

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА»

Институт «Информатики и кибернетики»
Специальность «Фотоника и оптоинформатика 6201-120303D»

Отчет по лабораторной работе № 1

Выполнил: студент Султанова А.М.,
группа 6201-120303D
Проверил: преподаватель Борисов Д.С.

Самара 2025

Задание 1

```
Microsoft Windows [Version 10.0.26100.5874]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\admin>java
Usage: java [java options...] <application> [application arguments...]

Where <application> is one of:
  <mainclass>           to execute the main method of a compiled main class
  -jar <jarfile>.jar     to execute the main class of a JAR archive
  -m <module>[/<mainclass>] to execute the main class of a module
  <sourcefile>.java     to compile and execute a source-file program

Where key java options include:
  -class-path <class path>
    where <class path> is a list of directories and JAR archives to search for class files, separated by ";"
  -module-path <module path>
    where <module path> is a list of directories and JAR archives to search for modules, separated by ";"
  -version
    to print product version to the error stream and exit

For additional help on usage:      java --help
For an interactive Java environment: jshell
```

```

C:\Users\admin>javac
Usage: javac <options> <source files>
where possible options include:
  @<filename>                Read options and filenames from file
  -Akey[=value]              Options to pass to annotation processors
  --add-modules <module>(<module>)*
                             Root modules to resolve in addition to the initial modules,
                             or all modules on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path>, -bootclasspath <path>
                             Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
                             Specify where to find user class files and annotation processors
  -d <directory>            Specify where to place generated class files
  -deprecation
                             Output source locations where deprecated APIs are used
  --enable-preview
                             Enable preview language features.
                             To be used in conjunction with either -source or --release.
  -encoding <encoding>      Specify character encoding used by source files
  -endorseddirs <dirs>      Override location of endorsed standards path
  -extdirs <dirs>           Override location of installed extensions
  -g                         Generate all debugging info
  -g:{lines,vars,source}    Generate only some debugging info
  -g:none                   Generate no debugging info
  -h <directory>
                             Specify where to place generated native header files
  --help, -help, -?         Print this help message
  --help-extra, -X          Print help on extra options
  -implicit:{none,class}
                             Specify whether to generate class files for implicitly referenced files
  -J<flag>                  Pass <flag> directly to the runtime system
  --limit-modules <module>(<module>)*
                             Limit the universe of observable modules
  --module <module>(<module>)*, -m <module>(<module>)*
                             Compile only the specified module(s), check timestamps
  --module-path <path>, -p <path>
                             Specify where to find application modules
  --module-source-path <module-source-path>
                             Specify where to find input source files for multiple modules
  --module-version <version>
                             Specify version of modules that are being compiled
  -nowarn                   Generate no warnings
  -parameters
                             Generate metadata for reflection on method parameters
  -proc:{none,only,full}
                             Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...]
                             Names of the annotation processors to run;
                             bypasses default discovery process
  --processor-module-path <path>
                             Specify a module path where to find annotation processors
  --processor-path <path>, -processorpath <path>
                             Specify where to find annotation processors
  -profile <profile>
                             Check that API used is available in the specified profile.
                             This option is deprecated and may be removed in a future release.
  --release <release>
                             Compile for the specified Java SE release.
                             Supported releases:
                                 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
  -s <directory>            Specify where to place generated source files
  --source <release>, -source <release>
                             Provide source compatibility with the specified Java SE release.
                             Supported releases:
                                 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
  --source-path <path>, -sourcepath <path>
                             Specify where to find input source files
  --system <jdk>|none        Override location of system modules
  --target <release>, -target <release>
                             Generate class files suitable for the specified Java SE release.
                             Supported releases:
                                 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
  --upgrade-module-path <path>
                             Override location of upgradeable modules
  -verbose                  Output messages about what the compiler is doing
  --version, -version        Version information
  -Werror                   Terminate compilation if warnings occur

```

```

C:\Users\admin>

```

Команды javac и java доступны в cmd, так как установлен JDK.

Задание 2

Создаем папку Task2 и открываем ее, с помощью команд mkdir и cd

```
Командная строка
Microsoft Windows [Version 10.0.26100.5074]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\admin>mkdir Task2

C:\Users\admin>cd Task2
```

Через Блокнот в этой папке создаём файл MyFirstProgram.java со следующим кодом:

```
MyFirstProgram.java
Файл  Изменить  Просмотр

class MyFirstClass {
}
```

Откомпилируем его с помощью компилятора javac

```
Командная строка
Microsoft Windows [Version 10.0.26100.5074]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\admin>mkdir Task2

C:\Users\admin>cd Task2

C:\Users\admin\Task2>javac MyFirstProgram.java
```

Результат: в папке появился файл MyFirstClass.class

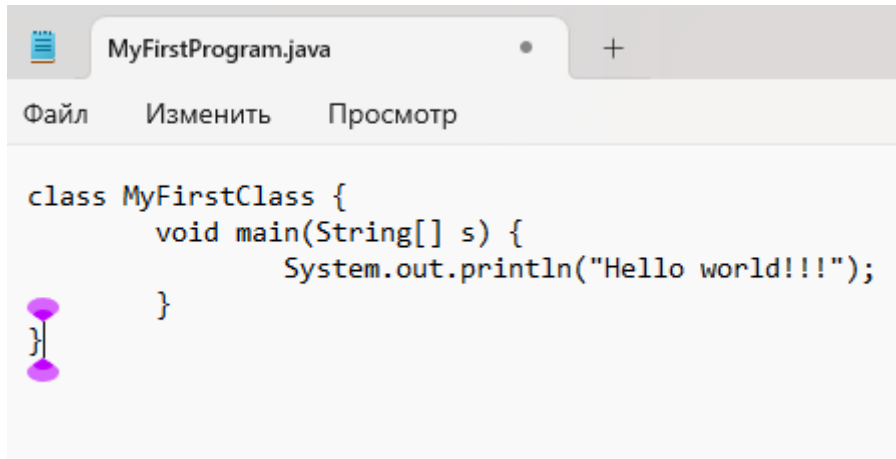
Имя	Дата изменения	Тип	Размер
MyFirstClass.class	13.09.2025 18:37	Файл "CLASS"	1 КБ
MyFirstProgram	13.09.2025 18:36	Файл "JAVA"	1 КБ

Попытка запуска и результат: ошибка

```
C:\Users\admin\Task2>java MyFirstClass
Error: Main method not found in class MyFirstClass, please define the main method as:
    public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\admin\Task2>
```

Меняем содержимое MyFirstProgram.java:



```
MyFirstProgram.java
Файл  Изменить  Просмотр

class MyFirstClass {
    void main(String[] s) {
        System.out.println("Hello world!!!");
    }
}
```

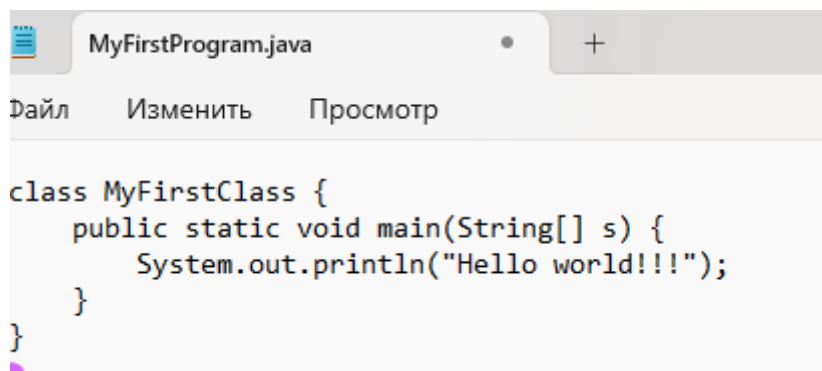
Компилируем исправленный файл:

```
C:\Users\admin\Task2>javac MyFirstProgram.java

C:\Users\admin\Task2>java MyFirstClass
Error: Main method not found in class MyFirstClass, please define the main method as:
    public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application

C:\Users\admin\Task2>
```

Мы снова получаем ошибку, поэтому вносим изменения в код, делая метод статическим:



```
MyFirstProgram.java
Файл  Изменить  Просмотр

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

Получаем ожидаемый результат:

```
C:\Users\admin\Task2>javac MyFirstProgram.java

C:\Users\admin\Task2>java MyFirstClass
Hello world!!!
```

Задание 3

Создаем папку Task3 и открываем ее, в файле MyFirstProgram.java меняем текст метода:

```
class MyFirstClass {
    public static void main(String[] s) {
        for (int i = 0; i < s.length; i++) {
            System.out.println(s[i]);
        }
    }
}
```

Откомпилируем и запустим программу, добавив в командную строку ряд аргументов:

```
C:\Users\admin>mkdir Task3

C:\Users\admin>cd Task3

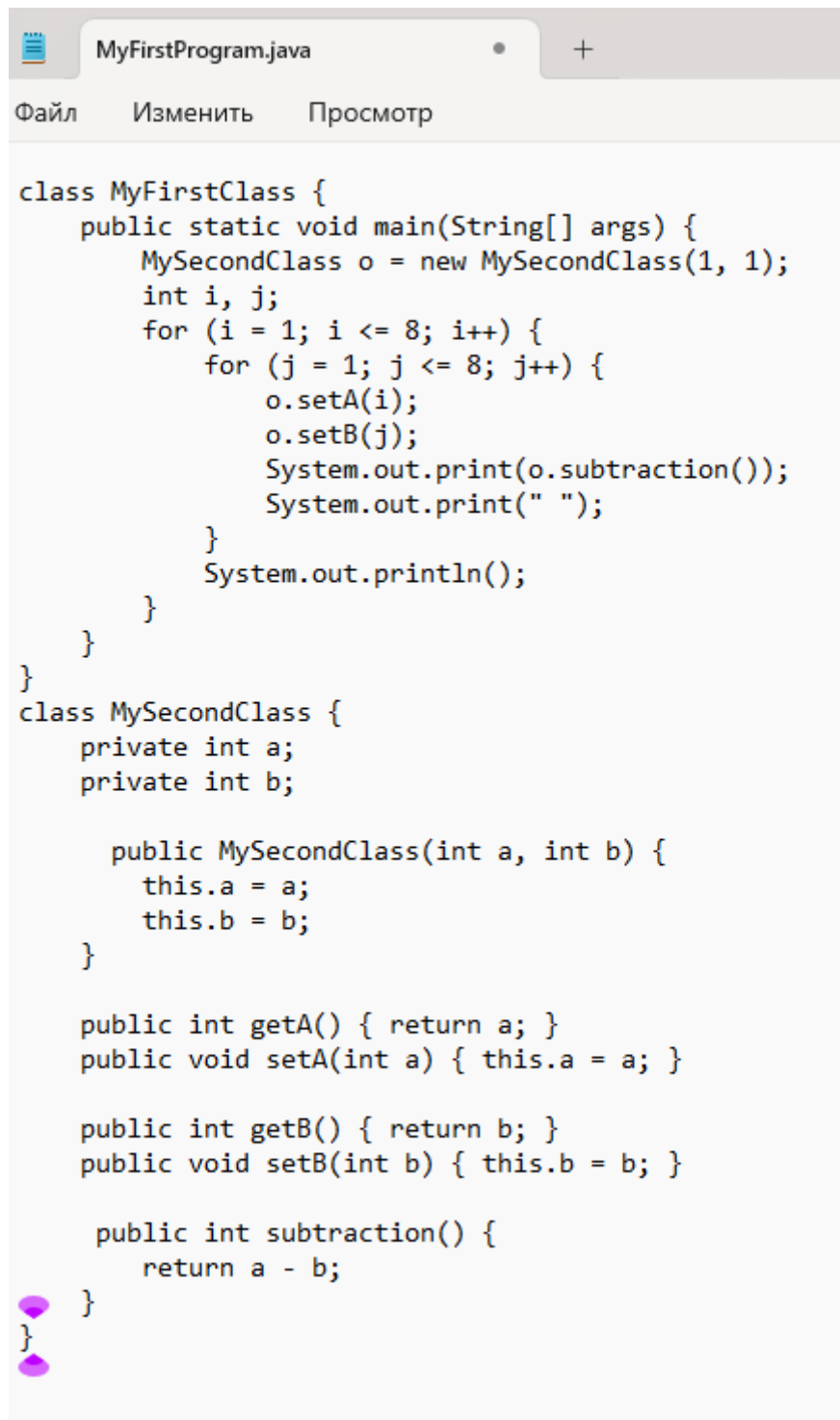
C:\Users\admin\Task3>javac MyFirstProgram.java

C:\Users\admin\Task3>java MyFirstClass arg1 arg2 arg3 arg4 arg5
arg1
arg2
arg3
arg4
arg5

C:\Users\admin\Task3>
```

Задание 4

Создаем папку Task4 ,открывая его , и файл MyFirstProgram.java, в котором теперь 2 класса:



```
class MyFirstClass {
    public static void main(String[] args) {
        MySecondClass o = new MySecondClass(1, 1);
        int i, j;
        for (i = 1; i <= 8; i++) {
            for (j = 1; j <= 8; j++) {
                o.setA(i);
                o.setB(j);
                System.out.print(o.subtraction());
                System.out.print(" ");
            }
            System.out.println();
        }
    }
}

class MySecondClass {
    private int a;
    private int b;

    public MySecondClass(int a, int b) {
        this.a = a;
        this.b = b;
    }

    public int getA() { return a; }
    public void setA(int a) { this.a = a; }

    public int getB() { return b; }
    public void setB(int b) { this.b = b; }

    public int subtraction() {
        return a - b;
    }
}
```

1. MyFirstClass

В этом классе находится метод `main`. Он создаёт объект второго класса и с помощью циклов от 1 до 8 формирует таблицу. На каждой итерации в объект записываются новые значения двух чисел (через методы `setA` и `setB`), а затем вызывается метод для вычисления результата (в моём случае вычитания). Полученные значения выводятся на экран, что в итоге даёт таблицу 8×8 .

2. MySecondClass

Внутри класса есть два приватных поля `a` и `b`. Приватные — значит, они закрыты от прямого доступа снаружи. Чтобы работать с этими полями, есть геттеры и сеттеры. Также есть конструктор, который позволяет создать объект сразу с начальными значениями `a` и `b`. Основной метод в этом классе — `subrtaction ()`. Он просто берёт два числа и возвращает их разность.

Компилируем и получаем результат:

```
C:\Users\admin>mkdir Task4

C:\Users\admin>cd Task4

C:\Users\admin\Task4>javac MyFirstProgram.java

C:\Users\admin\Task4>java MyFirstClass
0 -1 -2 -3 -4 -5 -6 -7
1 0 -1 -2 -3 -4 -5 -6
2 1 0 -1 -2 -3 -4 -5
3 2 1 0 -1 -2 -3 -4
4 3 2 1 0 -1 -2 -3
5 4 3 2 1 0 -1 -2
6 5 4 3 2 1 0 -1
7 6 5 4 3 2 1 0
```

Задание 5

Создаем папку `Task5` и открываем ее:

```
C:\Users\admin>mkdir Task5

C:\Users\admin>cd Task5

C:\Users\admin\Task5>
```

Удаляем все откомпилированные байт-коды классов, с помощью `del /s *.class`:

```
C:\Users\admin\Task5>del /s *.class
Удален файл - C:\Users\admin\Task5\MyFirstClass.class
Удален файл - C:\Users\admin\Task5\MySecondClass.class
```

Создаем поддиректорию `myfirstpackage`:

```
C:\Users\admin\Task5>mkdir myfirstpackage
```


Выносим код класса MySecondClass без изменений в отдельный файл с именем MyFirstPackage.java, и помещаем его в поддиректорию myfirstpackage. Компилируем:

```
C:\Users\admin\Task5>javac myfirstpackage/MyFirstPackage.java
C:\Users\admin\Task5>
```

Теперь компилируем MyFirstProgram.java:

```
C:\Users\admin\Task5>javac MyFirstProgram.java
MyFirstProgram.java:4: error: cannot find symbol
    MySecondClass o = new MySecondClass(1, 1);
    ^
  symbol:   class MySecondClass
  location: class MyFirstClass
MyFirstProgram.java:4: error: cannot find symbol
    MySecondClass o = new MySecondClass(1, 1);
                          ^
  symbol:   class MySecondClass
  location: class MyFirstClass
2 errors
C:\Users\admin\Task5>
```

Результат: ошибка

Добавляем в начало исходного кода в файле MyFirstProgram.java следующий код: `import myfirstpackage.*;`

Также нужно добавить `package myfirstpackage.*;` в начало кода файла MyFirstPackage.java, чтобы компилятор понимал, где этот класс искать.

Снова компилируем и получаем ошибку:

```
C:\Users\admin\Task5>javac MyFirstProgram.java
MyFirstProgram.java:4: error: cannot access MySecondClass
    MySecondClass o = new MySecondClass(1, 1);
    ^
  bad class file: .\myfirstpackage\MySecondClass.class
    class file contains wrong class: MySecondClass
    Please remove or make sure it appears in the correct subdirectory of the classpath.
MyFirstProgram.java:4: error: cannot find symbol
    MySecondClass o = new MySecondClass(1, 1);
                          ^
  symbol:   class MySecondClass
  location: class MyFirstClass
2 errors
```

Исправим ошибку сделав MySecondClass публичным, чтобы класс был доступен:

```
C:\Users\admin\Task5\myfirstpackage>javac MyFirstPackage.java
MyFirstPackage.java:1: error: class MySecondClass is public, should be declared in a file named MySecondClass.java
public class MySecondClass {
      ^
1 error
```

Исходя из ошибки переименовываем файл MyFirstPackage.java в название нашего класса MySecondClass.java и снова компилируем, получая ожидаемый результат:

```
C:\Users\admin\Task5>javac myfirstpackage/MySecondClass.java

C:\Users\admin\Task5>javac MyFirstProgram.java

C:\Users\admin\Task5>java MyFirstProgram.java
0 -1 -2 -3 -4 -5 -6 -7
1 0 -1 -2 -3 -4 -5 -6
2 1 0 -1 -2 -3 -4 -5
3 2 1 0 -1 -2 -3 -4
4 3 2 1 0 -1 -2 -3
5 4 3 2 1 0 -1 -2
6 5 4 3 2 1 0 -1
7 6 5 4 3 2 1 0

C:\Users\admin\Task5>
```

Задание 6

Создаем папку Task6 и открываем ее. Скопируем в рабочую папку, сохранив структуру каталогов, только файлы с расширением class, полученные в результате выполнения задания 5, с помощью команды xcopy:

```
C:\Users\admin>mkdir Task6

C:\Users\admin>cd Task6

C:\Users\admin\Task6>xcopy ..\Task5\*.class . /s
..\Task5\MyFirstClass.class
..\Task5\myfirstpackage\MySecondClass.class
Скопировано файлов: 2.

C:\Users\admin\Task6>
```

Создаём файл manifest.mf в папке Task6.

Используем команду jar для создания архива myfirst.jar, указав манифест:

```
C:\Users\admin\Task6>jar cfm myfirst.jar manifest.mf MyFirstClass.class myfirstpackage/  
C:\Users\admin\Task6>
```

После выполнения команды появится файл myfirst.jar

Создаем новую папку и перемещаем туда jar:

```
C:\Users\admin\Task6>mkdir MyJar
```

```
C:\Users\admin\Task6>move myfirst.jar MyJar\  
Перемещено файлов:      1.  
C:\Users\admin\Task6>
```

Запуск JAR-файла:

```
C:\Users\admin\Task6>java -jar MyJar/myfirst.jar  
0 -1 -2 -3 -4 -5 -6 -7  
1 0 -1 -2 -3 -4 -5 -6  
2 1 0 -1 -2 -3 -4 -5  
3 2 1 0 -1 -2 -3 -4  
4 3 2 1 0 -1 -2 -3  
5 4 3 2 1 0 -1 -2  
6 5 4 3 2 1 0 -1  
7 6 5 4 3 2 1 0  
C:\Users\admin\Task6>
```