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**

ОТЧЕТ

по лабораторной работе № 8

Название: Потоки

Дисциплина: Языки программирования для работы с большими данными

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

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

Москва, 2022

Файл Main.java:

```
public class BankAccount {
    private int balance;

    public BankAccount(int initBalance) {
        this.balance = initBalance;
    }

    public boolean withdraw(int amount) {
        boolean status = (amount > 0) && (this.balance - amount >= 0);
        this.balance = (status ? this.balance - amount : this.balance);
        return status;
    }

    public boolean deposit(int amount) {
        this.balance += amount;
        return true;
    }

    public int balance() { return this.balance; }
}
```

Файл Main.java:

```
public class BankClient extends Thread {
    private final int balanceChange;
    private final boolean isDeposit;
    private final BankAccount bankAccount;

    public BankClient(int balanceChange, BankAccount bankAccount) {
        this.isDeposit = balanceChange > 0;
        this.balanceChange = (isDeposit? balanceChange : -balanceChange);
        this.bankAccount = bankAccount;
    }

    @Override
    public void run() {
        synchronized (bankAccount) {
            System.out.println("Текущий баланс: " + bankAccount.balance() + " {Thread #" + getId() + "}");
            System.out.println("Операция " + (isDeposit ? "пополнения" : "снятия") + " на сумму " +
balanceChange);
            if (!isDeposit) {
                boolean status = bankAccount.withdraw(balanceChange);
                if (!status) {
                    System.out.println("Недостаточно денег!" + " {Thread #" + getId() + "}");
                }
            } else {
                bankAccount.deposit(balanceChange);
            }
        }
    }
}
```

Файл Main.java:

```
import java.util.concurrent.Callable;

/*
    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.
*/

public class Lab81 {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(1000);
        Thread client1 = new BankClient(-1500, account);
        Thread client2 = new BankClient(100, account);
        Thread client3 = new BankClient(1500, account);
        Thread client4 = new BankClient(-800, account);

        client1.start();
        client2.start();
        client3.start();
        client4.start();
    }
}
```

Файл Main.java:

```
/*
    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.
*/

public class Lab82 {
    public static void main(String[] args) {
        Robot robot = new Robot();

//        Thread controller1 = new RobotController(robot, "goLeft; goLeft; goRight; goLeft");
//        Thread controller2 = new RobotController(robot, "goRight; goLeft; goLeft");
//        Thread controller3 = new RobotController(robot, "goRight; goRight; goRight");
```

```
        Thread controller1 = new RobotController(robot, Robot.LEGS.LEFT, 15);
        Thread controller2 = new RobotController(robot, Robot.LEGS.RIGHT, 18);

        controller1.start();
        controller2.start();
    }
}
```

Файл Main.java:

```
public class RobotController extends Thread {
    private final Robot robot;
    private final Robot.LEGS leg;
    private int stepCount;

    public RobotController(Robot robot, Robot.LEGS leg, int stepCount) {
        this.robot = robot;
        this.leg = leg;
        this.stepCount = stepCount;
    }

    @Override
    public void run() {
//        synchronized (robot) {
            while (stepCount > 0) {
                switch (leg) {
                    case LEFT:
                        robot.goLeft();
                        break;
                    case RIGHT:
                        robot.goRight();
                        break;
                }
                stepCount--;
            }
//        }
    }
//    private final String algorithm;
//
//    public RobotController(Robot robot, String algorithm) {
//        this.robot = robot;
//        this.algorithm = algorithm;
//    }
//
//    @Override
//    public void run() {
//        String[] insns = algorithm.split("[;]\\s?");
//        synchronized (robot) {
//            for (String i : insns) {
//                if (i.equals("goLeft")) {
//                    robot.goLeft();
//                } else if (i.equals("goRight")) {
//                    robot.goRight();
//                }
//            }
//        }
//    }
}
```

Файл Main.java:

```
public class Robot {
    enum LEGS {
        LEFT,
        RIGHT
    };
    public Robot() {}

    public void goLeft() {
        System.out.println("LEFT {Thread #" + Thread.currentThread().getId() + "}");
    }

    public void goRight() {
        System.out.println("RIGHT {Thread #" + Thread.currentThread().getId() + "}");
    }
}
```