



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.04.01 Информатика и вычислительная техника
МАГИСТЕРСКАЯ ПРОГРАММА 09.04.01/12 Интеллектуальный анализ больших
данных в системах поддержки принятия решений

О Т Ч Е Т
по лабораторной работе № 1

Название: Введение, классы, объекты

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-23М
(Группа)

М.А. Гейне
(Подпись, дата) (И.О. Фамилия)

Преподаватель

П.В. Степанов
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

ЗАДАНИЕ

Вариант 1:

4. Создать приложение для ввода пароля из командной строки и сравнения его со строкой-образцом.

5. Создать программу ввода целых чисел как аргументов командной строки, подсчета их суммы (произведения) и вывода результата на консоль.

Вариант 2:

Ввести с консоли n целых чисел и поместить их в массив. На консоль вывести:

4. Числа, которые делятся на 5 и на 7.

5. Элементы, расположенные методом пузырька по убыванию модулей.

Создание проекта

Лабораторные работы было решено выполнять на языке программирования Scala по согласованию с преподавателем.

Для лабораторных работ был создан общий проект инструмента сборки sbt, в котором для каждой лабораторной работы создаётся отдельный под-проект (Multi-project build). В конфигурационном файле указаны параметры сборки как общие для всех проектов, так и частные для каждого под-проекта. К общим параметрам относятся версия Scala и название организации. Для под-проектов устанавливаются такие параметры, как имена их директорий, имена проектов, их зависимости и параметры для выполнения заданий над ними. Текст файла конфигурации сборки sbt приведён в листинге 1.

Листинг 1 – Код build.sbt

```
ThisBuild / scalaVersion := "3.2.2"
ThisBuild / organization := "ru.bmstu"

lazy val general = (project in file("general"))
  .settings(
    name := "General"
  )

lazy val lab1 = (project in file("lab1"))
  .settings(
    name := "Lab 1",
    assembly / assemblyJarName := "lab1.jar",
    assembly / target := file("exec/")
  )
  .dependsOn(general)
```

В настоящий момент объявлено 2 проекта: general и lab1. В проекте general планируется хранить общие для всех проектов функции, которые могут быть многократно использованы. Примером такого объявления является оператор «|>», используемый в некоторых функциональных языках программирования (например, OCaml) для связывания выхода функции слева от оператора с входом функции справа от оператора. Весь код проекта в настоящее время записан в 1 файле и приведён в листинге 2.

Листинг 2 – Исходный код General.scala

```
package bmstu.general

import scala.util.chaining._

// a pipe operator from OCaml PL
extension [A,B](a:A)
  infix def |>(f: A => B): B = a.pipe(f)
```

В проекте lab1 были выполнены задания лабораторных работ 1 и 2. Проект организован в стиле стандартных проектов на языке Scala. Структура проекта представлена в листинге 3.

Листинг 3 – Структура проекта lab1

```
lab1/
├── src
│   ├── main
│   │   └── scala
│   │       ├── main.scala
│   │       ├── var1
│   │       │   ├── task1.scala
│   │       │   └── task2.scala
│   │       └── var2
│   │           ├── task1.scala
│   │           └── task2.scala
└── target
    ├── scala-3.2.2
    │   └── *output omitted*
    └── streams
        └── *output omitted*
```

183 directories, 94 files

В директории var1 хранятся исходные коды решений задач ЛР1. Программа main.scala является начальной точкой исполнения всех задач; её код приведён в листинге 4.

Листинг 4 – Исходный код main.scala

```
import var1._
import var2._
import java.util.Date

@main def main(args: String* ): Unit =
  val arglist = args.toList
  arglist match {
    case "-p"::tasks::tail if isListOfInts(tail) => run_tasks(tasks,
(tail.map(_._toInt).toSeq))
    case "--help"::tail => print_help()
    case tail if isListOfInts(tail) => run_tasks("1234",
(tail.map(_._toInt).toSeq))
    case tail => illegal_args(tail.mkString(" "))
  }
  println(s""""|Developer: mikeGEINE
|Task recieved on: Fri Feb 17 15:39:00 MSK 2023
|Task completed (this run) on: ${new Date()}""").stripMargin)

def run_tasks(tasks: String, int: Seq[Int]) =
  tasks.split("").foreach { key =>
    key match {
      case "1" =>{
        println("Variant 1")
        println("Question 4")
        var1.task1.main()
      }
      case "2" => {
        println("Variant 1")
        println("Question 5")
        var1.task2.main(int)
      }
      case "3" => {
        println("Variant 2")
        println("Question 4")
        var2.task1.main()
      }
      case "4" => {
        println("Variant 2")
        println("Question 5")
        var2.task2.main()
      }
      case arg => illegal_args("Task %s "+arg)
    }
    println("-----")
  }
}
```

```

def isListOfInts(lst: List[String]) =
  lst.map(_.toIntOption).filter(_.isEmpty).isEmpty

def print_help() =
  println("""
lab1 [OPTIONS] [INTEGERS]
[OPTIONS]
  --help      Prints help message
  -p TASKS    Allows partial execution of tasks. TASK is a sting of digits
from 1 to 2.
[INTEGERS]
  List of integers separated by space; used for Task 2 (Sum and mul)
""")

def illegal_args(arg: Any) =
  println("Illegal arguments found: "+arg)
  println("Usage:")
  print_help()

```

Аннотация `@main` позволяет описать точку входа в программу в виде метода вместо класса, как было необходимо делать в более ранних версиях Scala. В действительности будет создан класс с именем, совпадающим с именем метода; в этом классе будет создан метод `main`, совпадающий с сигнатурой метода `main` из Java с `Array[String]` в качестве аргумента и возвращаемым типом `Unit`; в сгенерированном методе `main` будет вызван изначально написанный метод с аргументами, разобранными из `Array[String]`.

В файле описано простое приложение для работы в консольном режиме. Оно разбирает поступившие на вход аргументы командной строки, проверяет их на корректность и запускает нужные задания. Приложение позволяет запустить либо все задания одновременно, либо только некоторые из них, если будет задан ключ «-p». Программа также содержит краткое сообщение с инструкцией по использованию и сообщение об ошибке в случае, если что-то пошло не так. Далее рассмотрим решения задач ЛР1.

Задача 1

Текст задания: создать приложение для ввода пароля из командной строки и сравнения его со строкой-образцом.

В приложении задаётся строка-образец `password`. Приложение запрашивает у пользователя пароль, считывает его из командной строки и сравнивает с образцом. Результат сравнения выводится в терминал. Текст программы приведён в листинге 5.

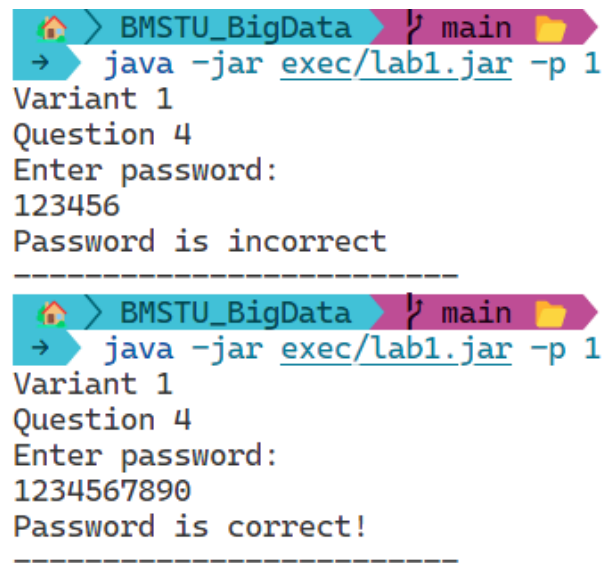
Листинг 5 – Текст решения задачи 1

```
package var1.task1
import bmstu.general._
import scala.io.StdIn

val password = "1234567890"

def main() =
  println("Enter password: ")
  val input = StdIn.readLine()
  (if password.equals(input) then "Password is correct!" else "Password is
incorrect") |> (println(_))
```

Результат работы программы приведён на рисунке 1.



```
> BMSTU_BigData main
→ java -jar exec/lab1.jar -p 1
Variant 1
Question 4
Enter password:
123456
Password is incorrect
-----
> BMSTU_BigData main
→ java -jar exec/lab1.jar -p 1
Variant 1
Question 4
Enter password:
1234567890
Password is correct!
-----
```

Рисунок 1 – Результат решения задачи 1

Задача 2

Текст задания: создать программу ввода целых чисел как аргументов командной строки, подсчета их суммы (произведения) и вывода результата на консоль.

Аргументы командной строки принимает основная программа. В ней предусмотрена проверка, что кроме аргументов ключа «-p» в аргументах командной строки содержатся только целые числа, выполняется преобразование аргументов в последовательность целых чисел (которая, теоретически, может быть и пустой), а затем вызывается нужная подпрограмма с этой последовательностью в аргументах.

Подпрограмма второй задачи в аргументах получает последовательность целых чисел и подсчитывает сумму и произведение с помощью функции `fold` и

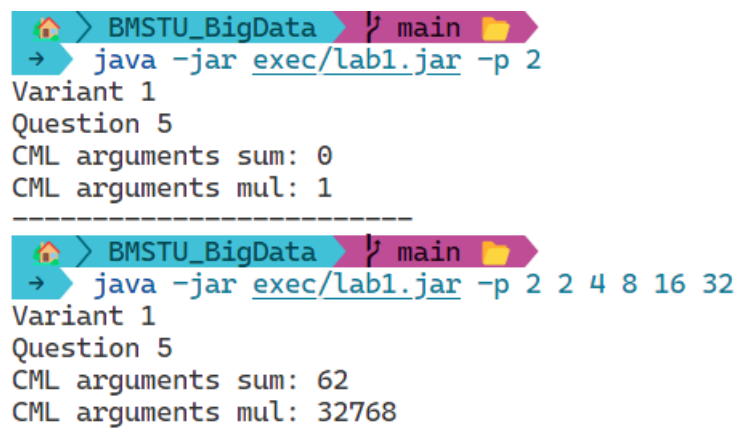
выводит в консоль. Первый аргумент fold должен быть нейтральным, т.е. для сложения 0, а для умножения 1. Текст решения приведён в листинге 6.

Листинг 6 – Текст решения задачи 2

```
package var1.task2
import scala.io.StdIn
import bmstu.general

def main(arr: Seq[Int] ): Unit =
  println(s"CML arguments sum: ${arr.fold(0)(_ + _)}")
  println(s"CML arguments mul: ${arr.fold(1)(_ * _)}")
```

Результат работы программы приведён на рисунке 2.



```
> BMSTU_BigData } main
→ java -jar exec/lab1.jar -p 2
Variant 1
Question 5
CML arguments sum: 0
CML arguments mul: 1
-----
> BMSTU_BigData } main
→ java -jar exec/lab1.jar -p 2 2 4 8 16 32
Variant 1
Question 5
CML arguments sum: 62
CML arguments mul: 32768
-----
```

Рисунок 2 – Решение задачи 2

Рисунок 2 иллюстрирует, что возможно исполнение программы без аргументов в виде последовательности целых чисел.

Задача 3

Текст задания: ввести с консоли n целых чисел и поместить их в массив.

На консоль вывести числа, которые делятся на 5 и на 7.

Пользователь получает приглашение на ввод строки чисел, разделённых пробелами. Программа разбивает строку по пробелам и конвертирует элементы полученного массива к типу Integer. Затем с помощью функции filter проверяется, что число делимо на 5 и на 7 (проверяется, что остаток от деления на число равен 0). Все числа, удовлетворяющие комбинированному условию, сохраняются в массив, объединяются в строку через пробел и выводятся в консоль. Текст решения приведён в листинге 7.

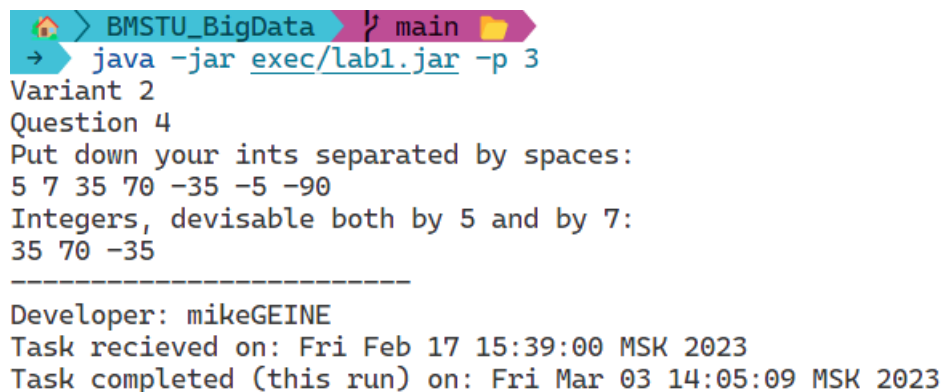
Листинг 7 – Решение задачи 3

```
package var2.task1
import scala.io.StdIn

def main() =
  println("Put down your ints separated by spaces:")
  val str = StdIn.readLine()
  val ints = str.split(" ").map(_.toInt)
  val filtered = ints.filter( i => (divisible_by(i, 5) && divisible_by(i, 7)))
  println(s"""Integers, devisable both by 5 and by 7:
            |${filtered.mkString(" ")}""").stripMargin)

def divisible_by(n: Int, div: Int) =
  n % div == 0
```

Результат работы программы приведён на рисунке 3.



```
> BMSTU_BigData main
→ java -jar exec/lab1.jar -p 3
Variant 2
Question 4
Put down your ints separated by spaces:
5 7 35 70 -35 -5 -90
Integers, devisable both by 5 and by 7:
35 70 -35
-----
Developer: mikeGEINE
Task recieved on: Fri Feb 17 15:39:00 MSK 2023
Task completed (this run) on: Fri Mar 03 14:05:09 MSK 2023
```

Рисунок 3 – Решение задачи 3

Задача 4

Текст задания: ввести с консоли n целых чисел и поместить их в массив.

На консоль вывести элементы, расположенные методом пузырька по убыванию модулей.

Ввод данных производится так же, как и в задаче 3. Элементы сортируются стандартным методом пузырька, в котором элемент меняется местами со следующим, если его абсолютное значение меньше абсолютного значения следующего. Текст решения задачи приведён в листинге 8.

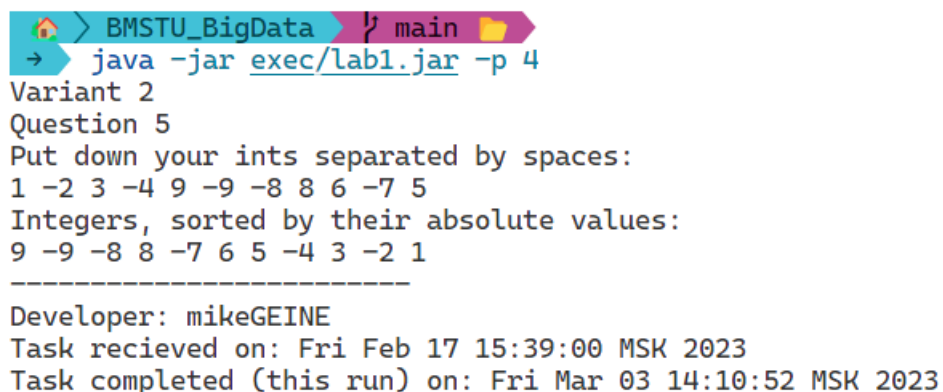
Листинг 8 – Решение задачи 4

```
package var2.task2
import scala.io.StdIn

def main() =
  println("Put down your ints separated by spaces:")
  val str = StdIn.readLine()
  val ints = str.split(" ").map(_.toInt)
  val sorted = bubbleSortByAbsDescending(ints)
  println(s"""Integers, sorted by their absolute values:
    |${sorted.mkString(" ")}""").stripMargin)

def bubbleSortByAbsDescending(arr: Array[Int]): Array[Int] = {
  var swapped = true
  var end = arr.length - 1
  while (swapped) {
    swapped = false
    for (i <- 0 until end) {
      if (math.abs(arr(i)) < math.abs(arr(i + 1))) {
        val temp = arr(i)
        arr(i) = arr(i + 1)
        arr(i + 1) = temp
        swapped = true
      }
    }
    end -= 1
  }
  arr
}
```

Результат работы программы приведён на рисунке 3.



```
> BMSTU_BigData main
→ java -jar exec/lab1.jar -p 4
Variant 2
Question 5
Put down your ints separated by spaces:
1 -2 3 -4 9 -9 -8 8 6 -7 5
Integers, sorted by their absolute values:
9 -9 -8 8 -7 6 5 -4 3 -2 1
-----
Developer: mikeGEINE
Task recieved on: Fri Feb 17 15:39:00 MSK 2023
Task completed (this run) on: Fri Mar 03 14:10:52 MSK 2023
```

Рисунок 4 – Решение задачи 4

ВЫВОД

Изучены базовые возможности и особенности языка программирования Scala.

Изучены способы работы со стандартным средством сборки проектов sbt. Изучены способы создания мульти-проектов, задания настроек для всего проекта и отдельно для под-проектов. Изучены методы сборки и запуска проектов.

Написано базовое консольное приложение, позволяющее производить частичное выполнение заданий, производящее разбор аргументов командной строки и имеющее встроенную справку. Изучены средства работы с вводом/выводом. Изучен ряд функций языка, в том числе функции высшего порядка map, fold, filter.