



**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 № 5

Название: Потоки

Дисциплина: Языки программирования для работы с большими данными

Студент

ИУ6-23М
(Группа)

(Подпись, дата)

А.О.Крейденко
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов
(И.О. Фамилия)

Москва, 2024

Цель: изучить потоки в java

Задание 1: 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 класса Leg:

```
// Класс, представляющий ногу робота
public class Leg implements Runnable {
    private final String legName;

    public Leg(String legName) {
        this.legName = legName;
    }

    @Override
    public void run() {
        // Имитируем шаги ноги робота
        for (int i = 0; i < 10; i++) {
            synchronized (Leg.class) {
                System.out.println(legName);
                try {
                    // Устанавливаем задержку для имитации
                    // движения ноги
                    Leg.class.notify();
                    Leg.class.wait();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
        synchronized (Leg.class) {
            Leg.class.notify(); // Убедиться, что другой поток
            // не останется заблокированным
        }
    }
}
```

Код класса Robot:

```
public class Robot {

    public static void main(String[] args) {
        // Создание объектов ног
        Leg leftLeg = new Leg("LEFT");
        Leg rightLeg = new Leg("RIGHT");

        // Создание потоков для каждой ноги
        Thread leftThread = new Thread(leftLeg);
        Thread rightThread = new Thread(rightLeg);
    }
}
```

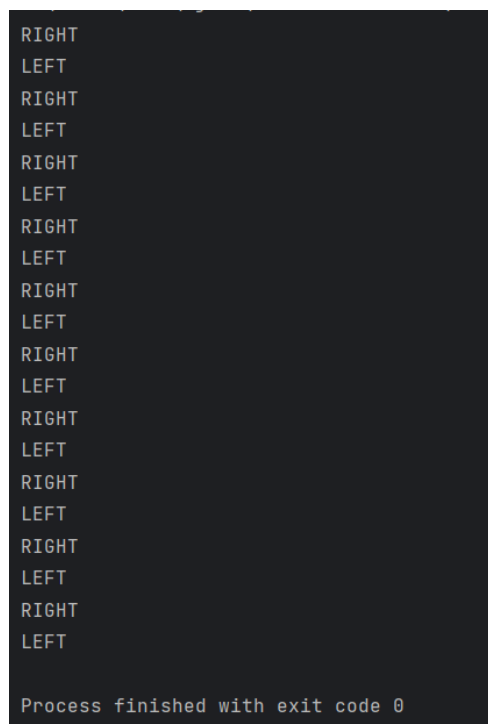
```

// Запуск потоков
leftThread.start();
rightThread.start();

// Ожидание завершения потоков
try {
    leftThread.join();
    rightThread.join();
} catch (InterruptedException e) {
    e.printStackTrace();
}
}
}

```

Работа программы показана на рисунке 1.



```

RIGHT
LEFT
RIGHT
LEFT
RIGHT
LEFT
RIGHT
LEFT
RIGHT
LEFT
RIGHT
LEFT
RIGHT
LEFT
RIGHT
LEFT
RIGHT
LEFT
RIGHT
LEFT
Process finished with exit code 0

```

Рисунок 1 – Работа программы 1

Задание 2: 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 Consumer:

```
import java.util.Random;
```

```

public class Consumer implements Runnable {
    private Store store;

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

    @Override
    public void run() {
        try {
            while (true) {
                store.consume(); // Забираем товар со склада
                Thread.sleep(new Random().nextInt(1000)); //
Имитация времени на потребление
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

Код класса Producer:

```

import java.util.Random;

public class Producer implements Runnable {
    private Store store;
    private Random random = new Random();

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

    @Override
    public void run() {
        try {
            while (true) {
                int product = random.nextInt(100); // Генерация
случайного товара
                store.produce(product); // Добавление товара на
склад
                Thread.sleep(random.nextInt(1000)); // Имитация
времени на производство
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

Код класса Store:

```
import java.util.LinkedList;
import java.util.Queue;

public class Store {
    private final int max_capacity; // Максимальная вместимость склада
    private Queue<Integer> products; // Очередь для хранения товаров

    public Store(int capacity) {
        this.max_capacity = capacity;
        this.products = new LinkedList<>();
    }
    // Метод для добавления товара на склад
    public synchronized void produce(int product) throws
    InterruptedException {
        while (products.size() == max_capacity) {
            System.out.println("Склад полон, производитель
ждет");
            wait(); // Если склад полон, производитель ждёт
        }
        products.add(product);
        System.out.println("Производитель добавил товар " +
product);
        notifyAll(); // Уведомляем все ожидающие потоки
    }

    // Метод для потребления товара со склада
    public synchronized int consume() throws InterruptedException
    {
        while (products.isEmpty()) {
            System.out.println("Товара не хватает в магазине,
покупатель ждет.");
            wait(); // Если товаров нет, покупатель ждёт
        }
        int product = products.poll();
        System.out.println("Покупатель забрал товар " +
product);
        notifyAll(); // Уведомляем все ожидающие потоки
        return product;
    }
}
```

Код класса StoreApp:

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Random;

public class StoreApp {
    public static void main(String[] args) {
        Store store = new Store(10);
```

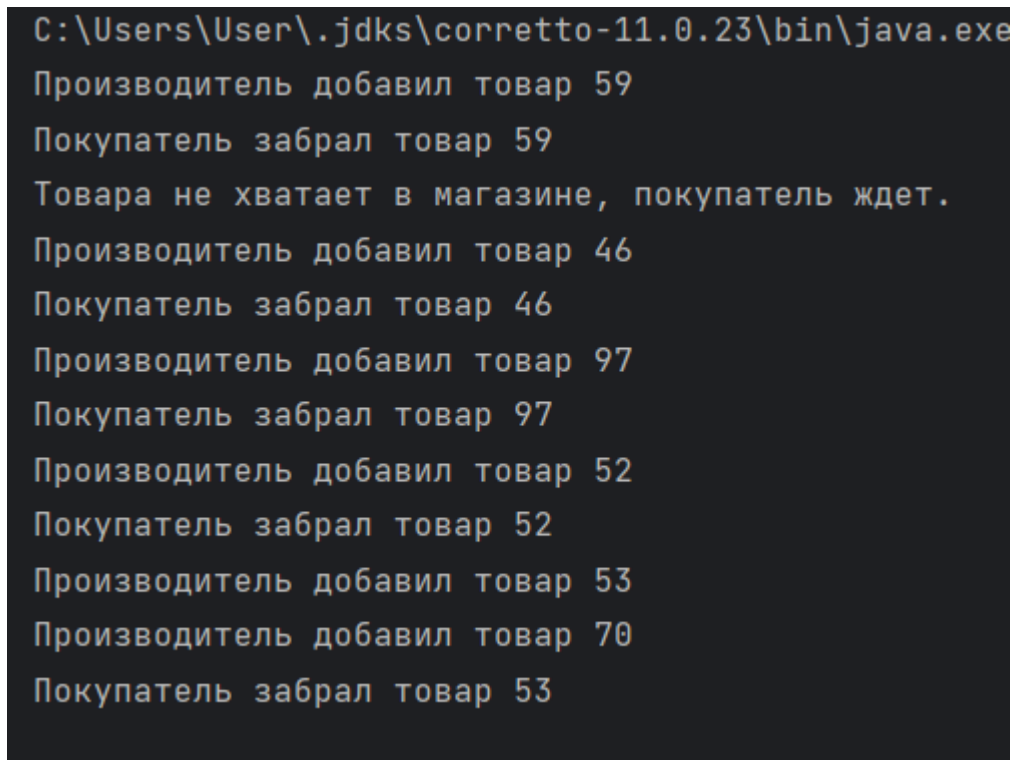
```

        Thread producerThread = new Thread(new Producer(store));
        Thread consumerThread = new Thread(new Consumer(store));

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

```

Работа программы показана на рисунке 2.



```

C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe
Производитель добавил товар 59
Попатель забрал товар 59
Товара не хватает в магазине, поатель ждет.
Производитель добавил товар 46
Попатель забрал товар 46
Производитель добавил товар 97
Попатель забрал товар 97
Производитель добавил товар 52
Попатель забрал товар 52
Производитель добавил товар 53
Производитель добавил товар 70
Попатель забрал товар 53

```

Рисунок 2 – Работа программы 2

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