Лабораторная работа №1

Выполнила: Джумашева Камилла Исфатовна

Студент 6203-010302D группы

Ход выполнения

Задача 1

В ходе работы мне было необходимо ознакомиться с утилитами javac и java, изучить структуру исходного кода, области видимости и использование пакетов.

Для ознакомления с параметрами компилятора javac была выполнена команда “javac” без параметров (рис. 1). Команда вывела подробную справку по использованию компилятора, включая основные опции:

- ➔ -d <directory> - указание директории для размещения скомпилированных class-файлов
- ➔ -cp <path> или -classpath <path> - указание путей к пользовательским классам
- ➔ -sourcepath <path> - указание путей к исходным файлам

- ➔ -encoding <encoding> - указание кодировки исходных файлов

```
PS C:\Users\Home> 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
```

Рис. 1

Аналогично была изучена утилита java для запуска программ (рис. 2).

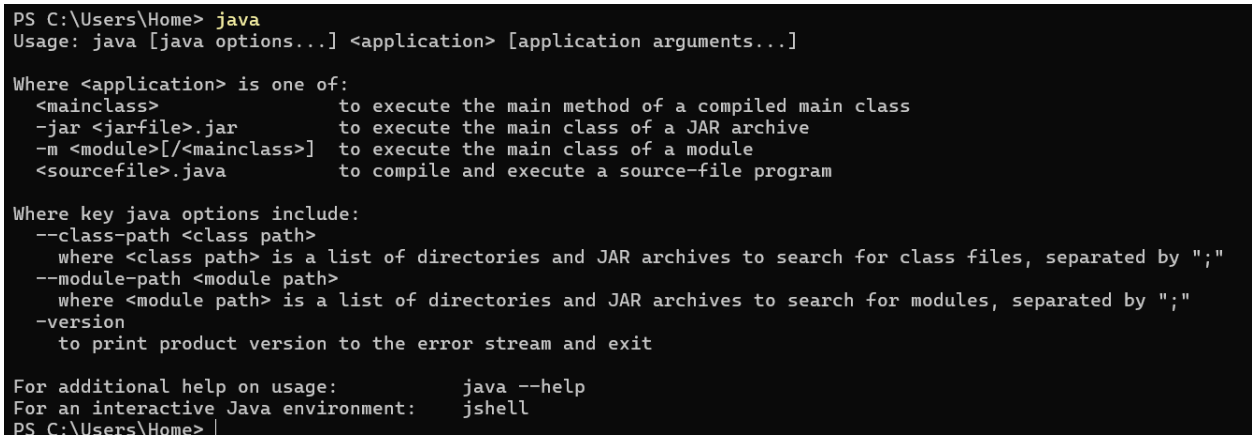

Рис. 2

Задание 2

В начале выполнения задания мы должны были попытаться скомпилировать файл MyFirstProgram.java, однако компилятор сообщил об ошибке - файл не был найден. Это произошло потому, что файл еще не был создан (См. рис. 3)

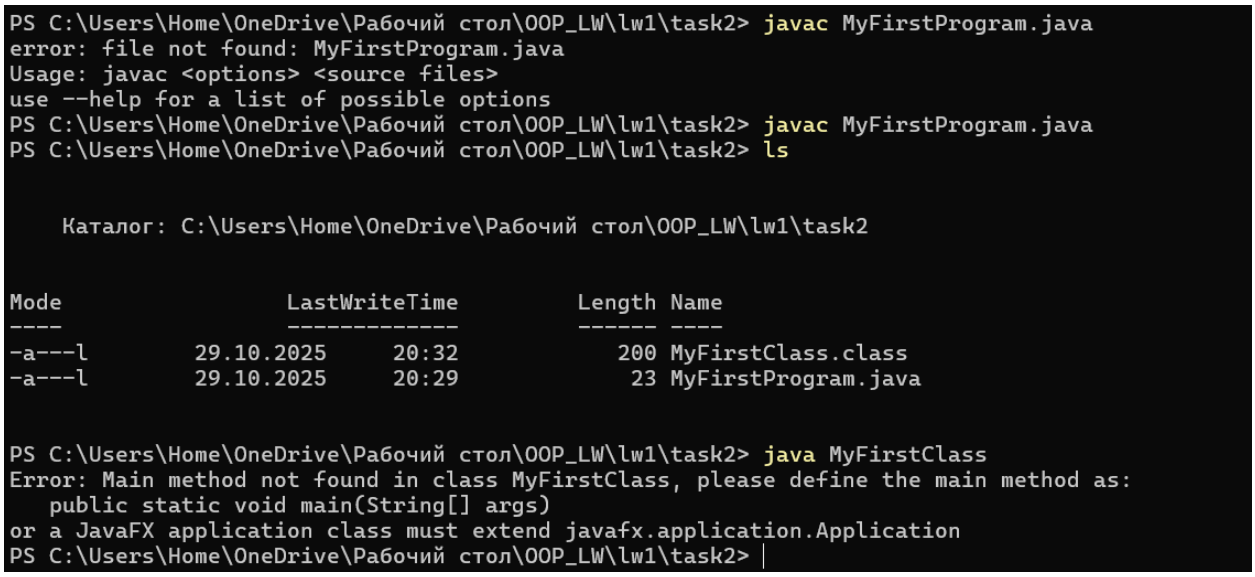

Рис. 3

Исходный код программы первоначально содержал пустой класс MyFirstClass без метода main. После компиляции был получен файл MyFirstClass.class. При попытке запуска программы с помощью команды java MyFirstClass виртуальная машина Java сообщила об ошибке - метод main не был найден в классе (См. рис. 4).

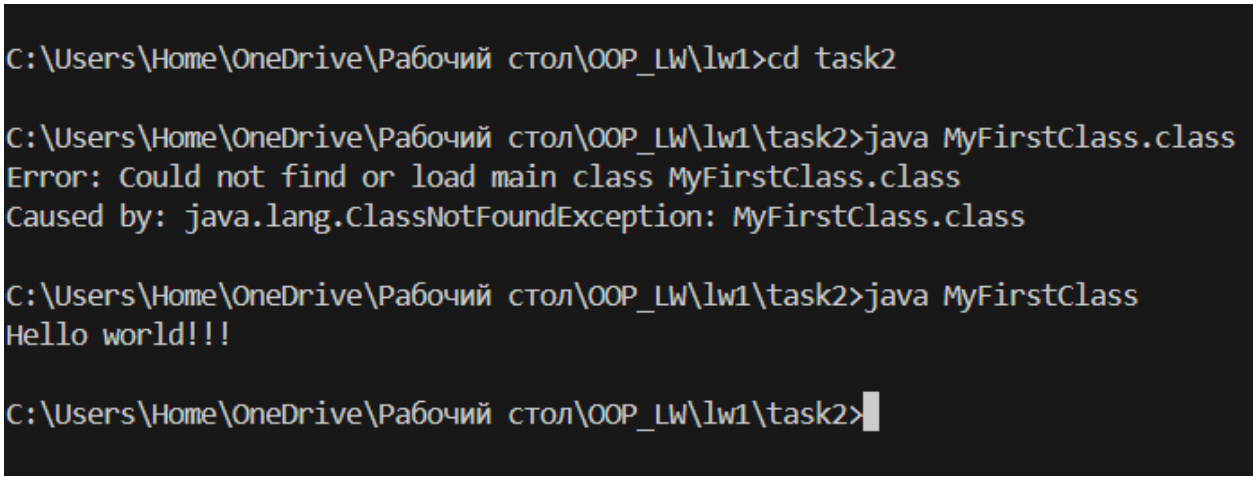

Рис. 4

Для исправления ошибки в класс MyFirstClass был добавлен метод main. Изначально метод был объявлен без модификатора static, что также вызывало ошибку при запуске.

Таким образом, “методом ошибок”, мы поняли, что для успешного запуска программы класс должен содержать метод main с правильной сигнатурой: public static void main(String[] args). Также мы заметили важное различие в формате команд для компилятора и виртуальной машины - при компиляции указывается имя файла с расширением .java, а при запуске - имя класса без расширения.

После исправления команды на java MyFirstClass программа была успешно запущена. Результат выполнения программы также продемонстрирован на рисунке 4.

На начальном этапе выполнения задания была предпринята попытка компиляции из корневой директории lw1, где файл MyFirstProgram.java отсутствовал, что закономерно привело к ошибке "file not found". После перехода в целевую директорию task2 с помощью команды cd task2 компиляция была выполнена успешно. Команда javac MyFirstProgram.java создала байт-код класса MyFirstClass в файле MyFirstClass.class.

Задание 3

Для выполнения задания мы изменили метод main класса MyFirstClass для обработки аргументов командной строки. Исходный код был изменен - программа теперь содержит цикл, который последовательно выводит все переданные аргументы.

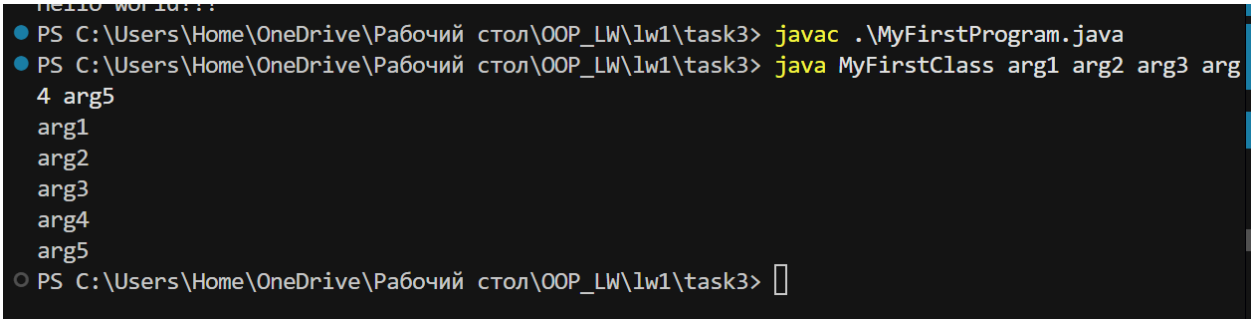

Рис. 5

Для демонстрации работы с аргументами программа была запущена с шестью параметрами: arg1, arg2, arg3, arg4, arg5. При выполнении команды “java MyFirstClass arg1 arg2 arg3 arg4 arg5” программа обработала массив строк String[] s, содержащий переданные аргументы, и вывела каждый из них на отдельной строке. (См. рис. 5)

Задание 4

В данном задании была расширена программа путем добавления второго класса MySecondClass. При первой попытке запуска программы была допущена ошибка в формате команды - указание пути к файлу с точкой и слешем .\MyFirstClass привело к ошибке загрузки класса (рис. 6). После исправления команды на корректный формат java MyFirstClass программа была успешно выполнена.

```
1615141312111109
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task4> java .\MyFirstClass
Error: Could not find or load main class .\MyFirstClass
Caused by: java.lang.ClassNotFoundException: /\MyFirstClass
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task4> javac .\MyFirstProgram.java
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\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
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task4>
```

Рис. 6

Класс MySecondClass был реализован с двумя приватными целочисленными полями, методами доступа к ним (геттерами и сеттерами), конструктором для инициализации и методом, выполняющим математическую операцию над полями. В данном случае операцией было выбрано вычитание второго числа из первого.

Основной класс MyFirstClass был модифицирован для создания объекта MySecondClass и использования его в двойном цикле для формирования матрицы 8x8.

В результате программа успешно вывела на экран матрицу 8x8, где каждый элемент представляет собой разность между номером строки и номером столбца.

Задание 5

На данном этапе класс MySecondClass был вынесен в отдельный файл и размещен в поддиректории myfirstpackage. Первоначальная попытка компиляции основного файла MyFirstProgram.java завершилась ошибками. Компилятор сообщил о невозможности доступа к классу MySecondClass и о том, что class-файл содержит неправильный класс (рис. 7).

```
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task5> javac .\myfirstpackage\MySecondClass
.java
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task5> javac .\MyFirstProgram.java
.\MyFirstProgram.java:7: error: cannot access MySecondClass
        MySecondClass o = new MySecondClass(0, 10);
        ^
  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:7: error: cannot find symbol
        MySecondClass o = new MySecondClass(0, 10);
                              ^
  symbol:   class MySecondClass
  location: class MyFirstClass
2 errors
```

Рис. 7

Ошибки возникли по двум причинам: во-первых, класс MySecondClass теперь должен быть объявлен в пакете myfirstpackage, а во-вторых, в основном классе необходимо добавить соответствующую директиву import для доступа к классу из пакета.

Класс MySecondClass был модифицирован - в начало файла добавлена строка package myfirstpackage;, указывающая принадлежность класса к пакету. Соответственно, в файле MyFirstProgram.java была добавлена директива import myfirstpackage.MySecondClass; для импорта класса из пакета.

После внесения необходимых изменений программа была успешно скомпилирована(См. рис. 8).

```
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task5> javac .\myfirstpackage\MySecondClass
.java
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task5> javac .\MyFirstProgram.java
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task5> 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
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task5>
```

Рис. 8

Задание 6

На начальном этапе выполнения задания была изучена утилита jar с помощью команды jar --help, которая вывела подробную справку по использованию архиватора (рис. 9).

```
PS C:\Users\Home\OneDrive\Рабочий стол\OOP_LW\lw1\task5> jar --help
Usage: jar [OPTION...] [ [--release VERSION] [-C dir] files] ...
jar creates an archive for classes and resources, and can manipulate or
restore individual classes or resources from an archive.

 Examples:
 # Create an archive called classes.jar with two class files:
 jar --create --file classes.jar Foo.class Bar.class
 # Create an archive using an existing manifest, with all the files in foo/:
 jar --create --file classes.jar --manifest mymanifest -C foo/ .
 # Create a modular jar archive, where the module descriptor is located in
 # classes/module-info.class:
 jar --create --file foo.jar --main-class com.foo.Main --module-version 1.0
     -C foo/ classes resources
 # Update an existing non-modular jar to a modular jar:
 jar --update --file foo.jar --main-class com.foo.Main --module-version 1.0
     -C foo/ module-info.class
 # Create a multi-release jar, placing some files in the META-INF/versions/9 directory:
 jar --create --file mr.jar -C foo classes --release 9 -C foo9 classes
```

Рис. 9

Для создания исполняемого JAR архива был подготовлен файл manifest.mf, содержащий необходимые метаданные: версию манифеста, информацию о создателе и главный класс программы. Файл манифеста был создан с соблюдением формата - с пустой строкой в конце, что является обязательным требованием.

Создание архива выполнено командой jar cfm MyFirstProgram.jar manifest.mf *.class myfirstpackage, где ключи cfm указывают на создание архива с использованием указанного файла манифеста. В архив были включены все class-файлы из текущей директории и папка myfirstpackage с ее содержимым.

В результате программа успешно упакована в JAR архив и запущена командой java -jar MyFirstProgram.jar. На экран выведена та же матрица 8x8 с результатами вычитания, что и в предыдущих заданиях, что подтверждает корректную работу архива (рис. 10). Все классы и пакеты были правильно упакованы, а указание главного класса в манифесте обеспечило возможность запуска архива напрямую.

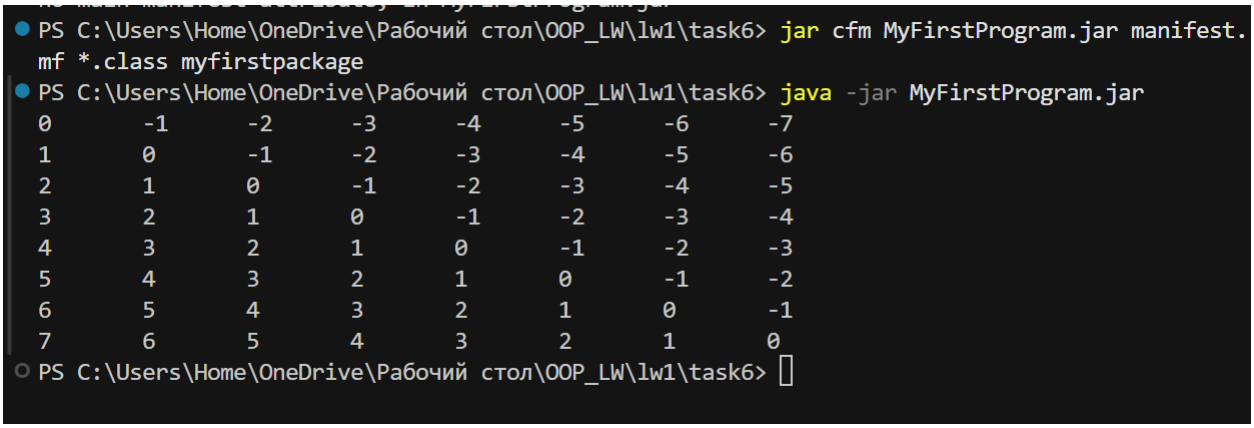

Рис. 10