Міністерство освіти і науки України  
Донецький Національний університет  
імені Василя Стуса  
Кафедра інформаційних технологій  
Факультет інформаційних та прикладних технологій  
  
  
  
  
  
  
  
  
  
  
ЗВІТ  
До лабораторної роботи №3  
З дисципліни: «Технології розподілених систем та паралельних обчислень»

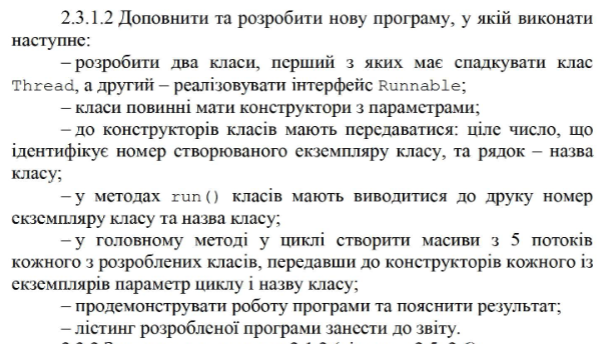
Виконав:  
студент 3-го курсу групи 122Б,  
Суханов А. О.  
  
  
 Перевірив:  
 доцент Кожем’яко А. В.

ВІННИЦЯ, 2022

Лабораторна робота №3

Тема: Багатопоточне програмування у Java

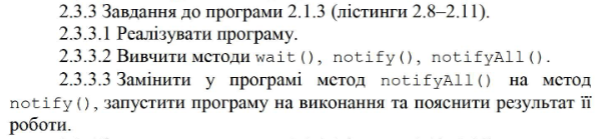
Завдання 1:



Програму виконано. Лістинг:

public class Main {  
  
  
 //1-ий клас, що спадковує Thread  
 static class NewThread extends Thread{  
  
 int number;  
 String name;  
  
 NewThread(int number, String name){  
 this.number = number;  
 this.name = name;  
 }  
  
 public void run(){  
 System.*out*.println("Number: " + this.number+ "; Name: " +this.name);  
 }  
 }  
  
 //2-ий клас, що реалізовує інтерфейс Runnable  
 static class NewRunnable implements Runnable {  
 int number;  
 String name;  
  
 NewRunnable(int number, String name){  
 this.number = number;  
 this.name = name;  
 }  
  
 //Метод run(), що виводить екземляр класу та його назву.  
 public void run(){  
 System.*out*.println("Number: " + this.number + "; Name: " +this.name);  
 }  
 }  
  
  
 public static void main(String[] args) {  
 NewThread[] ThisMassive = new NewThread[5];  
 NewRunnable[] NewMassive = new NewRunnable[5];  
 Thread[] tread = new Thread[5];  
  
 for(int i=0; i<5; i++){  
 ThisMassive[i] = new NewThread(i, "ThisMassive");  
 NewMassive[i] = new NewRunnable(i, "NewMassive");  
 tread[i] = new Thread(NewMassive[i]);  
 }  
  
 for(int i=0; i<5; i++){  
 ThisMassive[i].start();  
 tread[i].start();  
 }  
 }  
}

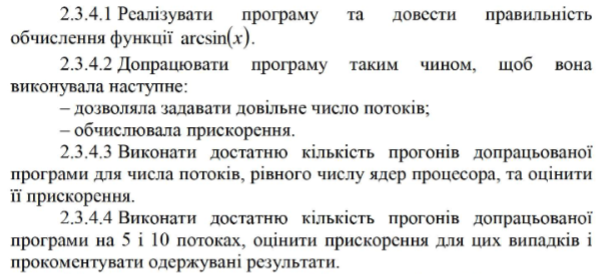
Завдання 3

:

Програму виконано. Лістинг:

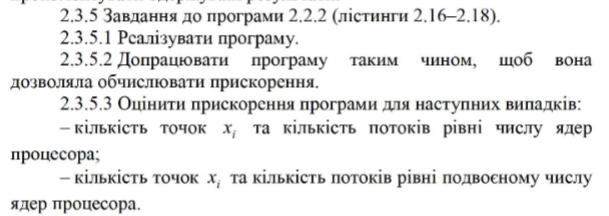
public class Main {  
  
 public static class Message{  
 private String msg;  
  
 public void setMessage(String msg){ this.msg = msg; }  
 public String getMessage(){ return msg; }  
 }  
  
 public static class Waiter implements Runnable{  
 private Message msg;  
  
 public Waiter(Message msg){ this.msg = msg; }  
  
 public void run(){  
 String name = Thread.currentThread().getName();  
 synchronized (msg){  
 try{  
 System.out.println(name + " waits for nitification " + System.currentTimeMillis());  
 msg.wait();  
 } catch(InterruptedException e){  
 e.printStackTrace();  
 }  
 System.out.println(name + " notified " + System.currentTimeMillis());  
 System.out.println(name + " received message " + msg.getMessage());  
 }  
 }  
 }  
  
 public static class Notifier implements Runnable{  
 private Message msg;  
  
 public Notifier(Message msg){  
 this.msg = msg;  
 }  
  
 public void run(){  
 String name = Thread.*currentThread*().getName();  
 System.*out*.println(name + " started");  
 try{  
 Thread.*sleep*(1500);  
 synchronized (msg){  
 msg.setMessage(name + " did his work");  
 msg.notifyAll();  
 }  
 } catch (InterruptedException e){  
 e.printStackTrace();  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 Message msg = new Message();  
 Waiter waiter1 = new Waiter(msg);  
 new Thread(waiter1, "waiter1").start();  
  
 Waiter waiter2 = new Waiter(msg);  
 new Thread(waiter2, "waiter2").start();  
  
 Notifier notifier = new Notifier(msg);  
 new Thread(notifier, "notifier").start();  
 System.out.println("all threads started");  
 }  
}

Завдання 4:



//завдання 4  
  
import java.math.BigDecimal;  
import java.util.Locale;  
import java.util.Scanner;  
  
public class Main {  
  
 public static class ComputeArcSin{  
 private BigDecimal arcsin;  
  
 private BigDecimal bigFact(BigDecimal k){  
 int a=k.compareTo(BigDecimal.*ZERO*);  
 return (a == 0) ? BigDecimal.*ONE* : k.multiply(bigFact(k.subtract(BigDecimal.*ONE*)));  
 }  
  
 public void bigArcSin(int n, double x, int threadNum, int numThreads){  
 BigDecimal fact1, fact2, numer, denom;  
 BigDecimal a = new BigDecimal(x);  
 BigDecimal arcsin = BigDecimal.*ZERO*;  
  
 for(int i = threadNum; i < n; i+=numThreads){  
 BigDecimal four = new BigDecimal(4);  
 fact1 = bigFact(new BigDecimal(i\*2));  
 fact2 = bigFact(new BigDecimal(i));  
 fact2 = fact2.multiply(fact2);  
 numer = a.pow(i\*2+1);  
 numer = numer.multiply(fact1);  
 denom = four.pow(i);  
 denom = denom.multiply(fact2);  
 denom = denom.multiply(new BigDecimal(2\*i+1));  
 numer = numer.divide(denom, BigDecimal.*ROUND\_HALF\_DOWN*);  
 arcsin = arcsin.add(numer);  
 }  
 this.arcsin = arcsin;  
  
 }  
  
 public BigDecimal getResult(){  
 return arcsin; }  
 }  
  
 public static class ParallelThread extends Thread {  
 private ComputeArcSin compute;  
 private int n;  
 private double x;  
 private int threadNum;  
 private int numThreads;  
  
 public ParallelThread(ComputeArcSin compute, int n, double x, int threadNum, int numThreads){  
 this.compute = compute;  
 this.n = n;  
 this.x = x;  
 this.threadNum = threadNum;  
 this.numThreads = numThreads;  
 }  
  
 public void run(){  
 compute.bigArcSin(n, x, threadNum, numThreads);  
 }  
 }  
  
  
 public static final int *TERMS\_COUNT* = 100;  
 public static final double *X* = 0.85;  
  
 public static void main(String[] args) {  
  
 Scanner inp = new Scanner(System.*in*);  
 System.*out*.println("Enter thread number:");  
 int numThreads = inp.nextInt();  
  
 //int numThreads = Runtime.getRuntime().availableProcessors();  
  
 long p1 = System.*nanoTime*();  
  
 ComputeArcSin[] compute = new ComputeArcSin[numThreads];  
  
 ParallelThread[] pth = new ParallelThread[numThreads];  
 for(int i=0; i<numThreads; i++){  
 compute[i] = new ComputeArcSin();  
 pth[i] = new ParallelThread(compute[i], *TERMS\_COUNT*, *X*, i, numThreads);  
 }  
  
 p1 = System.*nanoTime*() - p1;  
 long t = System.*nanoTime*();  
  
 for(int i = 0; i < numThreads; i++){  
 pth[i].start();  
 }  
 for(int i = 0; i < numThreads; i++){  
 try{  
 pth[i].join();  
 } catch (InterruptedException e){  
 e.printStackTrace(System.*err*);  
 }  
 }  
 BigDecimal y = new BigDecimal(0.);  
  
 t = System.*nanoTime*() - t;  
 long p2 = System.*nanoTime*();  
  
 for(int i = 0; i < numThreads; i++){  
 y = y.add(compute[i].getResult());  
 }  
 BigDecimal err = new BigDecimal(Math.*asin*(*X*));  
 err = err.subtract(y).abs();  
  
 p2 = System.*nanoTime*() - p2;  
  
 double p = (1.E-9 \*(double)(p1 + p2)) / (1.E-9 \*(double)(p1 + p2 + t));  
 double s = 1/(p+(1-p)/numThreads);  
  
 System.*out*.printf(Locale.*ENGLISH*, "x = %4.3f arcsin = %20.19f err = %.3e\n", *X*, y, err);  
 System.*out*.printf(Locale.*ENGLISH*, "tp = %4.3f\n", 1.E-9 \* (double) t);  
 System.*out*.printf(Locale.*ENGLISH*, "Speedup = %4.3f\n", s );  
 }  
}

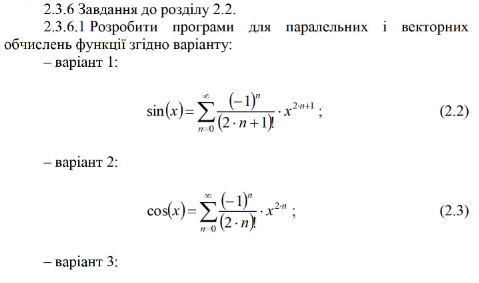
Завдання 5:

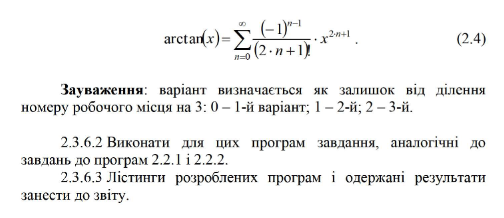


Завдання виконане, лістинг:

//завдання 5  
import java.math.BigDecimal;  
import java.util.Locale;  
  
public class Main {  
  
 public static class ComputeArcSin{  
 private BigDecimal arcsin;  
  
 private BigDecimal bigFact(BigDecimal k){  
 int a=k.compareTo(BigDecimal.ZERO);  
 return (a == 0) ? BigDecimal.ONE : k.multiply(bigFact(k.subtract(BigDecimal.ONE)));  
 }  
  
 public void bigArcSin(int n, double x){  
 BigDecimal fact1, fact2, numer, denom;  
 BigDecimal a = new BigDecimal(x);  
 BigDecimal arcsin = BigDecimal.ZERO;  
  
 for(int i = 0; i < n; i++){  
 BigDecimal four = new BigDecimal(4);  
 fact1 = bigFact(new BigDecimal(i\*2));  
 fact2 = bigFact(new BigDecimal(i));  
 fact2 = fact2.multiply(fact2);  
 numer = a.pow(i\*2+1);  
 numer = numer.multiply(fact1);  
 denom = four.pow(i);  
 denom = denom.multiply(fact2);  
 denom = denom.multiply(new BigDecimal(2\*i+1));  
 numer = numer.divide(denom, BigDecimal.ROUND\_HALF\_DOWN);  
 arcsin = arcsin.add(numer);  
 }  
 this.arcsin = arcsin;  
  
 }  
  
 public BigDecimal getResult(){  
 return arcsin; }  
 }  
  
 public static class VectorThread extends Thread {  
 private ComputeArcSin compute;  
 private int n;  
 private double x;  
  
 public VectorThread(ComputeArcSin compute, int n, double x){  
 this.compute = compute;  
 this.n = n;  
 this.x = x;  
 }  
  
 public void run(){  
 compute.bigArcSin(n, x);  
 }  
 }  
  
 public static final int TERMS\_COUNT = 100;  
 public static final int THREADS\_COUNT = 16;  
 public static final double DX = 0.85/THREADS\_COUNT;  
  
 public static void main(String[] args) {  
 long p1 = System.nanoTime();  
  
 double[] x = new double[THREADS\_COUNT];  
 for(int i=0; i<THREADS\_COUNT; i++){  
 x[i] = DX\*(i+1);  
 }  
  
 ComputeArcSin[] compute = new ComputeArcSin[THREADS\_COUNT];  
 VectorThread[] vth = new VectorThread[THREADS\_COUNT];  
 for(int i=0; i<THREADS\_COUNT; i++){  
 compute[i] = new ComputeArcSin();  
 vth[i] = new VectorThread(compute[i], TERMS\_COUNT, x[i]);  
 }  
  
 p1 = System.nanoTime() - p1;  
 long t = System.nanoTime();  
  
 for(int i = 0; i < *THREADS\_COUNT*; i++){  
 vth[i].start();  
 }  
 for(int i = 0; i < *THREADS\_COUNT*; i++){  
 try{  
 vth[i].join();  
 } catch (InterruptedException e){  
 e.printStackTrace(System.err);  
 }  
 }  
 BigDecimal[] y = new BigDecimal[THREADS\_COUNT];  
 BigDecimal[] err = new BigDecimal[THREADS\_COUNT];  
  
 t = System.nanoTime() - t;  
 long p2 = System.nanoTime();  
  
 for(int i = 0; i < THREADS\_COUNT; i++){  
 y[i] = compute[i].getResult();  
 err[i] = new BigDecimal(Math.asin(x[i]));  
 err[i] = err[i].subtract(y[i]).abs();  
 System.out.printf(Locale.ENGLISH, "x = %4.3f Arcsin = %20.19f err = %.3e\n", x[i], y[i], err[i]);  
 }  
  
 p2 = System.nanoTime() - p2;  
 double p = (1.E-9 \*(double)(p1 + p2)) / (1.E-9 \*(double)(p1 + p2 + t));  
 double s = 1/(p+(1-p)/THREADS\_COUNT);  
  
 System.out.printf(Locale.ENGLISH, "tp = %4.3f\n", 1.E-9 \* (double) t);  
 System.out.printf(Locale.ENGLISH, "Speedup = %4.3f\n", s );  
 }  
}

Завдання 6:





Програму виконано, лістинг:

import java.math.BigDecimal;  
import java.util.Locale;  
import java.util.Scanner;  
  
public class Main {  
  
 public static class ComputeCos {  
 private BigDecimal cos;  
  
 private BigDecimal bigFact(BigDecimal k){  
 int a=k.compareTo(BigDecimal.ZERO);  
 return (a == 0) ? BigDecimal.ONE : k.multiply(bigFact(k.subtract(BigDecimal.ONE)));  
 }  
  
 public void bigCos(int n, double x, int threadNum, int numThreads){  
 BigDecimal factor, numer, denom;  
 BigDecimal a = new BigDecimal(x);  
 BigDecimal cos = BigDecimal.ZERO;  
  
 for(int i = threadNum; i < n; i+=numThreads){  
 numer = new BigDecimal(-1);  
 numer = numer.pow(i);  
  
 factor = a.pow(i\*2);  
 numer = numer.multiply(factor);  
  
 denom = bigFact(new BigDecimal(i\*2));  
  
 numer = numer.divide(denom, BigDecimal.ROUND\_HALF\_DOWN);  
 cos = cos.add(numer);  
 }  
 this.cos = cos;  
  
 }  
  
 public BigDecimal getResult(){  
 return cos; }  
  
 }  
  
 public static class ParallelThread extends Thread {  
 private ComputeCos compute;  
 private int n;  
 private double x;  
 private int threadNum;  
 private int numThreads;  
  
 public ParallelThread(ComputeCos compute, int n, double x, int threadNum, int numThreads){  
 this.compute = compute;  
 this.n = n;  
 this.x = x;  
 this.threadNum = threadNum;  
 this.numThreads = numThreads;  
 }  
  
 public void run(){  
 compute.bigCos(n, x, threadNum, numThreads);  
 }  
  
 }  
  
  
 public static final int TERMS\_COUNT = 100;  
 public static final double X = 0.85;  
  
 public static void main(String[] args) {  
  
 Scanner inp = new Scanner(System.in);  
 System.out.println("Enter thread number:");  
 int numThreads = inp.nextInt();  
  
 long p1 = System.nanoTime();  
  
 ComputeCos[] compute = new ComputeCos[numThreads];  
  
 ParallelThread[] pth = new ParallelThread[numThreads];  
 for(int i=0; i<numThreads; i++){  
 compute[i] = new ComputeCos();  
 pth[i] = new ParallelThread(compute[i], TERMS\_COUNT, X, i, numThreads);  
 }  
  
 p1 = System.nanoTime() - p1;  
 long t = System.nanoTime();  
  
 for(int i = 0; i < numThreads; i++){  
 pth[i].start();  
 }  
 for(int i = 0; i < numThreads; i++){  
 try{  
 pth[i].join();  
 } catch (InterruptedException e){  
 e.printStackTrace(System.err);  
 }  
 }  
 BigDecimal y = new BigDecimal(0.);  
  
 t = System.nanoTime() - t;  
 long p2 = System.nanoTime();  
  
 for(int i = 0; i < numThreads; i++){  
 y = y.add(compute[i].getResult());  
 }  
 BigDecimal err = new BigDecimal(Math.cos(X));  
 err = err.subtract(y).abs();  
  
 p2 = System.nanoTime() - p2;  
  
 double p = (1.E-9 \*(double)(p1 + p2)) / (1.E-9 \*(double)(p1 + p2 + t));  
 double s = 1/(p+(1-p)/numThreads);  
  
 System.out.printf(Locale.ENGLISH, "x = %4.3f cos = %20.19f err = %.3e\n", X, y, err);  
 System.out.printf(Locale.ENGLISH, "tp = %4.3f\n", 1.E-9 \* (double) t);  
 System.out.printf(Locale.ENGLISH, "Speedup = %4.3f\n", s );  
 }  
}