Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Examen Final Teórico de la Asignatura
Febrero de 2014

Apellidos: Nombre:

D.N.I.: Grupo (A ó B):

1. Notas

1. Escriba su nombre, apellidos, D.N.I. y grupo en el espacio habilitado para ello, y en todos los folios blancos que utilice. Firme el documento en la esquina superior derecha de la primera página.

2. Dispone de diez minutos para leer los enunciados y formular preguntas o aclaraciones sobre ellos. Transcurrido ese tiempo, no se contestarán preguntas. Dispone de 2:00 horas para completar el ejercicio.

3. No complete el documento a lápiz. Utilice bolígrafo o rotulador. Escriba con letra clara y legible. No podemos corregir lo que no se no puede leer.

4. Utilice los folios blancos que se le proporcionan para resolver los enunciados, pero traslade a este documento únicamente la solución final que obtenga, utilizando el espacio específicamente habilitado para ello, sin sobrepasarlo en ningún caso, y sin proporcionar información o respuestas no pedidas. Entregue tanto el enunciado como los folios blancos. Únicamente se corregirá este documento.

2. Criterios de Corrección

1. El examen se calificará de cero a diez puntos, y ponderará en la calificación final al 40 % bajo los supuestos recogidos en la ficha de la asignatura.

2. Cada enunciado incluye información de la puntuación que su resolución correcta representa, incluida entre corchetes.

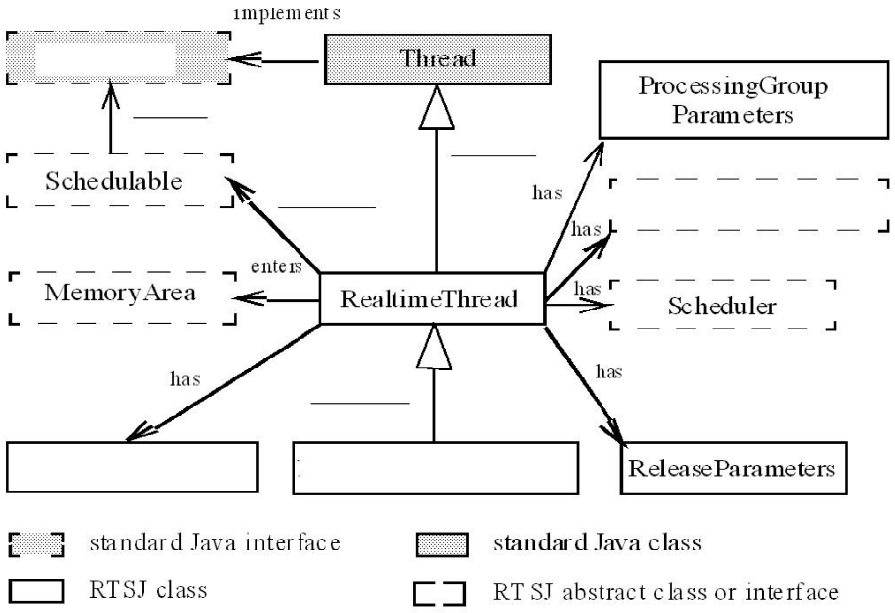
3. Un enunciado (cuestión teórica o problema) se considera correcto si la solución dada es correcta completamente. En cualquier otro caso se considera incorrecto y no puntúa.

4. Un enunciado de múltiples apartados (cuestión teórica o problema) es correcto si y solo si todos los apartados que lo forman se contestan correctamente. En cualquier otro caso se considera incorrecto y no puntúa.

3. Cuestiones de Desarrollo Corto

Conteste a las preguntas que se le formulan en el espacio habilitado para ello. Deberá razonar o justificar su respuesta simpre que se le indique. La ausencia del razonamiento o de la justificación invalidarán la respuesta al no ser esta completa.

1. El siguiente diagrama modela el API de Java-RT para creación y manipulación de objetos `Schedulable`. Complételo con la información que crea oportuna, y justifique la información añadida en el espacio habilitado para ello: [0.5 puntos]: Justifique la información añadida:



2. ¿Qué categorías establece la taxonomía de *Flynn*? ¿Qué relaciones guardan estas categorías con las siguiente arquitecturas? Procesadores MultiCore, Sistemas Distribuidos, Cluster de Procesadores, Procesador GPU. Escriba aquí su respuesta y las relaciones que propone, y justifíquelas: [0.5 puntos]

3. ¿Es posible implantar regiones críticas como primitiva de control de la concurrencia en C++11? [0.5 puntos] Escriba y justique su respuesta; proponga también un ejemplo en caso positivo:[1 punto]

4. ¿Qué sucede cuando una CPU lanza un *kernel* sobre una GPU nVidia que actúa como coprocesador? Escriba aquí su respuesta razonada: [0.5 puntos]

5. Considere el siguiente programa. Indique la salida -si la hay- que produce y el comportamiento que tiene. Justifique su respuesta. [1 punto]

```
public class E131402 extends Thread{
 public static Object[] locks;
 public static Integer n= new Integer(0);
 private int id;
 public E131402(int id){
   this.id=id;
 }
 public void run(){
   synchronized(locks[id]){
   n++;
   if(id%2==0)
     try{locks[id].wait();
     }catch(InterruptedException e){}
   System.out.println(n);
   }
 }
 public static void main(String[] args){
   locks = new Object[4];
   for(int i=0; i<4; i++)
     locks[i] = new Object();
   for(int i=0; i<4; i++)
     new E131402(i).start();
 }
}
```

Indique aquí la salida y el comportamiento, justificando ambos:

6. Considere el siguiente programa. Indique la salida -si la hay- que produce y el comportamiento que tiene. Justifique su respuesta. [1 punto]

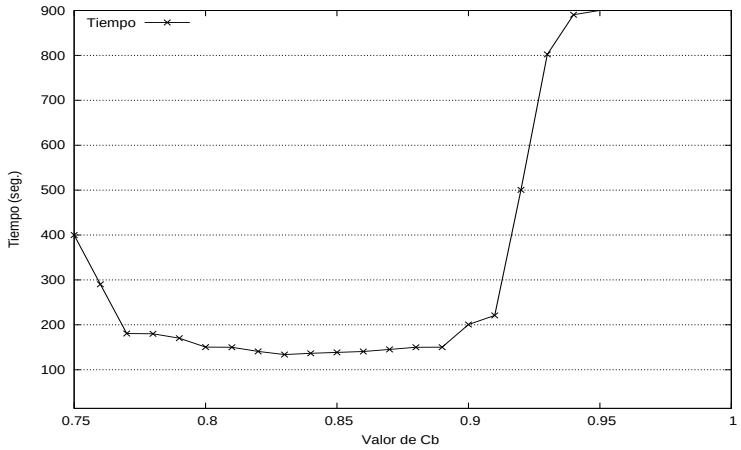
```
 import java.util.concurrent.locks.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.concurrent.atomic.*;

public class E131401 implements Runnable{
  private static AtomicInteger         s = new AtomicInteger();
  private static ReentrantLock         l = new ReentrantLock();
  private static int                   p = 2000;
  private static ArrayList<Condition>  c = new ArrayList<Condition>();
  private int                          n;
  public E131401(int n){
    this.n=n;
    c.add(n, l.newCondition());
  }
  public void run(){
    int j;
    for(int i=0; i<1000; i++);
    l.lock();
    try{ j=s.incrementAndGet();
         c.get(0).signalAll();
         try{c.get(n).await();
         }catch(InterruptedException e){}
       }finally{l.unlock();}
  }
  public static void main(String[] args){
    ExecutorService ept = Executors.newCachedThreadPool();
    for(int i=0; i<2000; i++)
      ept.submit(new E131401(i));
    ept.shutdown();
    System.out.print(s);
  }
}
```

Indique aquí la salida y el comportamiento, justificando ambos:

7. La siguiente curva ilustra el comportamiento temporal de una aplicación

paralela con latencia de red para diferentes valores del coeficiente de bloqueo: [1 punto]



Realice una intepretación de la misma de forma justificada:

8. Considere el siguiente programa escrito en C++1. Indique la salida -si la hay- que produce y el comportamiento que tiene. Justifique su respuesta. [1 punto]

```
#include <thread>
#include <mutex>
#include <iostream>
struct Comun {
    std::mutex clang;
    int q;
    Comun() : q(0) {}
    void oper_A(int x){
        std::lock_guard<std::mutex> cerrojo(clang);
        q -= x;
    }
    void oper_B(int x){
        std::lock_guard<std::mutex> cerrojo(clang);
        q++;
    }
    void oper_C(int x, int y){
```

```
        std::lock_guard<std::mutex> cerrojo(clang);
        oper_B(33);
        oper_A(56);
    }
};
int main(){
    Comun ref;
    ref.oper_B(5);
    ref.oper_C(6, 6);
    std::cout<<"Spock dice: Larga vida y prosperidad...";
    return 0;
}
```

Indique aquí la salida y el comportamiento, justificando ambos:

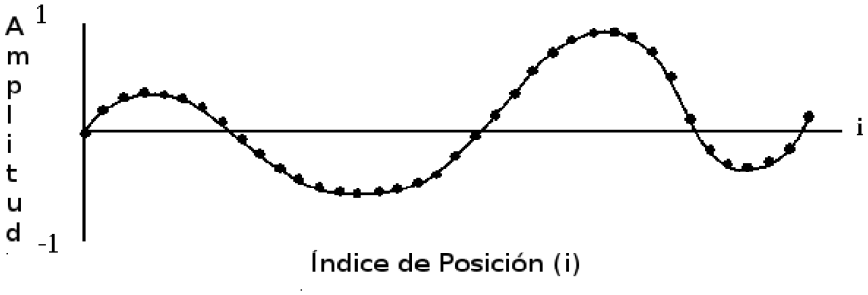
4. Problemas

1. Se desea disponer de un protocolo de traducción de monitores teóricos tipo *Hoare* a monitores redactados en C++11. Se pide:[1.5 puntos]

- Escribir el protocolo:

- Aplicarlo sobre el conocido monitor *Hoare* del problema del productor-consumidor:

2. Se desea calcular de forma paralela la amplitud de vibración de una cuerda en un número de nodos discreto y finito de la misma como una función del tiempo. Cada nodo tiene en tiempo t una amplitud comprendida en el intervalo $[-1, 1]$ (ver figura).[2 puntos]
Para efectuar el cálculo se utiliza la conocida ecuación de onda unidimensional,



que en su version discreta adopta la siguiente expresión:

$$A(i, t+1) = (2 * A(i, t) - A(i, t-1) + (c * (A(i-1, t) - (2 * A(i, t) + A(i+1, t)))$$

donde c es una constante real positiva. Escriba un programa paralelo `ecOnda.java` para resolver el problema en función del númeo de núcleos disponibles y del coeficiente de bloqueo que considere adecuado fijar, considerando 500 instantes de tiempo discreto y un número de nodos en la cuerda igual a 250. Suponga tareas `Runnable` y utilice un ejecutor `FixeThreadPool` para procesarlas.

(escriba aquí su programa)