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 **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

по лабораторной работе № 8

Вариант 18

Название: Потоки в Java

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-23М (Группа) _____ (Подпись, дата) Д.Н. Хныкин (И.О. Фамилия)

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

Москва, 2022

Цель работы

Получение навыков работы с потоками в Java.

Выполнение

Задание 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 “Магазин”. Вся цепочка: производитель-магазин-покупатель. Пока производитель не поставит на склад продукт, покупатель не может его забрать. Реализовать приход товара от производителя в магазин случайным числом. В том случае, если товара в магазине не хватает– вывести сообщение.

Листинг выполнения подзадачи 1

#n1.java

```
package com.aranei;

import java.util.Random;
import java.util.concurrent.TimeUnit;

public class n1 {
    static void ThreadsBank() throws InterruptedException {
        Bank bank = new Bank();

        Runnable moneyIn = () -> {
            while (true) {
                System.out.print("Производится заработок денег: ");
                bank.addMoney();
                System.out.println("Счет: " + bank.getMoney());
                try {
                    TimeUnit.SECONDS.sleep(5);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        };

        Runnable moneyOut = () -> {
            while (true) {
                System.out.print("Производится расходование средств: ");
                bank.deleteMoney();
                System.out.println("Счет: " + bank.getMoney());
                try {
                    TimeUnit.SECONDS.sleep(3);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        };

        Thread shopIn = new Thread(moneyIn);
        Thread shopOut = new Thread(moneyOut);
```

```
            shopIn.start();
            TimeUnit.SECONDS.sleep(1);
            shopOut.start();
        }



        private static class Bank {
            private int Money = 0;

            public synchronized void addMoney() {
                Random random = new Random();
                int temp = random.nextInt(1000 - 500) + 500;
                Money += temp;
                System.out.println(temp);
            }


            public synchronized void deleteMoney() {
                Random random = new Random();
                int temp = random.nextInt(800 - 500) + 500;
                if (Money - temp > 0) {
                    Money -= temp;
                    System.out.println(temp);
                } else {
                    System.out.println(temp + " - Казна пуста, Милорд!");
                }
            }
            public synchronized int getMoney() {
                return Money;
            }
        }
}
```

#n3.java

```
package com.aranei;

import java.util.Random;
import java.util.concurrent.TimeUnit;

public class n3 {
   static void ThreadsStore() throws InterruptedException {
      Store store = new Store();

      Runnable productsIn = () -> {
         while (true) {
            System.out.print("Производится поставка товаров: ");
            store.addProducts();
            System.out.println("Склад: " + store.getProducts());
            try {
               TimeUnit.SECONDS.sleep(5);
            } catch (InterruptedException e) {
               e.printStackTrace();
            }
         }
      };

      Runnable productsOut = () -> {
         while (true) {
            System.out.print("Производится покупка товаров: ");
            store.deleteProducts();
            System.out.println("Склад: " + store.getProducts());
            try {
               TimeUnit.SECONDS.sleep(3);
            } catch (InterruptedException e) {
               e.printStackTrace();
            }
         }
      };
      Thread shopIn = new Thread(productsIn);
      Thread shopOut = new Thread(productsOut);

      shopIn.start();
      TimeUnit.SECONDS.sleep(1);
      shopOut.start();
   }
```

```
        private static class Store {
            private int Products = 0;

            public synchronized void addProducts() {
                Random random = new Random();
                int temp = random.nextInt(10 - 5) + 5;
                Products += temp;
                System.out.println(temp);
            }


            public synchronized void deleteProducts() {
                Random random = new Random();
                int temp = random.nextInt(8 - 5) + 5;
                if (Products - temp > 0) {
                    Products -= temp;
                    System.out.println(temp);
                } else {
                    System.out.println(temp + " - Полки пусты, Милорд!");
                }
            }
            public synchronized int getProducts() {
                return Products;
            }
        }
    }
```

Ссылка на программное решение

Программное решение представлено в репозитории распределённой системы управления версиями Git:

https://github.com/Aranei99/labsJava/tree/main/LB8

Вывод

При выполнении лабораторной работы были получены навыки работы с потоками в Java.