Práctica 3 - Sistema concurrente

Análisis y Diseño de Software, 2023
Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación
ETSI de Telecomunicación
Universidad Politécnica de Madrid

Objetivos

- Ilustrar un enfoque a procesar grandes datos
- Desarrollar un sistema concurrente: gestión de hebras y su sincronización
- Ilustrar la integración de algoritmos y la concurrencia
- Depurar un programa, detectar y corregir errores.

Introducción

Esta práctica es una continuación del laboratorio 4. El contenido del proyecto proporcionado en este laboratorio es exactamente el que hay que trabajar en esta práctica. Igualmente, el contenido del enunciado del laboratorio 4 es aplicable y es imprescindible completar los ejercicios propuestos para esta práctica.

En el laboratorio 4, se usó la biblioteca (`monitor.jar`) que proporcionaba la clase `MonitorSegmentos`. En esta práctica hay que implementar justamente este monitor. El programa final deberá usar este código desarrollado para ejecutar el sistema. Evidentemente, no es posible usar la biblioteca propuesta (`monitor.jar`) en este trabajo.

En la versión proporcionada para la práctica, incluyan algunos cambios al laboratorio 4. Las modificaciones han sido necesarias para corregir algún error y preparar el código para la evaluación automática de la práctica:

- Se ha añadido un constructor que recibe un segmento con los registros de la red social.
- Se ha añadido el método público `getInversiones`.
- Se ha corregido un error en el método `integrarResultados` de la clase `AlmacenLab4`.

El contenido los métodos desarrollados en el laboratorio 4 se puede copiar y pegar en los métodos equivalentes en el proyecto publicado para esta práctica (`adsw-practica-3.zip`). Al hacer este cambio, se producirán errores de compilación en el código que se ha hecho en el laboratorio, pero debería ser sencillo corregirlos. Algunos cambios hechos:

- Se ha definido el interfaz `MonitorSegmentosInterface.java`. La clase `MonitorSegmentos.java` implementa este interfaz.
- Se ha añadido algún método público en `AlmacenLab4.java`.
- El método `procesarSegmentos.java` debe retornar una lista de los resultados obtenidos.
- Algunos métodos se han hecho públicos. Estos cambios no deberían cambiar el código desarrollado en el laboratorio.
- En el monitor de laboratorio 3 se han añadido `int id` en los métodos `putSegmento` y `getResultado` para facilitar la información de las trazas.

El fichero comprimido `adsw-practica-3.zip` incluye toda la información necesaria para la práctica. Igualmente, proporciona el programa necesario para entregar el trabajo.

Diagrama de clases

En un fichero adjuntado, muestra el diagrama de las clases en el sistema. En la documentación de este enunciado se proporciona el fichero `classdiagram.png` que incluye las clases del proyecto. Las clases están documentadas con javadoc. El diagrama y su contenido es igual a la documentación proporcionada en el laboratorio 4.

Especificación de la clase `MonitorSegmentos`

La clase `MonitorSegmentos` es un monitor que sincroniza objetos entre `AlmacenLab4` con hebras (`Hebra`). Hay que considerar los siguientes aspectos:

- Un método de la clase `AlmacenLab4` proporcionan segmentos y a las hebras los obtienen y procesan.
- Las hebras procesan y proporcionan resultados del procesamiento de los segmentos.
- La clase `AlmacenLab4` debe incluir, al menos un método para:
  - generar y enviar los segmentos.
  - retornar y procesar los resultados de procesar los segmentos.
- Las hebras solicitan segmentos hasta que el monitor haya finalizado. Entonces deben terminar.
- Hay que bloquear a las hebras cuando no hay segmentos disponibles. Hay que buscar a una hebra de `AlmacenLab4` (posiblemente la hebra `main`) cuando no hay resultados.
- El monitor finaliza cuando `AlmacenLab4` haya recibido tantos resultados, como segmentos haya enviado.
- Un resultado se representa como `Map<Localizacion, List<Integer>>`, donde `Localizacion` es una localización.

- Las estructuras de datos para almacenar segmentos o resultados no están acotadas.
- Hay que procesar una vez cada segmento y retornar un resultado asociado a un segmento.

putSegmento

```
public void putSegmento(SegmentoInterface segmento, int id)
```

Un método de la clase `AlmacenLab4` invoca a este método para proporcionar un segmento.

- **Parameters**
  - *segmento*: el segmento a procesar
  - *id*: identificador de la hebra. Se usa para conocer cuáles son las hebras que invoca este método en las trazas.

getSegmento

```
public SegmentoInterface getSegmento (int id) throws
java.lang.InterruptedException
```

Una hebra invoca a este método para obtener un segmento. Este método tiene que retornar un `null` si no hay segmentos pendientes de retornar a una hebra, cuando se ha solicitado la finalización el monitor.

- **Parameters**
  - *id*: identificador de la hebra. Se usa para conocer cuáles son las hebras que invoca este método en las trazas.
- **Returns**: Un segmento para procesarlo
- **Throws**
  - `java.lang.InterruptedException` - Esta excepción se lanza si se interrumpa a la hebra que ha invocado cuando esté bloqueada.

getResultado

```
public ResultadoInterface getResultado(int id) throws
InterruptedException;
```

Un método de la clase `AlmacenLab4` invoca a este método para obtener un resultado al procesar un segmento.

- **Parameters**
  - id: identificador de la hebra. Se usa para conocer cuáles son las hebras que invoca este método en las trazas.
- **Returns**: El resultado de procesar un segmento
- **Throws**:

- `java.lang.InterruptedException` - Esta excepción se lanza si se interrumpa a la hebra que ha invocado cuando esté bloqueada.

putResultado

```
public void putResultado( ResultadoInterface resultado, int id);
```

Una hebra invoca a este método para proporcionar un resultado al procesar un segmento.

- **Parameters**
  - *resultado*: El resultado
  - *id*: identificador de la hebra. Se usa para conocer cuáles son las hebras que invoca este método en las trazas.

finalizar

```
public void finalizar()
```

El método `MonitorSegmentos` invoca a este método para notificar que ya ha obtenido los resultados de los segmentos proporcionados. Este método debe activar a las hebras esperando segmentos.

estaFinalizado

```
public boolean estaFinalizado()
```

Las hebras consultan este método para conocer si se ha finalizado el monitor

- **Returns**: Indica si el monitor ha finalizado

Entrega de la práctica para la evaluación

Todos los alumnos tienen que subir su trabajo, si quieren evaluarlo.

El proyecto que hemos instalado incluye un script de entrega de prácticas, de forma similar al laboratorio 0. El proceso de entrega es el siguiente:

- El proyecto ADSW-practica3-2023 incluye un script de entrega (`Practica3Entrega.launch`) que podemos ejecutar seleccionando el script y ejecutando `Run > Run As > Practica3Entrega`. Este script nos va indicando en la consola de Eclipse los pasos que va dando (chequeos y compilaciones, ejecuciones de pruebas, cálculo de notas, comprimir entregas, subidas a moodle, ...). Muestra la estimación de nota calculada.

- Si hay errores de compilación, o si lo que se debe desarrollar en la práctica no se ajusta a lo que indica la guía (identificadores, signaturas de métodos, visibilidad de clases o métodos, ...) las pruebas no se pueden ejecutar y el fichero zip no se construirá. Si ha podido ejecutar pruebas dejará construido un fichero `ADSW-practica3-2023.zip` en la raíz del proyecto Eclipse. Para hacerlo

visible debemos refrescar el proyecto con el comando Refresh del menú de contexto del explorador de proyectos de Eclipse.

- Si tenemos un fichero entregable, se arranca un browser empotrado que nos permite logarnos en Moodle. Es la primera vez que utilizamos esta herramienta y es compleja porque depende de muchas cosas (versión de Eclipse, sistema operativo, versión Java que utilizamos). Si queremos hacer la entrega, nos logamos en moodle, y subirá la entrega, y subirá también la nota calculada. Si no queremos hacer la entrega simplemente cerramos el browser y terminará la entrega, sin subirse nada a Moodle.

- La versión actual de Moodle, algunas veces se queda paradas, y dejan de mandar datos al browser, y eso se puede arreglar algunas veces si recargamos la página en el browser empotrado.