|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**Дисциплина «Программирование на языке Джава»**

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №27, №28**

Выполнил студент группы ИНБО-02-20 Самойленко М. А.

Принял Степанов П.В.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2020г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2020г.

Отметка о выполнении

**Москва – 2020 г.**

**Практическое занятие №27-28**

**Задание :**

Напишите программу, которая уменьшает изображения на основе кода который приведен в данном пакете задания. Она должна запускать число потоков, равное количеству ядер процессора вашего компьютера.

Обязательно запустите исходный код проекта и осознайте что однопоточна обработка изображений медленная. Ваша основная задача сделать обработку большого колличества изображений быстрее за счет многопоточности, но без вылета ошибок невозможности создать поток и не перегружая слишком сильно систему юзера.

**Код программы:**

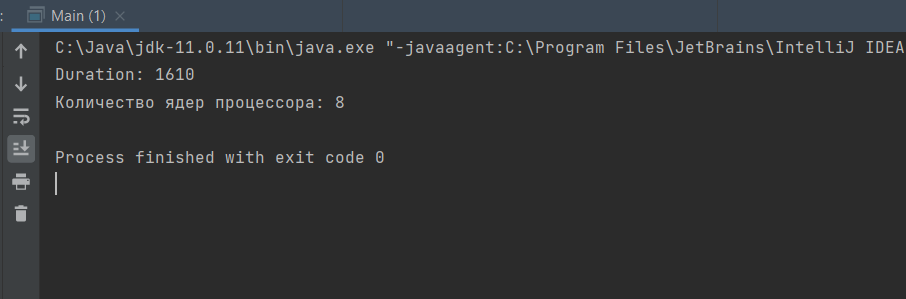
**Класс Main**

package ru.mirea.java.practice27.practice28;  
  
import java.io.File;  
import java.lang.String;  
  
public class Main  
{  
  
 public static void main(String[] args)  
 {  
 long start = System.*currentTimeMillis*();  
 // т.к. ядер на компьютере 8, пусть каждое ядро обрабатывает по 3 картинки из файла, в котором 24 картинки  
 MyThread thread0 = new MyThread(0);  
 MyThread thread1 = new MyThread(3);  
 MyThread thread2 = new MyThread(6);  
 MyThread thread3 = new MyThread(9);  
 MyThread thread4 = new MyThread(12);  
 MyThread thread5 = new MyThread(15);  
 MyThread thread6 = new MyThread(18);  
 MyThread thread7 = new MyThread(21);  
 thread0.start();  
 thread1.start();  
 thread2.start();  
 thread3.start();  
 thread4.start();  
 thread5.start();  
 thread6.start();  
 thread7.start();  
 try{ // подождем окончание каждого потока  
 thread0.join();  
 thread1.join();  
 thread2.join();  
 thread3.join();  
 thread4.join();  
 thread5.join();  
 thread6.join();  
 thread7.join();  
 }catch(InterruptedException e){}  
  
 System.*out*.println("Duration: " + (System.*currentTimeMillis*()-start));  
 System.*out*.println("Количество ядер процессора: " + Runtime.*getRuntime*().availableProcessors());  
 }  
}

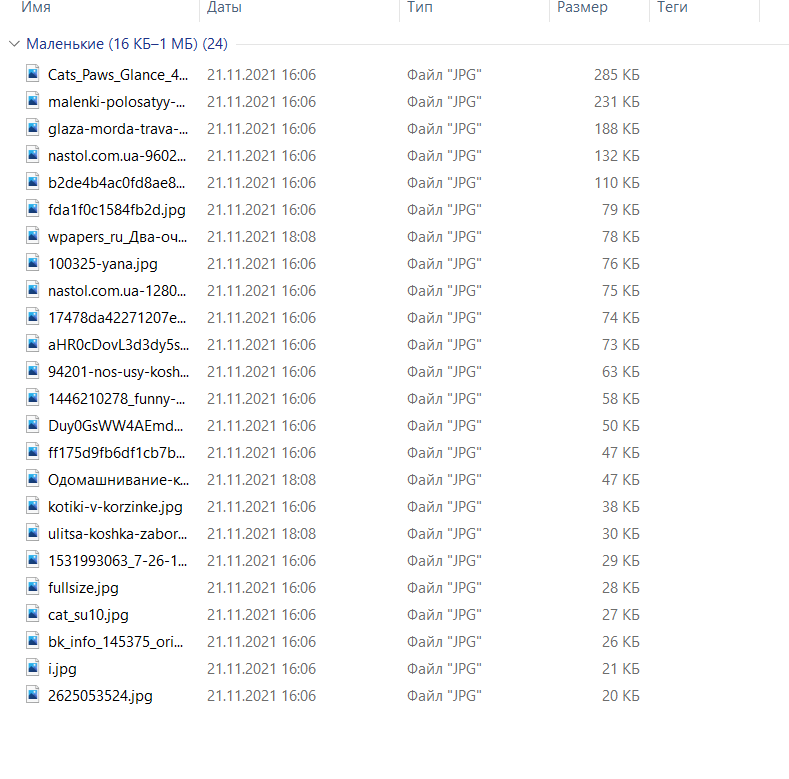
**Класс MyThread**

package ru.mirea.java.practice27.practice28;  
  
import javax.imageio.ImageIO;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
  
public class MyThread extends Thread{  
 private int num; // номер файла в images, с которого нужно начинать  
 MyThread(int num){  
 this.num = num;  
 }  
 @Override  
 public void run(){  
 String dstFolder = "src/ru/mirea/java/practice27/practice28/dst";  
 String srcFolder = "src/ru/mirea/java/practice27/practice28/images";  
 File srcDir = new File(srcFolder);  
 File[] files = srcDir.listFiles();  
 try  
 {  
 if (!Files.*exists*(Paths.*get*(dstFolder)))  
 {  
 Files.*createDirectories*(Paths.*get*(dstFolder));  
 }  
  
 for(int i = num; i < num + 3;i++)  
 {  
 BufferedImage image = ImageIO.*read*(files[i]);  
 if(image == null) {  
 continue;  
 }  
  
 int newWidth = image.getWidth() / 2;  
 int newHeight = (int) Math.*round*(  
 image.getHeight() / (image.getWidth() / (double) newWidth)  
 );  
 BufferedImage newImage = new BufferedImage(  
 newWidth, newHeight, BufferedImage.*TYPE\_INT\_RGB* );  
  
 int widthStep = image.getWidth() / newWidth;  
 int heightStep = image.getHeight() / newHeight;  
  
 for (int x = 0; x < newWidth; x++)  
 {  
 for (int y = 0; y < newHeight; y++) {  
 int rgb = image.getRGB(x \* widthStep, y \* heightStep);  
 newImage.setRGB(x, y, rgb);  
 }  
 }  
  
 File newFile = new File(dstFolder + "/" + files[i].getName());  
 ImageIO.*write*(newImage, "jpg", newFile);  
 }  
  
 }  
 catch (Exception ex) {  
 ex.printStackTrace();  
 }  
  
 }  
}

**Тесты**



**Рисунок 1 – тестирование**



**Рисунок 2 – сжатые файлы**

**Выводы**

Я изучил и научился использовать многопоточность в java, чтобы ускорить работу программы.