Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Examen Final de Prácticas
17 de Junio de 2022

1. Notas de Procedimiento

1. Dispone de 120 minutos para completar el ejercicio.
2. Puede utilizar el material bibliográfico (libros) y copia de API que estime convenientes.
3. 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 subcarpeta por cada enunciado del examen, la cual contendrá a su vez el conjunto de ficheros que den solución a ese enunciado en particular. Cada fichero de código debe incluir un comentario con su nombre, apellidos y D.N.I.
4. Los productos de examen deben entregarse **única y exclusivamente** mediante la tarea de entrega habilitada para ello en el Campus Virtual, y dentro del plazo de tiempo establecido. Entregas efectuadas incumpliendo lo anterior no serán consideradas a efectos de corrección.

2. Criterios Generales de Corrección

Los códigos entregados deben satisfacer los siguientes criterios[1]:

- compilar correctamente sin errores ($-n$);
- ejecutarse sin lanzamiento de excepciones u otras anomalías graves de ejecución ($-n$);
- poseer una semántica que da soporte a la solución pedida en términos de entrada/salida, seguridad, vivacidad, rendimiento, etc. ($-n$);

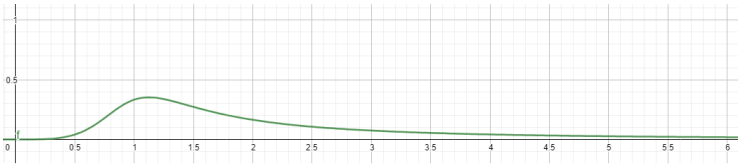
[1] Entre paréntesis, se indica la puntuación a descontar derivada del incumplimiento del criterio, para un enunciado valorado con n puntos.

- recoger íntegramente todas las especificaciones establecidas en el documento de examen e implementarlas tal y como se especifican ($-1{,}5$/especificación).
- estar contenidos en ficheros conformes en número, formato y nomenclatura de nombres con lo explicitado en este documento de examen ($-n$).

3. Enunciados de Examen

1. (5.5 puntos, Integración Paralela) Se desea disponer de un programa que calcule, aplicando el método de Monte-Carlo, la siguiente integral definida (el desarrollo de la función en el intervalo de integración $[0, 6]$ se ilustra en la Figura):

$$\int_0^6 \frac{2x^4}{(3x^6+3)} dx$$



Desarrolle utilizando el lenguaje Java un programa que permita calcular la integral de forma paralela de acuerdo al siguiente conjunto de especificaciones:

- No utilice paquetes (`package`);
- Guarde el código en `defIntegration.java`.
- Tareas paralelas modeladas mediante implementación de la interfaz `Runnable`. Cada tarea calcula la subsuperficie correspondiente a su subintervalo de integración de forma independiente, aplicando en el mismo el método de Monte-Carlo. AYUDA: la función a integrar está acotada superiormente en todo el intervalo de integración por uno (ver Figura).
- Cada tarea acumula la subsuperficie que ha calculado en una variable compartida de tipo `double` llamada `intParcial`, que al finalizar la ejecución contendrá la superficie (integral) buscada.
- Habrá exactamente seis tareas paralelas.
- La variable `intParcial` estará protegida por un protocolo de control de exclusión mutua que utilizará semáforos.
- Cada tarea debe disponer de su propio generador de números aleatorios de clase `Random`, independiente de los demás; los generadores estarán inicializados con diferentes semillas obligatoriamente.
- Cada una de las seis tareas paralelas utilizará un total de 3×10^6 puntos aleatorios. La tarea secuencial utilizará 18×10^6 puntos aleatorios.

- Las tareas se procesarán a través de un ejecutor de *pool* de hebras de clase `ThreadPoolExecutor`.
- El programa calculará la integral de forma secuencial y paralela, y terminará imprimiendo los tiempos en nanosegundos de cálculo secuencial y paralelo, los valores de la integral calculada secuencial y paralelamente, y finalmente el *speedup*, con un *output* como el siguiente:

```
Tiempo secuencial: n nanosegundos
Tiempo paralelo: m nanosegundos
Integracion secuencial: p (p es el valor calculado)
Integración paralela: q (q es el valor calculado)
Speedup: s
```

2. (4.5 puntos, MPJ-Express, Lotería) Una empresa de lotería desea desarrollar, empleando la biblioteca de paso de mensajes MPJ-Express, una aplicación que permita a los jugadores jugar de forma remota.

   - El juego estará formado por i procesos jugadores (*slaves*) y un proceso *master*, que será el encargado de dirigir el juego.
   - El número de jugadores vendrá determinado por el valor de los parámetros de lanzamiento (parámetro `-np`) de la aplicación y puede variar de una ejecución a otra. Cada jugador tendrá asociado un número único, que será asignado de forma consecutiva a partir del uno (jugador 1, jugador 2...)
   - Nada más empezar, el proceso *master* elegirá un número al azar de entre los cien primeros números naturales ($[0, 100)$) y lo imprimirá por pantalla. Será ese valor el que los jugadores tendrán que tratar de adivinar.
   - En cada ronda el proceso *master* quedará a la espera de que todos los jugadores le hagan llegar un número. Por cada uno de los números recibidos, el proceso *master* imprimirá un mensaje indicando el jugador que ha enviado el mensaje y el número que ha enviado, tal y como se muestra a continuación.

```
Número secreto: 44
Proceso 1 juega con el número 31
Proceso 2 juega con el número 38
Proceso 3 juega con el número 29
Proceso 4 juega con el número 23
Proceso 1 juega con el número 62
Proceso 2 juega con el número 62
Proceso 3 juega con el número 38
Proceso 4 juega con el número 12
Proceso 1 juega con el número 62
Proceso 2 juega con el número 44
Proceso 3 juega con el número 97
Proceso 4 juega con el número 32
El proceso 2 gana el juego
```

- Si alguno de los números enviados por parte de los jugadores coincide con el número secreto, el proceso *master* imprimirá un mensaje indicando el número de jugador que ha ganado el juego. A continuación enviará a todos los jugadores un mensaje con el número de jugador que ha ganado el juego para que dejen de jugar y finalicen su ejecución. Tras ese envío de mensajes el proceso *master* finalizará también su ejecución.
- Si ninguno de los jugadores ha acertado, el proceso *master* enviará a todos los jugadores el valor 0 y quedará a la espera de que todos los jugadores vuelvan a remitir un nuevo valor. Este proceso se repetirá indefinidamente hasta que alguno de los jugadores acierte el número secreto.
- Es importante que no se envíe ningún mensaje a los jugadores hasta que todos hayan enviado el número con el que desean jugar en la ronda.

- Los jugadores tienen muy poca memoria y son muy vagos para anotar los valores que han enviado previamente, así que en cada ocasión que tienen para enviar un número lo eligen siempre de forma completamente aleatoria. Un mismo jugador puede enviar varias veces el mismo número.
- Solo puede haber un vencedor del juego. En caso de que dos o más jugadores envíen el mismo número en una misma ronda y este coincida con el número secreto se podrá dar como vencedor a cualquiera de ellos. Queda a criterio del alumno cómo decidir el ganador en esta situación.
- Incluya en su programa como comentarios las órdenes de línea de comando necesarias para compilar y ejecutar su programa. Esta última debe incluir el valor del flag `-np` para crear el árbol de procesos que propone, suponiendo que en el juego participarán cuatro jugadores. La ausencia de estas órdenes invalida por completo el ejercicio, con independencia de los criterios generales de corrección.
- Guarde el código en `Loteria.java`.