











🏗️ **Paso 1: Comprender el Enunciado y Conceptos Clave**

Antes de empezar a programar, es fundamental tener claro:

✔️ El modelo de **Red de Petri** y su estructura (plazas, transiciones, buffers y recursos compartidos).

✔️ Las propiedades de concurrencia que deben analizarse: **deadlock, vivacidad, seguridad**.

✔️ Los **modos de procesamiento** y cómo deben gestionarse en la ejecución del sistema.

✔️ La necesidad de **paralelismo**, **gestión de hilos** y **políticas de procesamiento**.

💡 **Paso 2: Modelado y Análisis de la Red de Petri**

Utiliza la herramienta **PIPE** para analizar propiedades como:

✔️ **Deadlock:** Determinar si el sistema puede quedarse bloqueado sin poder continuar.

✔️ **Vivacidad:** Confirmar que todas las transiciones pueden ejecutarse en algún punto.

✔️ **Seguridad:** Evaluar si hay condiciones que pueden llevar a un comportamiento incorrecto.

Además, identifica:

✔️ **Invariantes de plaza**: Qué condiciones siempre deben cumplirse en el sistema.

✔️ **Invariantes de transición**: Qué reglas rigen la ejecución de transiciones y cómo se sincronizan.

🔧 **Paso 3: Diseño de la Solución en Java**

Se debe desarrollar la estructura del programa usando **orientación a objetos**, con las siguientes clases principales:

✔️ **Monitor** (gestiona la concurrencia y el disparo de transiciones).

✔️ **Política** (maneja la aleatoriedad o la prioridad en la elección de procesamiento).

✔️ **Main** (punto de entrada al programa).

✔️ **Clases auxiliares** para representar plazas, transiciones y buffers.

**Importante:**  
✔️ La clase **Monitor** debe ser **agnóstica** a la red de Petri que ejecuta.

✔️ El método **fireTransition()** será el único acceso público al monitor.

⚙️ **Paso 4: Implementación del Paralelismo y Hilos**

Para maximizar la concurrencia:

✔️ Determinar la cantidad **óptima** de hilos según los invariantes de transición.   
✔️ Gestionar casos de **conflictos y joins** en la ejecución.

✔️ Visualizar la responsabilidad de cada hilo con un **gráfico** (como la Figura 2 del enunciado).

🎭 **Paso 5: Implementación de las Políticas de Procesamiento**

Se debe desarrollar el objeto **Política** con:

✔️ **Modo aleatorio**: Elige un modo de procesamiento al azar.

✔️ **Modo priorizado**: Favorece el **procesamiento simple** sobre los otros.

⏳ **Paso 6: Semántica Temporal y Análisis de Tiempos**

✔️ Asignar tiempos a las **transiciones temporales**.

✔️ **Ejecutar múltiples pruebas** variando los tiempos y registrar los resultados.

✔️ Ajustar la configuración para que el programa dure entre **20 y 40 segundos**.

📊 **Paso 7: Registro y Validación de Resultados**

✔️ **Guardar un log** con los datos de ejecución.

✔️ **Verificar invariantes de plaza y transición** después de cada disparo de transición.

✔️ **Analizar logs** con expresiones regulares para validar el correcto funcionamiento.

📑 **Paso 8: Generación de Documentación y Entregables**

✔️ **Diagrama de clases** en alta calidad.

✔️ **Diagrama de secuencia** mostrando un disparo exitoso de transición.

✔️ **Código fuente** bien estructurado y documentado.

✔️ **Informe obligatorio** explicando la solución y los resultados obtenidos.