Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**по курсу «Программирование на языке JAVA»

по лабораторной работе №5

на тему: «Многопоточность в Java»  
Вариант 10.

Выполнил студент группы 20ВВП1:

Кодиров И. Н.

Приняла:

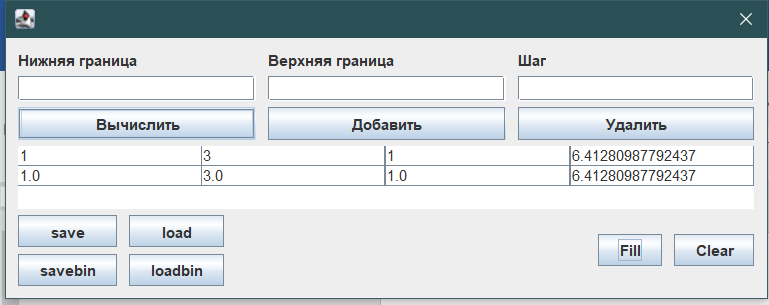
Юрова О. В.

Пенза 2023

**Цель работы:** научиться создавать многопоточные приложения c использованием стандартных средств языка Java.

**Задание на лабораторную работу:** Модифицировать приложение из предыдущей лабораторной работы, реализовав вычисление определенного интеграла в нескольких дополнительных потоках (число потоков определяется номером варианта), снимая нагрузку с основного потока и предотвращая "подвисание" графического интерфейса. Варианты с номерами до 5 включительно реализуют многопоточность путем наследования от класса Thread, остальные реализуют интерфейс Runnable. Оформление лабораторной работы должно быть выполнено в соответствии с требованиями, приведенными в Приложении 2.

**Результаты выполнения программы:**

****

**Листинг:**

public class Recintegral {  
 private double Min;  
 private double Max;  
 private double Step;  
 private double Result;  
  
 public Recintegral(double min, double max, double step, double result) {  
 Min = min;  
 Max = max;  
 Step = step;  
 Result = result;  
 }  
 public Recintegral() {  
 Min = 0;  
 Max = 0;  
 Step = 0;  
 Result = 0;  
 }  
 public Recintegral(double[] temp) throws InterruptedException{  
  
 Min = temp[0];  
 Max = temp[1];  
 Step = temp[2];  
 Result = Trap(temp[0],temp[1],temp[2]);  
 }  
 public double getMin() {  
 return Min;  
 }  
  
 public double getMax() {  
 return Max;  
 }  
  
 public double getStep() {  
 return Step;  
 }  
  
 public double getResult() {  
 return Result;  
 }  
  
 public void setMin(double min) {  
 Min = min;  
 }  
  
 public void setMax(double max) {  
 Max = max;  
 }  
  
 public void setStep(double step) {  
 Step = step;  
 }  
 public void setAllField(double a, double b,double step) throws InterruptedException  
 {  
 Min = a;  
 Max = b;  
 Step = step;  
 if(Min>Max)  
 {  
 setResultNull();  
 }else {  
 setResult(getResult());  
 }  
 }  
 public void setResult(double result) throws InterruptedException{  
  
 result = Trap(Min,Max,Step);  
 Result = result;  
 }  
 public void setResultNull() {  
  
 Result = 0;  
 }  
  
 public Object[] addMod() {  
 Object[] temp=new Object[4];  
 temp[0]=Min;  
 temp[1]=Max;  
 temp[2]=Step;  
 temp[3]=Result;  
 return temp;  
 }  
  
 public double Trap(double a,double b, double h) throws InterruptedException {  
 final double[] result = {0};  
 int n = (int)((a-h- b) / h);  
 result[0] += (InFunction(a) + InFunction(b)) / 2;  
 int chunkSize = n / 7; // Размер частей  
 Thread[] threads = new Thread[7];  
 for (int i = 0; i < 7; i++) {  
 int startIndex = i \* chunkSize +1;  
 int endIndex = (i +1) \* chunkSize;  
 if (i == 6) {  
 endIndex = n;  
 }  
 int finalEndIndex = endIndex;  
 //////////////////////////////////////////////////////////////////  
 Runnable task = new Runnable() {  
 public void run() {  
 double localResult = 0;  
 for (int j = startIndex; j <= finalEndIndex; j++) {  
 localResult += InFunction(a + h \* j);  
 }  
  
 synchronized(this) {  
 result[0] += localResult;  
 }  
 }  
 };  
 ////////////////////////////////////////////////////////////////  
 threads[i] = new Thread(task);  
 threads[i].start();  
 // threads[i].join();  
 }  
 for (Thread thread : threads) {  
 try {  
 thread.join(); // Ждём завершения всех потоков  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 return h\*result[0];  
 }  
 public static double InFunction(double x) //Подынтегральная функция  
 {  
 return ((Math.pow(Math.E, x))/x);  
 }  
 @Override  
 public String toString() {  
 return Min+","+Max+","+Step+","+Result;  
 }  
}

**Вывод:** научился разрабатывать приложения, обладающие графическим интерфейсом пользователя, с использованием библиотеки Swing.