

Лабораторная работа №1  
Коновалов Сергей Сергеевич 6204-010302D

## Задание 1

На рисунке 1 изображена информация, которая появляется при запуске команды `java`. Консоль показывает список команд, их синтаксис и описание

```
PS D:\Other\Study\GitHubLab1> 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
PS D:\Other\Study\GitHubLab1> |
```

Рисунок 1

На рисунке 2 изображена информация, которая появляется при запуске команды `javac`. Консоль показывает список команд, их синтаксис и описание

```
PS D:\Other\Study\GitHubLab1> 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
```

Рисунок 2

## Задание 2

Создал файл MyFirstProgram.java, изображенный на рисунке 3 и добавил следующий код:

```
class MyFirstClass {  
}
```



Рисунок 3

Затем откомпилировал этот файл, введя в консоль команду:

```
javac MyFirstProgram.java
```

и в папке появился файл MyFirstClass.class, изображенный на рисунке 4

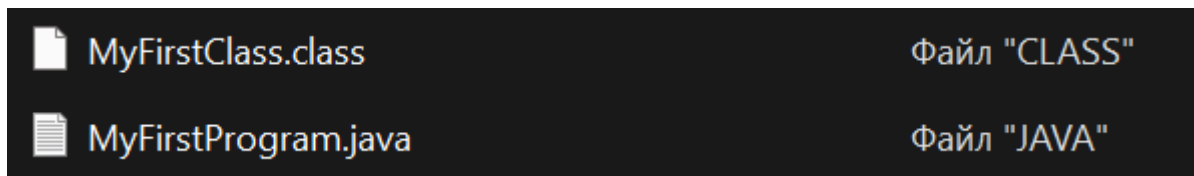


Рисунок 4

После этого я запустил программу командой в консоль

```
java MyFirstClass
```

Следуя заданию, добавил в класс следующий код:

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

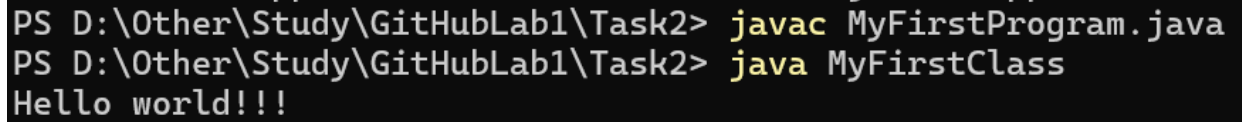
Затем также откомпилировал и запустил

Консоль выдала ошибку, сказав, что класс должен быть static, потому что на момент запуска программы нет никакого объекта типа MyFirstClass. Добавил public static перед void main(String[] s)

```
class MyFirstClass {  
    public static void main(String[] s) {
```

```
        System.out.println("Hello world!!!");  
    }  
}
```

После поправок видим результат(рисунок 5):



```
PS D:\Other\Study\GitHubLab1\Task2> javac MyFirstProgram.java  
PS D:\Other\Study\GitHubLab1\Task2> java MyFirstClass  
Hello world!!!
```

Рисунок 5

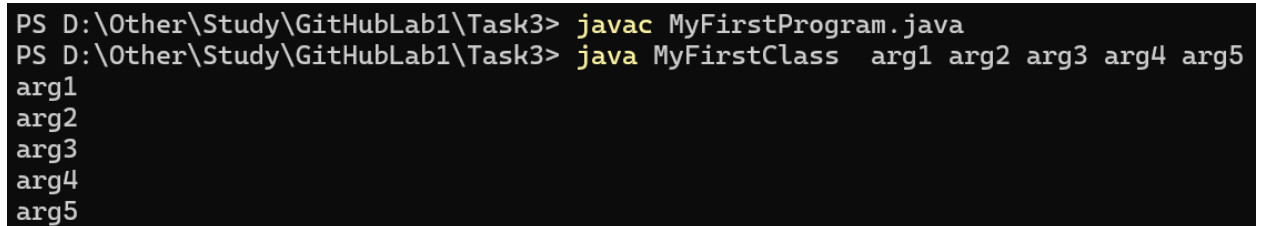
### Задание 3

Внесем некоторые изменения в метод main(), как требует задание:

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

После чего откомпилировал программу и запустил программу с добавлением аргументов: `java MyFirstClass arg1 arg2 arg3 arg4 arg5`

Результат на рисунке 6:



```
PS D:\Other\Study\GitHubLab1\Task3> javac MyFirstProgram.java  
PS D:\Other\Study\GitHubLab1\Task3> java MyFirstClass arg1 arg2 arg3 arg4 arg5  
arg1  
arg2  
arg3  
arg4  
arg5
```

Рисунок 6

### Задание 4

Согласно заданию, добавил второй класс в файл MyFirstProgram.java:

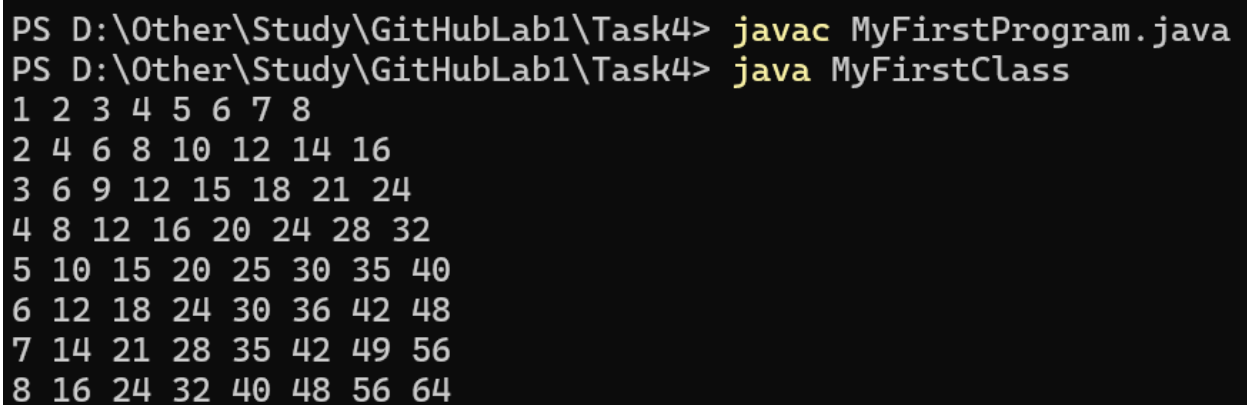
```
class MySecondClass {  
    private int firstNum;  
    private int secondNum;  
  
    MySecondClass(int first, int second) {  
        this.firstNum = first;  
        this.secondNum = second;  
    }  
  
    public int getFirstNum() {  
        return firstNum;  
    }  
  
    public int getSecondNum() {  
        return secondNum;  
    }  
  
    public void setFirstNum(int val) {  
        this.firstNum = val;  
    }  
  
    public void setSecondNum(int val) {  
        this.secondNum = val;  
    }  
  
    public int multiply() {  
        return firstNum * secondNum;  
    }  
}
```

Также изменил код в main() на следующие строки:

```
MySecondClass o = new MySecondClass(10, 10);
```

```
    int i, j;  
    for (i = 1; i <= 8; i++) {  
        for(j = 1; j <= 8; j++) {  
            o.setFirstNum(i);  
            o.setSecondNum(j);  
            System.out.print(o.multiply());  
            System.out.print(" ");  
        }  
        System.out.println();  
    }
```

Затем откомпилировал MyFirstProgram и увидел результат на рисунке 7:



```
PS D:\Other\Study\GitHubLab1\Task4> javac MyFirstProgram.java  
PS D:\Other\Study\GitHubLab1\Task4> java MyFirstClass  
1 2 3 4 5 6 7 8  
2 4 6 8 10 12 14 16  
3 6 9 12 15 18 21 24  
4 8 12 16 20 24 28 32  
5 10 15 20 25 30 35 40  
6 12 18 24 30 36 42 48  
7 14 21 28 35 42 49 56  
8 16 24 32 40 48 56 64
```

Рисунок 7

## Задание 5

По требованию задания создал папку myfirstpackage

Вынес MySecondClass в отдельный файл и назвал его:

MyFirstPackage.java

Затем добавил в MyFirstPackage.java следующую строку в начало:

```
package myfirstpackage;
```

в файл MyFirstClass.java добавил в начало:

```
import myfirstpackage.*;
```

Из-за нескольких ошибок я изменил код следующими изменениями:

- Добавил модификатор доступа public к MySecondClass
- Добавил модификатор доступа public к конструктору класса
- Переименовал MyFirstPackage.java в MySecondClass.java

После исправлений видим нужный результат на рисунке 8:

```
PS D:\Other\Study\GitHubLab1\Task5> javac myfirstpackage\MySecondClass.java
PS D:\Other\Study\GitHubLab1\Task5> javac MyFirstProgram.java
PS D:\Other\Study\GitHubLab1\Task5> java MyFirstClass
1 2 3 4 5 6 7 8
2 4 6 8 10 12 14 16
3 6 9 12 15 18 21 24
4 8 12 16 20 24 28 32
5 10 15 20 25 30 35 40
6 12 18 24 30 36 42 48
7 14 21 28 35 42 49 56
8 16 24 32 40 48 56 64
```

Рисунок 8

## Задание 6

Запустил jar для создание архива и получил формат записи для создания архива(Рисунок 9):

```
PS D:\Other\Study\GitHubLab1\Task6> jar
Usage: jar [OPTION...] [ [--release VERSION] [-C dir] files] ...
Try `jar --help' for more information.
```

Рисунок 9

После чего скопировал файлы с расширением .class в папку Task6 и создал файл:  
manifest.mf

В manifest.mf добавил следующий код:

Manifest-Version: 1.0

Created-By: Konovalov

Main-Class: MyFirstClass

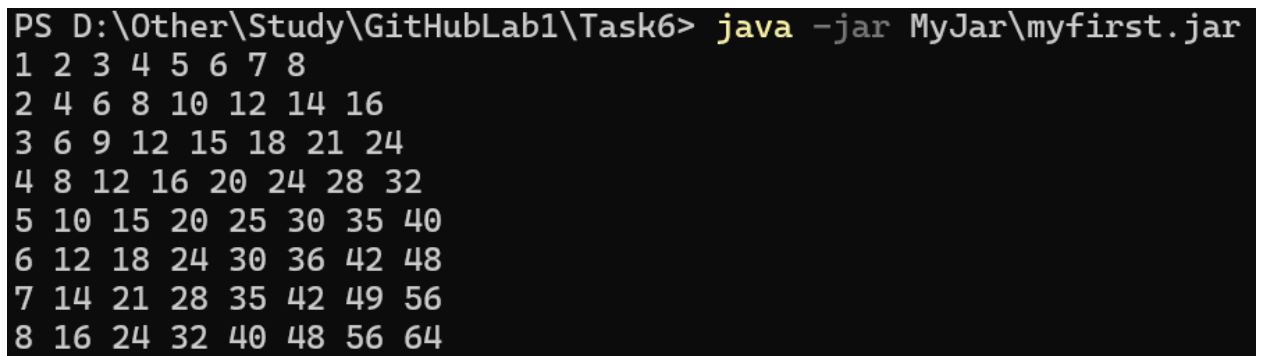
Обязательно с переносом строки в конце (после MyFirstClass).

Затем создал архив:

```
jar cfm myfirst.jar manifest.mf *.class myfirstpackage\*.class
```

После чего появился архив myfirst.jar, хранящий файлы с расширением .class и manifest.mf, который я перенес в папку MyJar.

Результат запуска myfirst.jar на рисунке 10:



```
PS D:\Other\Study\GitHubLab1\Task6> java -jar MyJar\myfirst.jar
1 2 3 4 5 6 7 8
2 4 6 8 10 12 14 16
3 6 9 12 15 18 21 24
4 8 12 16 20 24 28 32
5 10 15 20 25 30 35 40
6 12 18 24 30 36 42 48
7 14 21 28 35 42 49 56
8 16 24 32 40 48 56 64
```

Рисунок 10