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

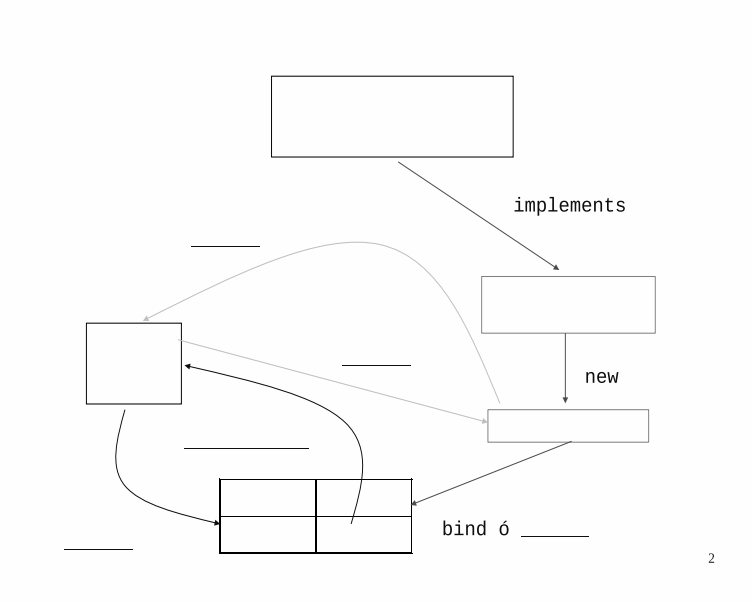
Conteste a las preguntas que se le formulan en el espacio habilitado para ello. Deberá razonar o justificar su respuesta simpre que se le indique. La ausencia del razonamiento o de la justificación invalidará la respuesta al no ser esta completa.

1. [0.5 puntos] ¿Cómo funciona el multihebrado en el lado del servidor de una arquitectura *RMI*? Justifique su respuesta.

2. [0.5 puntos] ¿Cuántas métricas de prioridad emplean los objetos de clase `NoHeapRealTimeThread`? ¿Por qué?.

3. [0.5 puntos] ¿Cuál es el principal inconveniente del modelo de regiones críticas? Justifique su respuesta.

4. [0.5 puntos] Complete el siguiente diagrama relativo a la arquitectura *RMI*. Debe rellenar los recuadros y las líneas horizontales con la información que crea adecuada.



5. [0.5 puntos] La clase `E35` que sigue no es un monitor. Reescríbala utilizando el API estándar para que lo sea, de forma que se verifiquen las condiciones de sincronización siguientes:

   - `valor` nunca puede exceder de `VMAX`
   - `valor` nunca puede disminuir de `VMIN`

```
public class E35 {
  private static final int VMAX = 10;
  private static final int VMIN =  0;
  private static       int valor =  0;

  public int incremento(){
    return (valor++);
  }

  public int decremento(){
    return (valor--);
  }
}
```

Escriba aquí la nueva versión de la clase `E35`:

6. [1.0 puntos] ¿Cual es la salida impresa del siguiente código Java?

```
import java.util.concurrent.locks.*;
import java.util.concurrent.*;
public class E33 implements Runnable{
  public static Integer val = new Integer(0);
  public int s;
  public static ReentrantLock lock = new ReentrantLock();
  public E33(int s) {this.s = s;}

  public void run(){
    Object val2=val;
    for(int i=0;i<s; i++){
      lock.lock();
      val++;
      lock.unlock();
      synchronized(val2){
        try{val2.wait();
        }catch(InterruptedException e){}
      }
    }
  }
  public static void main(String[] args)throws InterruptedException{
  E33 [] h = new E33[10];
  for(int i=0;i<h.length;i++)h[i]=new E33(100);
  int nP = Runtime.getRuntime().availableProcessors();
  ThreadPoolExecutor ept = new ThreadPoolExecutor(
                           nP*4, nP*4,
                           1000L,
```

```
                          TimeUnit.MILLISECONDS,
                          new LinkedBlockingQueue<Runnable>());
  ept.prestartAllCoreThreads();
  for(int i=0; i<h.length; i++) ept.execute(h[i]);
  ept.shutdown();
  do{} while(!ept.isTerminated());
  val++;
  System.out.print(val);
 }
}
```

Justifique su respuesta:

7. [1.0 puntos] ¿Cual es la salida impresa y el comportamiento del siguiente código Java?

```
public class E34 extends Thread{
  private String s;
  private static Object o = new Object();
  public E34(String s){this.s = s;}
  public void run(){
    synchronized(o){
      switch(this.s){
        case "ping":for(int i=0; i<10; i++){
               System.out.print(this.s+" ");
               o.notify();
               try{o.wait();}catch(InterruptedException e){}
                    }
        case "pong":for(int i=0; i<5; i++){
               try{o.wait();}catch(InterruptedException e){}
               System.out.print(this.s+" ");
               o.notify();
                    }
      }
    }
  }

  public static void main(String[] args)throws InterruptedException{
```

```
    E34 Lendl   = new E34("pong");
    E34 McEnroe = new E34("ping");
    Lendl.start();
    McEnroe.start();
    Lendl.join();
    McEnroe.join();
  }
}
```

Justifique su respuesta:

8. [1.0 puntos] Considere el siguiente código e indique cuál es la salida impresa que produce.

```
public class E31 extends Thread{
 private static Integer i = new Integer(1);
 private static Integer j = new Integer(1);
 private static int k=0;
 private E31 mi_thread;
 public E31(){k++;}
 public void run(){
   if(k<800)
     {mi_thread=new E31();
      synchronized(j){i--;}
      mi_thread.start();
      try{mi_thread.join();}catch (InterruptedException e){}
     }
 }

 public static void main(String[] args) throws Exception{
   E31 otro_thread = new E31();
```

```
    otro_thread.start();
    otro_thread.join();
    System.out.println(i.toString());
  }
}
```

Justifique su respuesta:

4. Problemas

1. [2 puntos] Problemas de presupuesto han obligado al Ministerio de Fomento a reducir a uno el número de carriles en dos tramos del nuevo viaducto sobre la Bahía de Cádiz, manteniendo los tramos restantes los cuatro carriles previstos en el proyecto original. Sobre los tramos monocarril solo puede haber un coche. Se pide programar utilizando **semáforos Djikstra** a los coches que desean cruzar el viaducto desde la orilla de Cádiz y la de Puerto Real, de forma que el viaducto no quede bloqueado.

(continúe escribiendo aquí su solución al problema número 1)

2. [1.5 puntos] Una familia de n crías de pájaro hambrientas comen de un plato común que al principio contiene F porciones de comida. Cada cría de pájaro repite un ciclo en el que come, se duerme un tiempo, y se despierta para repetir el ciclo. La madre pájaro duerme hasta que el plato de comida queda vacío. En ese momento se despierta, lo llena de comida y vuelve a dormirse. Puesto que las crías de pájaro pasan parte del tiempo durmiendo (usted decide qué parte), deberá estimar un coeficiente de bloqueo adecuado al problema. Suponga además que $N_{nd} = 8$. Escriba en java clases `criaPajaro.java`, `madrePajaro.java` y `plato.java` que modelen el problema. Debe utilizar de forma obligatoria tareas `Runnable` y el API de alto nivel para control -en su caso- de la sincronización. Las tareas serán procesadas a través de un objeto de clase `ThreadPoolExecutor`.

(continúe escribiendo aquí su solución al problema número 2)