Programación concurrente

¿Cómo implementar concurrencia en el paradigma orientado a objetos?

- se asocian las tareas a los métodos y los hilos que utilizan esas tareas a objetos

- un sistema es un conjunto de objetos autónomos que colaboran, activa y concurrentemente.

- modelos de objetos

  - Modelo activo
  - Modelo pasivo
  - Modelo mixto

Modelos de objetos

- Modelo Activo
  - Cada objeto es autónomo
  - Sistemas orientados a objetos distribuidos
  - Pasaje de mensajes via comunicación remota
- Modelo Pasivo
  - Objetos en modelo secuencial
  - Pasaje de mensajes por invocación de procedimientos
- Modelo Mixto (concurrencia en Java)
  - Objetos pasivos: muestran conciencia de los hilos protegiéndose via locks (cerrojos)
  - Objetos activos: hilos, que comparten el acceso a objetos pasivos

Modelo mixto

Programa concurrente

Colecciones de objetos que se coordinan para resolver un problema

- Objetos pasivos
- Objetos activos
- Objetos controladores

Modelo mixto

Objeto activo

Implementa codigo procedural que se ejecuta en hilos

Objeto pasivo/reactivo

Actua solo en respuesta a un requerimiento recibido desde un objeto activo, y controlan las interacciones entre los objetos activos

Diagrama de estados

Modelo mixto

- Pasos sugeridos para crear un programa concurrente
  - Descripción corta del problema a resolver
  - Identificar objetos y relaciones
    - Especificar si el objeto es activo o pasivo
  - Diseñar los objetos activos
  - Diseñar los objetos pasivos
    - Utilizar diagrama de estados
  - Implementar objetos activos como hilos
  - Implementar objetos pasivos – ***Thread safe***
  - Implementar objeto controlador

Modelo mixto

- Pasos sugeridos para crear un programa concurrente
  - Descripción corta del problema a resolver
  - Identificar objetos y relaciones
    - Especificar si el objeto es activo o pasivo
  - Diseñar los objetos activos
  - Diseñar los objetos pasivos
    - Utilizar diagrama de estados

Automata:
Nodos: estados del objeto
Arcos: metodos a ejecutar, pueden producir cambios de estdo

Modelo mixto

Problema del productor/consumidor

Descripcion del problema: un buffer con un numero finito de lugares para datos generados por un productor. El productor simplemente crea enteros y los envia al buffer. El buffer los almacenara en el proximo lugar vacio (si lo trabaja como una cola). El consumidor consumirá los enteros del buffer y los imprimira. El buffer tendrá la habilidad de almacenar solamente 3 items. Asi, si el productor crea mas items que los que el buffer puede mantener antes de que el consumidor los consuma, deberá esperar para que el consumidor tome el item del buffer para liberar un espacio antes de enviar otro item al buffer. Asi mismo el consumidor podría querer retirar mas items de los producidos y se encontrará con que no hay mas items disponibles. Cuando esto ocurre , el consumidor debera esperar hasta que el productor cree otro item y lo agregue al buffer.

Descripción corta del problema: crear un productor que cree datos y un consumidor que use esos datos y se coordinen para ello utilizando un buffer limitado.

Modelo mixto

Problema del productor/consumidor

Definir los objetos y las relaciones:

- identificar a los objetos activos de la solución
- identificar a los objetos pasivos de la solución
    - Se debe considerar objetos productor, consumidor y buffer.
    - El productor y el consumidor hacen cosas (producen y consumen datos), luego son objetos activos.
    - El buffer simplemente almacena y recupera datos segun sea requerido por el productor y el consumidor, luego es un objeto pasivo.
    - El productor debe crear un numero y enviarlo al buffer.
    - El consumidor debe recuperar un numero del buffer y mostrarlo por pantalla.
    - El buffer responde a 2 tipos de requerimientos: tomar un item del productor y agregarlo a su estructura, y tomar un item de su estructura y entregarlo al consumidor.

Modelo mixto

Problema del productor/consumidor

- Diseñar e implementar los objetos activos (productor y consumidor)

Se propone que realice repetidamente la accion de enviar un item al buffer, y consumir un item del buffer respectivamente.
Siempre vamos a considerar que los objetos activos son hilos, entonces las clases Productor y Consumidor deben tener un método run() que será donde se realicen las acciones indicadas anteriormente, y que además es el lugar donde el hilo comienza a ejecutarse. Se puede realizar por cualquiera de los 2 métodos vistos, ya sea creando una subclase de Thread o implementando Runnable.
El buffer es la variable compartida por los 2 objetos activos creados, el productor y el consumidor. El buffer será creado en el objeto controlador, o sea el que crea los hilos y los pone en estado de runnable, y que ademas tiene el main (en java). El acceso al buffer desde el productor o consumidor genera una sección crítica a tener en cuenta.

Modelo mixto

Problema del productor/consumidor

- Diseñar los objetos pasivos (el buffer)

El buffer tiene 2 métodos que permiten que los objetos activos productor y consumidor utilicen sus servicios. El método *agregar*, permite que un objeto agregue un item al buffer; y el método *sacar* permite que un objeto remueva un item del buffer. No alcanza con que estos método ssimplemente lleven a cabo la tarea para sus objetos activos llamadores cuando son llamados (que es lo que sucede normalmente en un situación secuencial, no concurrente). En un escenario concurrente, los objetos pasivos, mantienen un estado a través de las invocaciones a sus métodos. Este estado permite que mantengan un flujo de eventos cuando los métodos del objeto son utilizados. En el problema prod/cons, el buffer puede estar en uno de 3 estados: vacío, lleno, o disponible. Vamos a utilizar un diagrama de estados para modelar un objeto pasivo.Los estados son nodos y los métodos son transiciones de estado.

Modelo mixto

Problema del productor/consumidor

- El buffer responde a 2 tipos de requerimientos: tomar un item del productor y agregarlo a su estructura, y tomar un item de su estructura y entregarlo al consumidor.
- Entonces, el buffer tendrá 2 métodos: *agregar* y *sacar*.
- Se propone hacer una tabla indicando
    - Nombre del metodo
    - Estados en que puede ejecutarse
    - Precondición donde se verifique el estado para decidir si corresponde hacer *wait()*
    - Postcondicion donde se decide si se actualiza el estado para hacer el *notify()* o *notifyAll()*

Modelo mixto

Problema del productor/consumidor – buffer – diagrama de estado