

ЛАБОРАТОРНАЯ РАБОТА № 1

«Знакомство с языком программирования Java»

Цель: получить представление о написании программ на языке программирования Java с использованием командной строки, интегрированной среды разработки, а также решение задач на сайте timus.

Учебные вопросы

1 Установка необходимого программного обеспечения	2
2 Компиляция и запуск java программ из командной строки	3
2.1 Вывод на консоль	3
2.2 Ввод с консоли	6
3 Компиляция и запуск java программ в IntelliJ IDEA Community	10
4 Задания для самостоятельной работы.....	18
5 Описание результата выполнения лабораторной работы.....	19

Полезные ссылки:

1. [Как установить JDK?](#)
2. [Как добавить Java в PATH?](#)
3. [Как установить IntelliJ IDEA Community?](#)
4. [Как создать простую программу в IntelliJ IDEA Community?](#)
5. [Как скачать git и зарегистрироваться на github.com?](#)
6. [Как создать первый проект в IDEA с git?](#)
7. [Как создать один проект с ЛР и с git?](#)
8. [Как решать задачи с timus?](#)

1 Установка необходимого программного обеспечения

Для того, чтобы была возможность запускать программы на языке программирования java из командной строки, а также писать программы с помощью интегрированной среды разработки необходимо скачать Java Development Kit (Комплект разработчика приложений на Java) с сайта <https://www.oracle.com/>, пример установки доступен по ссылке <https://itlearn.ru/how-to-install-jdk> и необходимо скачать и установить интегрированную среду разработки IntelliJ IDEA Community, как это сделать описано в статье <https://itlearn.ru/how-to-install-intellij-idea-community>.

После того, как вы скачаете файл JDK необходимо его установить, после установки для того, чтобы была возможность вызова из командной строки компилятора **javac** (**Javac** — компилятор языка java) возможно потребуется добавить путь до JDK в переменные среды Path, но в явном добавлении может не быть необходимости, потому что современные пакеты установки делают это в автоматическом режиме. Если добавление в переменные Path все же потребуется, то описание как это сделать приведено в статье <https://www.java.com/ru/download/help/path.html>. Чтобы понять, что добавление в переменные среды Path не требуется вы можете выполнить команду `java -version` после установки JDK, и если не возникает ошибки и в командной строке выводится информация о версии Java, то добавление в переменные среды Path не требуется.

2 Компиляция и запуск java программ из командной строки

2.1 Вывод на консоль

Для компиляции программы в командной строке необходимо вызвать командную строку от имени администратора. Как показано на рисунке 1.

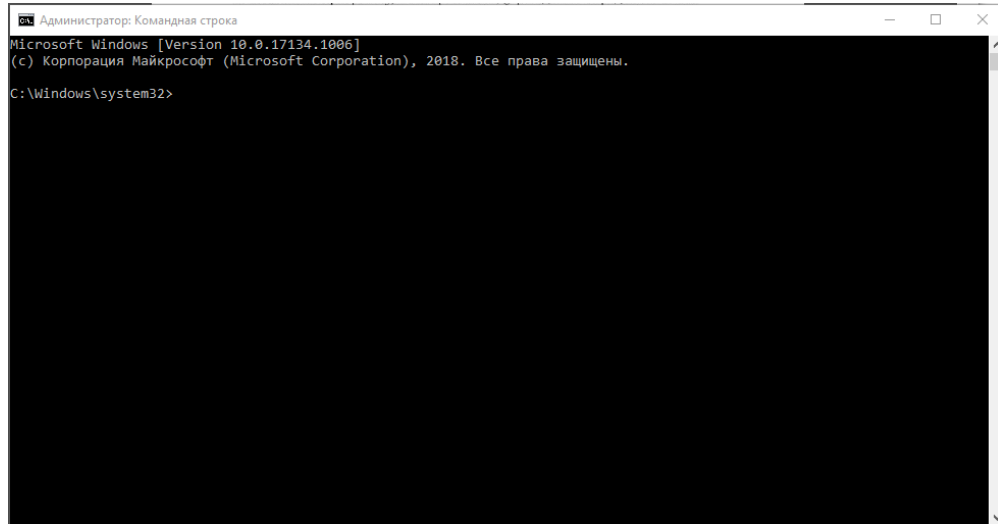


Рисунок 1. Командная строка

Затем необходимо создать папку, в которой будут храниться текстовые файлы с текстом программы. Путь к файлам не должен содержать каталогов с наименованием на кириллице. Например, создайте папку в корне диска C. Назовите её «*java_ex*», *ex* – сокращенное от *examples*. Введите в командную строку команду *cd C:\java_ex*. Как показано на рисунке 2.

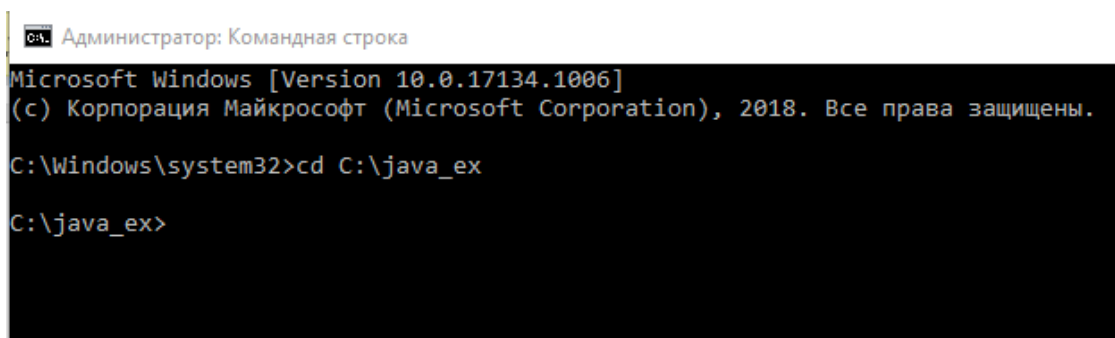


Рисунок 2. Результат выполнения команды *cd C:\java_ex*

После выполнения команды необходимо в каталоге *java_ex* создать текстовый файл «Блокнот», с расширением **.txt*, и назвать его *example1*, далее

изменить расширение файла с txt на java и скопировать в файл код, приведенный на листинге 1.

Листинг 1. Код программы «example1»

```
public class Example1 {  
  
    public static void main(String[] args) {  
  
        System.out.println("Hello world!");  
  
    }  
}
```

Затем вернитесь к командной строке введите в нее следующую команду **javac Example1.java**, где **javac** это обращение к компилятору, а **example1.java** это имя созданного вами текстового файла. Обратите внимание, в каталоге **C:\java_ex** появился скомпилированный файл «example1.class». Теперь чтобы запустить выполнение программы необходимо в командной строке ввести команду **java example1**.

Результат показан на рисунке 3.

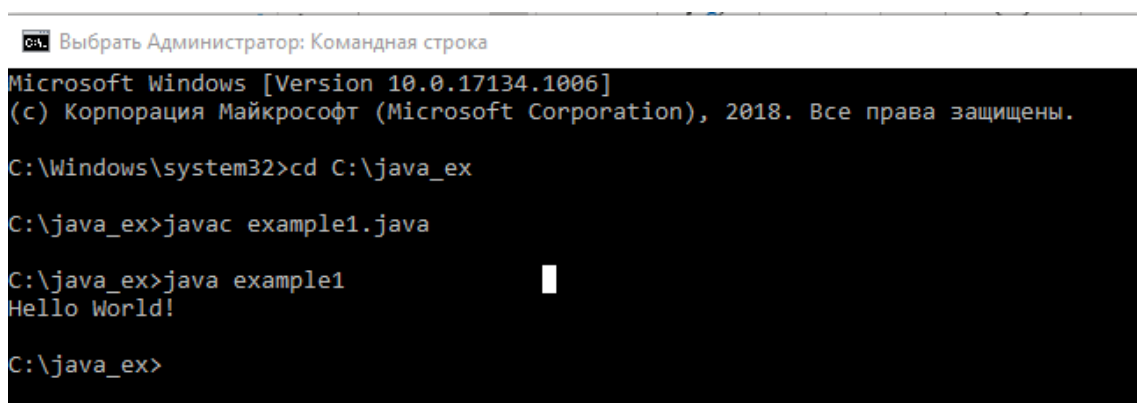


Рисунок 3. Результат выполнения программы example1

На листинге 2 представлена программа, в которой указанное число умножается на 2.

Листинг 2. Код программы «example2»

```
public class Example2 {  
  
    public static void main(String[] args) {  
        int num = 100;  
        System.out.println("num: " + num);  
        num = num * 2;  
  
        System.out.println("num * 2 = " + num);  
    }  
}
```

На листинге 3 показан пример программы с использованием условного оператора if.

Листинг 3. Код программы «example3»

```
public class Example3 {  
    public static void main(String[] args) {  
        int s, d;  
        s = 10;  
        d = 20;  
        if (s < d) {  
            System.out.println("S < D");  
        }  
        s = s * 2;  
        if (s == d) {  
            System.out.println("S = D");  
        }  
        s = s * d;  
        if (s > d) {  
            System.out.println("S > D");  
        }  
    }  
}
```

2.2 Ввод с консоли

Для получения ввода с консоли в классе **System** определен объект **In**. Однако непосредственно через объект **System. In** не очень удобно работать, поэтому, как правило, используют класс **Scanner**, который, в свою очередь использует **System.in**. Пример программы, осуществляющей ввод чисел показан на листинге 4.

Листинг 4. Код программы «example4»

```
import java.util.Scanner;

public class Example4 {
    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        System.out.println("Input a number:");
        int num = in.nextInt();

        System.out.println("Your number" + num);
        in.close();
    }
}
```

Так как класс **Scanner** находится в пакете **java.util**, то мы вначале его импортируем с помощью инструкции **import java.util.Scanner**.

Для создания самого объекта **Scanner** в его конструктор передается объект **System.in**. После этого мы можем получать вводимые значения. Например, в данном случае вначале выводим приглашение к вводу и затем получаем вводимое число в переменную **num**.

Чтобы получить введенное число, используется метод **in.nextInt()**, который возвращает введенное с клавиатуры целочисленное значение. На рисунке показан пример работы программы.

```
C:\java_ex>java example4
Input a number:1
Your number: 1
```

Рисунок 4. Пример работы программы «example4»

Класс Scanner имеет еще ряд методов, которые позволяют получить введенные пользователем значения:

next(): считывает введенную строку до первого пробела

nextLine(): считывает всю введенную строку

nextInt(): считывает введенное число int

nextDouble(): считывает введенное число double

nextBoolean(): считывает значение boolean

nextByte(): считывает введенное число byte

nextFloat(): считывает введенное число float

nextShort(): считывает введенное число short

Например, создадим программу для ввода информации о человеке показанной на листинге 5.

Листинг 5. Код программы «example5»

```
import java.util.Scanner;

public class Example5 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

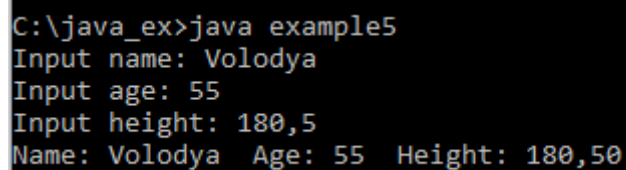
        System.out.println("Input name: ");
        String name = in.nextLine();

        System.out.println("Input age: ");
        int age = in.nextInt();

        System.out.println("Input height: ");
        float height = in.nextFloat();

        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Height: " + height);
        in.close();
    }
}
```

Здесь последовательно вводятся данные типов **string**, **int**, **float** и потом все введенные данные вместе выводятся на консоль. На рисунке показан пример работы программы:



```
C:\java_ex>java example5
Input name: Volodya
Input age: 55
Input height: 180,5
Name: Volodya Age: 55 Height: 180,50
```

Рисунок 5. Пример работы программы

Обратите внимание для ввода значения типа **float** (то же самое относится к типу **double**) применяется число "180,5", где разделителем является запятая, а не "180.5", где разделителем является точка. В данном случае все зависит от текущей языковой локализации системы. В моем случае русскоязычная локализация, соответственно вводить необходимо числа, где разделителем является запятая. То же самое касается многих других

локализаций, например, немецкой, французской и т.д., где применяется запятая.

3 Компиляция и запуск java программ в IntelliJ IDEA Community

После установки IntelliJ IDEA создайте простую программу <https://itlearn.ru/how-to-create-a-simple-program>.

Далее необходимо реализовать программы, описанные в разделе «Компиляция и запуск java программ из командной строки» в среде разработки.

Создайте новый проект или продолжите работу в уже созданном проекте. Создание нового проекта показано на рисунке 6.

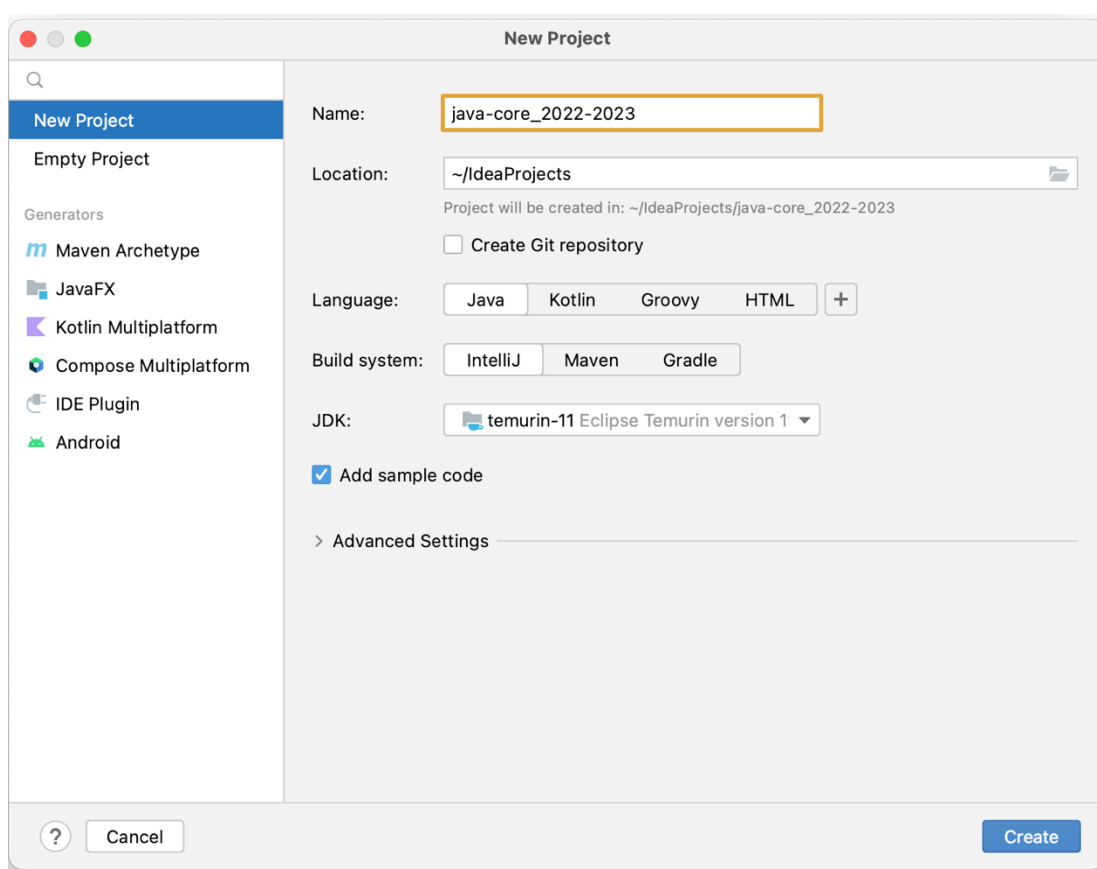


Рисунок 6. Создание нового проекта

Для того, чтобы удобнее было хранить код, все лабораторные работы рекомендуется выполнять в одном проекте, также в этом проекте можно хранить решенные задачи с сайта timus. Для логического разделения лабораторных работ рекомендую создавать отдельные пакеты с наименованием lr1 или laba1.

Создание пакета для первой лабораторной работы показано на рисунках 7 и 8.

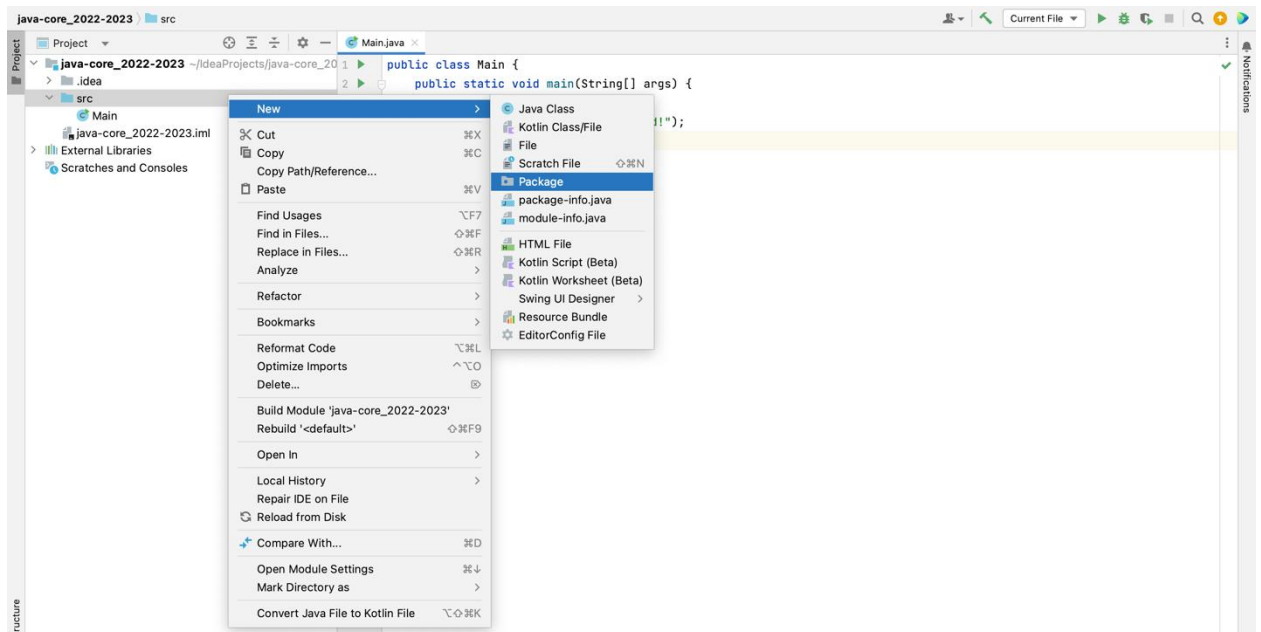


Рисунок 7. Создание пакета



Рисунок 8. Ввод названия для создаваемого пакета

После создания пакета необходимо создать класс с названием Example1. Для того, чтобы класс был создан в нужном пакете необходимо выбрать пакет и нажать правую клавишу мыши.

Создание класса с названием Example1 показано на рисунках 9 и 10.

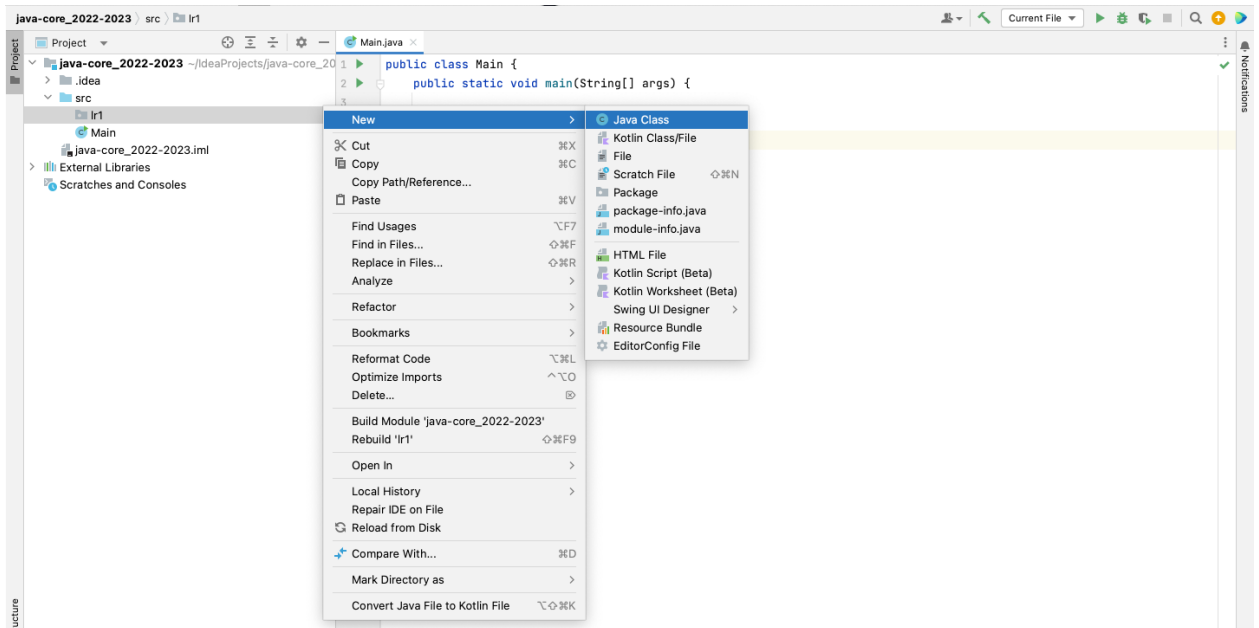


Рисунок 9. Создание класса

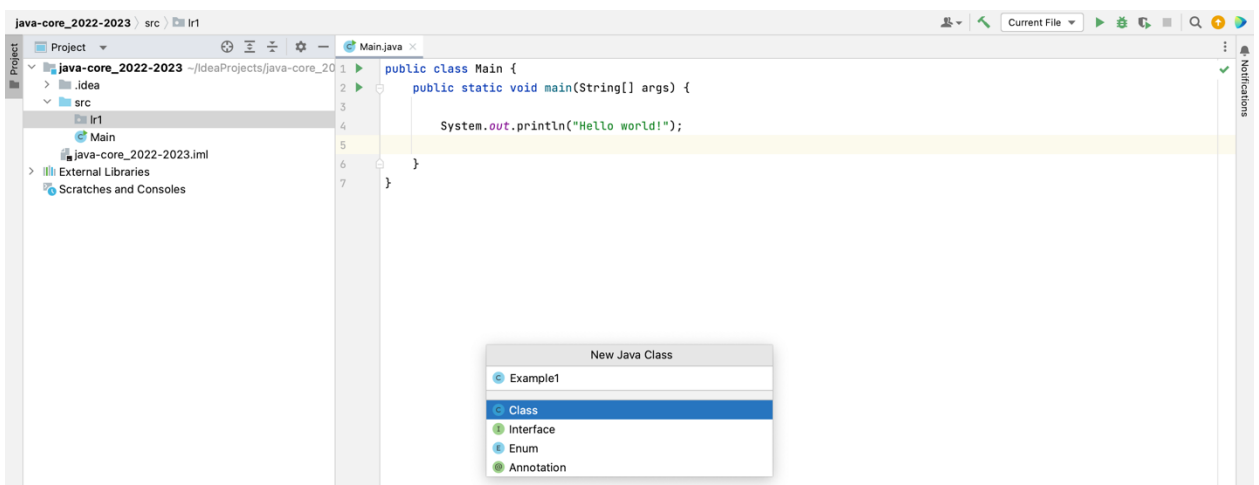
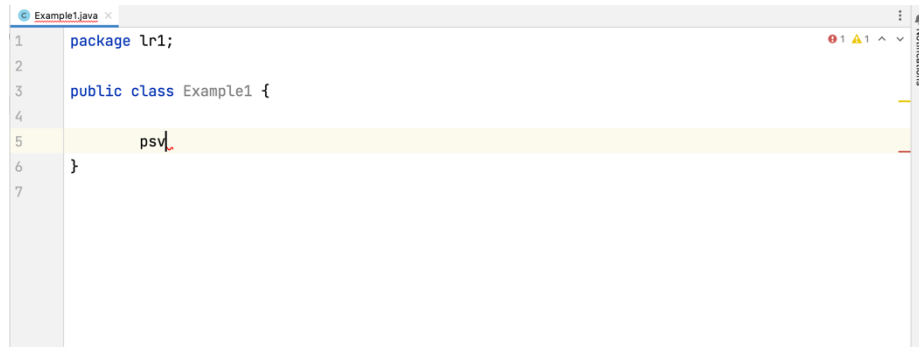


Рисунок 10. Ввод названия для создаваемого класса

В созданном классе введите `psv` и интеллектуальная подсказка IDEA предложит продолжить предложение и автоматически сгенерируется код главного метода.

Автоматическая генерация кода главного метода показана на рисунках 11 и 12.



```
1 package lr1;
2
3 public class Example1 {
4
5     psv
6 }
7
```

The screenshot shows a code editor window titled 'Example1.java'. The code contains a package declaration 'package lr1;', an opening curly brace for a public class 'Example1', and a line with 'psv' followed by a red squiggly line, indicating an error or an incomplete statement. The line number 5 is highlighted.

Рисунок 11. Автоматическая генерация главного метода

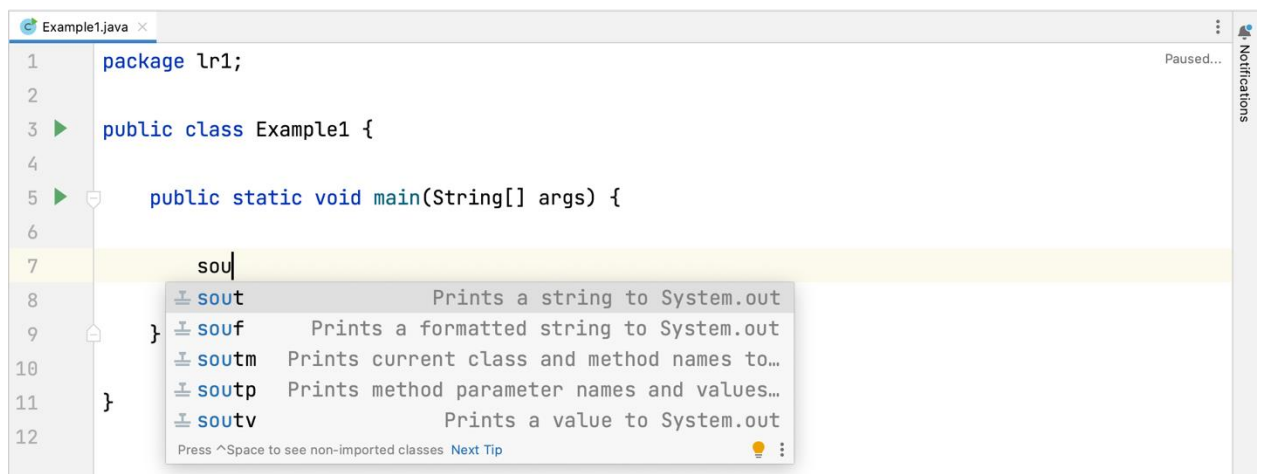


```
1 package lr1;
2
3 public class Example1 {
4
5     public static void main(String[] args) {
6
7     }
8 }
9
```

The screenshot shows the same code editor window after the automatic generation of the main method. The code now includes a 'public static void main(String[] args) {' block with an empty body, and the closing curly brace for the class. The line number 5 is highlighted.

Рисунок 12. Результат автоматической генерации главного метода

В созданном классе в теле главного метода введите `sout`, интеллектуальная подсказка IDEA предложит продолжить предложение и автоматически сгенерируется код печатающий в консоль текст. Автоматическая генерация кода печатающий в консоль текст показана на рисунках 13 и 14.



```
1 package lr1;
2
3 public class Example1 {
4
5     public static void main(String[] args) {
6
7         sout
8
9     }
10
11 }
12
```

The screenshot shows the code editor window with the 'sout' text entered on line 7. A tooltip is displayed, showing a list of suggestions for the 'sout' prefix: `sout` (Prints a string to System.out), `souf` (Prints a formatted string to System.out), `soutm` (Prints current class and method names to...), `soutp` (Prints method parameter names and values...), and `soutv` (Prints a value to System.out). The line number 7 is highlighted.

Рисунок 13. Автоматическая генерация кода, печатающего в консоль текст



```
1 package lr1;
2
3 public class Example1 {
4
5     public static void main(String[] args) {
6
7         System.out.println();
8
9     }
10
11 }
12
```

Рисунок 14. Результат генерации кода

Впишите в метод `println()` предложение «Привет мир!», как показано на рисунке 15.



```
1 package lr1;
2
3 public class Example1 {
4
5     public static void main(String[] args) {
6
7         System.out.println("Привет мир!");
8
9     }
10
11 }
12
```

Рисунок 15. Заполнение метода `println()`

Проверьте работоспособность программы, для этого нажмите на зеленый треугольник, после нажатия откроется контекстное меню, выберите **Run**, программа запустится и выведет в консоль сообщение **Привет мир!**

Запуск программы и вывод в консоль показаны на рисунках 16, 17 и 18.

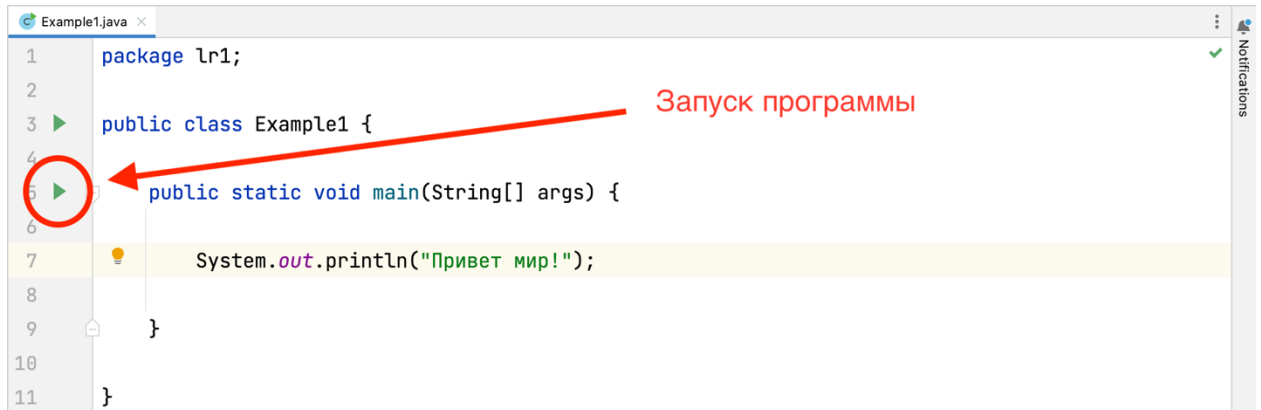


Рисунок 16. Начало запуска программы

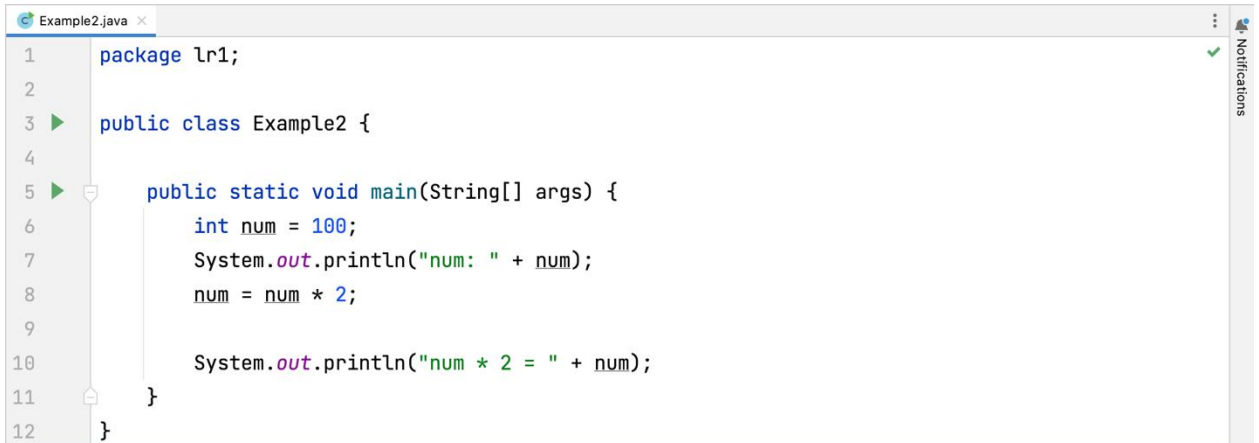


Рисунок 17. Запуск программы



Рисунок 18. Результат выполнения программы

Далее необходимо создать еще четыре класса в пакете `lr1`, классы `Example2`, `Example3`, `Example4`, `Example5`, соответствующие листингам и проверьте их работоспособность. Вышеописанные классы показаны на рисунках 19, 20, 21, 22 соответственно.



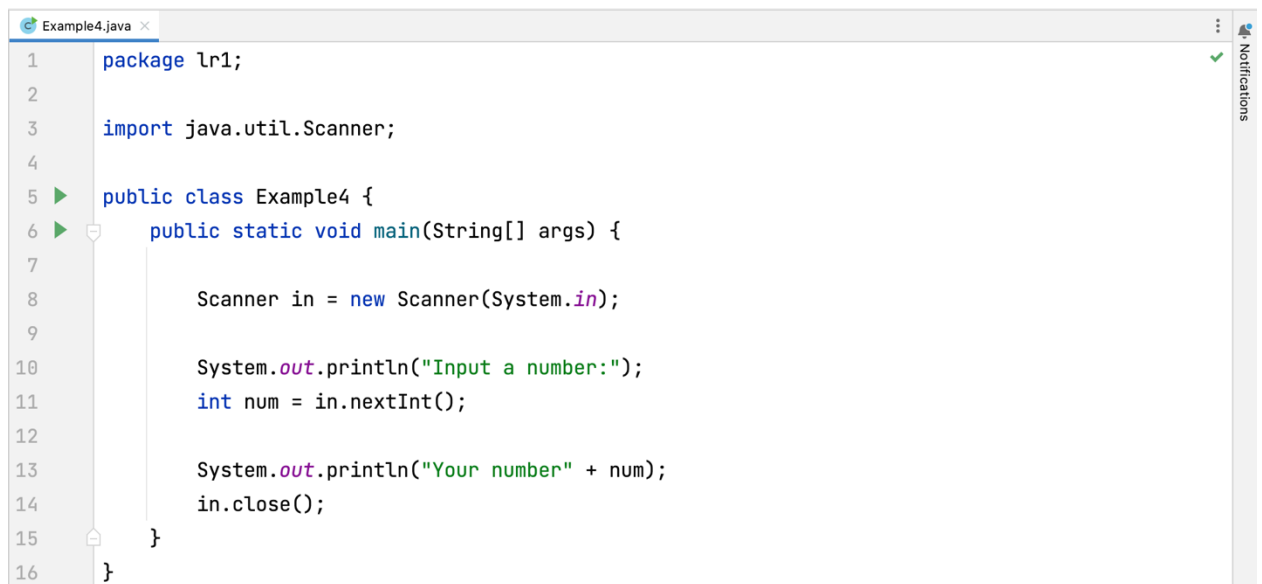
```
1 package lr1;
2
3 public class Example2 {
4
5     public static void main(String[] args) {
6         int num = 100;
7         System.out.println("num: " + num);
8         num = num * 2;
9
10        System.out.println("num * 2 = " + num);
11    }
12 }
```

Рисунок 19. Класс Example 2



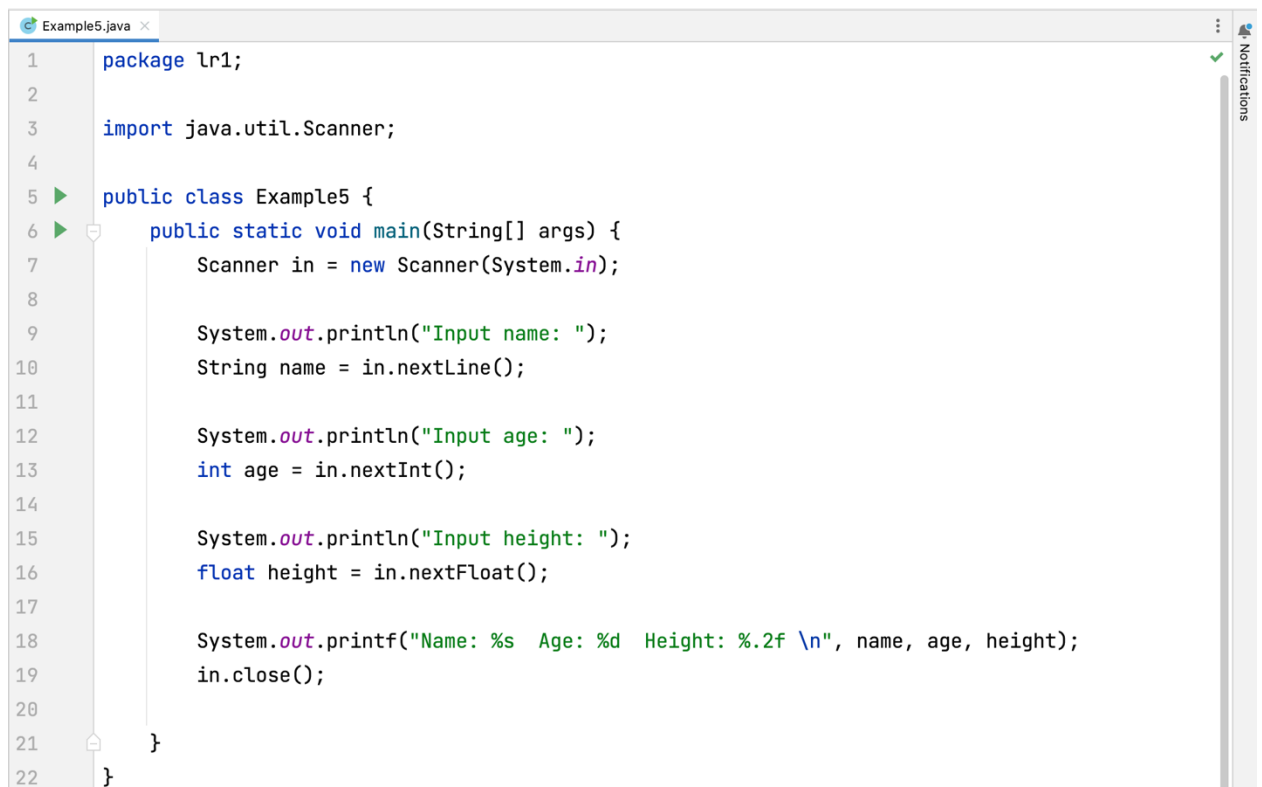
```
1 package lr1;
2
3 public class Example3 {
4     public static void main(String[] args) {
5         int s, d;
6         s = 10;
7         d = 20;
8         if (s < d) {
9             System.out.println("S < D");
10        }
11        s = s * 2;
12        if (s == d) {
13            System.out.println("S = D");
14        }
15        s = s * d;
16        if (s > d) {
17            System.out.println("S > D");
18        }
19
20    }
21 }
```

Рисунок 20. Класс Example 3



```
1 package lr1;
2
3 import java.util.Scanner;
4
5 public class Example4 {
6     public static void main(String[] args) {
7
8         Scanner in = new Scanner(System.in);
9
10        System.out.println("Input a number:");
11        int num = in.nextInt();
12
13        System.out.println("Your number" + num);
14        in.close();
15    }
16 }
```

Рисунок 21. Класс Example 4



```
1 package lr1;
2
3 import java.util.Scanner;
4
5 public class Example5 {
6     public static void main(String[] args) {
7
8         Scanner in = new Scanner(System.in);
9
10        System.out.println("Input name: ");
11        String name = in.nextLine();
12
13        System.out.println("Input age: ");
14        int age = in.nextInt();
15
16        System.out.println("Input height: ");
17        float height = in.nextFloat();
18
19        System.out.printf("Name: %s Age: %d Height: %.2f \n", name, age, height);
20        in.close();
21    }
22 }
```

Рисунок 22. Класс Example 5

4 Задания для самостоятельной работы

1. Напишите программу, в которой Пользователь вводит сначала фамилию, затем имя, затем отчество. После ввода программа выводит сообщение «Hello <фамилия, имя, отчество>».
2. Напишите программу, в которой Пользователь вводит имя и возраст. Программа отображает сообщение об имени и возрасте пользователя.
3. Напишите программу, в которой Пользователь последовательно вводит название текущего дня недели, название месяца и дату (номер дня в месяце). Программа выводит сообщение о сегодняшней дате (день недели, дата, месяц).
4. Напишите программу, в которой пользователю предлагается ввести название месяца и количество дней в этом месяце. Программа выводит сообщение о том, что соответствующий месяц содержит указанное количество дней.
5. Напишите программу, в которой по году рождения определяется возраст пользователя.
6. Напишите программу, в которой Пользователь вводит имя и год рождения, в программа отображает сообщение содержащее имя пользователя и его возраст.
7. Напишите программу, в которой по возрасту определяется год рождения.
8. Напишите программу для вычисления суммы двух чисел. Оба числа вводятся пользователем. Для вычисления суммы используйте оператор +.
9. Напишите программу, в которой пользователь вводит число, а программой отображается последовательность из четырех чисел: число, на единицу меньше введенного, введенное число и число, на единицу больше введенного. Четвертое число должно быть квадратом суммы первых трех чисел.
10. Напишите программу, в которой Пользователь вводит два числа, а программой вычисляется и отображается сумма и разность этих чисел.

5 Описание результата выполнения лабораторной работы

Результат выполнения ЛР:

1) Отчет о выполнении лабораторной работы приложенный к заданию, содержащий ссылку на репозиторий github.com;

2) Для описания части лабораторной работы, выполненной в командной строке должны быть приведены скриншоты шагов выполнения. Проект содержит пакеты `lr1` и `timus`. в пакете `lr1` содержится 15 классов с кодом, первые 5 классов соответствуют классам-примерам, приведенным ранее, остальные 10 классов — это решение задач для самостоятельной подготовки. В пакете `timus` содержится код решения задачи 1293. Пример итоговой структуры проекта показан на рисунке 23.

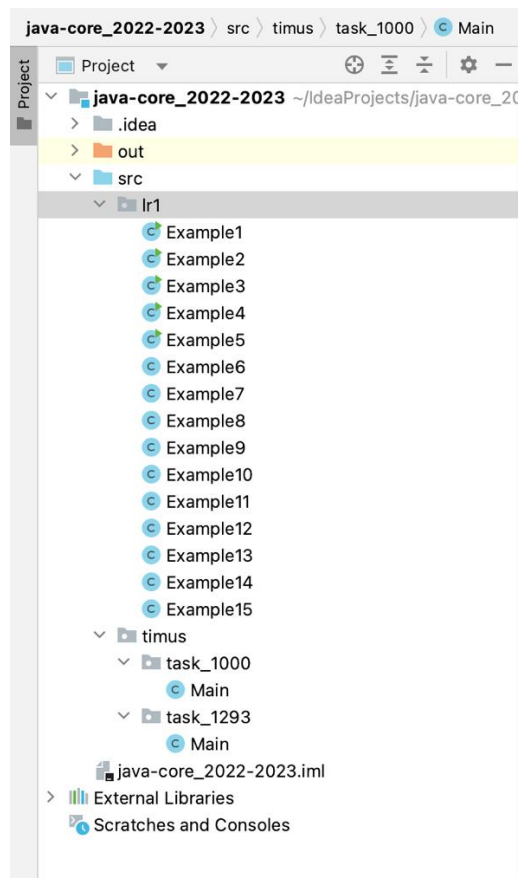


Рисунок 23. Пример итоговой структуры проекта

В отчете по лабораторной работе должны быть представлены:

1. Титульный лист;
2. Цель работы;
3. Описание задачи;
4. Ход выполнения (содержит код программы или скриншоты кода программы);
5. Ссылку на репозиторий git на сайте github.com;
6. Вывод;

Оформление:

- а) шрифт Times New Roman;
- б) размер шрифта 12 или 14;
- в) межстрочный интервал 1,5.

Отчет выполняется индивидуально.