Escuela Técnica Superior de Ingeniería

Prácticas
de
Programación Concurrente y Distribuida

3º Curso de Grado en Ingeniería Informática

Curso 2019-20

EXAMEN

Enero de 2020

CONSIDERACIONES PREVIAS:

- **No se permite el uso de ningún tipo de documentación.**
- **El acceso a Internet está desactivado conscientemente.**
- **Apague el teléfono móvil.**

ANTES DE COMENZAR EL EXAMEN:

- Cree una carpeta con su nombre y primer apellido en el **Escritorio** separados por un guión bajo (ejemplo: **`Pedro_Abad`).**
- En dicha carpeta aparecerá un proyecto por cada una de las preguntas del examen. Dichos proyectos se denominarán **`Proyecto1`, `Proyecto2`,…, `Proyecto4`.**

ENUNCIADO:

Una empresa de cromados dispone de un tanque para realizar el tratamiento. La empresa croma dos tipos de piezas: llantas y parachoques. En el tanque se pueden cromar a la vez dos parachoques, o cinco llantas, aunque también es posible mezclar un parachoques con hasta tres llantas.

PROYECTO 1.

Tiempo estimado: 40 minutos.
Puntos: 4

Será el proyecto base para solucionar el enunciado. Contendrá las siguientes clases:

- **`Tanque`**. La clase `Tanque` mantendrá el estado de ocupación del tanque e implementará los siguientes métodos:
  - **`Entrallanta`**. Que deberá ser invocado por las llantas cuando quieren acceder al tanque.
  - **`Salellanta`**. Que deberá ser invocado por las llantas al salir del tanque.
  - **`EntraPC`**. Que deberá ser invocado por los parachoques cuando quieren entrar en el tanque.
  - **`SalePC`**. Que deberá ser invocado por los parachoques al salir del tanque.
- **`Parachoque`**. Representará cada uno de los parachoques mediante un hilo. El hilo se creará heredando de la clase `Thread`. El hilo pondrá un mensaje de inicio indicando su identificador, intentará acceder al tanque usando la clase `Tanque`, permanecerá en él durante 4 segundos y saldrá.
- **`Llanta`**. Representará cada uno de las llantas mediante un hilo. El hilo se creará implementando el *interface* `Runnable`. El hilo pondrá un mensaje de inicio indicando su identificador, intentará acceder al tanque usando la clase `Tanque`, permanecerá en él un tiempo aleatorio entre 2 y 3 segundos y saldrá.
- **`Generador`**. Contendrá el método `main` y será quién comience la ejecución. Debe lanzar, de forma aleatoria, llantas y parachoques a intervalos de tiempo

entre 1 y 2 segundos. La frecuencia de llegada de las llantas será del 70% y la de parachoques del 30%. Deberá esperar a que finalicen todos los hilos para finalizar.

El control de la concurrencia y la sincronización se realizará en la clase `Tanque`, mediante las primitivas de Java `wait(), notify()` y/o `notifyAll()`.

PROYECTO 2.

Tiempo estimado: 20 minutos.
Puntos: 3

Se modificará el *Proyecto1* para que la clase `Tanque` controle la concurrencia mediante `ReentrantLocks` y `Conditions`.

En esta solución, los parachoques tendrán prioridad de acceso al tanque.

No podrá usarse el método `signalAll()` de las `Conditions`.

PROYECTO 3.

Tiempo estimado: 15 minutos.
Puntos: 2

Tomará como base el *Proyecto1* y se modificará la clase **`generador`** para que use un *ThreadPool* con un tamaño fijo de 6 hilos para lanzar las llantas. Al finalizar, cada llanta deberá devolver el tiempo que ha empleado en cromarse, y generador pondrá un mensaje final indicando el tiempo total de ocupación del tanque por las llantas.

PROYECTO 4.

Tiempo estimado: Depende de la implementación que se pretenda
Puntos: 1

Se creará un *Frame* que visualice de forma gráfica, mediante un *Canvas*, la situación del tanque y las colas de espera del *Proyecto 1*.