**Министерство науки и высшего образования Российской Федерации**  
  
Федеральное государственное автономное образовательное учреждение  
высшего образования «Уральский федеральный университет  
имени первого Президента России Б.Н. Ельцина»  
  
Институт радиоэлектроники и информационных технологий – РТФ  
Центр ускоренного обучения

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11**по дисциплине «Курсовое проектирование Java»

**Тема: «Введение в многопоточность»**

|  |  |
| --- | --- |
| Студент группы РИВ-410938у | М.О. Покуль |
| Преподаватель: | Н.А. Архипов |

**Екатеринбург 2024**

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc190896507)

[2 Описание задачи 4](#_Toc190896508)

[3 Ход выполнения работы. 5](#_Toc190896509)

[3.2 Программа, которая запускает поток и выводит на экран числа от 1 до 10 с задержкой в 1 секунду между выводами. 7](#_Toc190896510)

[3.3 Программа, которая создает два потока, которые выводят на экран числа от 1 до 10. Первый поток должен выводить только четные числа, а второй поток должен выводить только нечетные числа. 8](#_Toc190896511)

[3.4 Программа, которая создает 10 потоков и каждый поток выводит на экран свой номер. 10](#_Toc190896512)

[3.5 Функция, которая находит максимальный элемент в массиве целых чисел при помощи многопоточности. Количество потоков должно быть равно количеству ядер процессора. 11](#_Toc190896513)

[3.6 Функция, которая суммирует элементы в массиве целых чисел при помощи многопоточности. Количество потоков должно быть равно количеству ядер процессора. 13](#_Toc190896514)

[4 Вывод 15](#_Toc190896515)

1. Постановка задачи

Получить представление о механизме многопоточности в языке программирования Java.

1. Описание задачи
2. Написать программу, которая запускает два потока и выводит на экран их имена и текущее время в течение 10 секунд.
3. Написать программу, которая запускает поток и выводит на экран числа от 1 до 10 с задержкой в 1 секунду между выводами.
4. Написать программу, которая создает два потока, которые выводят на экран числа от 1 до 10. Первый поток должен выводить только четные числа, а второй поток должен выводить только нечетные числа.
5. Написать программу, которая создает 10 потоков и каждый поток выводит на экран свой номер.
6. Написать функцию, которая находит максимальный элемент в массиве целых чисел при помощи многопоточности. Количество потоков должно быть равно количеству ядер процессора.
7. Написать функцию, которая суммирует элементы в массиве целых чисел при помощи многопоточности. Количество потоков должно быть равно количеству ядер процессора.
8. Ход выполнения работы.
   1. **Программа, которая запускает два потока и выводит на экран их имена и текущее время в течение 10 секунд.**

package Lab12;  
  
import java.time.LocalTime;  
  
public class Task1 {  
 public static void main(String[] args) throws InterruptedException {  
 Thread t1 = new Thread(() -> {  
 for (int i = 0; i < 18; i++) {  
 System.*out*.println(Thread.*currentThread*().getName() + ": " + LocalTime.*now*());  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 );  
  
 Thread t2 = new Thread(() -> {  
 for (int i = 0; i < 18; i++) {  
 System.*out*.println(Thread.*currentThread*().getName() + ": " + LocalTime.*now*());  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 );  
  
 t1.start();  
 t2.start();  
  
 t1.join();  
 t2.join();  
 }  
}

* 1. Программа, которая запускает поток и выводит на экран числа от 1 до 10 с задержкой в 1 секунду между выводами.

package Lab12;  
  
public class Task2 {  
 public static void main(String[] args) throws InterruptedException {  
 Thread t = new Thread(() -> {  
 for (int i = 1; i <= 10; i++) {  
 System.*out*.println(i);  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
  
 t.start();  
 t.join();  
 }  
}

* 1. Программа, которая создает два потока, которые выводят на экран числа от 1 до 10. Первый поток должен выводить только четные числа, а второй поток должен выводить только нечетные числа.

package Lab12;  
  
public class Task3 {  
 private static final Object *lock* = new Object();  
 private static int *number* = 1;  
  
 public static void main(String[] args) {  
 Thread evenThread = new Thread(() -> {  
 synchronized (*lock*) {  
 while (*number* < 10) {   
 if (*number* % 2 == 0) {  
 System.*out*.println(Thread.*currentThread*().getName() + ": " + *number*);  
 *number*++;  
 }else{  
 try {  
 *lock*.wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 *lock*.notifyAll();  
 }  
 }  
 });  
  
 Thread oddThread = new Thread(() -> {  
 synchronized (*lock*) {  
 while (*number* < 10) {   
 if (*number* % 2 != 0) {  
 System.*out*.println(Thread.*currentThread*().getName() + ": " + *number*);  
 *number*++;  
 }else{  
 try {  
 *lock*.wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 *lock*.notifyAll();  
 }  
 }  
 });  
  
 evenThread.setName("Even Thread");  
 oddThread.setName("Odd Thread");  
  
 evenThread.start();  
 oddThread.start();  
 }  
}

* 1. Программа, которая создает 10 потоков и каждый поток выводит на экран свой номер.

package Lab12;  
  
public class Task4 {  
 public static void main(String[] args) {  
 Thread[] threads = new Thread[10];  
  
 for (int i = 0; i < 10; i++) {  
 final int threadNumber = i;  
 threads[i] = new Thread(() -> {  
 System.*out*.println("Поток номер: " + threadNumber);  
 });  
 }  
  
 for (Thread thread : threads) {  
 try {  
 thread.start();  
 thread.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 System.*out*.println("Все потоки завершены.");  
 }  
}

* 1. Функция, которая находит максимальный элемент в массиве целых чисел при помощи многопоточности. Количество потоков должно быть равно количеству ядер процессора.

package Lab12;  
  
import java.util.Random;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;  
import java.util.concurrent.Future;  
  
public class Task5 {  
 public static void main(String[] args) throws Exception {  
 int size = 100;  
 int[] array = new int[size];  
 Random random = new Random();  
 for (int i = 0; i < size; i++) {  
 array[i] = random.nextInt(1000);  
 }  
  
 System.*out*.println("Максимальный элемент в массиве: " + *findMaxInArray*(array));  
 }  
  
 public static int findMaxInArray(int[] array) throws Exception {  
 int numWorkers = Runtime.*getRuntime*().availableProcessors();  
 ExecutorService executor = Executors.*newFixedThreadPool*(numWorkers);  
 int chunkSize = array.length / numWorkers;  
  
 Future<Integer>[] futures = new Future[numWorkers];  
  
 for (int i = 0; i < numWorkers; i++) {  
 final int start = i \* chunkSize;  
 final int end = (i == numWorkers - 1) ? array.length : (i + 1) \* chunkSize; *//остаток в последнем фрагменте* futures[i] = executor.submit(() -> {  
 int max = array[start];  
 for (int j = start; j < end; j++) {  
 if (array[j] > max) {  
 max = array[j];  
 }  
 }  
 return max;  
 });  
 }  
  
 int max = Integer.*MIN\_VALUE*;  
 for (Future<Integer> future : futures) {  
 max = Math.*max*(max, future.get());  
 }  
  
 executor.shutdown();  
 return max;  
 }  
}

* 1. Функция, которая суммирует элементы в массиве целых чисел при помощи многопоточности. Количество потоков должно быть равно количеству ядер процессора.

package Lab12;  
  
import java.util.Random;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;  
import java.util.concurrent.Future;  
  
public class Task6 {  
 public static void main(String[] args) throws Exception {  
 int size = 100;  
 int[] array = new int[size];  
 Random random = new Random();  
 for (int i = 0; i < size; i++) {  
 array[i] = random.nextInt(100);  
 }  
  
 System.*out*.println("Сумма всех элементов массива: " + *findMaxInArray*(array));  
 }  
  
 public static int findMaxInArray(int[] array) throws Exception {  
 int numWorkers = Runtime.*getRuntime*().availableProcessors();  
 ExecutorService executor = Executors.*newFixedThreadPool*(numWorkers);  
 int chunkSize = array.length / numWorkers;  
  
 Future<Integer>[] futures = new Future[numWorkers];  
  
 for (int i = 0; i < numWorkers; i++) {  
 final int start = i \* chunkSize;  
 final int end = (i == numWorkers - 1) ? array.length : (i + 1) \* chunkSize; *//остаток в последнем фрагменте* futures[i] = executor.submit(() -> {  
 int sum = array[start];  
 for (int j = start; j < end; j++) {  
 sum += array[j];  
 }  
 return sum;  
 });  
 }  
  
 int sum = 0;  
 for (Future<Integer> future : futures) {  
 sum += future.get();  
 }  
  
 executor.shutdown();  
 return sum;  
 }  
}

1. Вывод

Получено представление о механизме многопоточности в языке программирования Java.