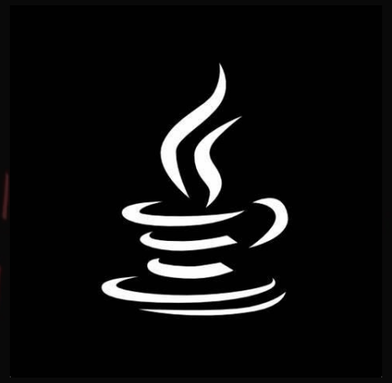
TESTING DE CONCURRENCIA CON JAVA



Mohamed Azougagh Asrih
Christopher Lucas Chinique
Manuel Jiménez Fernández

Definición de concurrencia

La concurrencia en Java se refiere al manejo simultáneo de múltiples tareas dentro de una misma aplicación. Esto permite ejecutar procesos paralelamente, aumentando la eficiencia y el rendimiento del software.

Casos de uso de la concurrencia

Procesamiento masivo de datos

La concurrencia permite dividir el trabajo en múltiples hilos, agilizando la manipulación de grandes volúmenes de información.

Interfaz de usuario sensible

Con la concurrencia, es posible ejecutar tareas en segundo plano sin afectar la capacidad de respuesta de la interfaz de usuario.

Servicios web

La concurrencia se utiliza para atender múltiples solicitudes simultáneamente, mejorando el rendimiento y la escalabilidad de los servicios web.

Juegos y simulaciones

La concurrencia permite realizar cálculos complejos en paralelo, brindando una experiencia más fluida y realista a los usuarios.

Ventajas e inconvenientes de la concurrencia en un testing

1

Ventajas

Aumento del rendimiento, mejor utilización de recursos, mayor capacidad de respuesta y mejor experiencia de usuario.

2

Inconvenientes

Complejidad de programación, mayor consumo de memoria, posibilidad de errores de concurrencia como condiciones de carrera y bloqueos.

Problemas potenciales de la concurrencia en el ámbito del Testing

Condiciones de carrera

Se producen cuando múltiples hilos intentan acceder y modificar una misma variable compartida, generando resultados inesperados.

Bloqueos y deadlocks

Ocurren cuando dos o más hilos quedan bloqueados esperando indefinidamente por recursos mutuamente excluyentes, impidiendo el progreso del programa.

Starvation y livelocks

La Starvation ocurre cuando un hilo no puede acceder a recursos debido a la alta prioridad de otros hilos. Livelocks son situaciones en las que los hilos se bloquean sin avanzar.

Ejemplos de concurrencia en Java

- Manejo de múltiples solicitudes REST en un servidor
- Procesamiento de imágenes en paralelo
- Simulación de tráfico en un sistema de gestión del transporte

Buenas prácticas de programación concurrente

1

Evitar condiciones de carrera

Utilizar sincronización adecuada y estructuras de datos concurrentes para evitar conflictos entre hilos.

2

Gestionar recursos compartidos

Utilizar bloqueos y semáforos adecuadamente para evitar bloqueos y liberar recursos cuando ya no sean necesarios.

3

Optimizar el número de hilos

No crear un número excesivo de hilos, ya que puede tener un impacto negativo en el rendimiento.