Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Examen Final de Prácticas
Junio de 2013

Apellidos: Nombre:

Grupo (A ó B):

1. Notas

1. Escriba su nombre y apellidos con letra clara y legible en el espacio habilitado para ello.
2. Firme el documento en la esquina superior derecha de la primera hoja y escriba debajo su D.N.I.
3. Dispone de $2:30'$ horas para completar el ejercicio.
4. Puede utilizar el material bibliográfico (libros) y copia de API que estime convenientes. Se prohíbe el uso de apuntes, *pen-drives* y similares.
5. Entregue sus productos, utilizando la tarea de entrega disponible en el Campus Virtual, en un fichero (`.rar, .zip`) de nombre

   `Apellido1_Apellido_2_Nombre`

   que contendrá una subcarperta por cada enunciado del examen, la cual contendrá a su vez el conjunto de ficheros que den solución a ese enunciado en particular.

2. Criterios de Corrección

1. El examen práctico se calificará de cero a diez puntos y ponderará en la calificación final al 50 % bajos los supuestos recogidos en la ficha de la asignatura.
2. Las condiciones que una solución a un enunciado de examen práctico debe cumplir para considerarse correcta son:

a) Los ficheros subidos a través del Campus Virtual que conforman el examen práctico se ajustan al número, formato y nomenclatura de nombres explicitados por el profesor en el documento de examen.

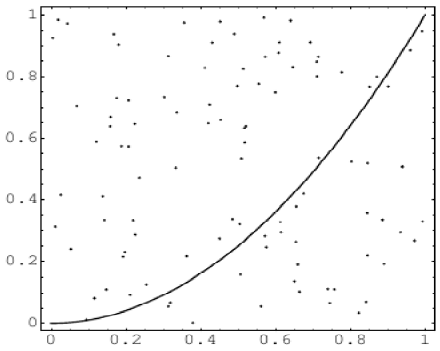
b) El contenido de los ficheros es el especificado por el profesor en el documento de examen en orden a solucionar el enunciado en cuestión.

c) Los programas elaborados por el alumno, se pueden abrir y procesar con el compilador del lenguaje, y realizan un procesamiento técnicamente correcto, según el enunciado de que se trate.

d) Se entiende por un procesamiento técnicamente correcto a aquél código de programa que compila correctamente sin errores, cuya semántica dá soporte a la solución pedida, y que ha sido escrito de acuerdo a las convenciones de estilo y eficiencia habituales en programación

3. Enunciados

1. (Integral Definida, 4 puntos) Se desea conocer de forma aproximada mediante un algoritmo numérico el valor de $\int_0^1 x^2\,dx$. Para ello, utilizaremos un método de Monte Carlo, que aproxima el valor de la integral pedida en el intervalo indicado mediante un cociente de superficies, tal y como se muestra en la figura, donde se ha inscrito la función x^2 en el cuadrado de superficie igual a la unidad:



Si se inscriben puntos aleatorios en el cuadrado, es claro que el número de puntos N es una aproximación a la superficie A del mismo. De la misma forma, el número de puntos N' que se encuadran bajo la curva es una aproximación a la superficie S que hay bajo la misma y, por tanto, al valor de la integral buscada. Así pues, puede concluirse que $\int_0^1 x^2 dx \simeq A \times \frac{N'}{N}$ siendo A la superficie del cuadrado, y que la aproximación mejora según se incrementa el número de puntos de prueba. Se pide escribir un programa multihebrado `intMontecarlo.java` que realice la aproximación, y que utilice tantos *threads* como núcleos disponibles haya. El número de tareas deberá ser determinado por el programa automáticamente en función del número de núcleos disponibles y el coficiente de bloque que usted asuma para el problema. Se utilizará la interfaz `Runnable` y los *threads* se tomarán de un `ThreadPoolExecutor`. Cuando el cálculo paralelo haya finalizado, el programa principal deberá imprimir en pantalla el valor obtenido para la integral definida.

2. (Sistema de Reservas Hoteleras, 2.5 puntos) Un hotel desea ofrecer un servicio de reservas *on-line*. El informático responsable del proyecto decide desarrollar la plataforma de reservas utilizando el *framework* RMI de java. Para ello, diseña la siguiente interfaz:

```
import java.rmi.*;
public interface iReservas extends Remote
public Integer[] disponibles() throws RemoteException;
public int hacerReserva(DatosReserva datos) throws RemoteException;
public void anularReserva(int codigo) throws RemoteException;
```

En ella, el método `disponibles()` permite al usuario cliente conocer qué habitaciones están libres, el método `hacerReserva(DatosReserva datos)` permite enviar los datos para reservar la habitación elegida (a cambio se obtiene un código de reserva) y el método `anularReserva(int codigo)` permite, utilizando ese código, eliminar una reserva ya creada. Se pide:

- Escribir una clase `DatosReserva.java` que incorpore los datos de la reserva: nombre, apellidos, DNI, teléfono y habitación.
- Escribir un servidor `sReservas.java` que corra en el puerto 1033 y que permita realizar reservas de forma concurrente desde varios clientes. Escriba el código, para simplificar, limitando las reservas a un solo día.
- Escribir un cliente `cReservas.java` que permita reservar y que ofreza un menú de usuario con las opciones ofrecidas por la interfaz.

3. (Aeropuerto, 3.5 puntos) Un aeropuerto dispone de tres pistas para despegue y de una para aterrizaje. Cuando un avión desea despegar, debe pedir pista a la torre de control, que le indica cuál de ellas le corresponde. Una vez que ha despegado, el avión indica a la torre que la pista está libre. Si no hay pistas libres, el avión debe esperar para poder despegar. El aterrizaje funciona de la misma forma pero, en este caso, con una sola pista. Escriba un monitor que simule a la torre de control en `Torre.java`, utilizando cerrojos de clase `ReentrantLock` y variables `Condition`. Modele los aviones mediante hilos en una clase `Avion.java` mediante herencia de `Thread` y ponga todo a funcionar en un programa ejecutable llamado `Aeropuerto.java`.