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 fábrica de fertilizantes produce dos tipos de abono, A y B, a partir de dos materias primas M1 y M2.

Para fabricar 1 saco de A hacen falta 3 kg de M1 y 2 kg de M2.
Para fabricar 1 saco de B hacen falta 1 kg de M1 y 3 kg de M2.

Un Robot *reponedor*, a intervalos de 4 segundos, introduce en un silo, cantidades aleatorias de producto M1 y M2 de entre 3 y 5 kg. Inicialmente los silos están vacíos.

Dos robots RA y RB, fabrican sacos de producto, tomando las cantidades de producto del silo. Cuando tienen las cantidades necesarias, tardan un tiempo aleatorio de 1 a 3 segundos para fabricar el fertilizante y posteriormente vuelven a por más. Los robots cogerán las cantidades totales de ambas materias prima o no cogerán nada.

PROYECTO1.

Tiempo estimado: 35 minutos.
Puntos: 4

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

- **`Silo`**. La clase `Silo` mantendrá el estado de ocupación del silo e implementará los siguientes métodos:
  - **`RACoge`**. Que deberá ser invocado por el robot RA cuando necesita los productos.
  - **`RBCoge`**. Que deberá ser invocado por el robot RB cuando necesita los productos.
  - **`Rellenar`**. Que deberá ser invocado por el robot *reponedor* cuando repone el silo.
- **`RobotA`**. Representará al robot RA mediante un hilo. El hilo se creará heredando de la clase `Thread`. El robot mezclará 10 sacos antes de acabar. Cada vez que retire los productos del silo lo indicará con un mensaje.
- **`RobotB`**. Representará al robot RB mediante un hilo. El hilo se creará implementando el *interface* `Runnable`. El robot mezclará 10 sacos antes de

acabar. Cada vez que retire los productos del silo lo indicará con un mensaje.

- **`RobotR`**. Representa al robot *reponedor*. El hilo se creará heredando de la clase `Thread`. Este robot será interrumpido por el generador cuando los otros acaben. Cada vez que rellene el silo, indicará con un mensaje la cantidad depositada.
- **`Generador`**. Contendrá el método `main` y será quién comience la ejecución. Debe lanzar los tres robots, esperar a que finalicen los robots RA y RB e interrumpir al robot *reponedor*.

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

PROYECTO 2.

Tiempo estimado: 25 minutos.
Puntos: 3

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

En esta solución, tras rellenar el silo, si ambos robots están esperando, tendrá prioridad el robot RA.

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

PROYECTO 3.

Tiempo estimado: 15 minutos.
Puntos: 2

Tomará como base el Proyecto 1 y **eliminará la clase `silo`**, haciendo que los robots controlen la toma de materias mediante **semáforos**.

Para este caso, se permite que los robots tomen el material de forma incremental (de kilo en kilo, empezando por la materia M1), y que lo hagan ambos simultáneamente.

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*.