**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 6**

**Название:** Потоки (Threads)

**Дисциплина:** Языки программирования для работы с большими данными

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

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

Москва, 2022

**Цель работы:** получение навыков работы с потоками в Java.

**Задание 1:**

1. 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.

2. 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.

**Подзадача 1.**

Код программы:

```
import java.util.Random;

public class Supplier extends Thread {
    private Shop shop;

    Supplier(Shop shop) {
        this.shop = shop;
    }

    @Override
    public void run() {
        Random random = new Random();

        try {
            while (true) {
                Thread.sleep(millis: 3000);

                shop.sendWeapon(random.nextInt(bound: 9));
            }
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}
```

```java
public class Weapon {
    int id;
    String name;

    Weapon(int id, String name) {
        this.id = id;
        this.name = name;
    }
}
```

```java
public class Shop extends Thread {
    BlockingQueue<Order> orders;
    ArrayList<Weapon> weaponStock;

    final String[] weapons = {
        "2-дюймовый миномёт",
        "10 cm K 14",
        "Type-97",
        "7,5 cm KwK 37",
        "AK-12",
        "3М6 \"Шмель\"",
        "8-дюймовая гаубица Mark I – V",
        "T-34",
        "Jagdpanzer 38"
    };

    Integer[] weaponCount = {0, 0, 0, 0, 0, 0, 0, 0, 0};

    Shop() {
       weaponStock = new ArrayList<Weapon>();
       orders = new LinkedBlockingDeque<Order>();
    }

    public void sendWeapon(int id) {
        weaponCount[id]++;

        System.out.println("В магазине появилось: " + weapons[id]);
    }

    public void viewStock() {
        for (int i = 0; i < weapons.length; i++) {
            System.out.println("[" + i + "] (" + weapons[i] + "), кол-во: " + weaponCount[i]);
        }
    }
```

```java
private void sell(int id, int count) {
    if (id < weapons.length) {
        if (weaponCount[id] >= count) {
            weaponCount[id] -= count;

            System.out.println(weapons[id] + " (" + count + ")" + " успешно приобретен! Поздравляем с покупкой!");
        }
        else {
            System.out.println("Товара [" + weapons[id] + "] недостаточно на складе");
        }
    }
    else {
        System.out.println(x: "Неверный ID товара");
    }
}

public void makeOrder(Order order) {
    orders.add(order);
}

@Override
public void run() {
    while (true) {
        try {
            Thread.sleep(millis: 1000);

            while (orders.size() > 0) {
                Order order = orders.take();
                sell(order.id, order.count);
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```java
import java.util.Scanner;

public class Laba1 {
    Run | Debug
    public static void main(String[] args) {
        Shop shop = new Shop();
        shop.start();

        Supplier supplier = new Supplier(shop);
        supplier.start();

        Scanner in = new Scanner(System.in);
        String command;

        do {
            command = in.nextLine();

            if ("weapons".equals(command)) shop.viewStock();
            if (command.startsWith(prefix: "order")) {
                String[] splitted = command.split(regex: " ");
                shop.makeOrder(new Order(Integer.parseInt(splitted[1]), Integer.parseInt(splitted[2])));
            }

        } while (!"exit".equals(command));

        in.close();

        shop.stop();
        supplier.stop();
    }
}
```

**Результат выполнения программы:**

```
В магазине появилось: ЗМб "Шмель"
В магазине появилось: Type-97
В магазине появилось: T-34
В магазине появилось: АК-12
цуВ магазине появилось: АК-12
аВ магазине появилось: Type-97
pon
weapons
[0] (2-дюймовый миномёт), кол-во: 0
[1] (10 cm K 14), кол-во: 0
[2] (Type-97), кол-во: 2
[3] (7,5 cm KwK 37), кол-во: 0
[4] (АК-12), кол-во: 2
[5] (ЗМб "Шмель"), кол-во: 1
[6] (8-дюймовая гаубица Mark I ? V), кол-во: 0
[7] (T-34), кол-во: 1
[8] (Jagdpanzer 38), кол-во: 0
В магазине появилось: Jagdpanzer 38
В магазине появилось: Type-97
В магазине появилось: Jagdpanzer 38
order 2 3
В магазине появилось: ЗМб "Шмель"
Type-97 (3) успешно приобретен! Поздравляем с покупкой!
orВ магазине появилось: 10 cm K 14
der 2 3
Товара [Type-97] недостаточно на складе
В магазине появилось: Type-97
В магазине появилось: Type-97
stop
sxitВ магазине появилось: 10 cm K 14

В магазине появилось: 2-дюймовый миномёт
В магазине появилось: 8-дюймовая гаубица Mark I ? V
exit
```

**Подзадача 2.**

Код программы:

```
public class Account extends Thread {
    String name;
    int moneyCount;
    Boolean hasChanged;

    Account() {
        Random random = new Random();
        this.name = UUID.randomUUID().toString().substring(beginIndex: 0, endIndex: 4);
        this.moneyCount = random.nextInt(bound: 100000) + 100000;
        this.hasChanged = false;
    }

    public int increase(int count) {
        this.hasChanged = true;

        return moneyCount += count;
    }

    public int decrease(int count) {
        this.hasChanged = true;

        return moneyCount -= count;
    }

    @Override
    public void run() {
        while (true) {
            try {
                Thread.sleep(millis: 1);

                if (hasChanged) {
                    System.out.println("Счет " + this.name + " стал равен " + moneyCount);
                    hasChanged = false;
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    public String toString() {
        return "[" + name + "]: " + moneyCount;
    }
}
```

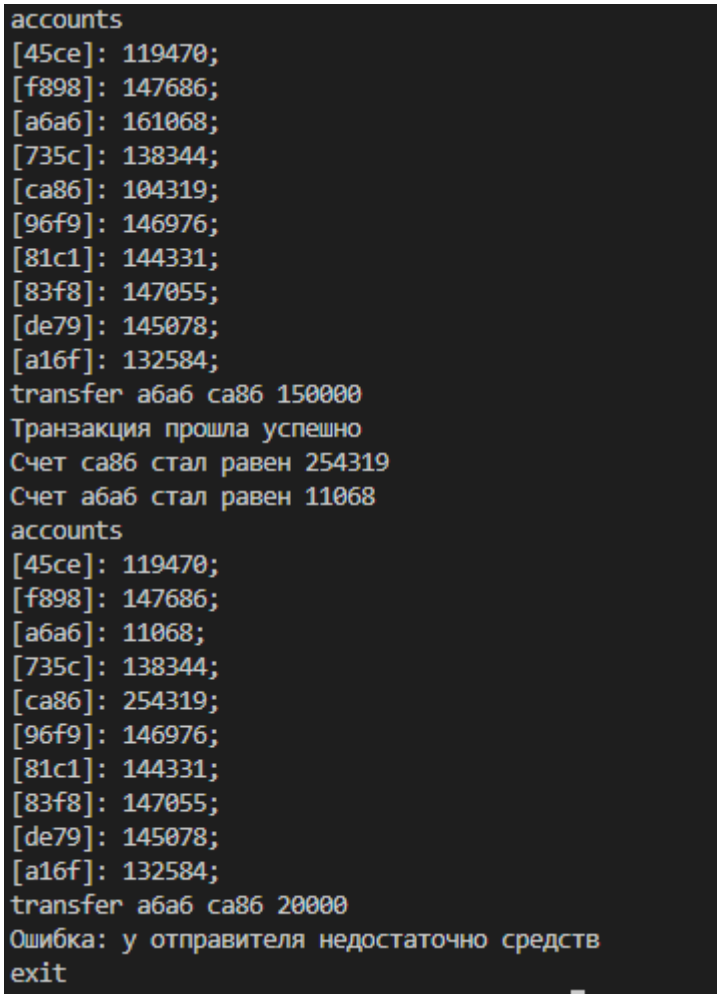
```java
public class Bank extends Thread {
    private BlockingQueue<Transaction> transactions;
    private ArrayList<Account> accounts;

    Bank() {
        this.transactions = new LinkedBlockingDeque<Transaction>();
        this.accounts = new ArrayList<Account>();
    }

    Bank(ArrayList<Account> accounts) {
        this.transactions = new LinkedBlockingDeque<Transaction>();
        this.accounts = accounts;
    }

    public void stopThreads() {
        for (Account account : accounts) account.stop();
        this.stop();
    }

    private void transfer(String from, String to, int count) {
        Account acc1 = accounts.stream().filter(a -> a.name.equals(from)).findAny().orElse(other: null);
        Account acc2 = accounts.stream().filter(a -> a.name.equals(to)).findAny().orElse(other: null);

        if (acc1 != null && acc2 != null) {
            if (acc1.moneyCount < count) {
                System.out.println(x: "Ошибка: у отправителя недостаточно средств");
            }
            else {
                acc1.decrease(count);
                acc2.increase(count);

                System.out.println(x: "Транзакция прошла успешно");
            }
        }
        else {
            System.out.println(x: "Ошибка: один или более из введенных аккаунтов не существуют");
        }
    }

    public void getAccounts() {
        accounts.forEach(a -> System.out.println(a.toString() + ";"));
    }
```

```java
public void makeTransaction(Transaction transaction) {
    transactions.add(transaction);
}

@Override
public void run() {
    while (true) {
        try {
            Thread.sleep(millis: 1000);

            if (transactions.size() != 0) {
                for (int i = 0; i < transactions.size(); i++) {
                    Transaction tr = transactions.poll();
                    transfer(tr.from, tr.to, tr.count);
                }
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```java
import java.util.Random;

public class Transaction {
    final int id;
    final String from;
    final String to;
    final int count;

    public Transaction(String from, String to, int count) {
        Random random = new Random();

        this.id = random.nextInt(random.nextInt(bound: 1000) + 1000);
        this.from = from;
        this.to = to;
        this.count = count;
    }
}
```

```java
import java.util.ArrayList;
import java.util.Scanner;

public class Laba2 {
    Run | Debug
    public static void main(String[] args) throws Exception {
        ArrayList<Account> accounts = new ArrayList<Account>();

        for (int i = 0; i < 10; i++) {
            Account account = new Account();
            accounts.add(account);
            account.start();
        }

        Bank bank = new Bank(accounts);
        bank.start();

        Scanner in = new Scanner(System.in);
        String command;

        do {
            command = in.nextLine();

            if ("accounts".equals(command)) {
                bank.getAccounts();
            }
            else if (command.startsWith(prefix: "transfer")) {
                String[] splitted = command.split(regex: " ");

                bank.makeTransaction(new Transaction(
                    splitted[1],
                    splitted[2],
                    Integer.parseInt(splitted[3])
                ));
            }

        } while (!"exit".equals(command));

        bank.stopThreads();
        in.close();
    }
}
```

**Результат выполнения программы:**




**Ссылка на программное решение:**

https://github.com/ArMaxik/BigDataLanguages/tree/main/lr8

**Вывод**: в ходе лабораторной работы были получены навыки работы с потоками в Java.