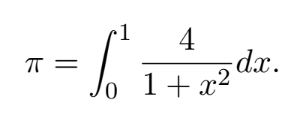
| EI1024/MT1024 “Programaci´on Concurrente y Paralela” 2022–23  Nombre y apellidos (1): Belén Ariño Bolinches  Nombre y apellidos (2): [Aleix Grau Bas](mailto:al394752@uji.es)  Tiempo empleado para tareas en casa en formato *h:mm* (obligatorio): . . . . . . . . . . . . . | Entregable  para  Laboratorio  la04 g |
| --- | --- |

Tema 06. El Problema de la Atomicidad en Java

Tema 07. *Thread Pools* e Interfaces Gráficas en Java

| 1 |
| --- |

Se desea calcular el número *π* mediante integración numérica de la siguiente función:

Este método no es el m´as r´apido para calcular el n´umero *π*, pero s´ı uno de las m´as simples. **Consiste en calcular la anterior integral mediante una aproximaci´on num´erica basada en el c´alculo y acumulaci´on del ´area de numerosos rect´angulos peque˜nos.** 

Uno de los parámetros más importantes es el número de rectángulos cuya área se va a sumar. En este caso, este parámetro será pasado en la línea de argumentos, después del número de hebras.

El siguiente programa realiza el c´alculo de forma secuencial. Con vistas a facilitar el desarrollo posterior de la versi´on paralela, este c´odigo secuencial contiene un fragmento de c´odigo comen tado, adem´as de la declaraci´on e inicializaci´on de la variables numHebras y numRectangulos. Ambas partes no son ´utiles en la versi´on secuencial, pero, la inclusi´on de este fragmento de c´odigo simplifica el desarrollo de la versi´on paralela.

*/∗*

*// ===========================================================================*

*c l a s s Acumula {*

*// ===========================================================================*

*d o u bl e suma ;*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−− Acumula ( ) {*

*// . . .*

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−− v o i d acumulaDato ( d o u bl e d a t o ) {*

*// . . .*

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−− d o u bl e dameDato ( ) {*

*// . . .*

*}*

*}*

*// ===========================================================================*

*c l a s s M iHebraMultAcumulac iones e x t e n d s Thread {*

*// =========================================================================== i n t miId , numHebras ;*

*l o n g numRec tangulos ;*

*Acumula a ;*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−− M iHebraMultAcumulac iones ( i n t miId , i n t numHebras , l o n g numRectangulos , Acumula a ) {*

*// . . .*

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−− p u b l i c v o i d run ( ) {*

*// . . .*

*}*

*}*

*// =========================================================================== c l a s s MiHebraUnaAcumulacion e x t e n d s Thread {*

*// =========================================================================== // . . .*

*}*

*// =========================================================================== c l a s s MiHebraMultAcumulacionAtomic e x t e n d s Thread {*

*// =========================================================================== // . . .*

*}*

*// =========================================================================== c l a s s MiHebraUnaAcumulacionAtomic e x t e n d s Thread {*

*// =========================================================================== // . . .*

*}*

*∗/*

*// ===========================================================================* c l a s s EjemploNumeroPI *{*

*// ===========================================================================*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public s t a t i c void main ( S t ri n g a r g s [ ] ) *{*

long numRectangulos ;

double b a seRec t an gul o , x , suma , pi ;

int numHebras ;

long t1 , t 2 ;

double tSec , tPar ;

*// Acumula a ;*

*// M iHebraMultAcumulac iones v t [ ] ;*

*// Comprobacion de l o s argumen tos de en t r a d a .*

i f ( a r g s . l e n g t h != 2 ) *{*

System . out . p r i n t l n ( ”ERROR: numero de argumentos i n c o r r e c t o . ” ) ; System . out . p r i n t l n ( ”Uso : j a v a programa *<*numHebras*> <*numRectangulos*>*” ) ; System . e x i t ( −1 ) ;

*}*

try *{*

numHebras = I n t e g e r . p a r s e I n t ( a r g s [ 0 ] ) ;

numRectangulos = Long . parseLong ( a r g s [ 1 ] ) ;

*}* catch ( NumberFormatException ex ) *{*

3

numHebras = −1;

numRectangulos = −1;

System . out . p r i n t l n ( ”ERROR: Numeros de en t r ad a i n c o r r e c t o s . ” ) ; System . e x i t ( −1 ) ;

*}*

System . out . p r i n t l n ( ) ;

System . out . p r i n t l n ( ” C al c ul o d el numero PI mediante i n t e g r a c i o n . ” ) ; *//*

*// C al c ul o d e l numero PI de forma s e c u e n c i a l .*

*//*

System . out . p r i n t l n ( ) ;

System . out . p r i n t l n ( ”Comienzo d el c a l c u l o s e c u e n c i a l . ” ) ;

t 1 = System . nanoTime ( ) ;

b a seRec t an gul o = 1. 0 / ( ( double ) numRectangulos ) ;

suma = 0 . 0 ;

for ( long i = 0 ; i *<* numRectangulos ; i++ ) *{*

x = b a seRec t an gul o ∗ ( ( ( double ) i ) + 0. 5 ) ;

suma += f ( x ) ;

*}*

pi = b a seRec t an gul o ∗ suma ;

t 2 = System . nanoTime ( ) ;

t S e c = ( ( double ) ( t 2 − t 1 ) ) / 1. 0 e9 ;

System . out . p r i n t l n ( ” Ve r si on s e c u e n c i a l . Numero PI : ” + pi ) ;

System . out . p r i n t l n ( ”Tiempo s e c u e n c i a l ( s . ) : ” + t S e c ) ; */∗*

*//*

*// C al c ul o d e l numero PI de forma p a r a l e l a :*

*// M u l t i p l e s acumul ac i one s por heb r a .*

*//*

*System . ou t . p r i n t l n ( ) ;*

*System . ou t . p r i n t ( ”Comienzo d e l c a l c u l o p a r a l e l o : ” ) ;*

*System . ou t . p r i n t l n ( ” M u l t i p l e s acumul ac i one s por heb r a . ” ) ;*

*t 1 = System . nanoTime ( ) ;*

*// . . .*

*t 2 = System . nanoTime ( ) ;*

*tPar = ( ( d o u bl e ) ( t 2 − t 1 ) ) / 1 . 0 e9 ;*

*System . ou t . p r i n t l n ( ” C al c ul o d e l numero PI : ” + p i ) ;*

*System . ou t . p r i n t l n ( ”Tiempo e j e c u c i o n ( s . ) : ” + tPar ) ;*

*System . ou t . p r i n t l n ( ” Incremen to v e l o c i d a d : ” + . . . ) ;*

*//*

*// C al c ul o d e l numero PI de forma p a r a l e l a :*

*// Una acumulac ion por heb r a .*

*// . . .*

*//*

*// C al c ul o d e l numero PI de forma p a r a l e l a :*

*// M u l t i p l e s acumul ac i one s por heb r a ( Atomica )*

*// . . .*

*//*

*// C al c ul o d e l numero PI de forma p a r a l e l a :*

*// Una acumulac ion por heb r a ( Atomica ) .*

*// . . .*

*∗/*

System . out . p r i n t l n ( ) ;

System . out . p r i n t l n ( ”Fin de programa . ” ) ;

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* s t a t i c double f ( double x ) *{*

return ( 4 . 0 / ( 1. 0 + x∗x ) ) ;

*}*

*}*

4

1.1) Estudia el código anterior y paralelízalo mediante el uso de hebras con una distribución c ́ıclica. Utiliza un objeto de la clase Acumula para almacenar el resultado. En esta versi´on paralela cada vez que las hebras calculan el ´area de un rect´angulo, deben acumularlo sobre el objeto compartido de la clase Acumula. Para un correcto manejo del programa, hay que asegurar que el acceso al objeto compartido sea *thread-safe*. No crees un nuevo programa. Haz que esta implementaci´on paralela se ejecute a continuaci´on de la versi´on secuencial dentro del mismo programa. Ello permitir´a obtener los tiempos y los incrementos de velocidad de forma m´as r´apida y automatizada.

Escribe a continuación la parte de tu código que realiza esta tarea: la definición de la clase **MiHebraMultAcumulaciones** y el código incluido en el programa principal que permite gestionar los objetos de esta clase.

//----------------------------------------MiHebraMultAcomulaciones--------------------------------------------

public class MiHebraMultAcumulaciones extends Thread{

int miId, numHebras;

long numRectangulos;

Acomula a;

double baseRectangulo;

MiHebraMultAcumulaciones(int miId, int numHebras, long numRectangulos, Acomula a){

this.miId=miId;

this.numHebras=numHebras;

this.numRectangulos=numRectangulos;

this.a=a;

baseRectangulo=1.0/((double)numRectangulos);

}

public void run(){

for (int i=miId;i<numRectangulos;i+=numHebras){

double x=baseRectangulo\*(((double) i)+0.5);

a.acomulaDato(*f*(x));

}

}

static double f(double x){

return (4.0/(1.0 + x\*x));

}

}

//----------------------------------------EjemploNumeroPI--------------------------------------------

System.*out*.println();

System.*out*.println("Comienzo del calculo paralelo multiples acomulaciones");

t1=System.*nanoTime*();

a=new Acomula();

MiHebraMultAcumulaciones[] vector=new MiHebraMultAcumulaciones[numHebras];

for (int i=0;i<numHebras;i++){

vector[i]= new MiHebraMultAcumulaciones(i,numHebras,numRectangulos,a);

vector[i].run();

}

for (int i=0;i<numHebras;i++){

vector[i].join();

}

pi=baseRectangulo\*suma;

t2=System.*nanoTime*();

tPar=((double) (t2-t1))/1.0e9;

System.*out*.println("Version paralela. Numero PI: "+pi);

System.*out*.println("Tiempo paralelo: "+tPar);

System.*out*.println("Incremento de la velocidad: "+tSec/tPar);

1.2) Modifica el programa anterior, de modo que en la versión paralela las hebras acumulen el área que han calculado en una variable local (suma), antes de sumarla al objeto compartido.

No crees un nuevo programa. Haz que esta implementaci´on paralela se ejecute a continua ci´on de la versi´on paralela desarrollada en el apartado anterior. Ello permitir´a obtener los tiempos y los incrementos de velocidad de forma m´as r´apida y automatizada.

Escribe a continuaci´on la parte de tu c´odigo que realiza esta tarea: la definici´on de la clase MiHebraUnaAcumulacion y el c´odigo incluido en el programa principal que permite gestionar los objetos de esta clase.

//----------------------------------------MiHebraUnaAcomulacion--------------------------------------------

public class MiHebraUnaAcumulacion extends Thread{

int miId, numHebras;

long numRectangulos;

Acomula a;

double baseRectangulo,suma;

MiHebraUnaAcumulacion(int miId, int numHebras, long numRectangulos, Acomula a){

this.miId=miId;

this.numHebras=numHebras;

this.numRectangulos=numRectangulos;

this.a=a;

baseRectangulo=1.0/((double)numRectangulos);

suma=0.0;

}

public void run(){

for (int i=miId;i<numRectangulos;i+=numHebras){

double x=baseRectangulo\*(((double) i)+0.5);

suma+=*f*(x);

}

a.acomulaDato(suma);

}

static double f(double x){

return (4.0/(1.0 + x\*x));

}

}

//----------------------------------------EjemploNumeroPI--------------------------------------------

System.*out*.println();

System.*out*.println("Comienzo del calculo paralelo una acomulacion ");

t1=System.*nanoTime*();

a=new Acomula();

MiHebraUnaAcumulacion[] vector2=new MiHebraUnaAcumulacion[numHebras];

for (int i=0;i<numHebras;i++){

vector2[i]= new MiHebraUnaAcumulacion(i,numHebras,numRectangulos,a);

vector2[i].run();

}

for (int i=0;i<numHebras;i++){

vector2[i].join();

}

pi=baseRectangulo\*suma;

t2=System.*nanoTime*();

tPar=((double) (t2-t1))/1.0e9;

System.*out*.println("Version paralela. Numero PI: "+pi);

System.*out*.println("Tiempo paralelo: "+tPar);

System.*out*.println("Incremento de la velocidad: "+tSec/tPar);

1.3) En las DOS versiones paralelas anteriores se ha utilizado un **objeto de la clase Acumula** que permite acumular n´umeros reales con precisión doble de forma atómica, pero tambi´en se podr´ıa realizar empleando clases y operadores atómicos, como **DoubleAdder**.

Para ello, define las clases MiHebraMultAcumulacionesAtomic y MiHebraUnaAcumulacionAtomic. Estas clases deben manejar un objeto de la clase DoubleAdder y acumular los valores con el m´etodo add, mientras que el valor final se obtiene con el m´etodo sum.

Escribe a continuaci´on los cambios realizados en el c´odigo.

Ten en cuenta que la mejor opci´on ser´ıa eliminar completamente la clase Acumula y en su lugar utilizar la clase atómica.

public class MiHebraMultAcumulacionAtomic extends Thread{

DoubleAdder adder;

int miId, numHebras;

long numRectangulos;

double baseRectangulo;

MiHebraMultAcumulacionAtomic(int miId, int numHebras, long numRectangulos, DoubleAdder a){

this.miId=miId;

this.numHebras=numHebras;

this.numRectangulos=numRectangulos;

this.adder=a;

baseRectangulo=1.0/((double)numRectangulos);

}

public void run(){

for (int i=miId;i<numRectangulos;i+=numHebras){

double x=baseRectangulo\*(((double) i)+0.5);

adder.add(*f*(x));

}

}

static double f(double x){

return (4.0/(1.0 + x\*x));

}

}

public class MiHebraUnaAcumulacionAtomic extends Thread{

DoubleAdder adder;

int miId, numHebras;

long numRectangulos;

double baseRectangulo,suma;

MiHebraUnaAcumulacionAtomic(int miId, int numHebras, long numRectangulos, DoubleAdder a){

this.miId=miId;

this.numHebras=numHebras;

this.numRectangulos=numRectangulos;

this.adder=a;

baseRectangulo=1.0/((double)numRectangulos);

suma=0.0;

}

public void run(){

for (int i=miId;i<numRectangulos;i+=numHebras){

double x=baseRectangulo\*(((double) i)+0.5);

suma+=*f*(x);

}

adder.add(suma);

}

static double f(double x){

return (4.0/(1.0 + x\*x));

}

}

//PARALELO distribución cíclica

//múltiples acumulaciones por hebra (Atómica)

System.*out*.println();

System.*out*.println("Comienzo del calculo paralelo multiples acomulaciones (Atomic)");

t1=System.*nanoTime*();

DoubleAdder adder= new DoubleAdder();

MiHebraMultAcumulacionAtomic[] vector3=new MiHebraMultAcumulacionAtomic[numHebras];

for (int i=0;i<numHebras;i++){

vector3[i]= new MiHebraMultAcumulacionAtomic(i,numHebras,numRectangulos,adder);

vector3[i].run();

}

for (int i=0;i<numHebras;i++){

vector3[i].join();

}

pi=baseRectangulo\*adder.sum();

t2=System.*nanoTime*();

tPar=((double) (t2-t1))/1.0e9;

System.*out*.println("Version paralela. Numero PI: "+pi);

System.*out*.println("Tiempo paralelo: "+tPar);

System.*out*.println("Incremento de la velocidad: "+tSec/tPar);

//PARALELO distribución cíclica

//una acumulacione por hebra (Atomic)

System.*out*.println();

System.*out*.println("Comienzo del calculo paralelo una acomulacion (Atomic)");

t1=System.*nanoTime*();

DoubleAdder adder2= new DoubleAdder();

MiHebraUnaAcumulacionAtomic[] vector4=new MiHebraUnaAcumulacionAtomic[numHebras];

for (int i=0;i<numHebras;i++){

vector4[i]= new MiHebraUnaAcumulacionAtomic(i,numHebras,numRectangulos,adder2);

vector4[i].run();

}

for (int i=0;i<numHebras;i++){

vector4[i].join();

}

pi=baseRectangulo\*adder2.sum();

t2=System.*nanoTime*();

tPar=((double) (t2-t1))/1.0e9;

System.*out*.println("Version paralela. Numero PI: "+pi);

System.*out*.println("Tiempo paralelo: "+tPar);

System.*out*.println("Incremento de la velocidad: "+tSec/tPar);

| 2 |
| --- |

Se dispone de una interfaz gráfica con un cuadro de texto y dos botones denominados **Comienza secuencia** y **Cancela secuencia**. Por el momento, la interfaz no hace nada cuando el usuario realiza alguna acción sobre los botones o sobre el cuadro de texto.

La interfaz est´a definida por el siguiente código:

import j a v a . awt . ∗ ;

import j a v a . awt . e ven t . ∗ ;

import j a v a x . swing . ∗ ;

import j a v a x . swing . e ven t . ∗ ;

*/∗*

*// =========================================================================== c l a s s Zona In tercamb io {*

*// =========================================================================== // . . .*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−− v o i d se tT iempo ( . . . ) {*

*// . . .*

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−− l o n g getT iempo ( . . . ) {*

*// . . .*

*}*

*}*

*∗/*

*// ===========================================================================* public c l a s s GUISecuenciaPrimos *{*

*// ===========================================================================* JFrame c o n t ai n e r ;

JPanel j p a n el ;

JTe x tField t x fM e n s a j e s ;

JButton btnComienzaSecuencia , b t nC an c el a Se c ue n ci a ;

J S l i d e r sl dE s p e r a ;

*// Heb r aC alc ul a d o r a t ; // E j e r c i c i o 2 . 2*

*// Zona In tercamb io z ; // E j e r c i c i o 2 . 3*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public s t a t i c void main ( S t ri n g a r g s [ ] ) *{*

GUISecuenciaPrimos g ui = new GUISecuenciaPrimos ( ) ;

S w i n g U t i l i t i e s . i n v o k eL a t e r (new Runnable ( ) *{*

public void run ( ) *{*

g ui . go ( ) ;

*}*

*}* ) ;

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* public void go ( ) *{*

*// C on s t an te s .*

f i n a l int valorMaximo = 1 0 0 0;

f i n a l int v al o rMedi o = 5 0 0;

*// V a r i a b l e s .*

JPanel tempPanel ;

*// Crea e l JFrame p r i n c i p a l .*

c o n t ai n e r = new JFrame ( ”GUI S e c u e n ci a de Primos ” ) ;

8

*// Cons igue e l p a n el p r i n c i p a l d e l Frame ” c o n t a i n e r ” .*

j p a n el = ( JPanel ) c o n t ai n e r . getContentPane ( ) ;

j p a n el . se tL a y ou t ( new GridLayout ( 3 , 1 ) ) ;

*// Crea e i n s e r t a l a e t i q u e t a y e l campo de t e x t o para l o s mensa jes .* t x fM e n s a j e s = new JTe x tField ( 20 ) ;

t x fM e n s a j e s . s e t E d i t a b l e ( f a l s e ) ;

tempPanel = new JPanel ( ) ;

tempPanel . se tL a y ou t ( new FlowLayout ( ) ) ;

tempPanel . add ( new JLabel ( ” S e c u e n ci a : ” ) ) ;

tempPanel . add ( t x fM e n s a j e s ) ;

j p a n el . add ( tempPanel ) ;

*// Crea e i n s e r t a l o s b o t o n e s de Comienza s e c u e n c i a y Cancela s e c u e n c i a .* btnComienzaSecuencia = new JButton ( ”Comienza s e c u e n ci a ” ) ;

b tnC a n c el a S e c u en ci a = new JButton ( ” Cancela s e c u e n ci a ” ) ;

tempPanel = new JPanel ( ) ;

tempPanel . se tL a y ou t ( new FlowLayout ( ) ) ;

tempPanel . add ( btnComienzaSecuencia ) ;

tempPanel . add ( b t nC a n cel aS e cu e n ci a ) ;

j p a n el . add ( tempPanel ) ;

*// Crea e i n s e r t a e l s l i d e r para c o n t r o l a r e l t iempo de e s p e r a .*

sl dE s p e r a = new J S l i d e r ( J S l i d e r .HORIZONTAL, 0 , valorMaximo , v al o rMedi o ) ; tempPanel = new JPanel ( ) ;

tempPanel . se tL a y ou t ( new BorderLayout ( ) ) ;

tempPanel . add ( new JLabel ( ”Tiempo de e s p e r a : ” ) ) ;

tempPanel . add ( sl dE s p e r a ) ;

j p a n el . add ( tempPanel ) ;

*// Ac t iv a i n i c i a l m e n t e l o s 2 b o t o n e s .*

btnComienzaSecuencia . se tEn abled ( true ) ;

b tnC a n c el a S e c u en ci a . se tEn abled ( true ) ;

*// Anyade c o d i g o para p r o c e s a r e l e v e n t o d e l b o t on de Comienza s e c u e n c i a .* btnComienzaSecuencia . a d dA c ti o nLi s t e n e r ( new A c ti o n Li s t e n e r ( ) *{*

public void a c ti o nP e r f o rm e d ( ActionEvent e ) *{*

*// . . .*

*}*

*}* ) ;

*// Anyade c o d i g o para p r o c e s a r e l e v e n t o d e l b o t on de Cancela s e c u e n c i a .* b tnC a n c el a S e c u en ci a . a d dA c ti o nLi s t e n e r ( new A c ti o n Li s t e n e r ( ) *{*

public void a c ti o nP e r f o rm e d ( ActionEvent e ) *{*

*// . . .*

*}*

*}* ) ;

*// Anyade c o d i g o para p r o c e s a r e l e v e n t o d e l s l i d e r ” Espera ” .* sl dE s p e r a . addChangeLi s tene r ( new Ch an geLi s tene r ( ) *{*

public void stateChanged ( ChangeEvent e ) *{*

J S l i d e r s l = ( J S l i d e r ) e . g e t S o u r c e ( ) ;

i f ( ! s l . g e tV al u e I s A d j u s ti n g ( ) ) *{*

long ti em p oMili s e g u n d o s = ( long ) s l . ge tV alue ( ) ;

System . out . p r i n t l n ( ” J S l i d e r v al u e = ” + ti em p oMili s e g u n d o s ) ;

*// . . .*

*}*

*}*

*}* ) ;

9

*// F i j a c a r a c t e r i s t i c a s d e l c o n t a i n e r .*

c o n t ai n e r . s e t D e f a ul t Cl o s eO p e r a ti o n ( JFrame . EXIT ON CLOSE ) ;

c o n t ai n e r . pack ( ) ;

c o n t ai n e r . s e t R e s i z a b l e ( f a l s e ) ;

c o n t ai n e r . s e t V i s i b l e ( true ) ;

System . out . p r i n t l n ( ” % End o f r o u ti n e : go . *\* n” ) ;

*}*

*// −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−* s t a t i c boolean esPrimo ( long num ) *{*

boolean primo ;

i f ( num *<* 2 ) *{*

primo = f a l s e ;

*}* e l s e *{*

primo = true ;

long i = 2 ;

while ( ( i *<* num )&&( primo ) ) *{*

primo = ( num % i != 0 ) ;

i ++;

*}*

*}*

return ( primo ) ;

*}*

*}*

2.1) Modifica la interfaz gr´afica para que los botones Comienza secuencia y Cancela secuencia se activen y desactiven (setEnabled) de acuerdo a la l´ogica de funcionamiento de la apli caci´on :

Inicialmente el botón Comienza secuencia debe estar activado y el botón Cancela secuencia debe estar desactivado (modificar método go).

Cuando se presione el botón Comienza secuencia, este se desactiva y se activa el botón Cancela secuencia (modificar ActionListener del primero).

Cuando se presione el bot´on Cancela secuencia, ´este se desactiva y se activa el bot´on Comienza secuencia (modificar ActionListener del primero).

// Activa inicialmente los 2 botones.

btnComienzaSecuencia.setEnabled( true );

btnCancelaSecuencia.setEnabled( false );

// Anyade codigo para procesar el evento del boton de Comienza secuencia.

btnComienzaSecuencia.addActionListener( new ActionListener() {

public void actionPerformed( ActionEvent e ) {

btnComienzaSecuencia.setEnabled(false);

btnCancelaSecuencia.setEnabled(true);

}

} );

// Anyade codigo para procesar el evento del boton de Cancela secuencia.

btnCancelaSecuencia.addActionListener( new ActionListener() {

public void actionPerformed( ActionEvent e ) {

btnCancelaSecuencia.setEnabled(false);

btnComienzaSecuencia.setEnabled(true);

}

} );

| ATENCION: Los ejercicios anteriores deben realizarse en casa. Los siguientes, en el aula. ´ |
| --- |

2.2) Modifica la anterior interfaz de tal forma que en cuanto el usuario pulse el botón Comienza secuencia el programa muestre la secuencia de números primos (comenzando por el 2, 3, 5, 7, 11, etc.) en el cuadro de texto. Para ello, la hebra ***event-dispatching***debería crear una hebra trabajadora (t) en la que delegar ́ a dicho trabajo.

En cuanto el usuario pulse el botón Cancela secuencia, la generación de la secuencia debe terminar. Para detener la hebra, se **fijar ́a un valor especial en un atributo de la hebra (fin), cuyo valor ser´a revisado por ´esta cada vez que se genere un nuevo n´umero primo**. Adem´as, cada vez que se presione el bot´on Comienza secuencia la secuencia se inicia desde el principio.

Seguidamente se muestra la estructura del cuerpo de la hebra.

*// E s t r u c t u r a d e l cuerpo de l a heb r a*

long i = 1L ;

while ( ! f i n ) *{*

i f ( esPrimo ( i ) ) *{*

*// imprime ( i ) ;*

*}*

i ++;

*}*

Una hebra trabajadora no puede llamar directamente a ning ́un m étodo de ning ́un objeto gr´afico, ya que ´estos s´olo pueden ser manejados por la *event-dispatching*. Por tanto, cuando la hebra trabajadora desee realizar alguna escritura sobre el cuadro de texto (txfMensajes), debe utilizar los m´etodos invokeAndWait o invokeLater, que indican a la *event-dispatching* que labores debe realizar.

Estos m´etodos ejecutan un objeto Runnable que reciben como par´ametro de entrada, el primer m´etodo bloquea a la hebras hasta que la *event-dispatching* finaliza, por lo que es necesario gestionar dos excepciones, mientras que el segundo no bloquea a la hebra. Escribe a continuaci´on la parte de tu c´odigo que realiza tal tarea: la definici´on de la clase HebraTrabajadora y el c´odigo a incluir en el programa principal que permite gestionar los objetos de esta clase.

--------------------------------------------GUISecuenciaPrimos--------------------------------------------

// Anyade codigo para procesar el evento del boton de Comienza secuencia.

btnComienzaSecuencia.addActionListener( new ActionListener() {

public void actionPerformed( ActionEvent e ) {

btnComienzaSecuencia.setEnabled(false);

btnCancelaSecuencia.setEnabled(true);

//habra auxiliar se crea en EDT

t= new HebraCalculadora(txfMensajes);

t.start();

}

} );

// Anyade codigo para procesar el evento del boton de Cancela secuencia.

btnCancelaSecuencia.addActionListener( new ActionListener() {

public void actionPerformed( ActionEvent e ) {

btnCancelaSecuencia.setEnabled(false);

btnComienzaSecuencia.setEnabled(true);

t.cambiaValor();

}

} );

-----------------------------------------------------HebraCalculadora-----------------------------------------

import static Ej2.GUISecuenciaPrimos.\*;

public class HebraCalculadora extends Thread{

boolean fin=false;

long num=0;

JTextField txMensajes;

HebraCalculadora(JTextField txfMensajes){

this.txMensajes=txfMensajes;

}

public void run() {

try {

while (!fin) {

if (*esPrimo*(num)) {

cambiaTexto(num);

}

num++;

}

} catch (Exception e) {

throw new RuntimeException(e);

}

}

public void cambiaTexto(long num){

SwingUtilities.*invokeLater*(

new Runnable() {

public void run() {

txMensajes.setText(Long.*valueOf*(num).toString());

}

});

}

public void cambiaValor(){

if(!fin){

fin=true;

}else{

fin=false;

num=0;

}

}

}

2.3) Modifica el programa anterior de tal forma que se muestre en pantalla una barra de des lizamiento horizontal (JSlider) con la que el usuario pueda determinar la velocidad de generaci´on de n ́umeros primos (ver código inicial).

Si la barra est´a en un **extremo**, la hebra deber ́a generar n´umeros primos intercalando una demora **(método sleep) de un segundo** tras la impresión en el cuadro de texto. Si la barra est´a en el otro extremo, la hebra deber´a generar n´umeros primos sin **ninguna demora**.

Se recomienda definir y emplear una nueva clase denominada Zona Intercambio, a través de la cual se comunicaran la hebra gráfica y la hebra calculadora. La hebra gr´afica escribir´a valores en un objeto de dicha clase y la hebra calculadora tomar´a valores de dicho objeto. Por ´ultimo comentar que el tiempo de espera se expresa en milisegundos, y que el valor inicial definido en el c´odigo es 500.

Escribe a continuaci´on la parte de tu c´odigo que realiza tal tarea: la definici´on de la clase ZonaIntercambio, el c´odigo para la gesti´on de la barra de desplazamiento, y los cambios en la clase HebraTrabajadora.

--------------------------------------------ZonaIntercambio--------------------------------------------

class ZonaIntercambio {

// ===========================================================================

// ...

volatile long tiempo = 0;

ZonaIntercambio(int valorMedio){

tiempo = (long) valorMedio;

}

// -------------------------------------------------------------------------

void setTiempo(long tm) {

// ...

tiempo = tm;

}

// -------------------------------------------------------------------------

long getTiempo() {

// ...

return tiempo;

}

}

--------------------------------------------GUISecuenciaPrimos--------------------------------------------

sldEspera.addChangeListener( new ChangeListener() {

public void stateChanged( ChangeEvent e ) {

JSlider sl = ( JSlider ) e.getSource();

if ( ! sl.getValueIsAdjusting() ) {

long tiempoMilisegundos = ( long ) sl.getValue();

System.*out*.println( "JSlider value = " + tiempoMilisegundos );

z.setTiempo(tiempoMilisegundos);

}

}

} );

-----------------------------------------------------HebraCalculadora-----------------------------------------

public class HebraCalculadora extends Thread{

boolean fin=false;

long num=0;

ZonaIntercambio z;

JTextField txMensajes;

HebraCalculadora(JTextField txfMensajes,ZonaIntercambio z){

this.txMensajes=txfMensajes;

this.z=z;

}

public void run() {

try {

while (!fin) {

if (*esPrimo*(num)) {

*sleep*(z.getTiempo());

cambiaTexto(num);

}

num++;

}

} catch (Exception e) {

throw new RuntimeException(e);

}

}

2.4) Se desea sustituir la barra de deslizamiento por dos botones adicionales: Un bot´on a˜nadir´a 0,1 segundos al tiempo de espera, mientras que el otro botón restar´a 0,1 segundos al tiempo de espera.

No hagas ninguna implementación, pero responde a la siguiente pregunta. ¿Se podrá realizar dicha modificación sólo con el operador volatile o habrá que recurrir al modificador synchronized? Justifica la respuesta.

Nos bastaría con solo volatile ya que realizamos una operación atómica sobre la variable tiempo

| 3 |
| --- |

Este ejercicio es una continuación del ejercicio 1.

3.1) Completa la siguiente tabla para 500 000 000 de rect ́angulos. Obt´en los resultados para 4 hebras en el ordenador del aula. Obt´en los resultados para 16 hebras en patan. Redondea los tiempos dejando s´olo tres decimales y redondea los incrementos dejando dos decimales. Los resultados utilizando la clase at´omica son optativos. Para obtenerlos, debes sustituir el objeto de la clase Acumula por un objeto de la clase at´omica propia de Java 8, y utilizar sus métodos en la actualizaci´on.

Justifica los resultados obtenidos.

Ejecuci´on con 500 000 000 rect´angulos

|  | 4 hebras | | 16 hebras | |
| --- | --- | --- | --- | --- |
|  | Tiempo | Incremento | Tiempo | Incermento |
| Secuencial | 4.64 | — |  | — |
| Paralela: Múltiples acumul. | 2.34 | 1.98 |  |  |
| Paralela: Una única acumul. | 1.25 | 3.69 |  |  |
| Paralela: Múltiples acumul. (clase atom.) | 1.42 | 3.24 |  |  |
| Paralela: Una ´unica acumul. (clase atom.) | 1.22 | 3.77 |  |  |

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .