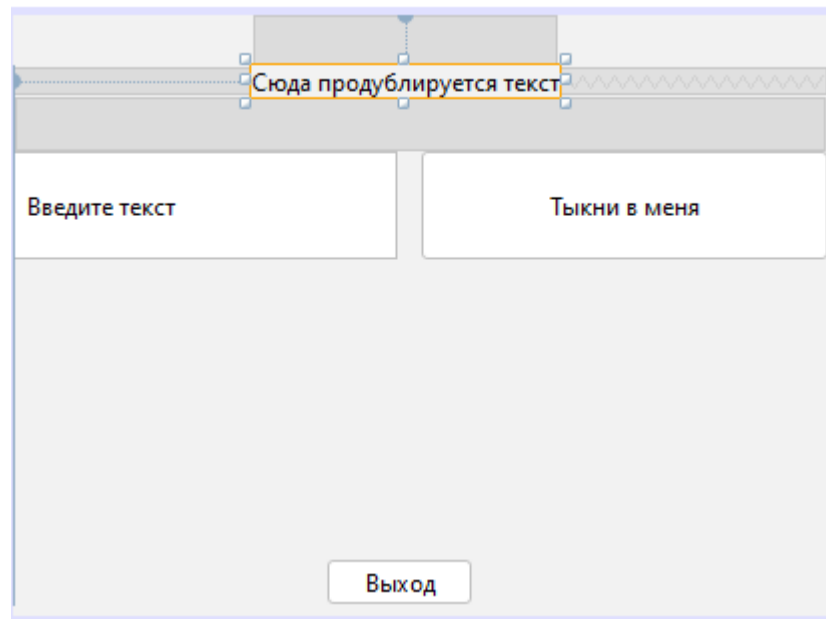


Рисунок. 47. Новые элементы

На рисунке 48 показан код в обработчике событий новой созданный кнопки.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    jLabel1.setText(jTextField1.getText());
}
```

Рисунок. 48. Обработчик события

При запуске программы в текстовое поле можно вводить значения, а нажав на кнопку, введенный текст перенесется в лейбл, как показано на рисунке 49.

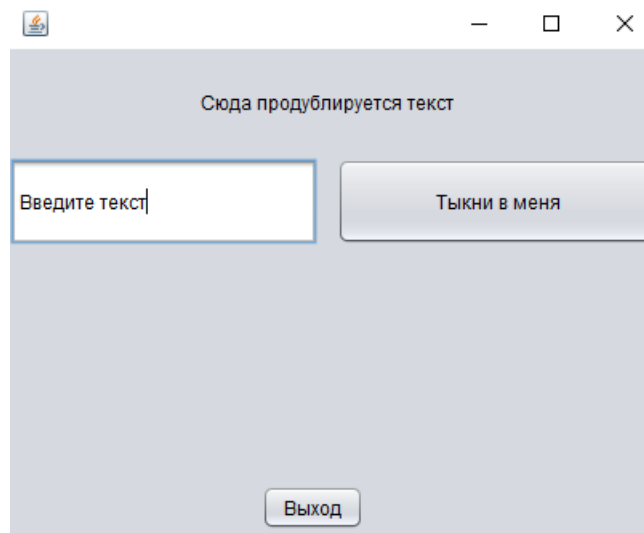


Рисунок. 49. Результат работы

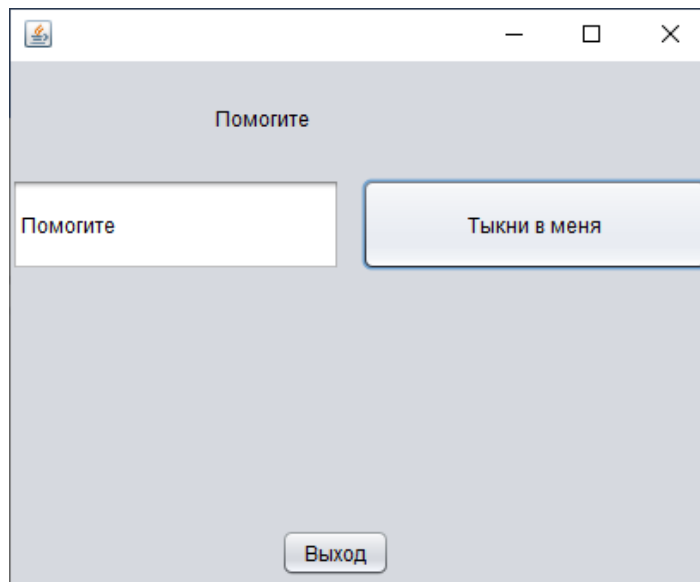


Рисунок. 50. Результат после нажатия

После нужно добавить еще элементы на форму, как показано на рисунке 51.

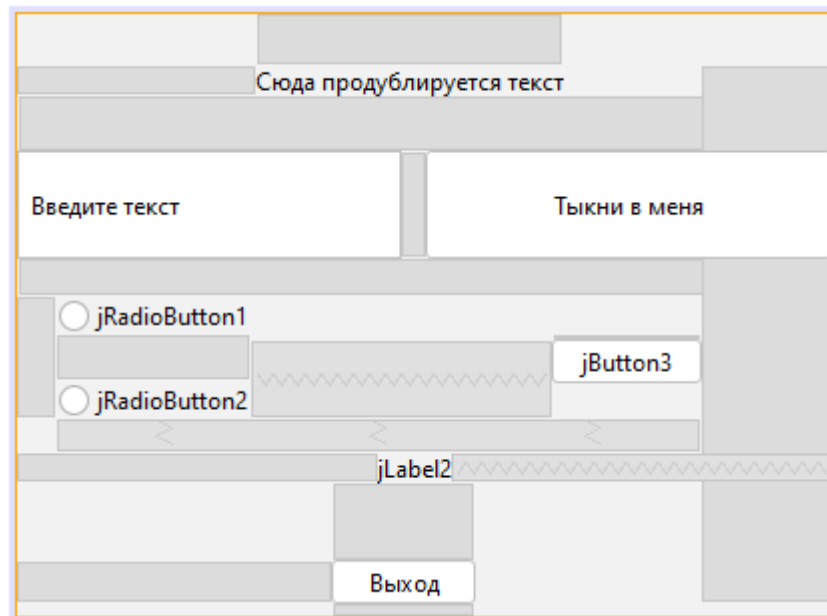


Рисунок. 51. Добавление новых элементов

Далее необходимо добавить элемент `buttonGroup`, который видно в свойствах, как показано на рисунке 52.

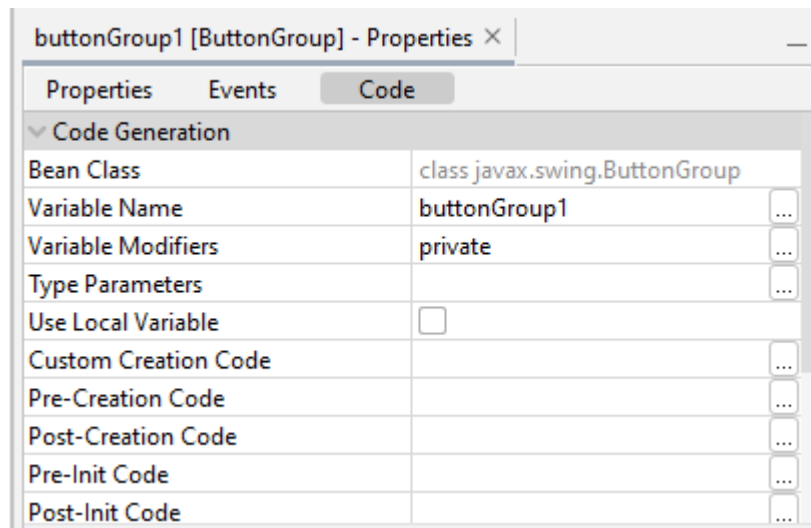


Рисунок. 52. Свойства группы

Для группировки переключателей необходимо им обоим в свойство `buttonGroup` добавить созданный элемент `buttonGroup1`, как показано на рисунке 53.

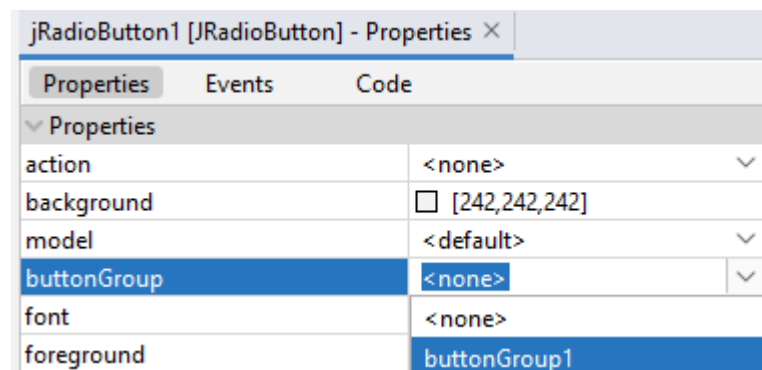


Рисунок. 53. Группировка

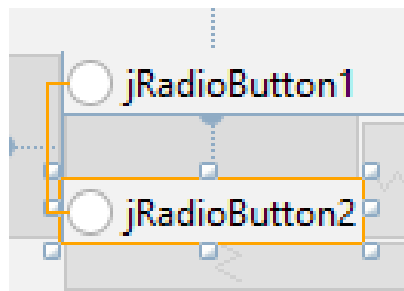


Рисунок. 54. Сгруппированы

Также нужно изменить текст элементов, как показано на рисунке 55.

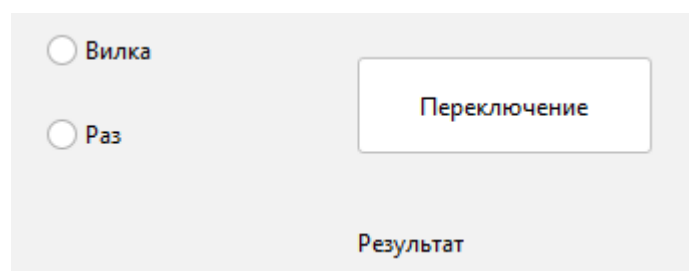


Рисунок. 55. Текст элементов

Код обработчика событий показан на рисунке 56.

```

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    if (jRadioButton1.isSelected()) {
        jLabel2.setText("Вилкой в глаз");
    }
    else if (jRadioButton2.isSelected()) {
        jLabel2.setText("Молодец");
    }
}

```

Рисунок. 56. Обработчик события

В результате при выборе переключателя и нажатия на кнопку будет меняться текст в метке, как показано на рисунках 57 и 58.

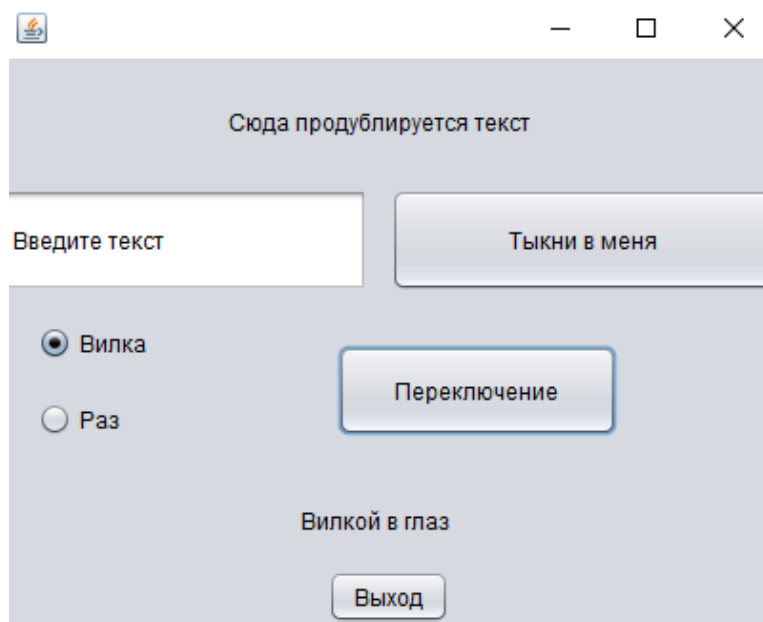


Рисунок. 57. Выбор первого радио

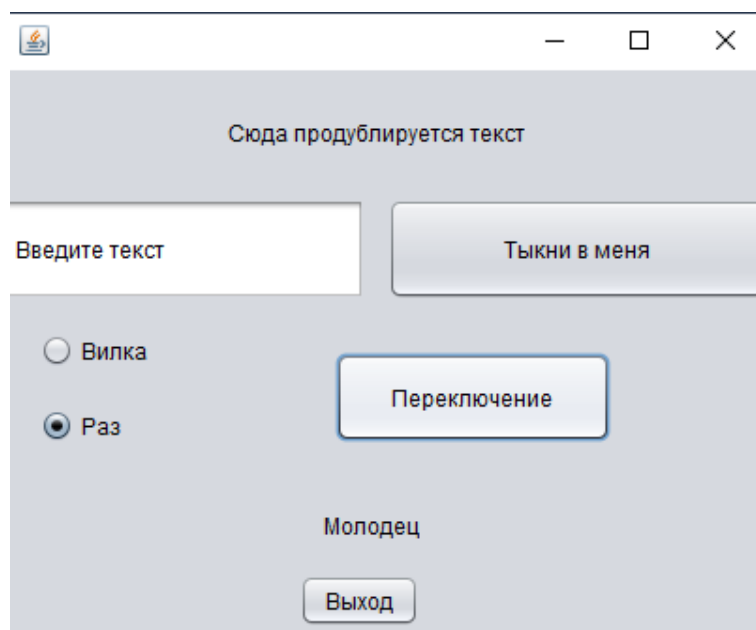


Рисунок. 58. Выбор второго радио

Готовое и запущенное приложение показано на рисунке 59.

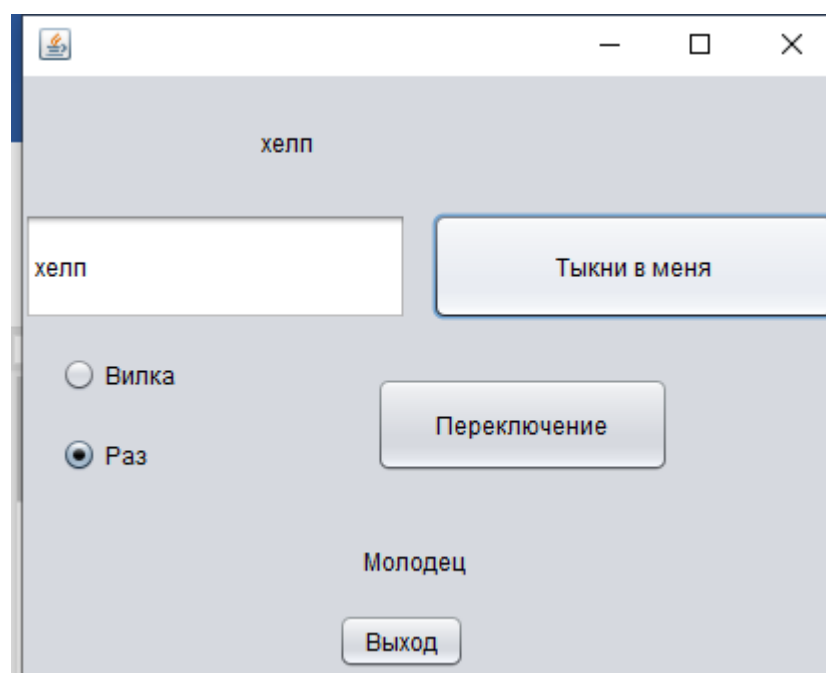


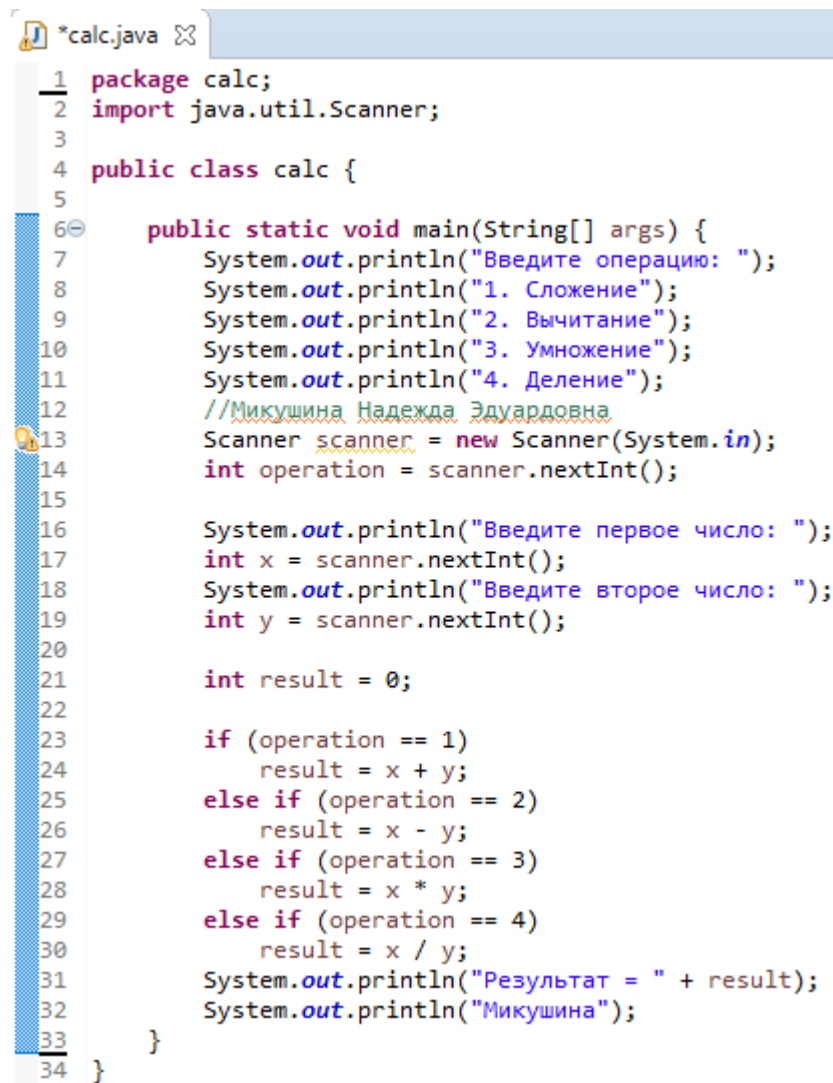
Рисунок. 59. Готовый запущенный проект с графическим интерфейсом



## Этап 4.

### Создание мини-калькулятора.

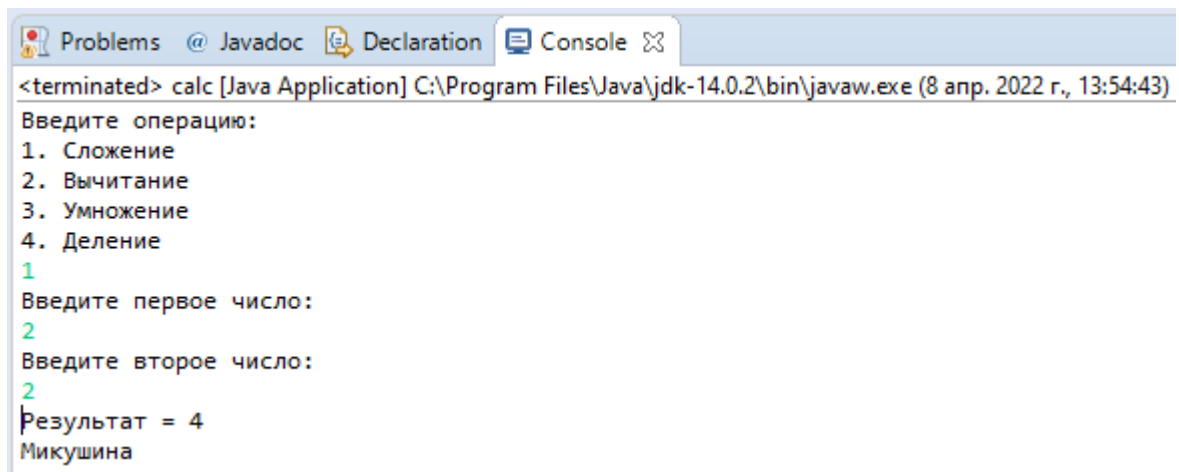
Для создания мини-калькулятора понадобится написание кода, показанного на рисунке 60.



```
1 package calc;
2 import java.util.Scanner;
3
4 public class calc {
5
6     public static void main(String[] args) {
7         System.out.println("Введите операцию: ");
8         System.out.println("1. Сложение");
9         System.out.println("2. Вычитание");
10        System.out.println("3. Умножение");
11        System.out.println("4. Деление");
12        //Микушина Надежда Эдуардовна
13        Scanner scanner = new Scanner(System.in);
14        int operation = scanner.nextInt();
15
16        System.out.println("Введите первое число: ");
17        int x = scanner.nextInt();
18        System.out.println("Введите второе число: ");
19        int y = scanner.nextInt();
20
21        int result = 0;
22
23        if (operation == 1)
24            result = x + y;
25        else if (operation == 2)
26            result = x - y;
27        else if (operation == 3)
28            result = x * y;
29        else if (operation == 4)
30            result = x / y;
31        System.out.println("Результат = " + result);
32        System.out.println("Микушина");
33    }
34 }
```

Рисунок. 60. Код программы

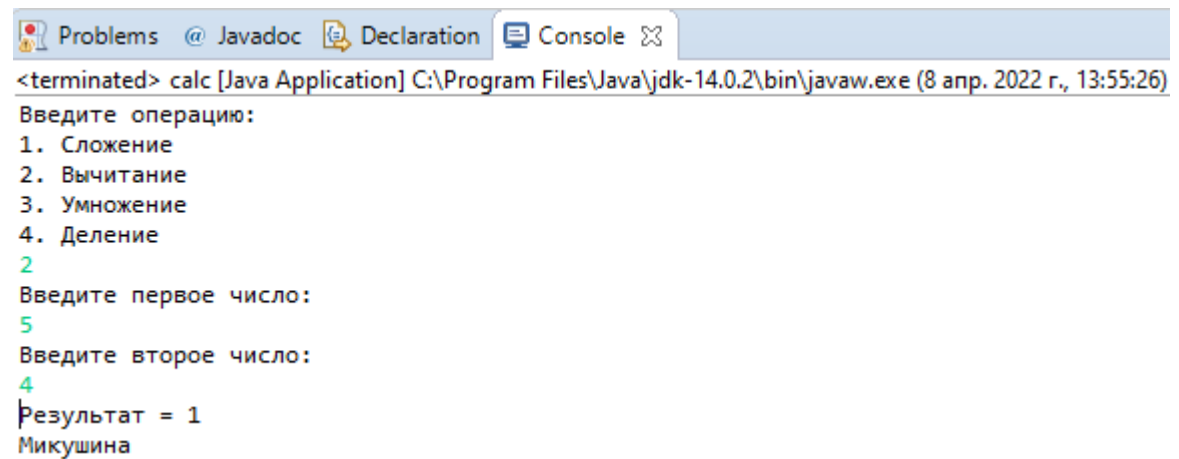
Результат работы программы при выборе операции «Сложение», показан рисунке 61.



```
<terminated> calc [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (8 апр. 2022 г., 13:54:43)
Введите операцию:
1. Сложение
2. Вычитание
3. Умножение
4. Деление
1
Введите первое число:
2
Введите второе число:
2
Результат = 4
Микушина
```

Рисунок. 61. Результат работы операции сложение

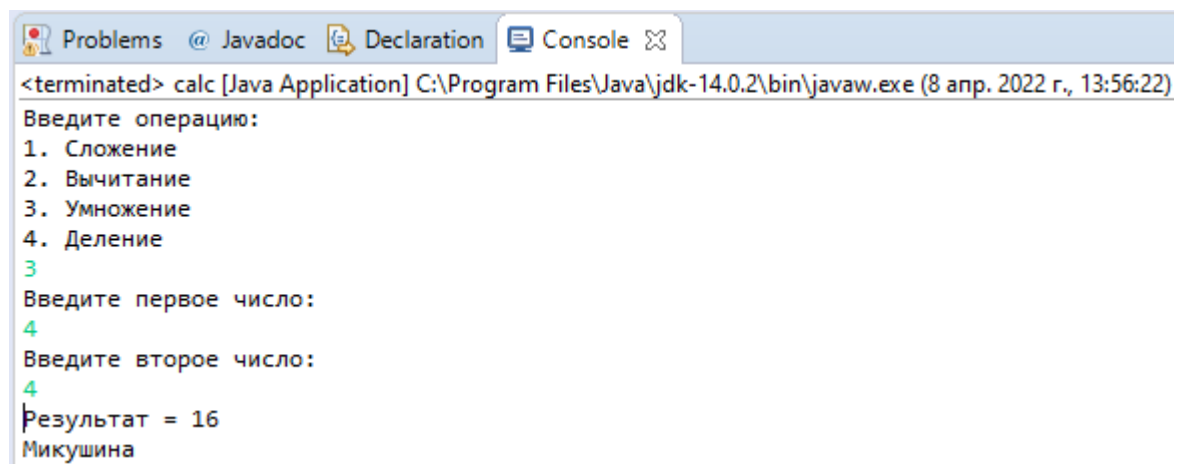
Результат работы программы при выборе операции «Вычитание», показан рисунке 62.



```
<terminated> calc [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (8 апр. 2022 г., 13:55:26)
Введите операцию:
1. Сложение
2. Вычитание
3. Умножение
4. Деление
2
Введите первое число:
5
Введите второе число:
4
Результат = 1
Микушина
```

Рисунок. 62. Результат работы операции вычитание

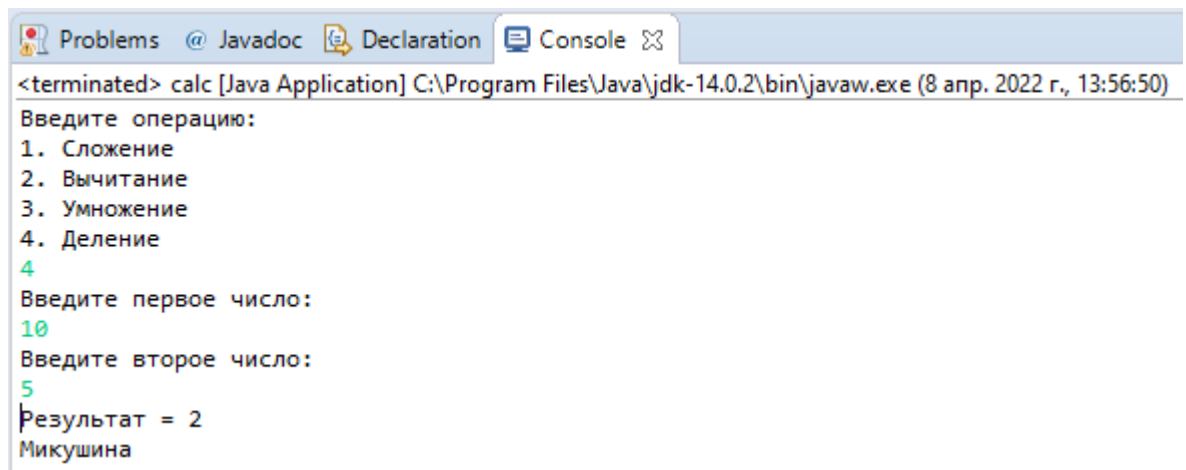
Результат работы программы при выборе операции «Умножение», показан рисунке 63.



```
<terminated> calc [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (8 апр. 2022 г., 13:56:22)
Введите операцию:
1. Сложение
2. Вычитание
3. Умножение
4. Деление
3
Введите первое число:
4
Введите второе число:
4
Результат = 16
Микушина
```

Рисунок. 63. Результат работы операции умножение

Результат работы программы при выборе операции «Деление», показан рисунке 64.



```
<terminated> calc [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (8 апр. 2022 г., 13:56:50)
Введите операцию:
1. Сложение
2. Вычитание
3. Умножение
4. Деление
4
Введите первое число:
10
Введите второе число:
5
Результат = 2
Микушина
```

Рисунок. 64. Результат работы операции деление

## Этап 5.

### Установка символов табуляции в качестве отступов

Перед началом работы необходимо запустить Eclipse, после чего в меню выбрать Window Preferences, как показано на рисунке 65.

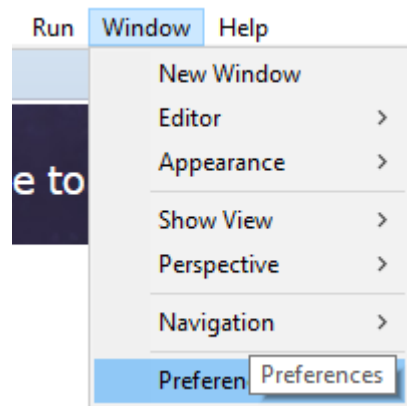


Рисунок. 65. Выбор Preferences

Далее будет открыто диалоговое окно, в котором нужно открыть Java -> Code Style -> Formatter, как показано на рисунке 66.

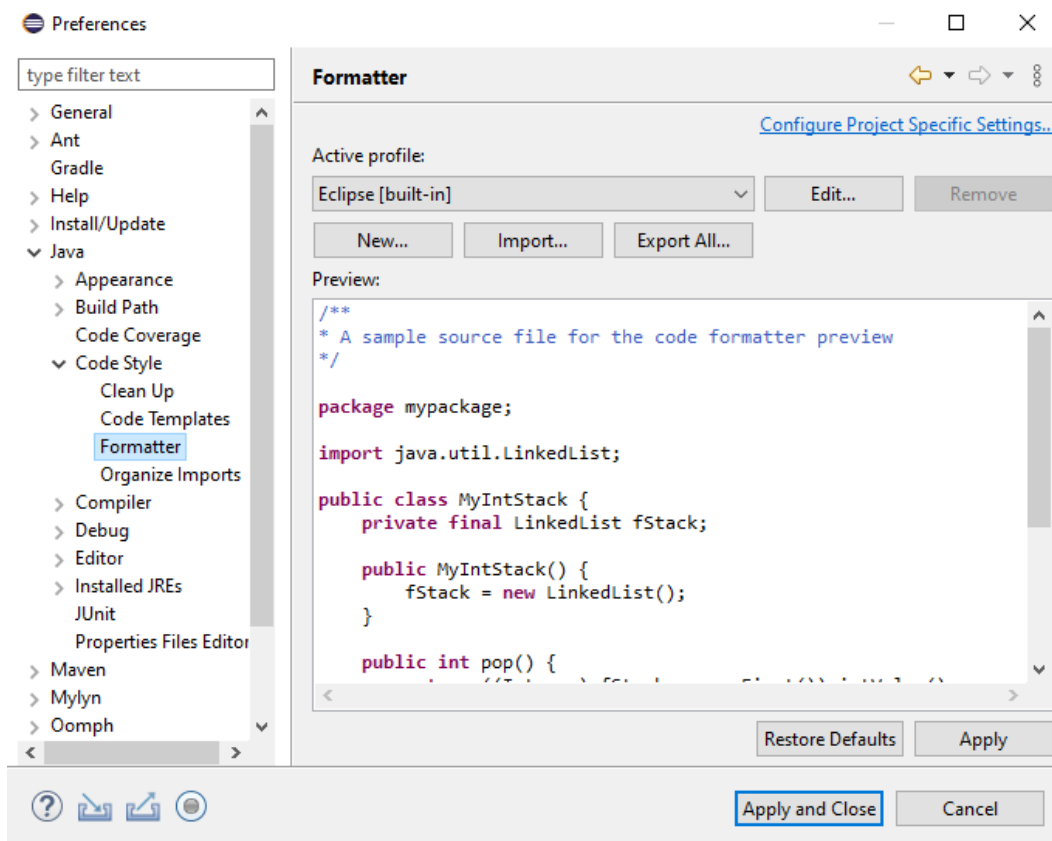


Рисунок. 66. Раздел Formatter

В качестве Active profile необходимо выбрать Java Conventions [built-in], как показано на рисунке 67.

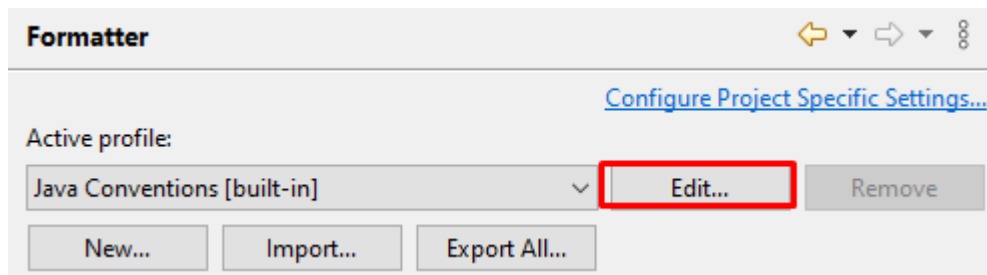


Рисунок. 67. Выбор Java Conventions

После чего будет открыто диалоговое окно, показанное на рисунке 68.

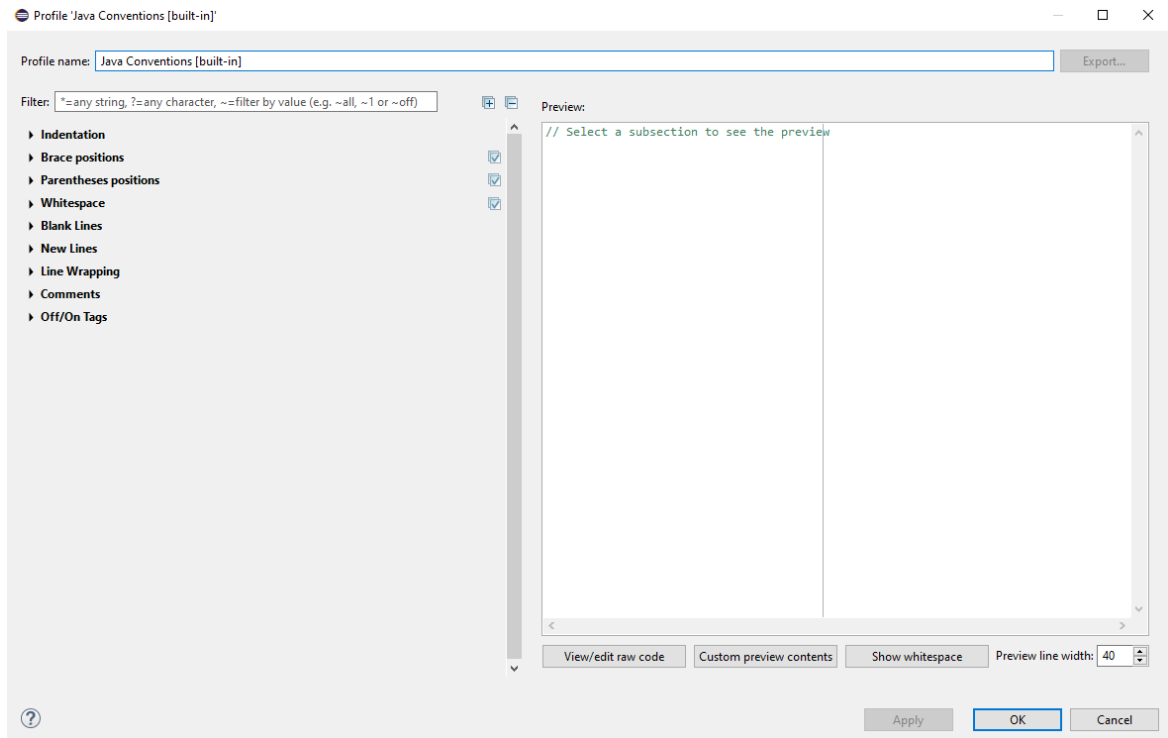


Рисунок. 68. Диалоговое окно Java Conventions

В нем нужно открыть вкладку Indentation и установить значение Tabs only, как показано на рисунке 69.

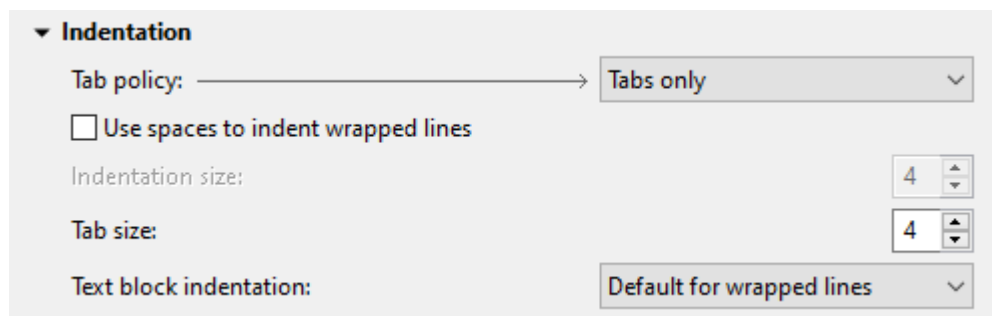


Рисунок. 69. Установка Tabs only

После этого нужно закрыть диалоговое окно Preferences, как показано на рисунке 70.

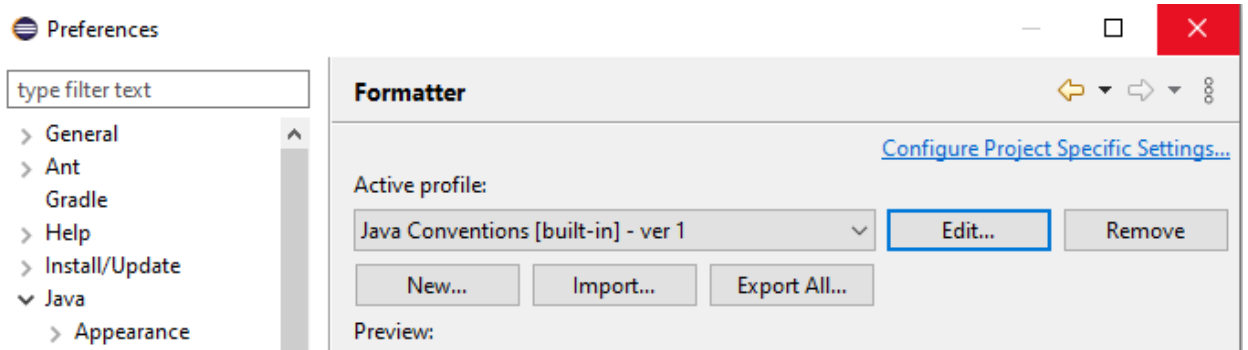


Рисунок. 70/ Закрытие диалогового окна Preferences

В результате отступами в коде приложения будут только символы табуляции.

## Переименование элементов проекта

Для того, чтобы переименовать класс нужно нажать на файл класса правой кнопкой мыши, затем Refactor -> Rename, как показано на рисунке 71.

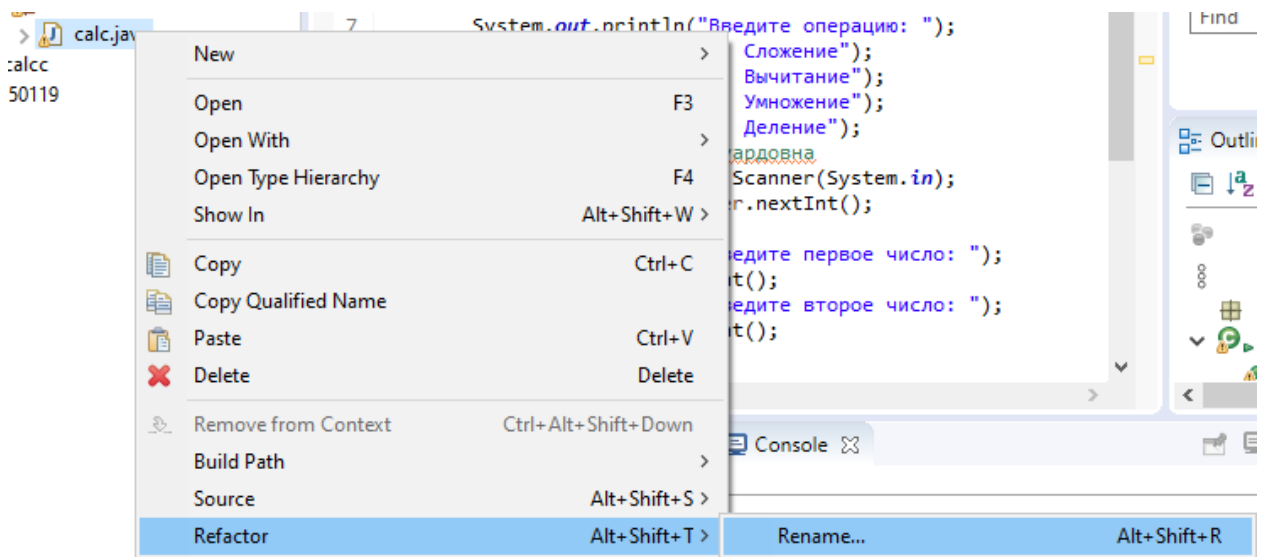


Рисунок. 71. Выбор Rename

После этого появится диалоговое окно, в котором нужно записать новое имя класса, как показано на рисунке 72.

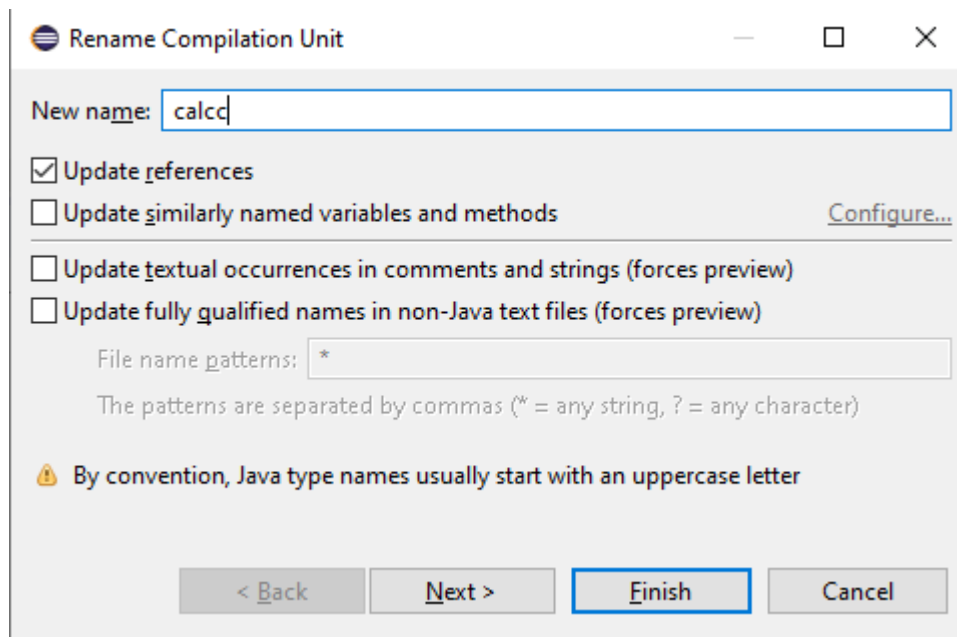


Рисунок. 72. Изменение имени класса

После появления предупреждающее о возможных проблемах в проекте, как показано на рисунке 73.

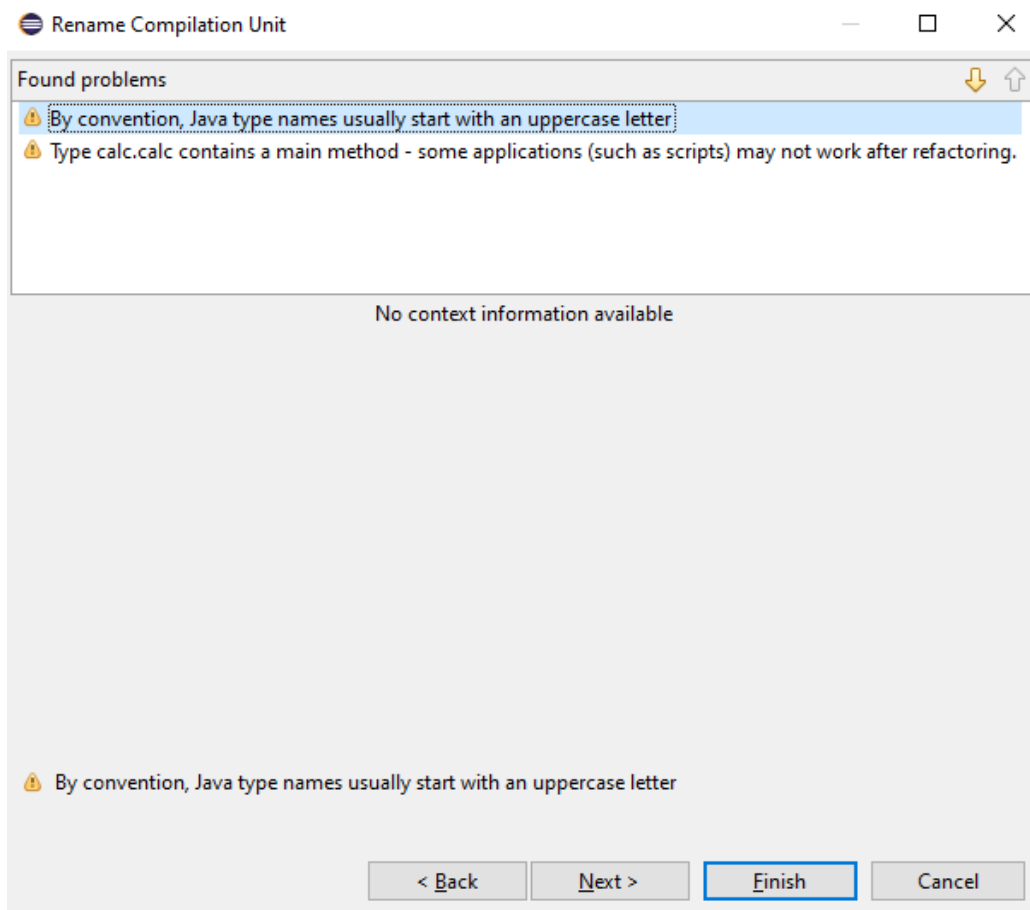


Рисунок. 73. Диалоговое окно предупреждения

Далее нужно переименовать переменную, выделив название, нажать правой кнопкой мыши, затем Rename, как показано на рисунке 74.

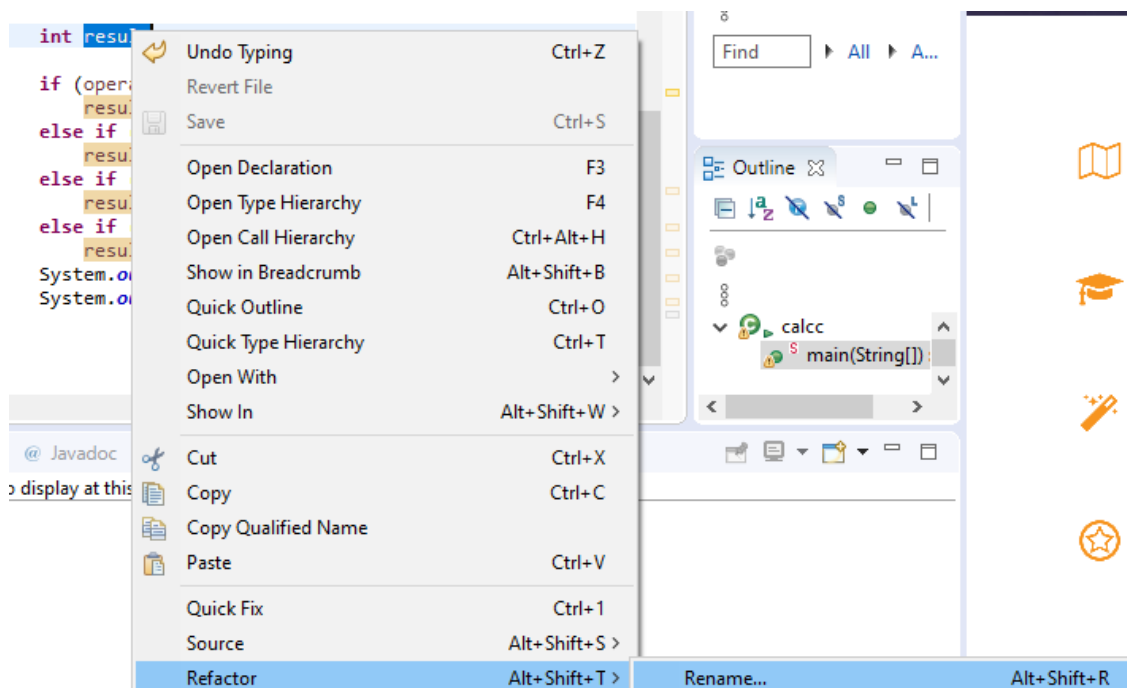


Рисунок. 74. Выбор Rename



После этого надо ввести новое название переменной и после этого нажать Enter (рисунок 93).

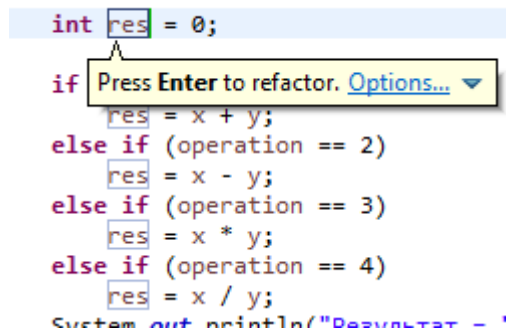


Рисунок. 75. Изменение названия переменной

## Создание нового пакета

Для того, чтобы создать новый пакет нужно выбрать проект, затем New, Package, как показано на рисунке 76.

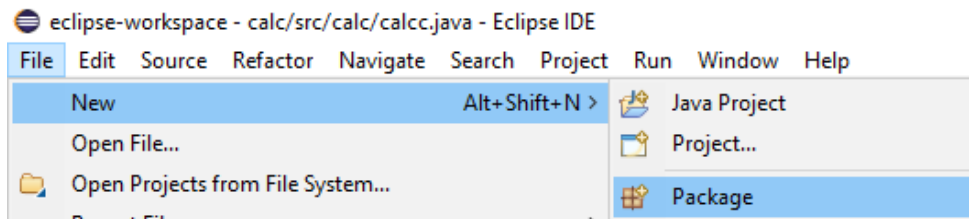


Рисунок. 76. Выбор создания нового пакета

После этого будет открыто диалоговое окно, в котором нужно ввести название нового пакета.

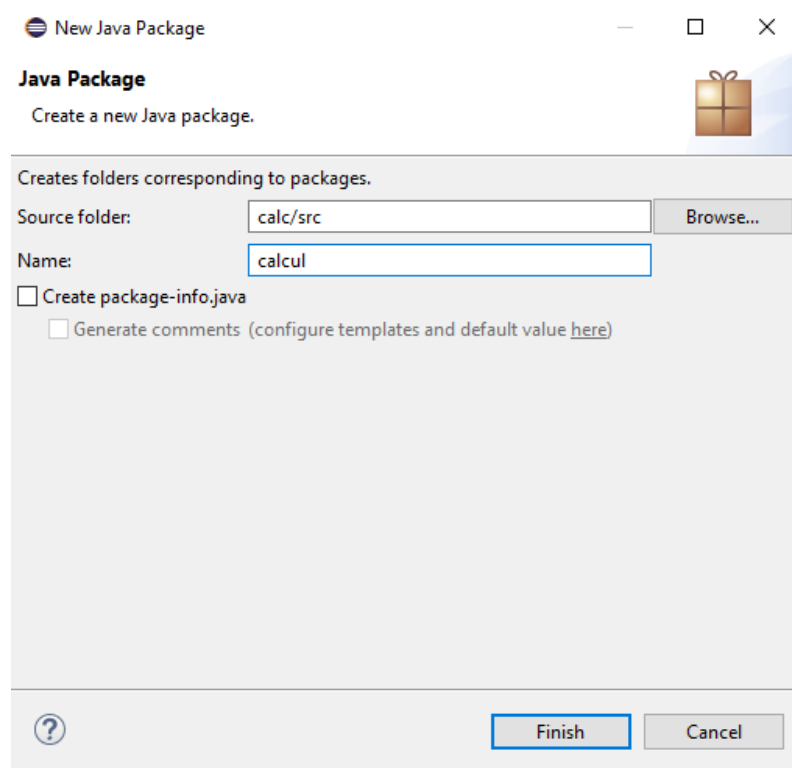


Рисунок. 77. Диалоговое окно создания пакета

После этого новый пакет будет создан.

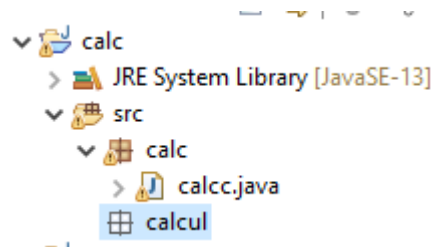


Рисунок. 78. Новый пакет создан

## Перемещение класса из одного пакета в другой

Для того, чтобы переместить класс из одного пакета в другой, требуется нажать правой кнопкой мыши на название класса, потом Refactor, Move, как показано на рисунке 79.

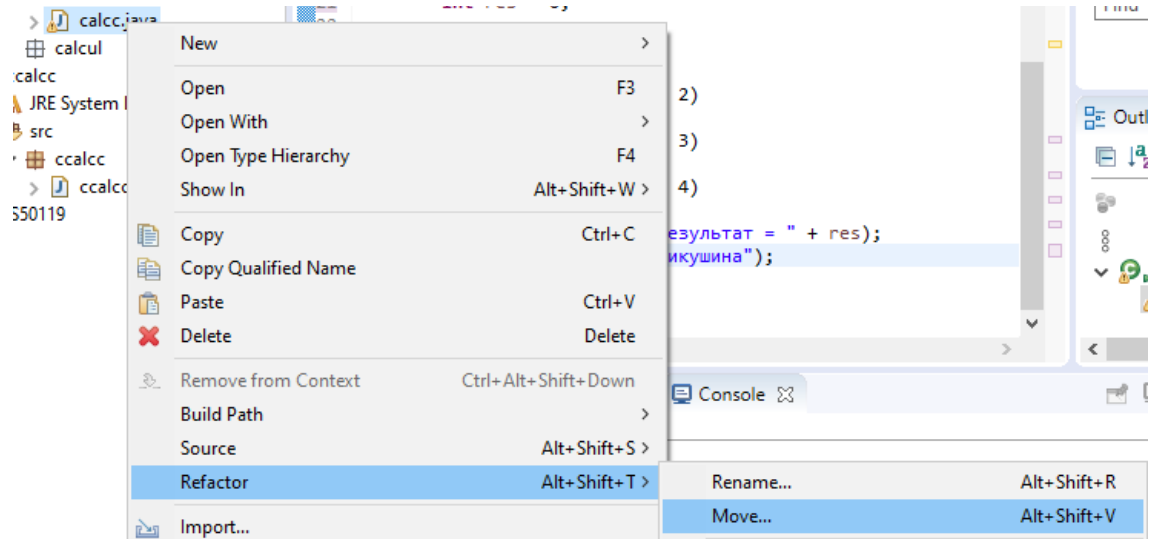


Рисунок. 79. Выбор Move

После этого будет открыто диалоговое окно, в котором нужно выбрать пакет, в который будет перенесен класс, как показано на рисунке 80.

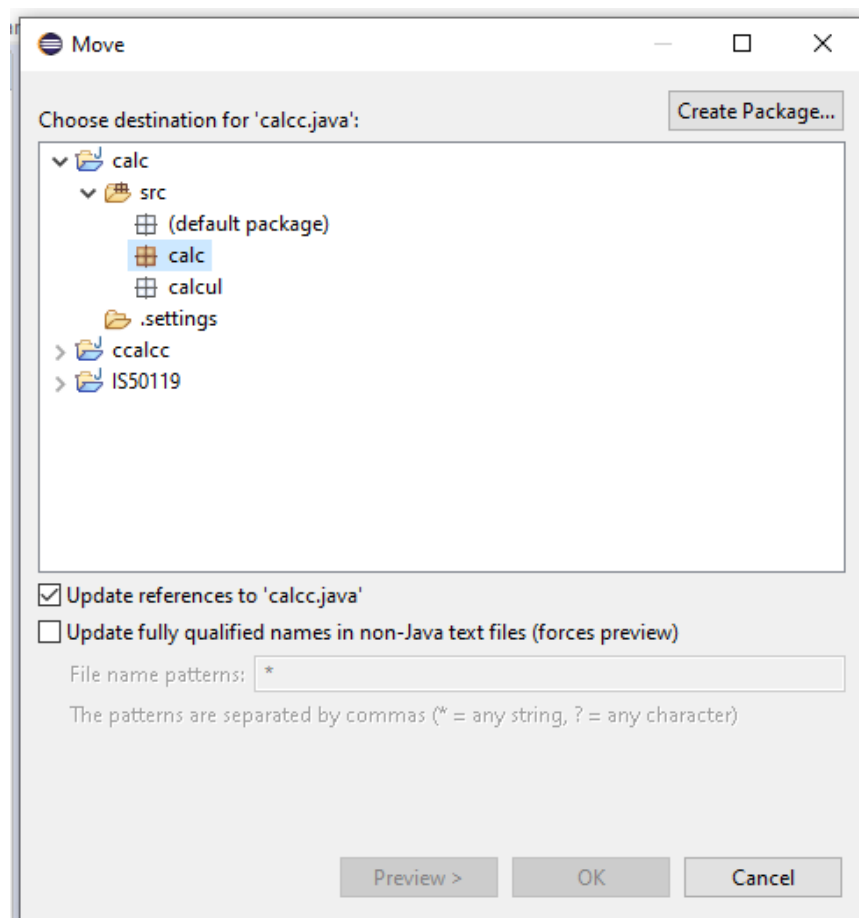
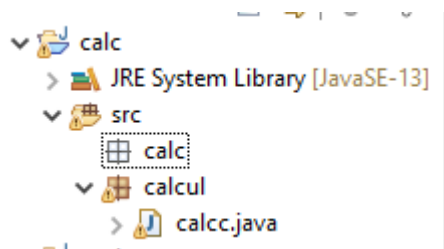


Рисунок. 80. Выбор пакета

На рисунке 81 можно увидеть, что класс был перенесен успешно.



*Рисунок. 81. Класс перенесен*

## Экспорт проекта в архив zip

Для экспорта необходимо выбрать Export, как показано на рисунке 82.

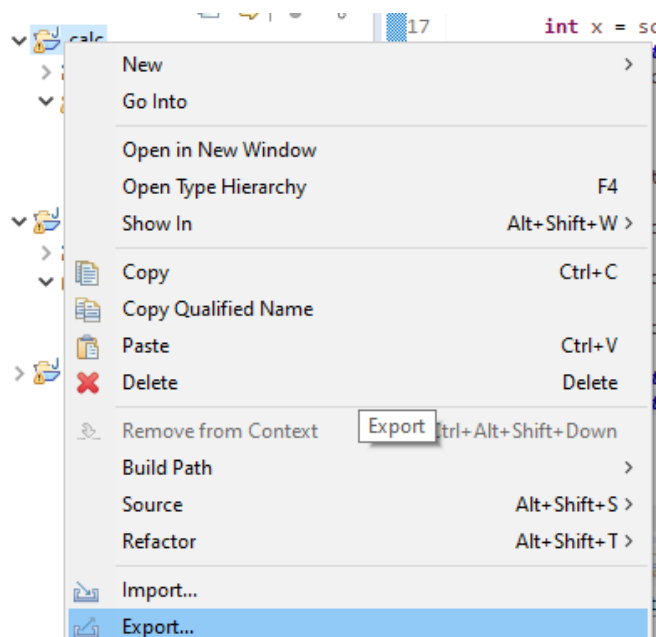


Рисунок. 82. Выбор Export

После чего будет открыто диалоговое окно, в котором нужно выбрать General -> Archive File.

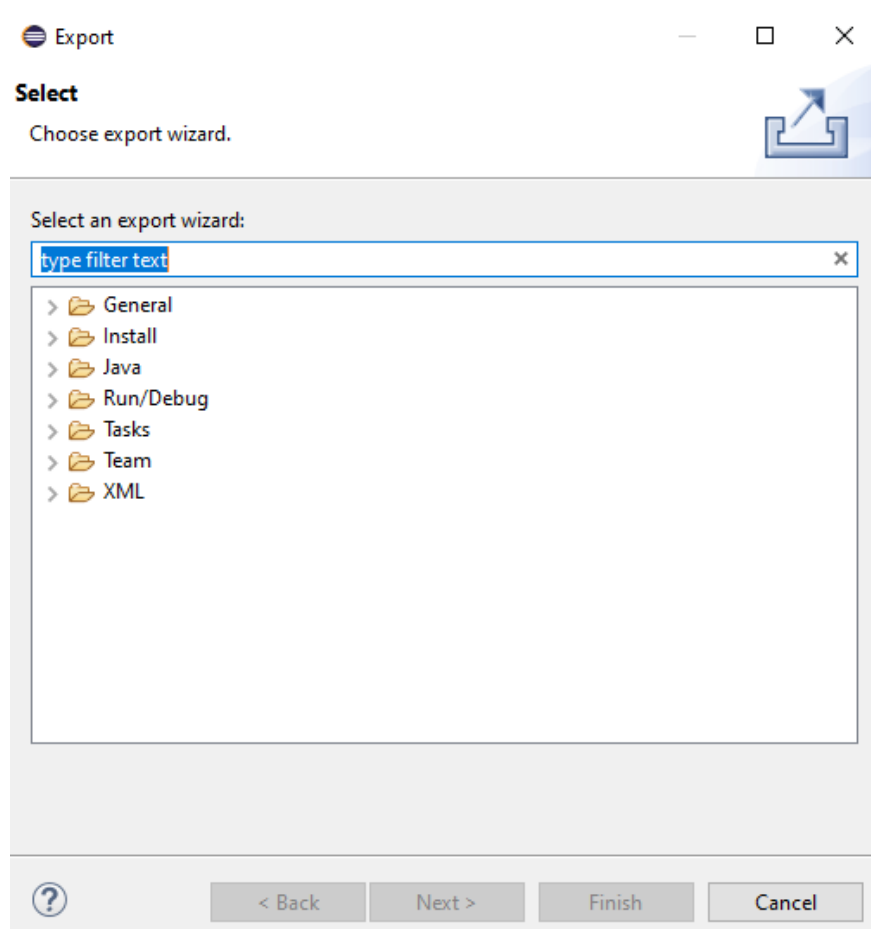


Рисунок. 83. Выбор Archive File

После этого необходимо выбрать путь и название создаваемого архива, как показано на рисунке 84.

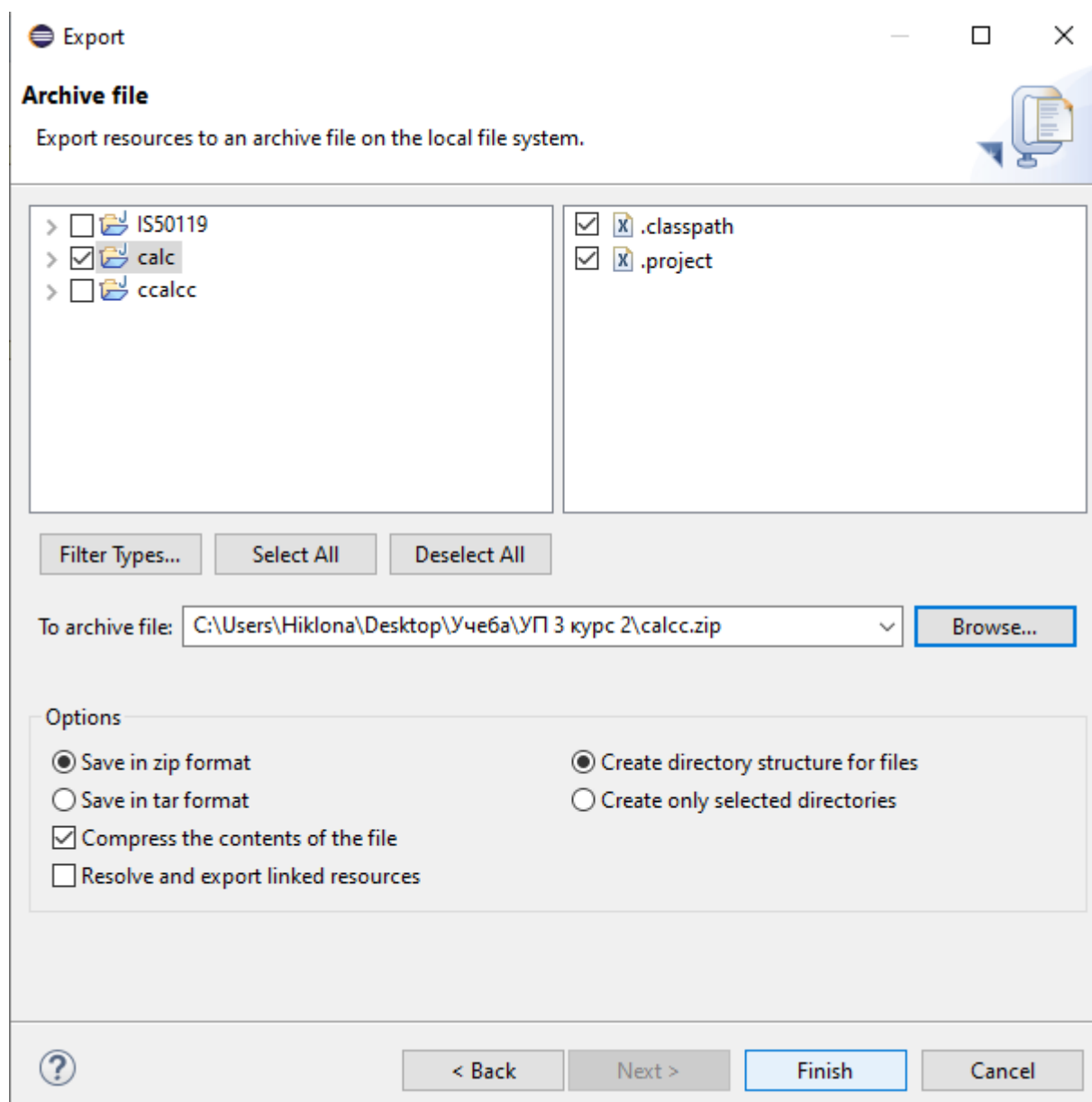


Рисунок. 84. Выбор пути

На рисунке 85 показано, что архив был создан успешно.

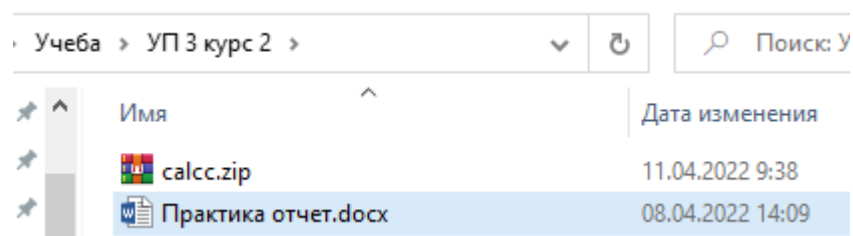


Рисунок. 85. Созданный архив

## Удаление проекта

Для удаления проекта необходимо выбрать в контекстном меню Delete, как показано на рисунке 86.

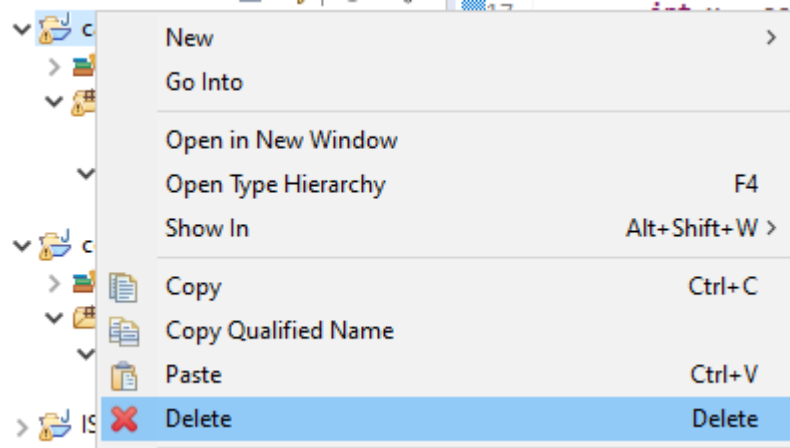


Рисунок. 86. Выбор Delete

В открывшемся окне необходимо установить галочку, как показано на рисунке 87.

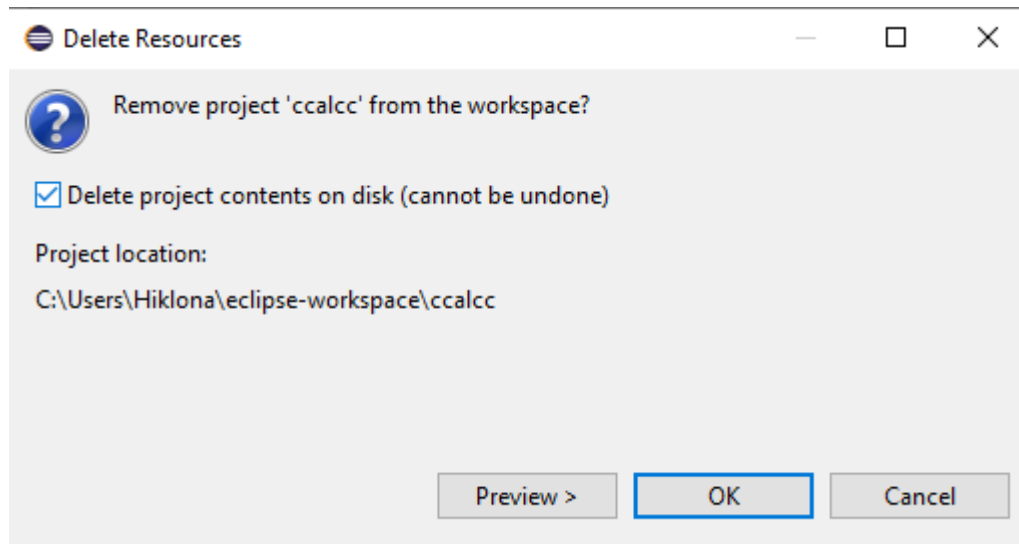


Рисунок. 87. Диалоговое окно удаления

## Импорт проекта из архива zip

Для импорта проекта необходимо в программе нажать File, а затем Import, как показано на рисунке 88.

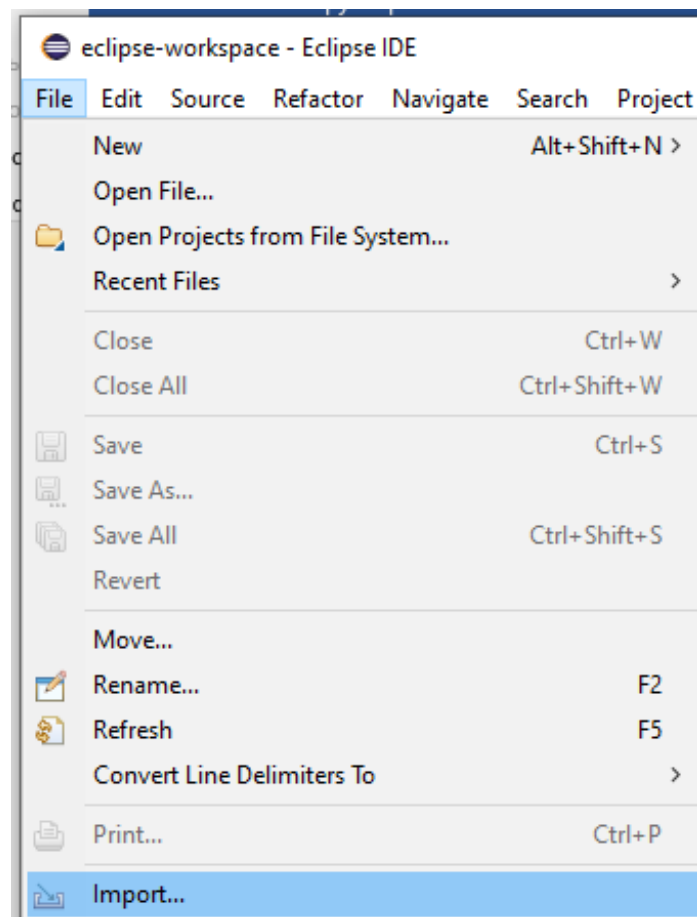


Рисунок. 88. Импорт

После этого в открывшемся окне нужно выбрать General, Existing project into Workspace, как показано на рисунке 89.

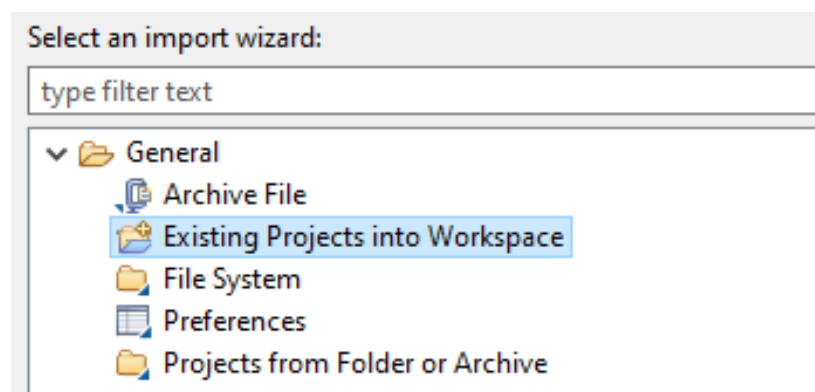


Рисунок. 89. Выбор импорта существующего проекта



Далее необходимо выбрать путь к архиву с проектом, как показано на рисунке 90.

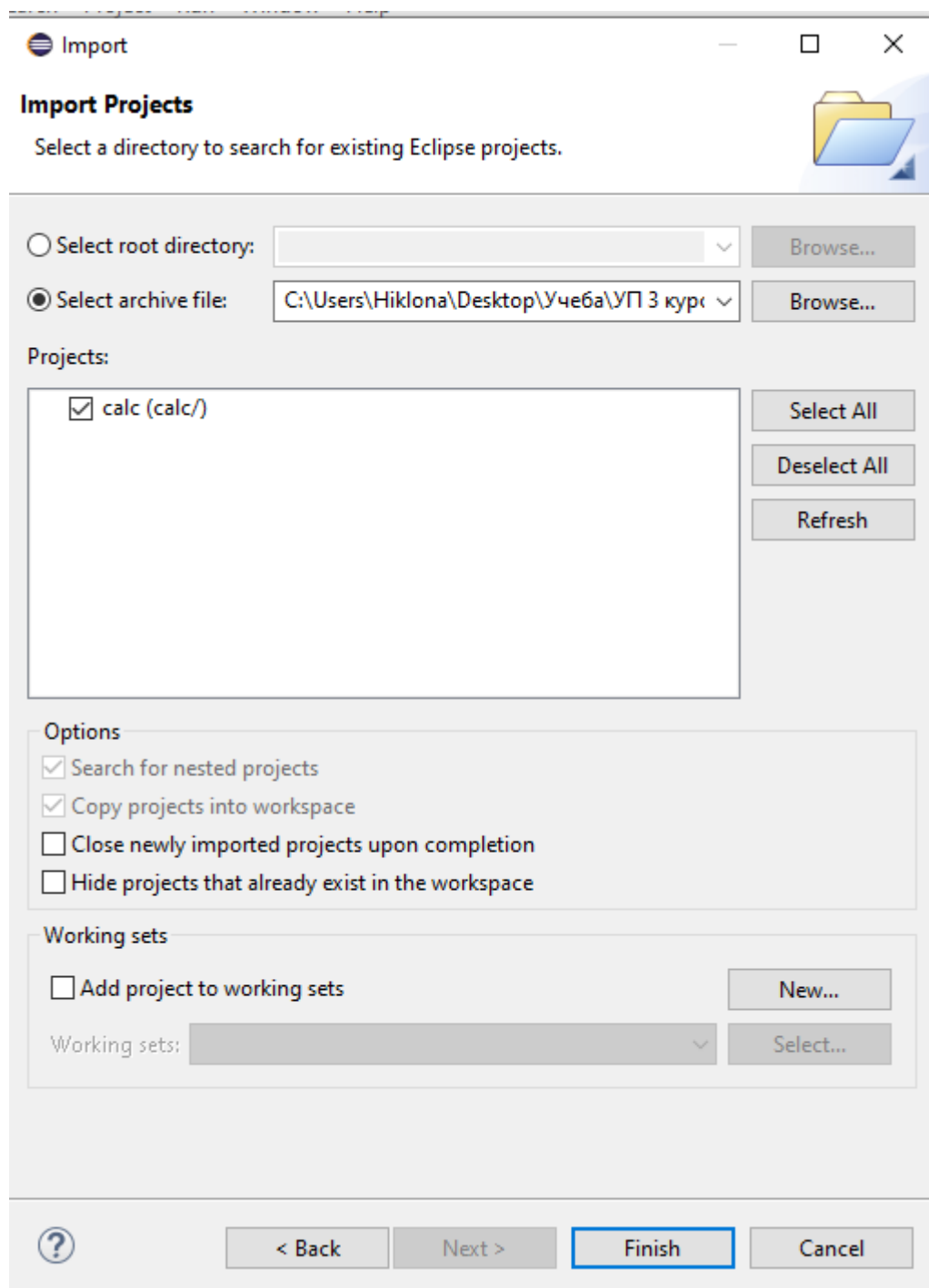


Рисунок. 90. Выбор пути к архиву с проектом

После проект будет импортирован в программу, как показано на рисунке 91.

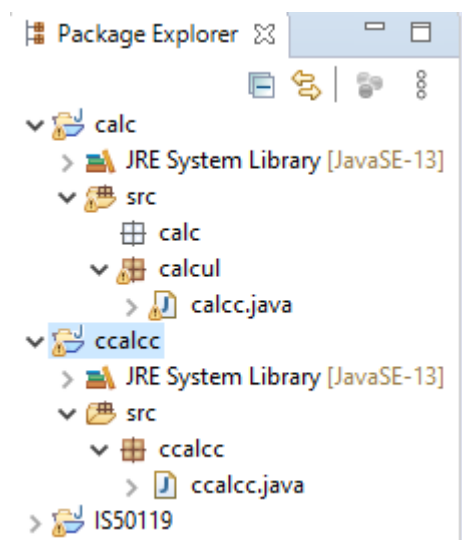


Рисунок. 91. Импортированный проект

## Этап 6.

### Часть 1. Создание овала.

Перед началом работы необходимо создать новый проект, показанный на рисунке 92.

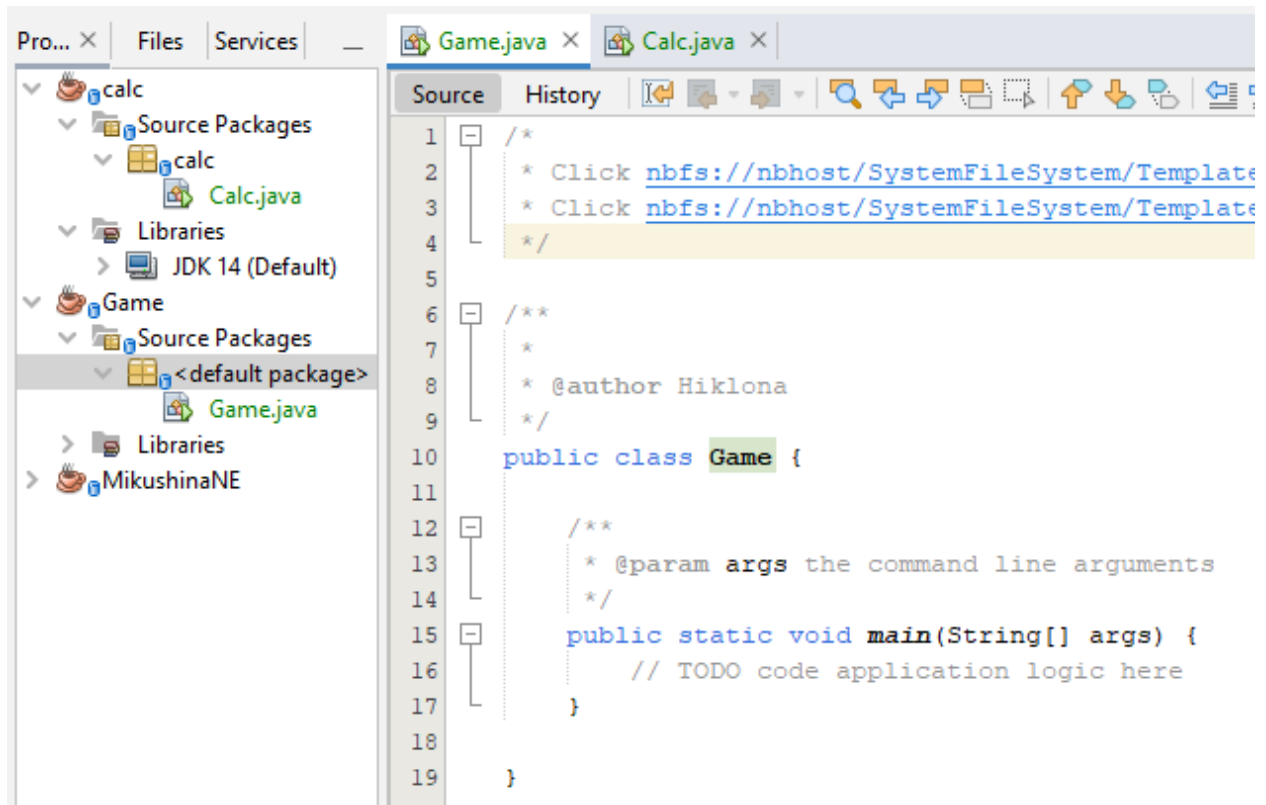


Рисунок. 92. Созданный проект

Добавление строк кода для создания класса Game, строка с импортом библиотеки, созданием переменной и созданием объекта показано на рисунке 93.

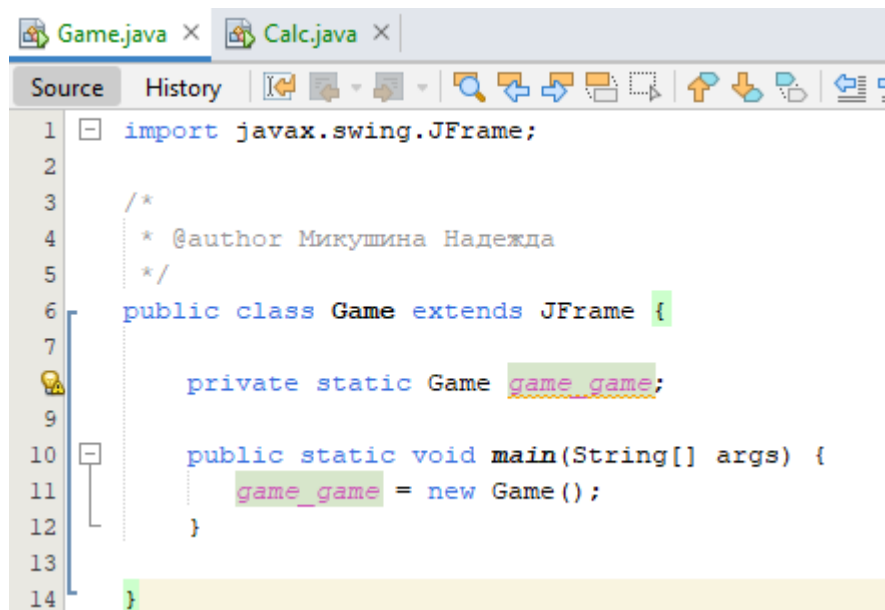


Рисунок. 93. Наполнение кода проекта

Далее необходимо добавить строки кода настройки объекта, как показано на рисунке 94.

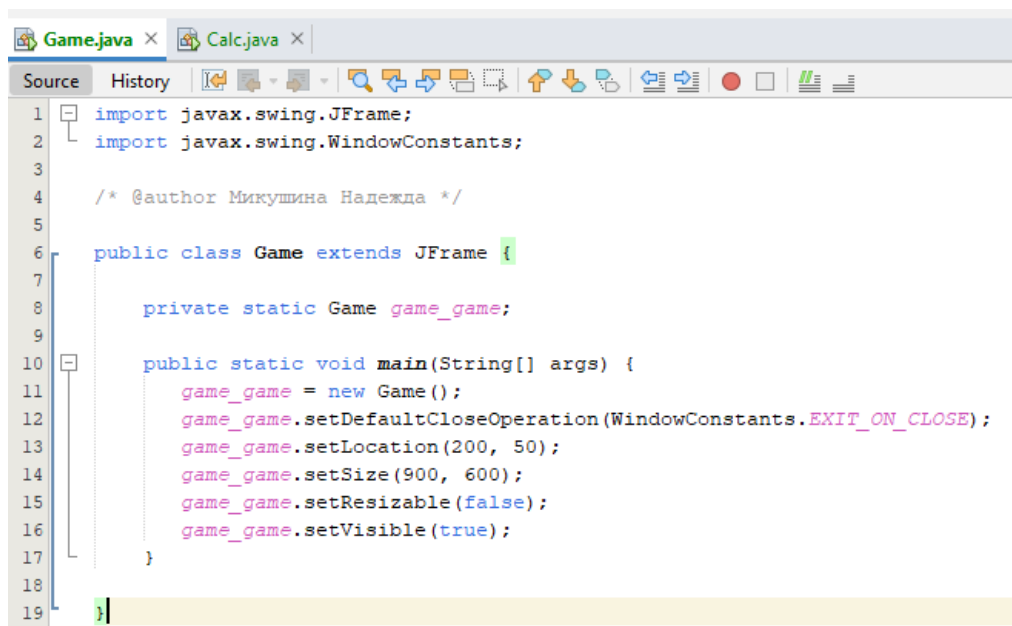


Рисунок. 94. Настройки окна программы

После этого можно запустить проект, как показано на рисунке 95.

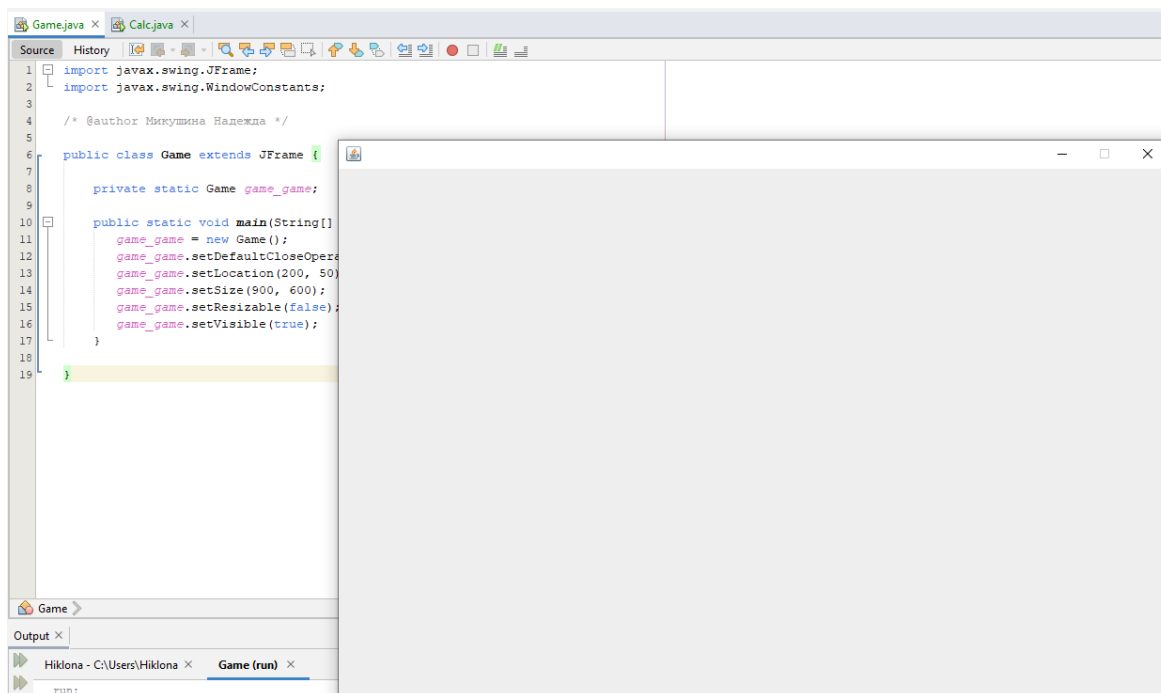


Рисунок. 95. Окно проекта

После необходимо создать методы onRepaint и GameField, как показано на рисунке 96.

```

/* @author Микушина Надежда */

public class Game extends JFrame {

    private static Game game_game;

    public static void main(String[] args) {
        game_game = new Game();
        game_game.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        game_game.setLocation(200, 50);
        game_game.setSize(900, 600);
        game_game.setResizable(false);
        GameField game_field = new GameField();
        game_game.add(game_field);
        game_game.setVisible(true);
    }

    public static void onRepaint(Graphics g) {

    }

    public static class GameField extends JPanel {

        @Override
        protected void paintComponent(Graphics g) {

            super.paint(g);
            onRepaint(g);
        }
    }
}
  
```

Рисунок. 96. Создание методов

После этого можно написать код для пробной отрисовки овала, как показано на рисунке 97.

```

/* @author Микушина Надежда */

public class Game extends JFrame {

    private static Game game_game;

    public static void main(String[] args) {
        game_game = new Game();
        game_game.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        game_game.setLocation(200, 50);
        game_game.setSize(900, 600);
        game_game.setResizable(false);
        GameField game_field = new GameField();
        game_game.add(game_field);
        game_game.setVisible(true);
    }

    public static void onRepaint(Graphics g){
        g.fillOval(10, 10, 200, 100);
    }
}

```

Рисунок. 97. Отрисовка овала

На рисунке 98 показан результат работы и отрисованный овал.

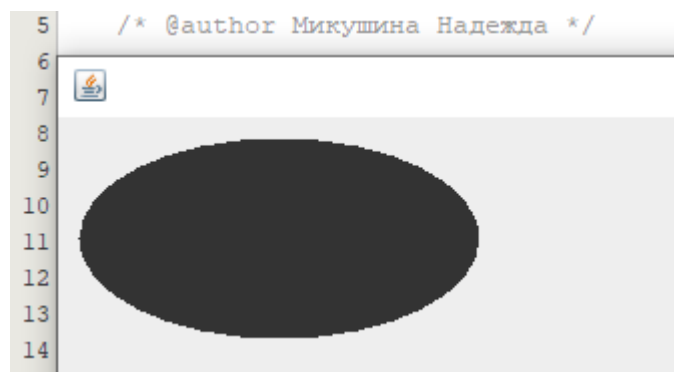


Рисунок. 98. Результат.

## Часть 2. Работа с изображениями

Перед началом работы в папку с проектом необходимо добавить картинки, как показано на рисунке 99.

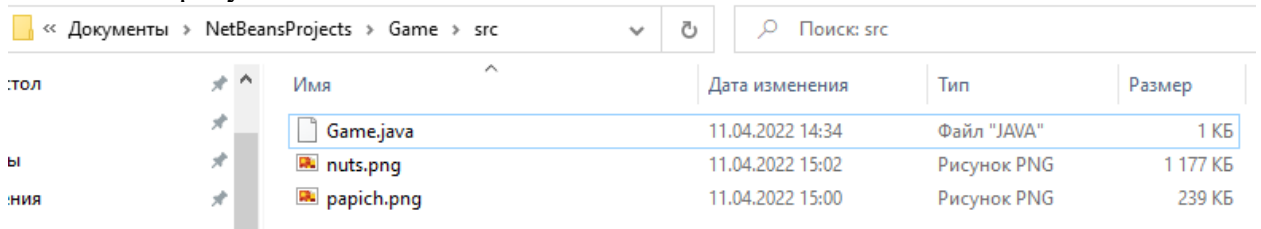


Рисунок. 99. Добавление картинок

После этого нужно создать переменные для картинок, как показано на рисунке 100, после чего загрузить их.

```
public class Game extends JFrame {  
  
    private static Game game_game;  
    private static Image papich;  
    private static Image nuts;
```

Рисунок. 100. Новые переменные

```
public static void main(String[] args) throws IOException {  
    papich = ImageIO.read(Game.class.getResourceAsStream("papich.png"));  
    nuts = ImageIO.read(Game.class.getResourceAsStream("nuts.png"));
```

Рисунок. 101. Загрузка картинок

Далее нужно добавить код для отрисовки картинок в проекте, как показано на рисунке 102.

```
private static void onRepaint(Graphics g){  
    g.drawImage(nuts, 100, 100, null);  
    g.drawImage(papich, 300, 300, null);  
}
```

Рисунок. 102. Расположение картинок

Запущенный проект с отрисованными картинками показан на рисунке 103.



Рисунок. 103. Запущенный проект с отрисованными картинками

После чего необходимо закомментировать код с отрисовкой падающего объекта, как показано на рисунке 104.

```
private static void onRepaint(Graphics g) {
    //g.drawImage(nuts, 100, 100, null);
    g.drawImage(papich, 300, 300, null);
}
```

Рисунок. 104. Комментарий

Далее нужно добавить переменные координат точек углов падающего объекта.

```
public class Game extends JFrame {

    private static Game game_game;
    private static Image papich;
    private static Image nuts;
    private static float drop_left = 200;
    private static float drop_top = 200;
```

Рисунок. 105. Координаты точек углов

На рисунке 106 координаты отрисовки заменены на переменные.

```
private static void onRepaint(Graphics g) {
    g.drawImage(papich, 0, 0, null);
    g.drawImage(nuts, (int) drop_left, (int) drop_top, null);
    g.drawImage(end, 300, 300, null);
}
```

Рисунок. 106. Изменение координат на переменные

Результат, который будет после запуска программы изображен на рисунке 107.





Рисунок. 107. Результат

После нужно изменить код в некоторых местах, как показано на рисунках 108-115.

```
private static class GameField extends JPanel{

@Override
protected void paintComponent(Graphics g){

    super.paintComponent(g);
    onRepaint(g);
    repaint();|
}
}
```

Рисунок. 108. Дополнение кода метода отрисовки

```
private static Game game_game;
private static long last_frame_time;|
private static Image papich;
private static Image nuts;
private static Image end;
private static float drop_left = 200;
private static float drop_top = 200;
```

Рисунок. 109. Добавление переменной для времени

```

public static void main(String[] args) throws IOException {
    papich = ImageIO.read(Game.class.getResourceAsStream("papich.
    nuts = ImageIO.read(Game.class.getResourceAsStream("nuts.png"
    end = ImageIO.read(Game.class.getResourceAsStream("end.png"))
    game_game = new Game();
    game_game.setDefaultCloseOperation(WindowConstants.EXIT_ON_CI
    game_game.setLocation(200, 50);
    game_game.setSize(1200, 800);
    game_game.setResizable(false);
    last_frame_time = System.nanoTime();
    GameField game_field = new GameField();
    game_game.add(game_field);
    game_game.setVisible(true);
}

```

Рисунок. 110. Присвоение значения времени переменной

```

private static void onRepaint(Graphics g){
    long current_time = System.nanoTime();
    float delt_time = (current_time - last_frame_time) * 0.000000001f;
    last_frame_time = current_time;
}

```

Рисунок. 111. Написание игрового цикла

```

private static Game game_game;
private static long last_frame_time;
private static Image papich;
private static Image nuts;
private static Image end;
private static float drop_left = 200;
private static float drop_top = 200;
private static float drop_v = 200;

```

Рисунок. 112. Добавление переменной скорости



Рисунок. 113. Запущенная программа

```

private static Game game_game;
private static long last_frame_time;
private static Image papich;
private static Image nuts;
private static Image end;
private static float drop_left = 200;
private static float drop_top = -100;
private static float drop_v = 200; |

```

Рисунок. 114. Изменение начального значения переменной верхней координаты



Рисунок. 115. Запущенная программа

### Часть 3. Анимация.

Далее необходимо сделать условие отрисовки падающего объекта, как показано на рисунке 116.

```
private static void onRepaint(Graphics g){
    long current_time = System.nanoTime();
    float delt_time = (current_time - last_frame_time) * 0.000000001f;
    last_frame_time = current_time;
    g.drawImage(papich, 0, 0, null);
    g.drawImage(nuts, (int) drop_left, (int) drop_top, null);
    if(drop_top > game_game.getHeight()) g.drawImage(end, 210, 150, null);
}
```

Рисунок. 116. Условие отрисовки падающего объекта

Код обработчика события нажатия на падающий объект показан на рисунке 117.

```
GameField game_field = new GameField();
game_field.addMouseListener(new MouseAdapter() {
    @Override
    public void mousePressed(MouseEvent e) {
        int x = e.getX();
        int y = e.getY();
        float drop_right = drop_left + end.getWidth(null);
        float drop_bottom = drop_top + end.getHeight(null);
        boolean is_drop = x >= drop_left && x <= drop_right && y >= drop_top && y <= drop_bottom;

        if (is_drop) {
            drop_top = -100;
            drop_left = (int) (Math.random() * (game_field.getWidth() - end.getWidth(null)));
            drop_v = drop_v + 10;
            score++;
            game_game.setTitle("Score: " + score);
        }
    }
});
```

Рисунок. 117. Обработчик события нажатия на кнопку мыши

На рисунке 118 показано добавление количества очков.

```
public class Game extends JFrame {

    private static Game game_game;
    private static long last_frame_time;
    private static Image papich;
    private static Image nuts;
    private static Image end;
    private static float drop_left = 200;
    private static float drop_top = -100;
    private static float drop_v = 200;
    private static int score = 0;
```

Рисунок. 118. Добавление очков

Код, который увеличивает значения переменной количества очков и отражает его в заголовке окна, показан на рисунке 139.

```
if (is_drop) {  
    drop_top = -100;  
    drop_left = (int) (Math.random() * (game_field.getWidth() - end.getWidth(null)));  
    drop_v = drop_v + 10;  
    score++;  
    game_game.setTitle("Score: " + score);  
}
```

Рисунок. 119. Добавление кода увеличения очков

## Готовая программа



Рисунок. 120. Программа в работе

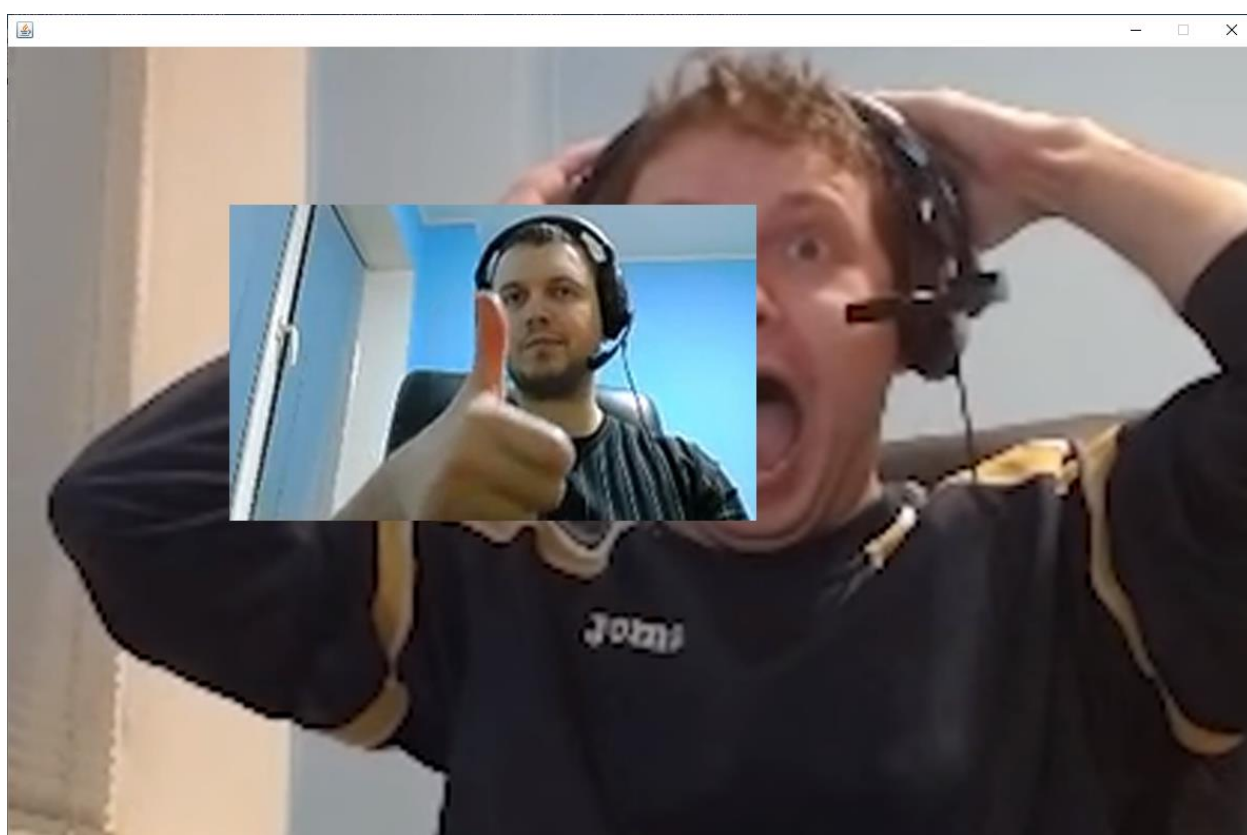


Рисунок. 121.Окно при завершении работы программы



## Готовый код программы

```
import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

/* @author Микушина Надежда */

public class Game extends JFrame {

    private static Game game_game;
    private static long last_frame_time;
    private static Image papich;
    private static Image nuts;
    private static Image end;
    private static float drop_left = 200;
    private static float drop_top = -100;
    private static float drop_v = 200;
    private static int score = 0;

    public static void main(String[] args) throws IOException {
        papich = ImageIO.read(Game.class.getResourceAsStream("papich.png"));
        nuts = ImageIO.read(Game.class.getResourceAsStream("nuts.png"));
        end = ImageIO.read(Game.class.getResourceAsStream("end.png"));
        game_game = new Game();
        game_game.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        game_game.setLocation(200, 50);
        game_game.setSize(1200, 800);
        game_game.setResizable(false);
        last_frame_time = System.nanoTime();
        GameField game_field = new GameField();
        game_field.addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                int x = e.getX();
                int y = e.getY();
            }
        });
    }
}
```

```

float drop_right = drop_left + end.getWidth(null);
float drop_bottom = drop_top + end.getHeight(null);
boolean is_drop = x >= drop_left && x <= drop_right && y >= drop_top && y <= drop_bottom;

if (is_drop) {
    drop_top = -100;
    drop_left = (int) (Math.random() * (game_field.getWidth() - end.getWidth(null)));
    drop_v = drop_v + 10;
    score++;
    game_game.setTitle("Score: " + score);
}
}
});

game_game.add(game_field);
game_game.setVisible(true);
}

private static void onRepaint(Graphics g){
    long current_time = System.nanoTime();
    float delt_time = (current_time - last_frame_time) * 0.000000001f;
    last_frame_time = current_time;
    drop_top = drop_top + drop_v * delt_time;
    g.drawImage(papich, 0, 0, null);
    g.drawImage(nuts, (int) drop_left, (int) drop_top, null);
    if(drop_top > game_game.getHeight()) g.drawImage(end, 210, 150, null);
}

private static class GameField extends JPanel{
    @Override
    protected void paintComponent(Graphics g){

        super.paintComponent(g);
        onRepaint(g);
        repaint();
    }
}
}

```



## Кейс-задание

Проанализировав и повторив действия «этапа 6», вам необходимо создать приложение для ранее выбранной предметной области в определенной тематике.

Требования:

1. Использовать изображение или логотип компании (можно разработать и нарисовать самостоятельно или взять готовый);
2. Реализовать анимацию (показать наличие перемещающихся изображений);
3. Сделать проект в определенной тематике (например, в новогодней, показав падающие снежинки, мигающую огнями новогоднюю ель на примере сайта выбранной компании и т.д.);
4. Подтвердить последовательность выполненных действий скриншотами;
5. Представить готовый программный код.

# GitHub

Перед началом работы необходимо создать удаленный репозиторий на GitHub, как показано на рисунке 122.

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*



HiKlonA ▾

Repository name \*



UP.03.01



Great repository names are short and memorable. Need inspiration? How about [expert-spoon?](#)

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.



Add a README file

This is where you can write a long description for your project. [Learn more.](#)



Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)



Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)



You are creating a private repository in your personal account.

Create repository

Рисунок. 122. Создание репозитория

После чего необходимо перейти в папку с проектами практики и инициализировать новый репозиторий, как показано на рисунке 123.

```

Hiklona@HOME-PC MINGW64 ~/Мои документы/NetBeansProjects (master)
$ cd ..

Hiklona@HOME-PC MINGW64 ~/Мои документы (master)
$ cd NetBeansProjects/

Hiklona@HOME-PC MINGW64 ~/Мои документы/NetBeansProjects (master)
$ git init
Initialized empty Git repository in C:/Users/Hiklona/Documents/NetBeansProjects/.git/

```

Рисунок. 123. Инициализация проектов

Далее необходимо добавить проекты в индекс и осуществить первый коммит изменений, как показано на рисунках 124 и 125.

```

Hiklona@HOME-PC MINGW64 ~/Мои документы/NetBeansProjects (master)
$ git add calc

Hiklona@HOME-PC MINGW64 ~/Мои документы/NetBeansProjects (master)
$ git commit -m 'calc 1'
[master (root-commit) 66e3ce7] calc 1
9 files changed, 1996 insertions(+)
create mode 100644 calc/build.xml
create mode 100644 calc/manifest.mf
create mode 100644 calc/nbproject/build-impl.xml
create mode 100644 calc/nbproject/genfiles.properties
create mode 100644 calc/nbproject/private/private.properties
create mode 100644 calc/nbproject/private/private.xml
create mode 100644 calc/nbproject/project.properties
create mode 100644 calc/nbproject/project.xml
create mode 100644 calc/src/calc/Calc.java

```

Рисунок. 124. Calc

```

Hiklona@HOME-PC MINGW64 ~/Мои документы/NetBeansProjects (master)
$ git add Game

Hiklona@HOME-PC MINGW64 ~/Мои документы/NetBeansProjects (master)
$ git commit -m 'Game 1'
[master 1be4efc] Game 1
23 files changed, 2058 insertions(+)
create mode 100644 Game/build.xml
create mode 100644 Game/build/classes/.netbeans_automatic_build
create mode 100644 Game/build/classes/.netbeans_update_resources
create mode 100644 Game/build/classes/Game$1.class
create mode 100644 Game/build/classes/Game$GameField.class
create mode 100644 Game/build/classes/Game.class
create mode 100644 Game/build/classes/Game.rs
create mode 100644 Game/build/classes/end.png
create mode 100644 Game/build/classes/nuts.png
create mode 100644 Game/build/classes/papich.png
create mode 100644 Game/build/classes/Без имени-1.psd
create mode 100644 Game/manifest.mf
create mode 100644 Game/nbproject/build-impl.xml
create mode 100644 Game/nbproject/genfiles.properties
create mode 100644 Game/nbproject/private/private.properties
create mode 100644 Game/nbproject/private/private.xml
create mode 100644 Game/nbproject/project.properties
create mode 100644 Game/nbproject/project.xml
create mode 100644 Game/src/Game.java
create mode 100644 Game/src/end.png
create mode 100644 Game/src/nuts.png
create mode 100644 Game/src/papich.png
create mode 100644 Game/src/Без имени-1.psd

```

Рисунок. 125. Game

После чего нужно добавить локальному репозиторию ссылку на удаленный репозиторий, затем можно сделать команду push в удаленный репозиторий `git push origin master`, тем самым отправив локальную ветку master на сервер origin, как показано на рисунке 126.

```

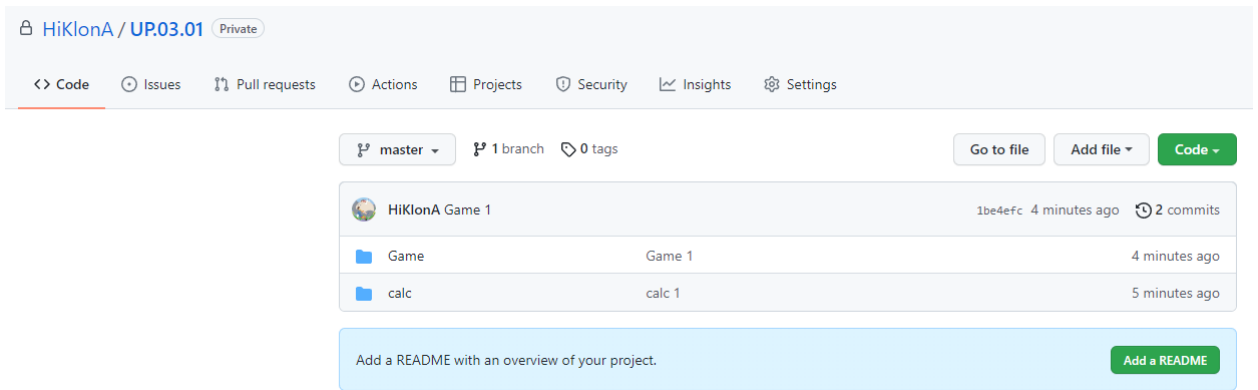
Hiklona@HOME-PC MINGW64 ~/Мои документы/NetBeansProjects (master)
$ git remote add origin https://github.com/Hiklona/UP.03.01.git

Hiklona@HOME-PC MINGW64 ~/Мои документы/NetBeansProjects (master)
$ git push origin master
Enumerating objects: 40, done.
Counting objects: 100% (40/40), done.
Delta compression using up to 8 threads
Compressing objects: 100% (34/34), done.
Writing objects: 100% (40/40), 1.61 MiB | 159.00 KiB/s, done.
Total 40 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/Hiklona/UP.03.01.git
* [new branch]      master -> master

```

Рисунок. 126. Отправка на удаленный репозиторий

Теперь добавленные файлы можно увидеть на GitHub.



*Рисунок. 127. GitHub*