

Лабораторная работа №1

Валиневич Владислав Александрович

группа: 6204-010302D

Цель лабораторной работы: В процессе написания тестовых заданий ознакомиться со структурой исходного кода для Java, изучить особенности областей видимости и использования пакетов.

Ход выполнения:

Задание 1:

Запустим компилятор `javac` и `java` без параметров

```
vladislavvalinevich@MacBook-Air-Vladislav ~ % 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 file
  --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 f
  -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 only mandatory 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:

vladislavvalinevich@MacBook-Air-Vladislav ~ % 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
vladislavvalinevich@MacBook-Air-Vladislav ~ %
```

Задание 2:

Создадим файл `MyFirstProgram.java`, содержащий пустой класс с именем `MyFirstClass`. Попробуем скомпилировать файл запустив команду `javac MyFirstProgram.java` в консоли, у нас создастся файл `MyFirstClass.class` - скомпилированный байт-код. После запустим сам класс командой `java MyFirstClass`, консоль выдаст ошибку, т. к. отсутствует точка входа программы, которым является метод `main`.

Добавим метод `main`, программа при запуске выдаст ошибку, так как `main` должен быть статический и публичный.

Также нужно поставить правильные кавычки.

Доработаем программу, чтобы она была рабочей и получим нужный результат

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

```
vladislavvalinevich@MacBook-Air-Vladislav ~ % javac MyFirstProgram.java
vladislavvalinevich@MacBook-Air-Vladislav ~ % java MyFirstClass
Hello world!!!
vladislavvalinevich@MacBook-Air-Vladislav ~ %
```

Задание 3:

Изменяем метод `main` для обработки аргументов, массив `String[] s` в методе `main` содержит все аргументы, переданные при запуске, `s.length` - количество переданных аргументов, а `s[i]` - доступ к конкретному аргументу по индексу

Запустим файл также через команду `javac MyFirstProgram.java`, но при запуске класса в команде перечислим нужные нам аргументы `java MyFirstClass arg1 arg2 arg3 arg4 arg5`

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

Hello world!!!
vladislavvalinevich@MacBook-Air-Vladislav ~ % javac MyFirstProgram.java
vladislavvalinevich@MacBook-Air-Vladislav ~ % java MyFirstClass arg1 arg2 arg3 arg4 arg5
arg1
arg2
arg3
arg4
arg5
vladislavvalinevich@MacBook-Air-Vladislav ~ %
```

В результате в консоли мы получим вывод всех аргументов по одному с новой строки.

Задание 4:

Добавим в файл новый класс `MySecondClass`, который реализует следующую функциональность:

```
class MySecondClass {
    int i;
    public int get_i(){
        return i;
    }
    public int get_j(){
        return j;
    }
    public void set_i(int x){
        i=x;
    }
    public void set_j(int x){
        j=x;
    }
    public MySecondClass(int x, int y){
        i=x;
        j=y;
    }
    public int ymnoi(){
        return i*j;
    }
}
```

Два приватных поля типа `int`

Методы для получения и модификации их значений

Конструктор, создающий объект и инициализирующий значения полей

Метод с возвращаемым типом `int`, реализующий умножение

Также изменим и дополним метод main как требуется в задании:

```
class MyFirstClass {
    public static void main(String[] s) {
        int i, j;
        i=0;
        j=0;
        MySecondClass o = new MySecondClass(i,j);

        for (i = 1; i <= 8; i++) {
            for(j = 1; j <= 8; j++) {
                o.set_i(i);
                o.set_j(j);
                System.out.print(o.ymnoj());
                System.out.print(" ");
            }
            System.out.println();
        }
    }
}
```

Попробуем откомпилировать программу, у нас создастся новый файл в папке MySecondClass.class - скомпилированный байт-код. В результате получим успешно рабочую программу, создающую объект и использующая его методы для генерации таблицы умножения.

```
class MyFirstClass {
    public static void main(String[] s) {
        int i, j;
        i=0;
        j=0;
        MySecondClass o = new MySecondClass(i,j);

        for (i = 1; i <= 8; i++) {
            for(j = 1; j <= 8; j++) {
                o.set_i(i);
                o.set_j(j);
                System.out.print(o.ymnoj());
                System.out.print(" ");
            }
            System.out.println();
        }
    }
}

class MySecondClass {
    int i,j;
    public int get_i(){
        return i;
    }
    public int get_j(){
        return j;
    }
    public void set_i(int x){
        i=x;
    }
    public void set_j(int x){
        j=x;
    }
    public MySecondClass(int x, int y){
        i=x;
        j=y;
    }
    public int ymnoj(){
        return i*j;
    }
}
```

```
[vladislavvalinevich@MacBook-Air-Vladislav ~ % javac MyFirstProgram.java
[vladislavvalinevich@MacBook-Air-Vladislav ~ % 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
vladislavvalinevich@MacBook-Air-Vladislav ~ %
```

Задание 5:

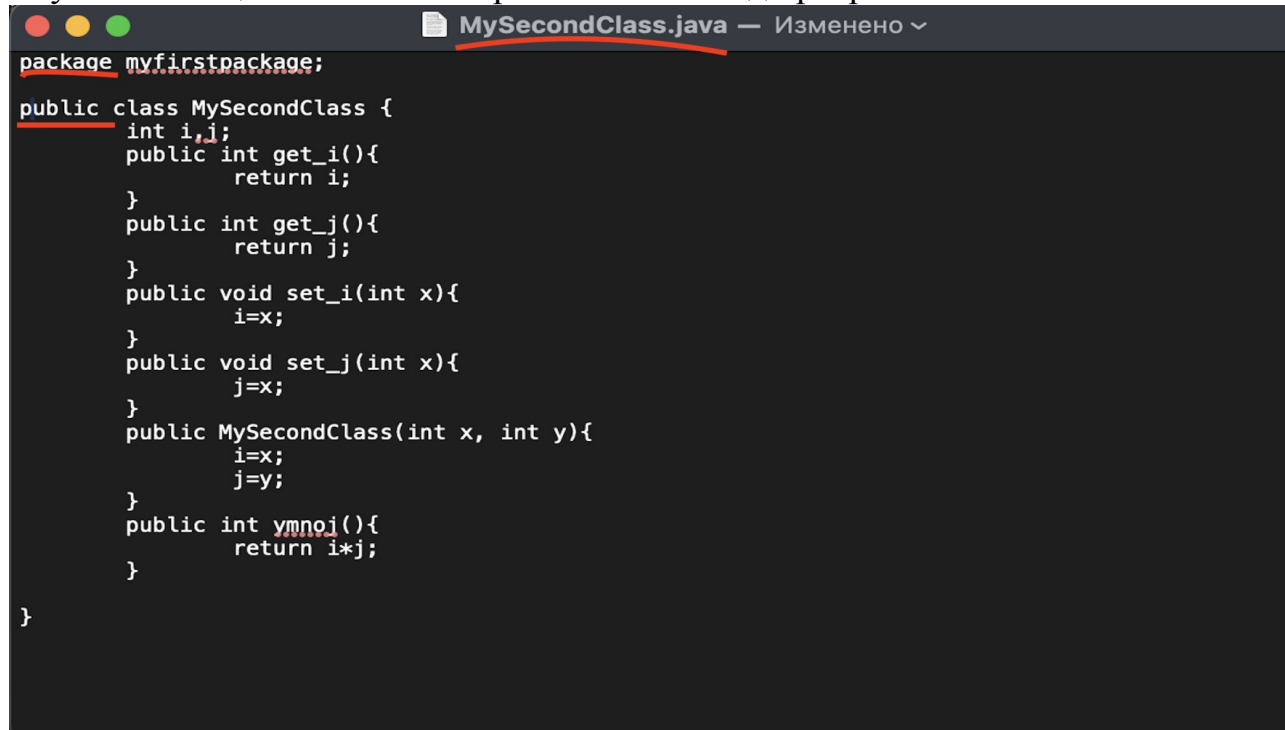
Перенесем код *MySecondClass* отдельно в файл *MyFirstPackage.java* в поддиректорию *myfirstpackage*.

Также добавим в исходный код: *import myfirstpackage.**;

Попробуем откомпилировать:

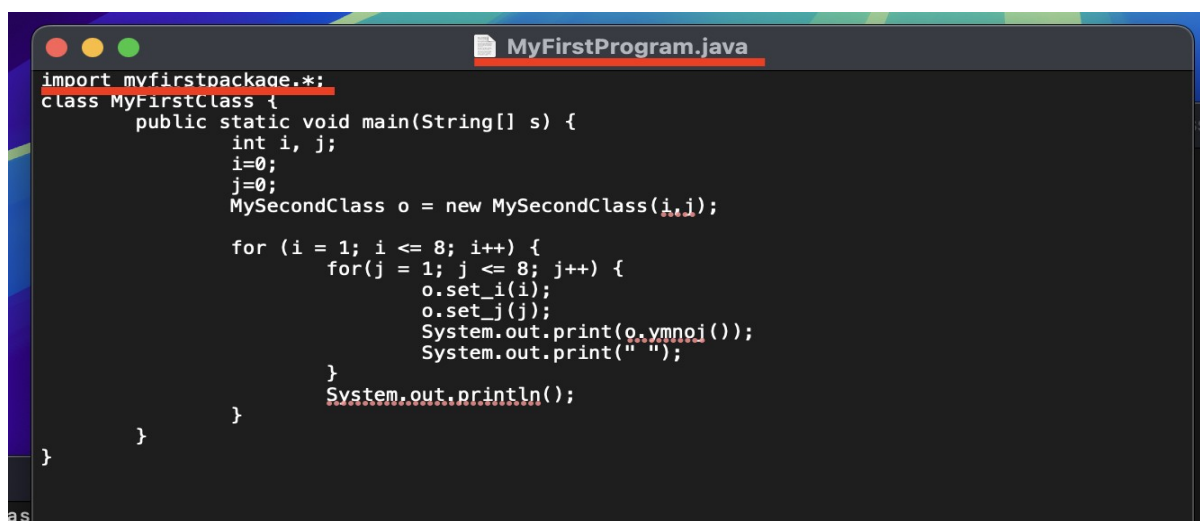
```
2 errors
vladislavvalinevich@MacBook-Air-Vladislav task 5 % javac MyFirstProgram.java
MyFirstProgram.java:7: error: cannot find symbol
    MySecondClass o = new MySecondClass(i,j);
                        ^
  symbol:   class MySecondClass
  location: class MyFirstClass
MyFirstProgram.java:7: error: cannot find symbol
    MySecondClass o = new MySecondClass(i,j);
                        ^
  symbol:   class MySecondClass
  location: class MyFirstClass
2 errors
vladislavvalinevich@MacBook-Air-Vladislav task 5 % javac MyFirstPackage.java
error: file not found: MyFirstPackage.java
Usage: javac <options> <source files>
vladislavvalinevich@MacBook-Air-Vladislav task 5 % javac myfirstpackage/MyFirstPackage.java
myfirstpackage/MyFirstPackage.java:2: error: class MySecondClass is public, should be declared in a file named MySecondClass.java
public class MySecondClass {
      ^
1 error
vladislavvalinevich@MacBook-Air-Vladislav task 5 %
```

Изучим сообщения компилятора и изменим код программы.



```
package myfirstpackage;

public class MySecondClass {
    int i,j;
    public int get_i(){
        return i;
    }
    public int get_j(){
        return j;
    }
    public void set_i(int x){
        i=x;
    }
    public void set_j(int x){
        j=x;
    }
    public MySecondClass(int x, int y){
        i=x;
        j=y;
    }
    public int ymnoji(){
        return i*j;
    }
}
```

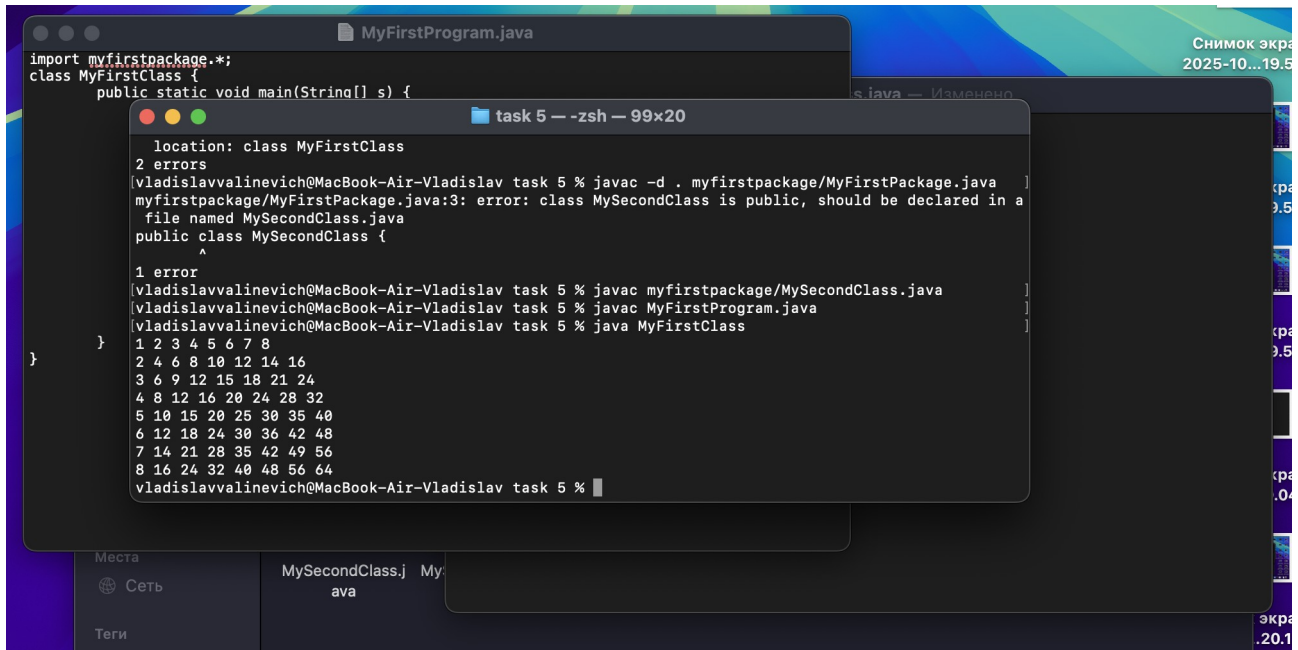


```
import myfirstpackage.*;

class MyFirstClass {
    public static void main(String[] s) {
        int i, j;
        i=0;
        j=0;
        MySecondClass o = new MySecondClass(i,i);

        for (i = 1; i <= 8; i++) {
            for(j = 1; j <= 8; j++) {
                o.set_i(i);
                o.set_j(j);
                System.out.print(o.ymnoji());
                System.out.print(" ");
            }
            System.out.println();
        }
    }
}
```

Программа успешно скомпилировалась



```
MyFirstProgram.java
import myfirstpackage.*;
class MyFirstClass {
    public static void main(String[] s) {
        location: class MyFirstClass
        2 errors
vladislavvalinevich@MacBook-Air-Vladislav task 5 % javac -d . myfirstpackage/MyFirstPackage.java
myfirstpackage/MyFirstPackage.java:3: error: class MySecondClass is public, should be declared in a
file named MySecondClass.java
    public class MySecondClass {
        ^
    1 error
vladislavvalinevich@MacBook-Air-Vladislav task 5 % javac myfirstpackage/MySecondClass.java
vladislavvalinevich@MacBook-Air-Vladislav task 5 % javac MyFirstProgram.java
vladislavvalinevich@MacBook-Air-Vladislav task 5 % 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
vladislavvalinevich@MacBook-Air-Vladislav task 5 %
```

Задание 6:

Создадим манифест, а после создадим jar архив с помощью команды в консоле:
jar cfm myfirst.jar manifest.mf MyFirstClass.class myfirstpackage/MySecondClass.class

```
vladislavvalinevich@MacBook-Air-Vladislav task 6 % ls MyJar
vladislavvalinevich@MacBook-Air-Vladislav task 6 % jar cfm myfirst.jar manifest.mf MyFirstClass.class myfirstpackage/MySecondClass.class
vladislavvalinevich@MacBook-Air-Vladislav task 6 % java -jar 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
vladislavvalinevich@MacBook-Air-Vladislav task 6 %
```

Вывод: Я успешно ознакомился со структурой исходного кода для Java, изучил особенности областей видимости и использования пакетов, научился пользоваться git и сайтом github.