Progrmación Concurrente, 23 -I, CG01, June 7, 2023

Java Communicating Sequential Processes (JCSP)

Martínez Buenrostro Jorge Rafael.

Unidad Iztapalapa
Universidad Autónoma Metropolitana
*molap*96@*gmail.com*

Resumen: En este reporte se describirá y mostrará el uso de concurrencia de JCSP.
Palabras Clave: Concurrencia, Sincronización, Entidad Activa, Comunicación.

Introducción

JCSP (Communicating Sequential Processes for Java) es una librería de concurrencia diseñada para el lenguaje de programación Java. Proporciona un modelo de concurrencia basado en procesos secuenciales que se comunican entre sí mediante canales. Esta librería es una implementación del modelo CSP (Communicating Sequential Processes) creado por Tony Hoare.

Descripción general del lenguaje

JCSP se integra con el lenguaje de programación Java, que es ampliamente utilizado en el desarrollo de aplicaciones empresariales y de propósito general. Java es un lenguaje de programación orientado a objetos, seguro y portátil, que se ejecuta en la Java Virtual Machine (JVM). La integración de JCSP con Java permite aprovechar todas las características y bibliotecas existentes de Java, mientras se agrega el modelo de concurrencia basado en CSP.

Descripción general del modelo de concurrencia

El modelo de concurrencia de JCSP se basa en el paradigma de programación concurrente conocido como CSP. En este modelo, los procesos secuenciales se comunican entre sí mediante canales, que actúan como medios de comunicación unidireccionales. Los procesos se sincronizan en los canales, lo que garantiza una comunicación segura y sin condiciones de carrera.

JCSP proporciona abstracciones para crear procesos concurrentes y canales de comunicación. Los procesos se implementan como clases Java que extienden la interfaz CSProcess. Los canales se definen mediante interfaces específicas, como ChannelInput y ChannelOutput, que permiten la comunicación entre los procesos.

Soporte para la creación de EA

JCSP ofrece soporte para la creación de Elementos Activos (EA) dentro de su modelo de concurrencia. Un Elemento Activo es un objeto concurrente que encapsula su propio proceso de ejecución y se comunica con otros Elementos Activos a través de canales.

Para crear un Elemento Activo en JCSP, se define una clase que implementa la interfaz Active. Esta interfaz proporciona métodos para la inicialización, ejecución y finalización del Elemento Activo. El proceso de ejecución se realiza en paralelo con otros procesos y se sincroniza mediante canales para lograr una comunicación segura.

Modelos de comunicación que soporta

JCSP soporta varios modelos de comunicación para facilitar la interacción entre los procesos concurrentes. Algunos de los modelos de comunicación que ofrece JCSP son:

- Comunicación síncrona: Los procesos se sincronizan en los canales, de modo que el emisor se bloquea hasta que el receptor esté listo para recibir el mensaje. - Comunicación asíncrona: Los procesos no se bloquean al enviar o recibir mensajes a través de los canales. Esto permite una comunicación más rápida, pero puede dar lugar a condiciones de carrera si no se realiza una sincronización adecuada. - Comunicación por eventos: JCSP también admite la comunicación basada en eventos, donde los procesos pueden enviar y recibir eventos para coordinar su ejecución.

Herramientas de sincronización

JCSP proporciona diversas herramientas de sincronización para facilitar la coordinación y la comunicación entre los procesos concurrentes. Algun

as de estas herramientas son:

- Semáforos: JCSP incluye una implementación de semáforos que permite controlar el acceso a recursos compartidos entre los procesos concurrentes. - Barreras: Las barreras en JCSP permiten sincronizar la ejecución de múltiples procesos, de modo que todos los procesos deben alcanzar la barrera antes de que puedan continuar. - AltingBarrier: Esta herramienta combina una barrera con una espera alternativa (alt), lo que permite a los procesos esperar en la barrera o realizar otras acciones según las condiciones de comunicación.

Ejemplo mínimo

A continuación se muestra un ejemplo mínimo de cómo utilizar JCSP para crear dos procesos que se comunican entre sí:

```
import org.jcsp.lang.*;
```

```
public class EjemploJCSP
{
  public static void main(String[] args)
  {
    // Crear canales de comunicacion
        One2OneChannel channel1 = Channel.one2one();
        One2OneChannel channel2 = Channel.one2one();

        // Crear procesos
        new ProcessA(channel1.out(), channel2.in()).run();
        new ProcessB(channel2.out(), channel1.in()).run();
  }

  static class ProcessA implements CSProcess
  {
      private ChannelOutput out;
          Private ChannelInput in;

    public ProcessA(ChannelOutput out, ChannelInput in)
    {
      this.out = out;
          this.in = in;
    }

    public void run()
    {
      // Enviar mensaje al proceso B
          out.write("Hola, proceso B!");

      // Recibir respuesta del proceso B
      String respuesta = (String) in.read();
      System.out.println("Proceso A recibe: "+respuesta);
    }
  }

  static class ProcessB implements CSProcess
  {
    private ChannelOutput out;
        private ChannelInput in;

        public ProcessB(ChannelOutput out, ChannelInput in)
        {
          this.out = out;
          this.in = in;
        }
```

```
        public void run()
        {
          // Recibir mensaje del proceso A
          String mensaje = (String) in.read();
          System.out.println("Proceso B recibe: " + mensaje);

          // Enviar respuesta al proceso A
          out.write("Hola, proceso A!");
        }
  }
}
```

En este ejemplo, se crean dos procesos: ProcessA y ProcessB. Utilizan dos canales de comunicación, channel1 y channel2, para enviar mensajes entre sí. ProcessA envía un mensaje a ProcessB a través de channel1 y recibe la respuesta a través de channel2. ProcessB recibe el mensaje de ProcessA a través de channel2 y envía una respuesta a través de channel1.

Conclusiones

JCSP proporciona una librería sólida y eficiente para la programación concurrente en Java, basada en el modelo de concurrencia CSP. Permite crear procesos concurrentes y establecer comunicación segura entre ellos mediante canales. Además, ofrece herramientas de sincronización y soporte para la creación de Elementos Activos. Utilizar JCSP puede mejorar la concurrencia y la eficiencia de las aplicaciones Java, evitando problemas comunes asociados con la concurrencia, como condiciones de carrera y bloqueos.