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 (Threads)

Дисциплина: Языки программирования для работы с большими данными

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

Москва, 2023

Задания:

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.

Код для решения задания 1:

```
package bdjava.lab8;
import java.util.Random;

public class Part1 {
    public static void main(String[] args) {
        Bank bank = new Bank(1000);

        Thread depositThread = new Thread(() -> {
            Random random = new Random();

            while (true) {
                int amount = random.nextInt(100) + 1;
                bank.deposit(amount);

                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        });

        Thread withdrawThread = new Thread(() -> {
            Random random = new Random();

            while (true) {
                int amount = random.nextInt(100) + 1;
                bank.withdraw(amount);

                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
```

```
        });

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



class Bank {
    private int balance;

    public Bank(int balance) {
        this.balance = balance;
    }

    public synchronized void deposit(int amount) {
        balance += amount;
        System.out.println("Пополнение на " + amount + " рублей. Текущий баланс: "
+ balance + " рублей.");
    }

        public synchronized void withdraw(int amount) {
        if (balance < amount) {
            System.out.println("Недостаточно денежных средств на счете. Текущий
баланс: " + balance + " рублей.");
        } else {
            balance -= amount;
            System.out.println("Снятие " + amount + " рублей. Текущий баланс: " +
balance + " рублей.");
        }
    }
}
```

Код для решения задания 2:

```
package bdjava.lab8;
import java.util.Random;

public class Part1 {
    public static void main(String[] args) {
        Bank bank = new Bank(1000);

        Thread depositThread = new Thread(() -> {
            Random random = new Random();

            while (true) {
                int amount = random.nextInt(100) + 1;
                bank.deposit(amount);

                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
```

```
            }
        });

        Thread withdrawThread = new Thread(() -> {
            Random random = new Random();

            while (true) {
                int amount = random.nextInt(100) + 1;
                bank.withdraw(amount);

                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        });

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



class Bank {
    private int balance;

    public Bank(int balance) {
        this.balance = balance;
    }

    public synchronized void deposit(int amount) {
        balance += amount;
        System.out.println("Пополнение на " + amount + " рублей. Текущий баланс: "
+ balance + " рублей.");
    }

        public synchronized void withdraw(int amount) {
        if (balance < amount) {
            System.out.println("Недостаточно денежных средств на счете. Текущий
баланс: " + balance + " рублей.");
        } else {
            balance -= amount;
            System.out.println("Снятие " + amount + " рублей. Текущий баланс: " +
balance + " рублей.");
        }
    }
}
```

Вывод:

Для задания "Банк" было реализовано многопоточное приложение, которое позволяет пополнять и снимать деньги со счета банка. При каждой операции

выводится текущий баланс счета, а если денежных средств недостаточно, выводится сообщение об ошибке. Реализация синхронизации пополнения и снятия денежных средств позволяет избежать ошибок при одновременном доступе к счету из разных потоков. Для задания "Робот" было реализовано многопоточное приложение, которое позволяет роботу ходить. За движение каждой его ноги отвечает отдельный поток, который выводит в консоль сообщение о шаге (LEFT или RIGHT). Реализация многопоточности позволяет роботу двигаться более плавно и естественно, а также дает возможность контролировать каждую ногу отдельно.