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

Вариант 6

Название: Потоки (Threads)

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-23М		Г.Л. Кушнир
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Цель: Изучить потоки в java.

Задание 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.

Код класса Main:

```
public class Main {
    private int balance;
    private static boolean finished = false;

    public Main(int initialBalance) {
        this.balance = initialBalance;
    }

    public synchronized void deposit(int amount) {
        balance += amount;
        System.out.println("Top up amount " + amount + " rub. Current
balance: " + balance + " rub");
    }

    public synchronized void withdraw(int amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawal amount " + amount + " rub. Current
balance: " + balance + " rub");
        } else {
            System.out.println("Insufficient funds for withdrawal " + amount
+ " rub");
        }
    }

    public static void main(String[] args) {
        Main account = new Main(1000);

        Thread depositThread = new Thread(() -> {
            Random rand = new Random();
            synchronized (account) {
                for (int i = 0; i < 5; i++) {
                    int amount = rand.nextInt(501); // случайная сумма от 0
до 500
                    account.deposit(amount);
                    account.notify();
                    try {
                        account.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                finished = true;
                account.notify(); // Для завершения работы
            }
        });
```

```
        Thread withdrawThread = new Thread(() -> {
            Random rand = new Random();
            synchronized (account) {
                while (!finished) {

                    int amount = rand.nextInt(1501); // случайная сумма от 0
до 500
                    account.withdraw(amount);
                    account.notify();
                    try {
                        account.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                account.notify();
            }
        });

        depositThread.start();
        withdrawThread.start();

        try {
            depositThread.join();
            withdrawThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Operations were stopped.");

    }

}
```

Вариант 3: 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.

Код класса Main:

```
public class Main {

    private int amountOfProduct;
    private static boolean finished = false;

    public Main(int initialAmmount){
        this.amountOfProduct = initialAmmount;
    }

    public synchronized void addProduct (int amount){
        amountOfProduct += amount;
        System.out.println("Adding " + amount + " of product to the shop.
Current amount of product in the shop: " + amountOfProduct);
    }
```

```
    public synchronized void buyProduct (int amount){
        if (amountOfProduct >= amount){
            amountOfProduct -= amount;
            System.out.println("Buying " + amount + " products from the shop.
Current amount of product in the shop: " + amountOfProduct);
        }
        else{
            System.out.println("Can't buy " + amount + " products");
        }
    }

    public static void main(String[] args) {

        Main shop = new Main(10);

        Thread AddingToTheShop = new Thread(() -> {
            Random rand = new Random();
            synchronized (shop){
                for (int i = 0; i < 10; i++){
                    int amount = rand.nextInt(5)+1;
                    shop.addProduct(amount);
                    shop.notify();
                    try {
                        shop.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                finished = true;
                shop.notify();
            }
        });

        Thread BuyingFromTheShop = new Thread(() -> {
            Random rand = new Random();
            synchronized (shop) {
                while (!finished) {

                    int amount = rand.nextInt(7)+1; // случайная сумма от 0
до 500
                    shop.buyProduct(amount);
                    shop.notify();
                    try {
                        shop.wait();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                shop.notify();
            }
        });

        AddingToTheShop.start();
        BuyingFromTheShop.start();

        try {
            AddingToTheShop.join();
            BuyingFromTheShop.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Shop is closed.");
```

```
    }

}
```

Работа программы представлена на рисунках 1-2.

```
Insufficient funds for withdrawal 1028 rub
Top up amount 288 rub. Current balance: 1288 rub
Withdrawal amount 429 rub. Current balance: 859 rub
Top up amount 212 rub. Current balance: 1071 rub
Withdrawal amount 754 rub. Current balance: 317 rub
Top up amount 400 rub. Current balance: 717 rub
Insufficient funds for withdrawal 782 rub
Top up amount 51 rub. Current balance: 768 rub
Insufficient funds for withdrawal 857 rub
Top up amount 169 rub. Current balance: 937 rub
Insufficient funds for withdrawal 1273 rub
Operations were stopped.
```

Рисунок 1 – Работа программы для 1 задания

```
Buying 5 products from the shop. Current amount of product in the shop: 5
Adding 4 of product to the shop. Current amount of product in the shop: 9
Buying 5 products from the shop. Current amount of product in the shop: 4
Adding 3 of product to the shop. Current amount of product in the shop: 7
Buying 3 products from the shop. Current amount of product in the shop: 4
Adding 3 of product to the shop. Current amount of product in the shop: 7
Buying 6 products from the shop. Current amount of product in the shop: 1
Adding 1 of product to the shop. Current amount of product in the shop: 2
Can't buy 3 products
Adding 5 of product to the shop. Current amount of product in the shop: 7
Buying 3 products from the shop. Current amount of product in the shop: 4
Adding 3 of product to the shop. Current amount of product in the shop: 7
Buying 5 products from the shop. Current amount of product in the shop: 2
Adding 4 of product to the shop. Current amount of product in the shop: 6
Buying 3 products from the shop. Current amount of product in the shop: 3
Adding 5 of product to the shop. Current amount of product in the shop: 8
Buying 5 products from the shop. Current amount of product in the shop: 3
Adding 5 of product to the shop. Current amount of product in the shop: 8
Buying 4 products from the shop. Current amount of product in the shop: 4
Adding 2 of product to the shop. Current amount of product in the shop: 6
Buying 4 products from the shop. Current amount of product in the shop: 2
Shop is closed.
```

Рисунок 2 – Работа программы для 2 задания

Вывод: Во время выполнения лабораторной работы были изучены потоки в java.