Actividad

Actividad AE3_T3_Multihilo

Ubicación

Tema 3 - Programación multihilo

Objetivos

- Comprender el concepto de hilo y programación multihilo.
- Saber crear múltiples hilos a partir de una aplicación Java.
- Aprender a gestionar la comunicación entre hilos.
- Desarrollar aplicaciones multihilo para resolución de problemas complejos.

Temporalización

La duración prevista para esta actividad es de seis sesiones lectivas.

Instrucciones

En los juegos de estrategia tipo Age of Empires, Warcraft, Command & Conquer, etc. hay una parte del juego que consiste en que unas unidades especializadas recolecten recursos de algún elemento del juego (minas, bosques, granjas, etc.). Estos elementos tienen una capacidad concreta de recursos (por ejemplo: 1.000 oros), que va decreciendo a medida que las unidades especializadas los recolectan. Se pide desarrollar un programa que tenga 3 clases:

- Clase “Mina” con un atributo que sea “stock” y un método constructor que inicialice el stock a un valor determinado pasado como parámetro al método (por ejemplo, 10.000 oros).
- Clase “Minero” con dos atributos enteros que sean “bolsa” (donde guarda los recursos que recolecta) y “tiempoExtraccion” (tiempo de trabajo necesario para extraer un recurso) y dos métodos: un método constructor que inicialice la bolsa a 0; y un método “extraerRecurso” que extraiga de la mina un recurso en cada turno y lo guarde en su bolsa. Para simular un turno de trabajo se deberá incluir una pausa (definida por la variable tiempoExtraccion, en milisegundos -por ejemplo 1000 ms-) cada vez que se ejecute el método “extraerRecurso”.
- Clase “App” con un método “main” que cree un objeto Mina y un pequeño ejército de hilos mineros (no menos de 10) que se dediquen a extraer los recursos de la mina. El programa finalizará cuando ya no queden recursos para extraer. Al finalizar deberá mostrar la suma de todo lo recolectado por los mineros.

El programa debe ser “thread-safe”, es decir, debe asegurar que cuando un minero accede a la mina y descuenta 1 oro, no haya inconsistencias en el stock, sino que la parte correspondiente del método “extraerRecurso” esté debidamente protegida. Para comprobar que funciona correctamente, la suma de todo lo recolectado deberá ser igual al stock original.

Como ampliación se propone introducir un sistema de ventilación en la mina para cumplir con el protocolo COVID-19. Para ello se pide crear una nueva clase “Ventilador” que tendrá como

atributos un booleano para determinar el modo de funcionamiento (encendido/apagado) y un tiempo (entero) que será igual para el modo encendido y para el modo apagado. Deberá tener dos métodos: “encenderVentilador” y “apagarVentilador”, que se ejecutarán desde dos hilos distintos, de manera similar al funcionamiento del programa del Productor-Consumidor. El hilo de encendido y el hilo de apagado deberán ir alternándose (utilizando “wait()” y “notify()”) indefinidamente de acuerdo al valor de tiempo especificado, mostrando por pantalla un mensaje que indique el estado en que se encuentra el ventilador. El objeto Ventilador y los hilos para encendido y apagado se instanciarán en el método “main” de la case “App”.

Observaciones

- Se entregará en Florida Oberta el fichero ZIP con el código completo de la actividad y un fichero txt que incluya el enlace al repositorio Github, cuyo nombre seguirá el siguiente formato:
  - <APELLIDO1>_<APELLIDO2>_<NOMBRE>__AE<N>.zip donde “N” es el número de actividad.
- La actividad también deberá subirse a una rama de Git “AE<N>” que partirá de “master”, donde “N” es el número de actividad.

Evaluación

La actividad es obligatoria. Para la evaluación se tendrá en cuenta el funcionamiento de los programas, la codificación adecuada y la documentación. Se puede solicitar al alumno que explique parte de su código así como que realice pequeñas modificaciones.

Recursos

Material de módulo (Florida Oberta).