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 № 8

Название: Потоки (Threads)

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-23М (Группа) _____ (Подпись, дата) М.А. Гейне (И.О. Фамилия)

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

Москва, 2023

ЗАДАНИЕ

Вариант 1:

1. Реализовать многопоточное приложение “Банк”. Имеется банковский счет. Сделать синхронным пополнение и снятие денежных средств на счет/со счет случайной суммой. При каждой операции (пополнения или снятие) вывести текущий баланс счета. В том случае, если денежных средств недостаточно – вывести сообщение.
2. Реализовать многопоточное приложение “Робот”. Надо написать робота, который умеет ходить. За движение каждой его ноги отвечает отдельный поток. Шаг выражается в выводе в консоль LEFT или RIGHT.

Задача 1

Для решения задачи описывается банк с несколькими потоками, в которых запускаются процессы снятия и внесения случайных сумм. Потоки запускаются одновременно и при исполнении ожидают завершения других потоков перед началом работы. Код решения приведён ниже.

```
import scala.util.Random

object Bank:
  def run(): Unit =
    val account = new BankAccount()
    val threads = Seq(
      new Thread(new Depositor(account)),
      new Thread(new Withdrawer(account)),
      new Thread(new Depositor(account)),
      new Thread(new Withdrawer(account)),
      new Thread(new Depositor(account)),
      new Thread(new Withdrawer(account))
    )
    threads.foreach(_.start())
    threads.foreach(_.join())

  class BankAccount(var balance: Int = 0):
    def deposit(amount: Int): Unit = synchronized {
      balance += amount
      println(s"Deposited $amount, balance is now $balance")
    }

    def withdraw(amount: Int): Unit = synchronized {
      if (balance < amount)
        println("Insufficient funds")
      else
        balance -= amount
        println(s"Withdrew $amount, balance is now $balance")
    }

  class Depositor(account: BankAccount) extends Runnable:
    override def run(): Unit =
      for (_ <- 1 to 5) {
        val amount = Random.nextInt(100)
        account.deposit(amount)
        Thread.sleep(1000)
      }

  class Withdrawer(account: BankAccount) extends Runnable:
    override def run(): Unit =
      for (_ <- 1 to 5) {
```

```
        val amount = Random.nextInt(100)
        account.withdraw(amount)
        Thread.sleep(1000)
      }

def test(): Unit =
  Bank.run()
```

Результат решения задачи представлен ниже.

```
Variant 1
Question 1
Insufficient funds
Insufficient funds
Insufficient funds
Deposited 48, balance is now 48
Deposited 68, balance is now 116
Deposited 41, balance is now 157
Withdrew 62, balance is now 95
Deposited 48, balance is now 143
Deposited 7, balance is now 150
Deposited 86, balance is now 236
Withdrew 86, balance is now 150
Withdrew 38, balance is now 112
Withdrew 1, balance is now 111
Deposited 7, balance is now 118
Deposited 28, balance is now 146
Deposited 82, balance is now 228
Withdrew 97, balance is now 131
Withdrew 14, balance is now 117
Withdrew 65, balance is now 52
Deposited 85, balance is now 137
Deposited 93, balance is now 230
Deposited 52, balance is now 282
Withdrew 99, balance is now 183
Withdrew 72, balance is now 111
Withdrew 84, balance is now 27
Deposited 7, balance is now 34
Deposited 13, balance is now 47
Deposited 58, balance is now 105
Withdrew 74, balance is now 31
Insufficient funds
-------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 31 15:05:00 MSK 2023
Task completed (this run) on: Thu May 25 20:38:20 MSK 2023
```

Задача 2

Для решения задачи описывается класс `Robot` с двумя ногами и двумя потоками, по одному на каждую ногу. Каждая нога ходит до тех пор, пока не

будет остановлена. После совершения шага поток останавливается на произвольный период времени. Обе ноги останавливаются через 5 секунд с начала работы программы. Решение задачи приводится ниже.

```
import scala.util.Random

object Robot:
  def run(): Unit =
    val leftLeg = new Leg("LEFT")
    val rightLeg = new Leg("RIGHT")

    val leftThread = new Thread(new Runnable {
      override def run(): Unit =
        leftLeg.walk()
    })

    val rightThread = new Thread(new Runnable {
      override def run(): Unit =
        rightLeg.walk()
    })

    leftThread.start()
    rightThread.start()

    Thread.sleep(5000) // Allow 5 seconds for the robot to walk

    leftLeg.stop()
    rightLeg.stop()

class Leg(name: String):
  private var isWalking = true

  def walk(): Unit =
    while (isWalking) {
      println(s"$name step")
      Thread.sleep(Random.nextInt(1000)) // Random delay up to 1 second
    }

  def stop(): Unit =
    isWalking = false

def test(): Unit =
  Robot.run()
```

Результат работы программы приведён ниже.

```
Variant 1
Question 2
```

```
LEFT step
RIGHT step
LEFT step
RIGHT step
RIGHT step
RIGHT step
LEFT step
RIGHT step
LEFT step
RIGHT step
LEFT step
RIGHT step
RIGHT step
LEFT step
RIGHT step
RIGHT step
LEFT step
LEFT step
RIGHT step
LEFT step
RIGHT step
RIGHT step
-------------------------
Developer: mikeGEINE
Task recieved on: Fri Mar 31 15:05:00 MSK 2023
Task completed (this run) on: Thu May 25 20:41:52 MSK 2023
```

ВЫВОДЫ

Изучены способы работы с потоками в Scala.

Изучены способы запуска потоков, назначения функций исполнения в них и их синхронизации.