Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Examen Final Teórico de la Asignatura
Febrero de 2013

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.

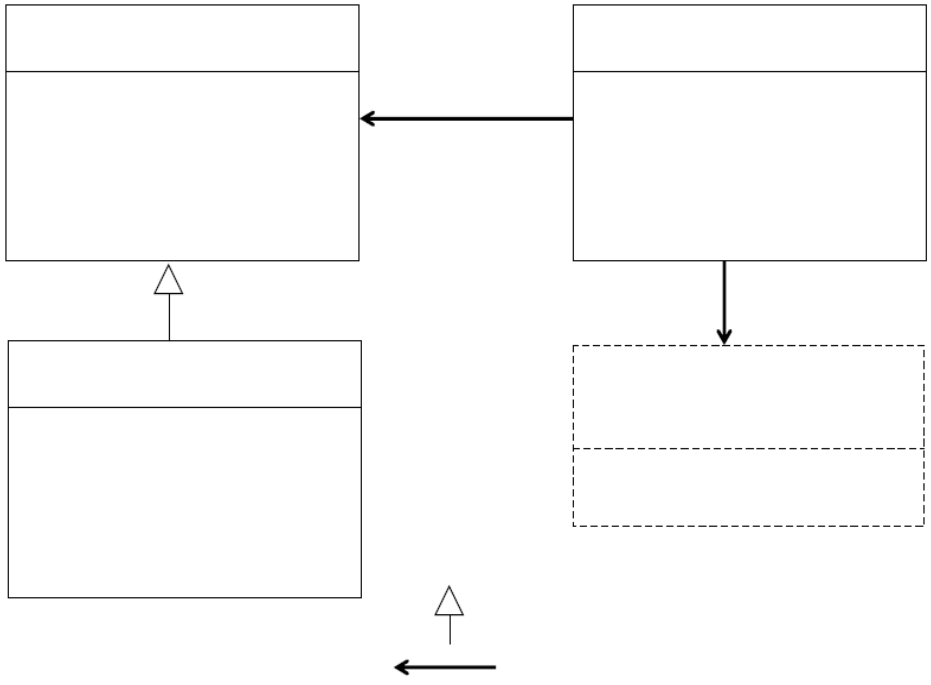
3. Un enunciado (cuestión teórica o problema) se considera correcto si la solución dada es correcta completamente. En cualquier otro caso se considera incorrecto y no puntúa.

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. **Se muestran algunos ejemplos de cuestiones cortas**.

1. El siguiente diagrama modela el API básico de java para creación y manipulación de *Threads*. Complételo con la información que crea oportuna, y justifique la información añadida en el espacio habilitado para ello: [0.5 puntos]

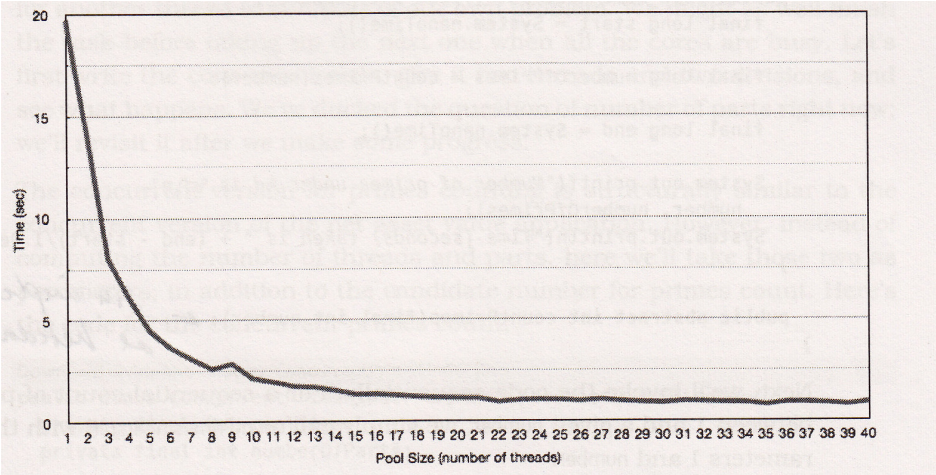


Justifique la información añadida:

2. ¿Cuál es la diferencia entre una región crítica y un monitor? [0.5 puntos]
Escriba aquí su respuesta:

3. ¿Qué diferencia a los *threads* estándar de los *threads-RT*? [0.5 puntos]
Escriba aquí su respuesta:

4. Considere el conocido problema de encontrar números primos en un rango dado, y una solución *muti-core* al mismo que utiliza un *pool* de *Threads*. Conforme aumentamos el tamaño del *pool*, el tiempo de cálculo cambia de acuerdo a siguiente curva: [0.5 puntos]



Explique, justificando sus ideas, qué está sucediendo.

5. La siguiente, es una solución al problema de la exclusión mutua que emplea variables compartidas. Discuta la corrección parcial de esta solución. Si es correcta, indique por qué y si no lo es, describa escenarios de ejecución que indiquen los entrelazados que llevan a condiciones patológicas: [1 punto]

```
Procedure Pi
 Begin
  Repeat
    Turno:=i;
    While Turno<>i Do;
      Sección_Crítica_i;
    Turno:=j;
  Forever
 End;
```

Escriba aquí su respuesta:

6. Considere el siguiente programa concurrente. Estudie su corrección total, y diga si es o no totalmente correcto. Justifique su respuesta. [1 punto]

```
import java.util.concurrent.locks.*;
public class Correc extends Thread
{
 public static int nH = 10;
 public static int n = 0;
 public static ReentrantLock l = new ReentrantLock();
 public Condition v;
 public Correc() {v = l.newCondition();}
 public void run()
```

```
 {
     l.lock();try{
         try{v.await();} catch(InterruptedException e){}
         n++;
     }finally {l.unlock();}
 }

 public static void main(String[] args)
 throws InterruptedException
 {
 Correc [] h = new Correc[nH];
 for(int i=0; i<h.length;i++)
 {h[i]=new Correc(); h[i].start();}
 for(int i=0; i<h.length;i++)
 {h[i]=new Correc(); h[i].join();}
 n++;
 System.out.print(n);
 }
}
```

Escriba y justifique aquí su respuesta:

7. Describa el resultado de ejecutar el siguiente programa. Razone su respuesta. [1 punto]

```
import java.util.concurrent.*;
 public class E17 extends Thread {
 static int x = 0;
 Semaphore cerrojo = new Semaphore(1);
 int t;

 public E17(int t){this.t=t;}
 public void run(){
 switch(t){
     case 0:{try{cerrojo.acquire();}
          catch(InterruptedException e){} x++;cerrojo.release();}
     case 1:{try{cerrojo.acquire();}
          catch(InterruptedException e){} x--;cerrojo.release();}
 }
 }
```

```
 public static void main(String[] args) throws Exception{
 E17[] lista = new E17[10000];
 for(int i=0;i<lista.length;i++){
   lista[i] = new E17(1);
   lista[i].start();
 }
 for(int i=0;i<lista.length;i++){
   lista[i].join();
 }
 System.out.print(x);
 }
}
```

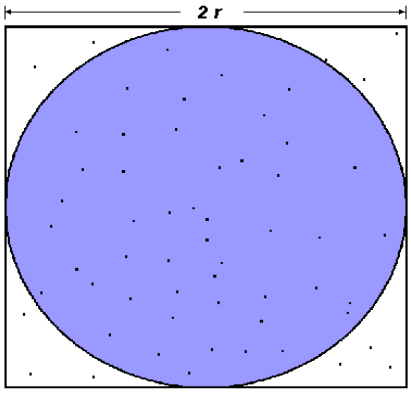
Escriba y razone aquí su respuesta:

4 Problemas

Se muestra un ejemplo de problema.

...

2. Un método de tipo *Monte Carlo* para calcular de forma aproximada el valor de π es inscribir un círculo en un cuadrado según se muestra en la figura y generar puntos aleatorios. [2 puntos]



Si p es el número de puntos inscritos en el círculo dividido por el número de puntos inscritos en el cuadrado, se sabe que $\pi \simeq 4 \times p$ y que a mayor número de puntos generados se obtiene una aproximación a π de mayor precisión. Se desea escribir un programa multihebrado en Java que permita estimar π con la precisión deseada. Para ello, debe determinar y explicar:

- ¿Qué coeficiente C_b de bloqueo cree que tiene este problema? ¿Por qué?

- Si se sabe que $N_{nd} = 4$, y en función de lo anterior, ¿cuántos hilos N_t debería tener su programa?

- ¿Cómo va a particionar el problema para lograr paralelismo *multi-core*?

- Escriba ahora una clase `piMonteCarloParalelo` que herede de `Thread` y que dé soporte al paralelismo de datos indicado (vuelva la página).

- Escriba finalmente el trozo de programa principal que crea los hilos, los lanza, y muestra el resultado de la aproximación a π obtenida.