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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ 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

Название: Потоки

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-22М		М.А. Зотов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023 г.

Цель: получить опыт работы с потоками на языке Java.

Задание 1

Условие: 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:

```
import java.util.Random;

class BankAccount {
    private double balance;

    public BankAccount(double balance) {
        this.balance = balance;
    }

    public synchronized void deposit(double amount) {
        balance += amount;
        System.out.println("Пополнение: " +
String.format("%.2f",amount)
                + ", баланс: " + String.format("%.2f",balance));
    }

    public synchronized void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Снятие: " +
String.format("%.2f",amount)
                    + ", баланс: " +
String.format("%.2f",balance));
        } else {
            System.out.println("Недостаточно средств для снятия:
" + String.format("%.2f",amount)
                    + ", баланс: " +
String.format("%.2f",balance));
        }
    }
}

class BankTransaction implements Runnable {
    private final BankAccount account;

    public BankTransaction(BankAccount account) {
        this.account = account;
    }

    @Override
    public void run() {
```

```
        Random random = new Random();

        while (true) {
            // Случайная сумма для операции
            double amount = random.nextDouble() * 1000;

            // С вероятностью 50% проводим либо операцию
пополнения, либо операцию снятия
            if (random.nextBoolean()) {
                account.deposit(amount);
            } else {
                account.withdraw(amount);
            }

            // Задержка между операциями
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class ex1 {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(1000);

        Thread producerThread = new Thread(new
BankTransaction(account));
        Thread consumerThread = new Thread(new
BankTransaction(account));

        producerThread.start();
        consumerThread.start();
    }
}
```

Результат выполнения:

```
Пополнение: 590,86, баланс: 1590,86
Пополнение: 85,35, баланс: 1676,20
Снятие: 491,42, баланс: 1184,78
Снятие: 987,88, баланс: 196,90
Недостаточно средств для снятия: 825,86, баланс: 196,90
Недостаточно средств для снятия: 278,36, баланс: 196,90
Пополнение: 68,16, баланс: 265,06
```

Рисунок 1 – Результат выполнения 1

Задание 3

Условие: 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:

```
import java.util.Random;

class Store {
    private int products = 0;
    private final int maxProducts = 10;
    private final int minProducts = 1;

    public synchronized void get() {
        Random random = new Random();
        int newProducts = random.nextInt(maxProducts -
minProducts + 1) + minProducts;
        while (products < newProducts) {
            System.out.println("Покупатель не может купить " +
newProducts + " товаров, так как на складе лежит " + products +
" товаров");
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        products -= newProducts;
        System.out.println("Покупатель купил " + newProducts + "
товаров");
        System.out.println("Товаров на складе: " + products);
        notify();
    }

    public synchronized void put() {
        Random random = new Random();
        int newProducts = random.nextInt(maxProducts -
minProducts + 1) + minProducts;

        products += newProducts;
        System.out.println("Производитель поставил " +
newProducts + " товаров");
        notify();
    }
}

class Producer implements Runnable {
    private final Store store;

    public Producer(Store store) {
        this.store = store;
    }

    @Override
```

```
    public void run() {
        while (true) {
            store.put();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Consumer implements Runnable {
    private final Store store;

    public Consumer(Store store) {
        this.store = store;
    }

    @Override
    public void run() {
        while (true) {
            store.get();
            try {
                Thread.sleep(1500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class ex3 {

    public static void main(String[] args) {
        Store store = new Store();
        Producer producer = new Producer(store);
        Consumer consumer = new Consumer(store);

        new Thread(producer).start();
        new Thread(consumer).start();
    }

}
```

Результат выполнения:

```
Производитель поставил 6 товаров
Покупатель не может купить 9 товаров, так как на складе лежит 6 товаров
Производитель поставил 7 товаров
Покупатель купил 9 товаров
Товаров на складе: 4
Производитель поставил 5 товаров
Покупатель купил 3 товаров
Товаров на складе: 6
Производитель поставил 10 товаров
```

Рисунок 2 – Результат выполнения 3

Вывод: в ходе выполнения лабораторной работы был получен опыт работы с потоками на языке Java.