Concorrencia e Distribución

Teoría

Concorrencia ou Paralelismo?

Programas, Procesos e Fíos.

Práctica

Práctica 1

Parte 1

2. Utiliza la forma con Runnable para implementar un programa que cree y ejecute tantos hilos como se le indique a través de línea de comando.

```
public class CDI_P1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int nThreads = Integer.parseInt(args[0]);
        MyFirstRunnable r;
        Thread t;
        for (int i = 0; i < nThreads; i++) {
            r = new MyFirstRunnable();
            t = new Thread(r);
            t.start();
        }
        System.out.println("Program of exercise 2 has terminated");
    }

}

class MyFirstRunnable implements Runnable {
```

```
    @Override
    public void run() {
        System.out.println("Hello World!");
    }

}
```

3. Utilizando tu programa del apartado anterior realiza las implementaciones necesarias para
    - que cada hilo imprima en pantalla un mensaje como “Hello world, I’m the java thread number X.”
    - y después de uno o varios segundos (usa otro argumento vía línea de comando que indique el número de segundos) un mensaje “Bye from thread number X.”.

    ¿Las salidas del programa reflejan lo que esperabas?

```
public class CDI_P1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int nThreads = Integer.parseInt(args[0]);
        int sleepTime = Integer.parseInt(args[1]) * 1000;
        MyFirstRunnable r;
        Thread t;
        for (int i = 1; i <= nThreads; i++) {
            r = new MyFirstRunnable(i, sleepTime);
            t = new Thread(r);
            t.start();
        }
        System.out.println("Program of exercise 3 has terminated");
    }

}
```

```
class MyFirstRunnable implements Runnable {

    private int number;
    private int sleepTime;

    public MyFirstRunnable(int number, int sleepTime) {
        this.number = number;
        this.sleepTime = sleepTime;
    }

    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + this.numb
        try {
            Thread.sleep(this.sleepTime);
        } catch (InterruptedException e) {
            System.out.println("An interruption occurred in the thread");
        }
        System.out.println("Bye from thread number " + this.number + ".");
    }

}
```

```
public class CDI_P1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int nThreads = Integer.parseInt(args[0]);
        int sleepTime = Integer.parseInt(args[1]) * 1000;
        List<Thread> threadList = new ArrayList<>(nThreads);
        MyFirstRunnable r;
        Thread t;
        for (int i = 1; i <= nThreads; i++) {
            r = new MyFirstRunnable(i, sleepTime);
```

```
            t = new Thread(r);
            threadList.add(t);
            t.start();
        }
        System.out.println("Program of exercise 4 has terminated");
    }

}

class MyFirstRunnable implements Runnable {

    private int number;
    private int sleepTime;

    public MyFirstRunnable(int number, int sleepTime) {
        this.number = number;
        this.sleepTime = sleepTime;
    }

    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + this.numb
        try {
            Thread.sleep(this.sleepTime);
        } catch (InterruptedException e) {
            System.out.println("An interruption occurred in the thread");
        }
        System.out.println("Bye from thread number " + this.number + ".");
    }

}
```

Parte 2

```
public class CDI_P1 {
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int nThreads = Integer.parseInt(args[0]);
        List<MyThread> threadList = new ArrayList<>(nThreads);
        MyThread t;
        for (int i = 1; i <= nThreads; i++) {
            t = new MyThread(String.valueOf(i));
            threadList.add(t);
            t.start();
        }

        for (MyThread t : threadList) {
                while (t.isAlive()) {
                        Thread.sleep(5);
                }
        }

        System.out.println("All the other threads have terminated");
        System.out.println("Program of exercise 1 has terminated");
    }

}

class MyThread extends Thread {

    public MyThread(String name) {
        super(name);
    }

    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + this.getNa
    }
}
```

```
public class CDI_P1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int nThreads = Integer.parseInt(args[0]);
        List<MyThread> threadList = new ArrayList<>(nThreads);
        MyThread t;
        for (int i = 1; i <= nThreads; i++) {
            t = new MyThread(String.valueOf(i));
            threadList.add(t);
            t.start();
        }

        for (int i = 0; i < nThreads; i++) {
            try {
                threadList.get(i).join();
            } catch (InterruptedException e) {
                System.out.println("An interruption occurred in the thread");
            }
        }

        System.out.println("All the other threads have terminated");
        System.out.println("Program of exercise 1 has terminated");
    }

}

class MyThread extends Thread {

    public MyThread(String name) {
        super(name);
    }

    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + this.getNa
```

```
    }
}
```

```
public class CDI_P1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int nThreads = Integer.parseInt(args[0]);
        List<MyThread> threadList = new ArrayList<>(nThreads);
        MyThread t;
        long time;
        for (int i = 1; i <= nThreads; i++) {
            time = System.currentTimeMillis();
            t = new MyThread(String.valueOf(i), time);
            threadList.add(t);
            System.out.println("Thread number " + i + " created at " + time);
            t.start();

        }

        for (int i = 0; i < nThreads; i++) {
            try {
                threadList.get(i).join();
            } catch (InterruptedException e) {
                System.out.println("An interruption occurred in the thread");
            }
        }

        System.out.println("All the other threads have terminated");
        System.out.println("Program of exercise 2 has terminated");
    }

}
```

```
class MyThread extends Thread {

    private long timeCreated;
    private long timeExecuted;
    private long timeTerminated;

    public MyThread(String name, long timeCreated) {
        super(name);
        this.timeCreated = timeCreated;
    }

    @Override
    public void run() {
        this.timeExecuted = System.currentTimeMillis();
        System.out.println("Thread number " + this.getName() + " executed " + S
        for (int i = 0; i < 1000000; ++i) {
            double d = Math.tan(Math.atan(
                    Math.tan(Math.atan(
                            Math.tan(Math.atan(
                                    Math.tan(Math.atan(
                                            Math.tan(Math.atan(123456789.123456789))
                                    ))
                            ))
                    ))
            ));
            Math.cbrt(d);
        }
        this.timeTerminated = System.currentTimeMillis();
        System.out.println("Thread number " + this.getName() + " terminated " +
    }
}
```

Práctica 2

Parte 1

```
public class CDI_P2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int number_of_threads = Integer.parseInt(args[0]);
        List<MyThread> myThreadList = new ArrayList<>(number_of_threads);
        List<Thread> threadList = new ArrayList<>(number_of_threads);

        for (int i = 1; i <= number_of_threads; i++) {
            MyThread myT = new MyThread(Integer.parseInt(args[1]));
            myThreadList.add(myT);
            Thread t = new Thread(myT, String.valueOf(i));
            threadList.add(t);
            t.start();
        }

        for (Thread thread : threadList) {
            try {
                thread.join();
            } catch (InterruptedException e) {
                System.out.println("Thread number " + thread.getName() + " was int
            }
        }

        System.out.println("Program of exercise 3 has terminated");
    }

}

class MyThread implements Runnable {

    private int mySum;
    private int n;

    public MyThread(int n) {
        this.n = n;
```

```
    }

    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + Thread.cu
        for (int i = 1; i <= n; i++) {
            mySum++;
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                System.out.println("Thread number " + Thread.currentThread().getN
            }

        }
        System.out.println("Bye from thread number " + Thread.currentThread().
    }

}
```

```
public class CDI_P2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int number_of_threads = Integer.parseInt(args[0]);
        List<MyThread> myThreadList = new ArrayList<>(number_of_threads);
        List<Thread> threadList = new ArrayList<>(number_of_threads);
        MyThread myT = new MyThread(Integer.parseInt(args[1]));
        myThreadList.add(myT);

        for (int i = 1; i <= number_of_threads; i++) {
            Thread t = new Thread(myT, String.valueOf(i));
            threadList.add(t);
            t.start();
        }
```

```
        for (Thread thread : threadList) {
            try {
                thread.join();
            } catch (InterruptedException e) {
                System.out.println("Thread number " + thread.getName() + " was int
            }
        }

        System.out.println("Program of exercise 4 has terminated");
    }

}

class MyThread implements Runnable {

    private int mySum;
    private int n;

    public MyThread(int n) {
        this.n = n;
    }

    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + Thread.cu
        for (int i = 1; i <= n; i++) {
            this.mySum++;
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                System.out.println("Thread number " + Thread.currentThread().getN
            }

        }
        System.out.println("Bye from thread number " + Thread.currentThread().
    }
```

```
}
```

```
public class CDI_P2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int number_of_threads = Integer.parseInt(args[0]);
        List<MyThread> myThreadList = new ArrayList<>(number_of_threads);
        List<Thread> threadList = new ArrayList<>(number_of_threads);
        MyThread myT = new MyThread(Integer.parseInt(args[1]));
        myThreadList.add(myT);

        for (int i = 1; i <= number_of_threads; i++) {
            Thread t = new Thread(myT, String.valueOf(i));
            threadList.add(t);
            t.start();
        }

        for (Thread thread : threadList) {
            try {
                thread.join();
            } catch (InterruptedException e) {
                System.out.println("Thread number " + thread.getName() + " was int
            }
        }

        System.out.println("Program of exercise 5 has terminated");
    }

}

class MyThread implements Runnable {
```

```
    private final ThreadLocal<Integer> mySum;
    private int n;

    public MyThread(int n) {
        this.n = n;
        this.mySum = ThreadLocal.withInitial(() -> 0);
    }

    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + Thread.cu
        for (int i = 1; i <= n; i++) {
            this.mySum.set(mySum.get()+1);
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                System.out.println("Thread number " + Thread.currentThread().getN
            }

        }
        System.out.println("Bye from thread number " + Thread.currentThread().
    }

}
```

Parte 2

```
public class CDI_P2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int number_of_threads = Integer.parseInt(args[0]);
        List<MyThread> myThreadList = new ArrayList<>(number_of_threads);
        List<Thread> threadList = new ArrayList<>(number_of_threads);
        MyThread myT = new MyThread(Integer.parseInt(args[1]));
        myThreadList.add(myT);
```

```
        int index = Integer.parseInt(args[2]);

        for (int i = 1; i <= number_of_threads; i++) {
            Thread t = new Thread(myT, String.valueOf(i));
            threadList.add(t);
            t.start();
        }

        threadList.get(index).interrupt();

        for (Thread thread : threadList) {
            try {
                thread.join();
            } catch (InterruptedException e) {
                System.out.println("Thread number " + thread.getName() + " was int
            }
        }

        System.out.println("Program of exercise 5 has terminated");
    }

}

class MyThread implements Runnable {

    private final ThreadLocal<Integer> mySum;
    private int n;

    public MyThread(int n) {
        this.n = n;
        this.mySum = ThreadLocal.withInitial(() -> 0);
    }

    /*
    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + Thread.cu
        try {
```

```
            for (int i = 1; i <= n; i++) {
                this.mySum.set(mySum.get()+1);
                Thread.sleep(5);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread number " + Thread.currentThread().getNan
        }
        System.out.println("Bye from thread number " + Thread.currentThread().
    }
    */

    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + Thread.cu
        int i = 1;
        while (!Thread.currentThread().isInterrupted() && i <= n) {
            this.mySum.set(mySum.get() + 1);
        }

        System.out.println("Thread number " + Thread.currentThread().getName
        System.out.println("Bye from thread number " + Thread.currentThread().
    }

}
```

Práctica 3

```
package cdi_p3;

/*
* (c) copyright 2017-2023 Universidade de Vigo. All rights reserved.
* formella@uvigo.es, http://formella.webs.uvigo.es
 */
/**
 * \file \brief working with an image
 */
// These are the packages we need for the example.
```

```
import java.awt.image.*;
import java.io.File;
import java.util.ArrayList;
import java.util.List;
import javax.imageio.ImageIO;

class CDI_P3 {

   static BufferedImage img;

   public static void main(String[] args) {
      System.out.println("starting...");
      try {
         int number_of_threads = Integer.parseInt(args[0]);
         File file_in = new File("C:\\Users\\Estudante\\Desktop\\CDI_P3\\small.jp
         img = ImageIO.read(file_in);

         System.out.println("type: " + img.getType());
// Prepare output image with the same size and type as the input image.
         final int width = img.getWidth();
         final int height = img.getHeight();
         BufferedImage out_img = new BufferedImage(width, height, img.getTy

         List<Thread> threadList = new ArrayList<>(number_of_threads);
         int aux = 0;
         int salto = img.getHeight() / number_of_threads;
         for (int i = 0; i < number_of_threads; i++) {
            Thread t;
            if (i == number_of_threads-1) {
               t = new Worker(img, out_img, aux, img.getHeight(), i);
            } else {
               t = new Worker(img, out_img, aux, aux + salto, i);
               aux += salto;
            }
            threadList.add(t);
            t.start();
         }
```

```
            for (Thread t: threadList) {
                t.join();
            }


// Write the image always in png-format with a fixed name.
            File file_out = new File("my_image.png");
            ImageIO.write(out_img, "png", file_out);
        } catch (Exception e) {
// Here, we just exit the program, something more useful should
// be implemented...
            e.printStackTrace();
            System.out.println(e.toString());
            System.exit(1);
        }
// As always: out last line.
        System.out.println("exiting...");
    }
}
```

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.t
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 */
package cdi_p3;

import java.awt.Color;
import java.awt.image.BufferedImage;

/**
 *
 * @author Estudante
 */
public class Worker extends Thread {

    private int number;
    private int sleepTime;
```

```
    BufferedImage originalImage;
    BufferedImage outImage;
    int startRow;
    int endRow;

    public Worker(BufferedImage originalImage, BufferedImage outImage, int st
        this.number = number + 1;
        this.originalImage = originalImage;
        this.outImage = outImage;
        this.startRow = startRow;
        this.endRow = endRow;
    }

    @Override
    public void run() {
        System.out.println("Hello world, I'm the java thread number " + this.numb

        for (int x = 0; x < originalImage.getWidth(); x++) {
            for (int y = startRow; y < endRow; y++) {
                Color originalColor = new Color(originalImage.getRGB(x, y));
// Calcular la luminancia percibida para el efecto de blanco y negro
                int luminance = (int) (originalColor.getRed() * 0.299 + originalColor.g
                        * 0.587 + originalColor.getBlue() * 0.114);
                Color bwColor = new Color(luminance, luminance, luminance);
                outImage.setRGB(x, y, bwColor.getRGB());
            }
        }
        System.out.println("Bye from thread number " + this.number + ".");
    }
}
```

Práctica 4

Parte 1.1.

```
public class CDI_P4 {

    /**
```

```
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int number_of_threads = Integer.parseInt(args[0]);

        Counter c = new Counter();
        for (int i = 0; i < number_of_threads; i++) {
            MyTask t = new MyTask(String.valueOf(i), c);
            t.start();
        }
    }

}
```

```
public class Counter {
    private int counter;

    public Counter() {
        this.counter = 0;
    }

    public int increment() {
        return ++this.counter;
    }

    public int getCounter() {
        return counter;
    }
}
```

```
import java.util.Arrays;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
```

```
 * @author Estudante
 */
public class MyTask extends Thread {
    Counter counter;
    int sleepTime;

    public MyTask(String name, Counter counter) {
        Random r = new Random();
        this.sleepTime = r.nextInt(0, 100);
        this.counter = counter;
        this.setName(name);
    }

    @Override
    public void run() {
        System.out.println("Hello world, Im the java thread number " + this.getNa
        try {
            Thread.sleep(this.sleepTime);
        } catch (InterruptedException ex) {
            System.out.println(Arrays.toString(ex.getStackTrace()));
        }
        this.counter.increment();
        System.out.println("Bye from thread number " + this.getName() + ". Coun
    }
}
```

Parte 1.2.

La primera forma es mucho más mantenible, en cualquier parte del programa en la que se llame a Counter.increment() estará sincronizado. Si lo hacemos de la segunda forma deberemos asegurarnos de que en todas las zonas del código en las que se incremente el valor del contenedor esten sincronizadas.

```
public class Counter {
    private int counter;

    public Counter() {
        this.counter = 0;
```

```
    }

    public synchronized int increment() {
        return ++this.counter;
    }

    public int getCounter() {
        return counter;
    }
}
```

```
public class MyTask extends Thread {
    Counter counter;
    int sleepTime;

    public MyTask(String name, Counter counter) {
        Random r = new Random();
        this.sleepTime = r.nextInt(0, 100);
        this.counter = counter;
        this.setName(name);
    }

    @Override
    public void run() {
        System.out.println("Hello world, Im the java thread number " + this.getNa
        try {
            Thread.sleep(this.sleepTime);
        } catch (InterruptedException ex) {
            System.out.println(Arrays.toString(ex.getStackTrace()));
        }
        synchronized(this.counter) {
            this.counter.increment();
        }

        System.out.println("Bye from thread number " + this.getName() + ". Coun
    }
}
```

Parte 1.3.

AtomicInteger:

```
import java.util.concurrent.atomic.AtomicInteger;

/**
 *
 * @author Estudante
 */
public class CDI_P4 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int number_of_threads = Integer.parseInt(args[0]);
        AtomicInteger c = new AtomicInteger();
        for (int i = 0; i < number_of_threads; i++) {
            MyTask t = new MyTask(String.valueOf(i), c);
            t.start();
        }
    }

}
```

```
import java.util.Arrays;
import java.util.Random;
import java.util.concurrent.atomic.AtomicInteger;

/**
 *
 * @author Estudante
 */
public class MyTask extends Thread {
    AtomicInteger counter;
    int sleepTime;
```

```
    public MyTask(String name, AtomicInteger counter) {
        Random r = new Random();
        this.sleepTime = r.nextInt(0, 100);
        this.counter = counter;
        this.setName(name);
    }

    @Override
    public void run() {
        System.out.println("Hello world, Im the java thread number " + this.getNa
        try {
            Thread.sleep(this.sleepTime);
        } catch (InterruptedException ex) {
            System.out.println(Arrays.toString(ex.getStackTrace()));
        }

        System.out.println("Bye from thread number " + this.getName() + ". Coun
    }
}
```

LongAdder:

```
import java.util.concurrent.atomic.LongAdder;

/**
 *
 * @author Estudante
 */
public class CDI_P4 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int number_of_threads = Integer.parseInt(args[0]);
        LongAdder c = new LongAdder();
        for (int i = 0; i < number_of_threads; i++) {
            MyTask t = new MyTask(String.valueOf(i), c);
```

```
            t.start();
        }
    }

}
```

```
import java.util.Random;
import java.util.concurrent.atomic.LongAdder;

/**
 *
 * @author Estudante
 */
public class MyTask extends Thread {
    LongAdder counter;
    int sleepTime;

    public MyTask(String name, LongAdder counter) {
        Random r = new Random();
        this.sleepTime = r.nextInt(0, 100);
        this.counter = counter;
        this.setName(name);
    }

    @Override
    public void run() {
        System.out.println("Hello world, Im the java thread number " + this.getNa
        try {
            Thread.sleep(this.sleepTime);
        } catch (InterruptedException ex) {
            System.out.println(Arrays.toString(ex.getStackTrace()));
        }
        this.counter.increment();
        System.out.println("Bye from thread number " + this.getName() + ". Coun
    }
}
```

Parte 1.4.

```
import java.util.Arrays;
import java.util.Random;
import java.util.concurrent.locks.Lock;

/**
 *
 * @author Estudante
 */
public class MyTask extends Thread {
    Counter counter;
    int sleepTime;
    final Lock lock;

    public MyTask(String name, Counter counter, Lock lock) {
        Random r = new Random();
        this.sleepTime = r.nextInt(0, 100);
        this.counter = counter;
        this.lock = lock;
        this.setName(name);
    }

    @Override
    public void run() {
        System.out.println("Hello world, Im the java thread number " + this.getNa
        try {
            Thread.sleep(this.sleepTime);
        } catch (InterruptedException ex) {
            System.out.println(Arrays.toString(ex.getStackTrace()));
        }
        lock.lock();
        try {
            System.out.println("Bye from thread number " + this.getName() + ". Co
        } finally {
            lock.unlock();
        }
```

```
    }
}
```