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"Системное программирование"

Лабораторная работа № 7 Вариант № 29

По дисциплине

“Информационные технологии и программирование”

Выполнил: студент гр. БВТ2201
Шамсутдинов Р.Ф.

Москва, 2023 г

Первое задание:

```
import java.util.Arrays;

public class CountSum extends Thread{

    private int[] array;
    private int sum = 0;

    public CountSum(int[] array){
        this.array = array;
    }

    @Override
    public void run(){
        calculateSum();
    }
    private void calculateSum(){
        for (int item: array){
            sum += item;
            System.out.println(sum + " из " + Arrays.toString(array));

        }
    }
    public int getSum(){
        return sum;
    }
}
```

Второе задание:

```
import java.lang.reflect.Array;
import java.util.Arrays;

class MaxElementFinder implements Runnable {
    private int[] row;
    private int maxElement = Integer.MIN_VALUE;

    public MaxElementFinder(int[] row) {
        this.row = row;
    }
    @Override
    public void run() {
        findMax();
    }
    private void findMax(){
        for (int item : row) {
            if (item > maxElement) {
                maxElement = item;
            }
        }
    }
    public int getMaxElement() {
        System.out.println(maxElement + " наибольшее в " +
Arrays.toString(row));
        return maxElement;
    }
}
```

Третье задание:

```
import java.util.concurrent.CyclicBarrier;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
import java.util.Random;


public class WarehouseTransfer {
    private static final int NUM_LOADERS = 3;
    private static final int MAX_WEIGHT = 150;
    private static final int NUM_PRODUCTS = MAX_WEIGHT *
NUM_LOADERS;
    private static AtomicInteger index = new AtomicInteger(0); // Use
AtomicInteger

    public static void main(String[] args) {
        int[] weights = generateWeights(NUM_PRODUCTS);

        Lock weightsLock = new ReentrantLock();
        CyclicBarrier barrier = new CyclicBarrier(NUM_LOADERS, () ->
{
            System.out.println("Все грузчики доставили товары на
другой склад");
//          try {
//              Thread.sleep(300);
//          } catch (InterruptedException e) {
//              throw new RuntimeException(e);
//          }
        });

        Thread[] loaders = new Thread[NUM_LOADERS];
        for (int i = 0; i < NUM_LOADERS; i++) {
            loaders[i] = new Thread(new Loader(weightsLock, barrier,
MAX_WEIGHT, weights));
            loaders[i].start();
        }

        for (Thread loader : loaders) {
            try {
                loader.join();
//              Thread.sleep(300);
            } catch (InterruptedException e) {
```

```
                throw new RuntimeException(e);
            }
        }
//        System.out.println(Arrays.toString(weights));
        System.out.println("Все грузчики завершили работу");
    }

    static class Loader implements Runnable {
        private final Lock weightsLock;
        private final CyclicBarrier barrier;
        private final int maxWeight;
        private final int[] weights;
        private boolean needToLoad = true;
        public Loader(Lock weightsLock, CyclicBarrier barrier, int
maxWeight, int[] weights) {
            this.weightsLock = weightsLock;
            this.barrier = barrier;
            this.maxWeight = maxWeight;
            this.weights = weights;
        }

        @Override
        public void run() {
            int currentWeight = 0;

            while (needToLoad) {
//                System.out.println(Thread.currentThread().getName());
                weightsLock.lock();
                try {

                    int currentIndex = index.getAndIncrement();
                    if (currentIndex < NUM_PRODUCTS && currentWeight +
weights[currentIndex] <= maxWeight) {
                        currentWeight += weights[currentIndex];
//                        Thread.sleep(50);
                        System.out.println("Грузчик " +
Thread.currentThread().getName() + " поднял " +
weights[currentIndex] + " кг товара, всего " + currentWeight);
                        // Удаляем вес товара из массива, чтобы другие
грузчики не могли взять его
                        weights[currentIndex] = 0;

                    } else {
                        needToLoad = false;
```

```
                }
//              } catch (InterruptedException e) {
//                  throw new RuntimeException(e);
              } finally {
                  weightsLock.unlock();
              }
          }

          try {
              barrier.await(); // Грузчик ждет остальных перед
разгрузкой на другом складе
              System.out.println("Грузчик " +
Thread.currentThread().getName() + " разгружает товары на
другом складе");
          } catch (Exception e) {
              e.printStackTrace();
          }
      }
   }

   private static int[] generateWeights(int numProducts) {
      int[] weights = new int[numProducts];
      for (int i = 0; i < numProducts; i++) {
          Random rand = new Random();
          int randomNum = rand.nextInt(30) + 1;
          weights[i] = randomNum;
      }
//       System.out.println(Arrays.toString(weights));
      return weights;
   }
}
```

Вызов всех задание в мейн файле:

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        System.out.println("1) First task: ");
        int[] array = {1,2,3,4,5,6,7,8,9,10};
        int[] firstHalfOfArray = Arrays.copyOfRange(array, 0,
array.length/2);
        int[] secondHalfOfArray = Arrays.copyOfRange(array,
array.length/2, array.length);

        CountSum firstThread = new CountSum(firstHalfOfArray);
        CountSum secondThread = new
CountSum(secondHalfOfArray);

        firstThread.start();
        secondThread.start();
        try {
            firstThread.join();
            secondThread.join();
        } catch (InterruptedException e){
            e.printStackTrace();
        }

        int totalSum = firstThread.getSum() +
secondThread.getSum();
        System.out.println("Сумма чисел в массиве равна " +
totalSum);


        System.out.println("2) Second task: ");

        int[][] matrix = {
                {1, 2, 3, 99, 500, -3},
                {4, 5, 6, 18, 44, 192},
                {7, 8, 9, 10000, 32, 8},
                {1, 2, 3, 88, 56, 31},
                {4, 5, 6, 444, 404, 403},
                {7, 8, 9, 228, 322, 911},
        };

        int maxElement = Integer.MIN_VALUE;
```

```
        MaxElementFinder[] maxFinders = new
MaxElementFinder[matrix.length];
        Thread[] threads = new Thread[matrix.length];

        for (int i = 0; i<matrix.length; i++){
            maxFinders[i] = new MaxElementFinder(matrix[i]);
            threads[i] = new Thread(maxFinders[i]);
            threads[i].start();
        }
        for (int i = 0; i<matrix.length; i++){
            try {
                threads[i].join();
                int result = maxFinders[i].getMaxElement();
                if (result > maxElement) {
                    maxElement = result;
                }
            } catch (InterruptedException e){
                e.printStackTrace();
            }
        }
        System.out.println("Самый большой элемент в " +
Arrays.deepToString(matrix) + " это " + maxElement);

        System.out.println("3) Third task: ");
        WarehouseTransfer.main(new String[]{" "});

    }

}
```

Вывод результата в консоль:

1) First task:
6 из [6, 7, 8, 9, 10]
1 из [1, 2, 3, 4, 5]
13 из [6, 7, 8, 9, 10]
3 из [1, 2, 3, 4, 5]
21 из [6, 7, 8, 9, 10]
6 из [1, 2, 3, 4, 5]
30 из [6, 7, 8, 9, 10]
10 из [1, 2, 3, 4, 5]
40 из [6, 7, 8, 9, 10]
15 из [1, 2, 3, 4, 5]
Сумма чисел в массиве равна 55
2) Second task:
500 наибольшее в [1, 2, 3, 99, 500, -3]
192 наибольшее в [4, 5, 6, 18, 44, 192]
10000 наибольшее в [7, 8, 9, 10000, 32, 8]
88 наибольшее в [1, 2, 3, 88, 56, 31]
444 наибольшее в [4, 5, 6, 444, 404, 403]
911 наибольшее в [7, 8, 9, 228, 322, 911]
Самый большой элемент в [[1, 2, 3, 99, 500, -3], [4, 5, 6, 18, 44, 192], [7, 8, 9, 10000, 32, 8], [1, 2, 3, 88, 56, 31], [4, 5, 6, 444, 404, 403], [7, 8, 9, 228, 322, 911]] это 10000
3) Third task:
Грузчик Thread-8 поднял 26 кг товара, всего 26
Грузчик Thread-8 поднял 12 кг товара, всего 38
Грузчик Thread-8 поднял 3 кг товара, всего 41
Грузчик Thread-8 поднял 16 кг товара, всего 57
Грузчик Thread-8 поднял 28 кг товара, всего 85
Грузчик Thread-8 поднял 19 кг товара, всего 104
Грузчик Thread-8 поднял 6 кг товара, всего 110
Грузчик Thread-8 поднял 1 кг товара, всего 111
Грузчик Thread-8 поднял 25 кг товара, всего 136
Грузчик Thread-9 поднял 6 кг товара, всего 6
Грузчик Thread-9 поднял 23 кг товара, всего 29
Грузчик Thread-9 поднял 15 кг товара, всего 44
Грузчик Thread-9 поднял 11 кг товара, всего 55
Грузчик Thread-9 поднял 4 кг товара, всего 59
Грузчик Thread-9 поднял 17 кг товара, всего 76
Грузчик Thread-9 поднял 3 кг товара, всего 79
Грузчик Thread-9 поднял 22 кг товара, всего 101
Грузчик Thread-9 поднял 25 кг товара, всего 126
Грузчик Thread-9 поднял 3 кг товара, всего 129
Грузчик Thread-9 поднял 20 кг товара, всего 149

Грузчик Thread-10 поднял 24 кг товара, всего 24
Грузчик Thread-10 поднял 22 кг товара, всего 46
Грузчик Thread-10 поднял 3 кг товара, всего 49
Грузчик Thread-10 поднял 13 кг товара, всего 62
Грузчик Thread-10 поднял 10 кг товара, всего 72
Грузчик Thread-10 поднял 17 кг товара, всего 89
Грузчик Thread-10 поднял 27 кг товара, всего 116
Грузчик Thread-10 поднял 1 кг товара, всего 117
Грузчик Thread-10 поднял 8 кг товара, всего 125
Грузчик Thread-10 поднял 4 кг товара, всего 129
Грузчик Thread-10 поднял 17 кг товара, всего 146
Все грузчики доставили товары на другой склад
Грузчик Thread-8 разгружает товары на другом складе
Грузчик Thread-9 разгружает товары на другом складе
Грузчик Thread-10 разгружает товары на другом складе
Все грузчики завершили работу

Process finished with exit code 0

Вывод:

Проделав данную работу я познакомился с реализацией многопоточности в языке программирования Java