МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена трудового Красного Знамени федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе №5.

Выполнил: студент группы БВТ2402

Безматерных Иван Алексеевич

Москва, 2025

Цель работы: Приобретение практических навыков разработки многопоточных приложений на Java путем изучения базовых концепций, современных средств управления потоками и механизмов синхронизации для решения задач параллельных вычислений и корректного доступа к общим ресурсам.

Задания:

Задание 1: Реализация многопоточной программы для вычисления суммы элементов массива.

Создать два потока, которые будут вычислять сумму элементов массива по половинкам, после чего результаты будут складываться в главном потоке.

Код программы:

```java
public class Task1 {

    static class SumThread extends Thread {
        int[] arr;
        int start, end;
        long sum = 0;

        SumThread(int[] arr, int start, int end) {
            this.arr = arr;
            this.start = start;
            this.end = end;
        }

        public void run() {
            for (int i = start; i < end; i++) {
                sum += arr[i];
            }
        }
    }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {

        int[] numbers = new int[10];
        for (int i = 0; i < 10; i++) {
            numbers[i] = i + 1;
        }

        SumThread t1 = new SumThread(numbers, start: 0, end: 5);
        SumThread t2 = new SumThread(numbers, start: 5, end: 10);

        t1.start();
        t2.start();

        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        long total = t1.sum + t2.sum;
        System.out.println("Сумма 1-ой половины: " + t1.sum);
        System.out.println("Сумма 2-ой половины: " + t2.sum);
        System.out.println("Итого: " + total);
    }
}
```

Вывод программы:

```
Сумма 1-ой половины: 15
Сумма 2-ой половины: 40
Итого: 55
```

Задание 2: Реализация многопоточной программы для поиска наибольшего элемента в матрице.

Создать несколько потоков, каждый из которых будет обрабатывать свою строку матрицы. После завершения работы всех потоков результаты будут сравниваться в главном потоке для нахождения наибольшего элемента.

Код программы:

```java
public class Task2 {

    static class MaxThread extends Thread {
        int[] row;
        int max;

        MaxThread(int[] row) {
            this.row = row;
        }

        public void run() {
            max = row[0];
            for (int i = 1; i < row.length; i++) {
                if (row[i] > max) {
                    max = row[i];
                }
            }
        }
    }

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        int[][] matrix = {
            {1, 5, 3, 8, 2},
            {9, 4, 7, 1, 6},
            {3, 2, 9, 4, 5},
            {7, 8, 6, 2, 1}
        };

        MaxThread[] threads = new MaxThread[matrix.length];

        for (int i = 0; i < matrix.length; i++) {
            threads[i] = new MaxThread(matrix[i]);
            threads[i].start();
        }

        try {
            for (int i = 0; i < threads.length; i++) {
                threads[i].join();
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        int globalMax = threads[0].max;
        for (int i = 1; i < threads.length; i++) {
            if (threads[i].max > globalMax) {
                globalMax = threads[i].max;
            }
        }

        System.out.println("Наибольший элемент в матрице: " + globalMax);
    }
}
```

Вывод программы:

```
Наибольший элемент в матрице: 9
```

Задание 3: У вас есть склад с товарами, которые нужно перенести на другой склад. У каждого товара есть свой вес. На складе работают 3 грузчика. Грузчики могут переносить товары одновременно, но суммарный вес товаров, переносимый ими за одну итерацию, не может превышать 150 кг. Как только грузчики соберут 150 кг товаров, они отправятся на другой склад и начнут разгружать товары. Напишите программу на Java, используя многопоточность, которая реализует данную ситуацию.

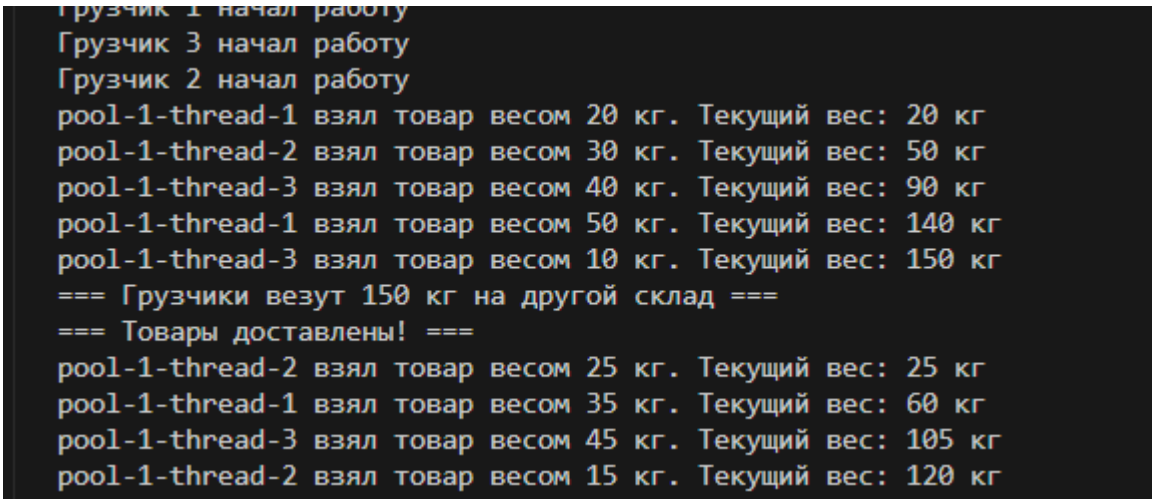
Использование Thread. Создайте классы Товар, Склад, и Грузчик. Каждый грузчик должен быть представлен в виде отдельного потока.

Код программы:

```java
import java.util.Random;
import java.util.concurrent.atomic.AtomicInteger;

class Product {
    private int weight;

    public Product(int weight) {
        this.weight = weight;
    }

    public int getWeight() {
        return weight;
    }
}

class Warehouse {
    private Product[] products;
    private AtomicInteger currentIndex = new AtomicInteger(initialValue: 0);

    public Warehouse(Product[] products) {
        this.products = products;
    }

    public Product takeProduct() {
        int index = currentIndex.getAndIncrement();
        if (index < products.length) {
            return products[index];
        }
        return null;
    }

    public boolean hasProducts() {
        return currentIndex.get() < products.length;
    }
}

class Loader extends Thread {
    private Warehouse warehouse;
    private static int totalWeight = 0;
    private static final Object lock = new Object();
    private static final int MAX_WEIGHT = 150;

    public Loader(String name, Warehouse warehouse) {
        super(name);
        this.warehouse = warehouse;
    }

    @Override
    public void run() {
```

```
    public void run() {
        while (warehouse.hasProducts()) {
            Product product = warehouse.takeProduct();
            if (product == null) break;

            synchronized (lock) {
                if (totalWeight + product.getWeight() <= MAX_WEIGHT) {
                    totalWeight += product.getWeight();
                    System.out.println(getName() + " взял товар весом " + product.getWeight() + " кг. Текущий вес: " + totalWeight);
                } else {
                    System.out.println(getName() + " не может взять товар " +
                                       product.getWeight() + " кг. Превышен лимит " + MAX_WEIGHT);
                    deliver();
                    continue;
                }

                if (totalWeight == MAX_WEIGHT) {
                    deliver();
                }
            }

            try {
                Thread.sleep(millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        if (totalWeight > 0) {
            deliver();
        }
    }

    private void deliver() {
        synchronized (lock) {
            if (totalWeight > 0) {
                System.out.println("=== " + getName() + " везет " + totalWeight + " кг на другой склад ===");
                try {
                    Thread.sleep(millis: 100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                System.out.println("=== " + getName() + " разгрузил " + totalWeight + " кг ===");
                totalWeight = 0;
            }
        }
    }
}

public class Task3 {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) throws InterruptedException {
        Random random = new Random();
        Product[] products = new Product[30];

        for (int i = 0; i < products.length; i++) {
            products[i] = new Product(random.nextInt(bound: 50) + 10);
        }

        Warehouse warehouse = new Warehouse(products);

        Loader loader1 = new Loader(name: "Грузчик 1", warehouse);
        Loader loader2 = new Loader(name: "Грузчик 2", warehouse);
        Loader loader3 = new Loader(name: "Грузчик 3", warehouse);

        System.out.println(x: "Начинаем погрузку...");
        loader1.start();
        loader2.start();
        loader3.start();

        loader1.join();
        loader2.join();
        loader3.join();

        System.out.println(x: "Все товары перенесены!");
    }
}
```

Вывод программы:



Вывод: в ходе выполнения лабораторной работы были приобретенены практические навыки разработки многопоточных приложений на Java путем изучения базовых концепций, современных средств управления потоками и механизмов синхронизации для решения задач параллельных вычислений и корректного доступа к общим ресурсам.