

Proyecto 1

Problema seleccionado:

Simulador de una barbería.

Planteamiento del problema.

En una barbería hay un solo barbero (Tony) que atiende a los clientes cortándoles el cabello. Si no hay clientes, el barbero duerme. Sin embargo, cuando no está cortando el cabello ni durmiendo, debe realizar otras tareas como limpiar su zona de trabajo (barrer los cabellos) o dar cambio a los clientes antes de atender al siguiente.

La barbería tiene tres sillas de espera y una silla de corte, lo que significa que en la barbería puede haber hasta cuatro personas a la vez: tres esperando y una siendo atendida.

El flujo de trabajo es el siguiente:

1. Un cliente llega y, si hay espacio disponible, espera su turno; si no, se va.
2. Si el barbero está libre, atiende inmediatamente al cliente.
3. Cuando el corte termina, el cliente paga y el barbero le da el cambio.
4. Antes de atender a otro cliente, el barbero debe limpiar su área de trabajo.
5. El barbero solo puede dormir si no hay clientes esperando ni en proceso de atención.

Entorno de desarrollo

El programa fue desarrollado en Java (versión 1.8.0_431) en una computadora con macOS, específicamente en una Mac con chip M2.

Se implementaron mecanismos de sincronización mediante hilos, semáforos y candados (locks) para gestionar la concurrencia entre los clientes y el barbero.

Dado que Java es un lenguaje multiplataforma, el programa puede ejecutarse sin problemas en otros sistemas operativos compatibles.

Estrategias de sincronización usadas

Para poder simular correctamente la dinámica de una barbería con múltiples clientes y un solo barbero, desarrollé varios mecanismos de sincronización que garantizan un comportamiento ordenado y libre de condiciones de carrera. Todos los clientes son hilos que se ejecutan concurrentemente, por lo que me aseguré de coordinar el acceso a los recursos compartidos.

Semáforo (sillasDisponibles)

Implementé este semáforo para controlar el acceso a las sillas de espera en la barbería. Decidí establecer un máximo de 3 sillas disponibles. Cuando todas están ocupadas, los clientes que llegan después no pueden esperar y tienen que irse.

Semáforo (barberoLibre)

Diseñé este semáforo para gestionar la disponibilidad del barbero, asegurando que solo un cliente sea atendido a la vez. Controla el acceso exclusivo al barbero.

Candado o Lock (lockParaColaClientes)

Implementé este candado para proteger el acceso concurrente a la cola de clientes que esperan turno. Evita condiciones de carrera al modificar la cola de espera.

Candado o Lock (lockParaProcesoPago)

Desarrollé este candado específicamente para proteger la operación de pago entre el cliente y el barbero. Evita que múltiples pagos se mezclen o interfieran.

Cola de Espera (clientesEnEspera)

Diseñé esta estructura FIFO protegida para almacenar los clientes en orden de llegada. Esto mantiene un orden justo de atención.

Esta implementación logra un buen entre eficiencia usando estructuras concurrentes y una lógica fácil de seguir, manteniendo todas las reglas del problema inicial.

Lógica de operación

Identificación del estado compartido (variables y estructuras globales)

En el código de la barbería, se utilizaron varias estructuras de datos y mecanismos de sincronización para gestionar el estado compartido entre los hilos de los clientes y el barbero.

Capacidad de sillas de espera (`CAPACIDAD_MAXIMA_SILLAS_ESPERA`): Determina el número máximo de clientes que pueden esperar en la barbería (3 sillas).

Semáforo `sillasDisponibles`: Controla la cantidad de sillas de espera disponibles.

Semáforo `barberoLibre`: Indica si el barbero está disponible para atender a un cliente (1 = disponible, 0 = ocupado).

Lock `lockParaColaClientes`: Protege el acceso concurrente a la cola de clientes (`clientesEnEspera`).

Lock `lockParaProcesoPago`: Asegura que el proceso de pago no tenga condiciones de carrera.

Cola de clientes `clientesEnEspera`: Mantiene la lista de clientes que están esperando su turno.

Estado `barberoEstaDurmiendo` (volatile boolean): Indica si el barbero está dormido y necesita ser despertado.

Instancia única de Barbero: Implementado con el patrón Singleton para que solo exista un barbero (Tony).

Descripción algorítmica del avance de cada hilo/proceso

Clientes (Cliente implementa Runnable)

Cada cliente llega a la barbería en un momento aleatorio.

Intenta sentarse en una silla de espera (si hay disponible).

Si hay sillas disponibles, se sienta y espera su turno.

Si no hay sillas disponibles, se va.

Cuando llega su turno, el cliente es atendido por el barbero.

Una vez terminado el corte, el cliente realiza el pago.

Finaliza su ejecución.

Barbero (Barbero - Patrón Singleton)

Si no hay clientes, el barbero se duerme.

Cuando hay clientes en espera, el barbero se despierta.

Atiende al primer cliente en la cola.

Corta el cabello del cliente (tiempo aleatorio entre 1.5 y 2.5 segundos).

Espera el pago del cliente.

Limpia su área de trabajo.

Si hay más clientes, repite el proceso; si no, vuelve a dormir.

Gestión de la barbería (Barberia)

Controla la entrada de clientes.

Administra las sillas de espera con un semáforo.

Despierta al barbero si está dormido.

Gestiona la cola de clientes con un Lock.

Ejecuta el servicio de corte de cabello en un hilo separado.

Gestiona el pago del cliente protegiendo la operación con un Lock.

Descripción de la interacción entre ellos (sincronización y comunicación)

En el programa se usan varios mecanismos de sincronización para evitar condiciones de carrera y garantizar un comportamiento ordenado:

Semáforo sillasDisponibles: Controla la cantidad de sillas en la barbería, asegurando que solo se sienten clientes si hay espacio.

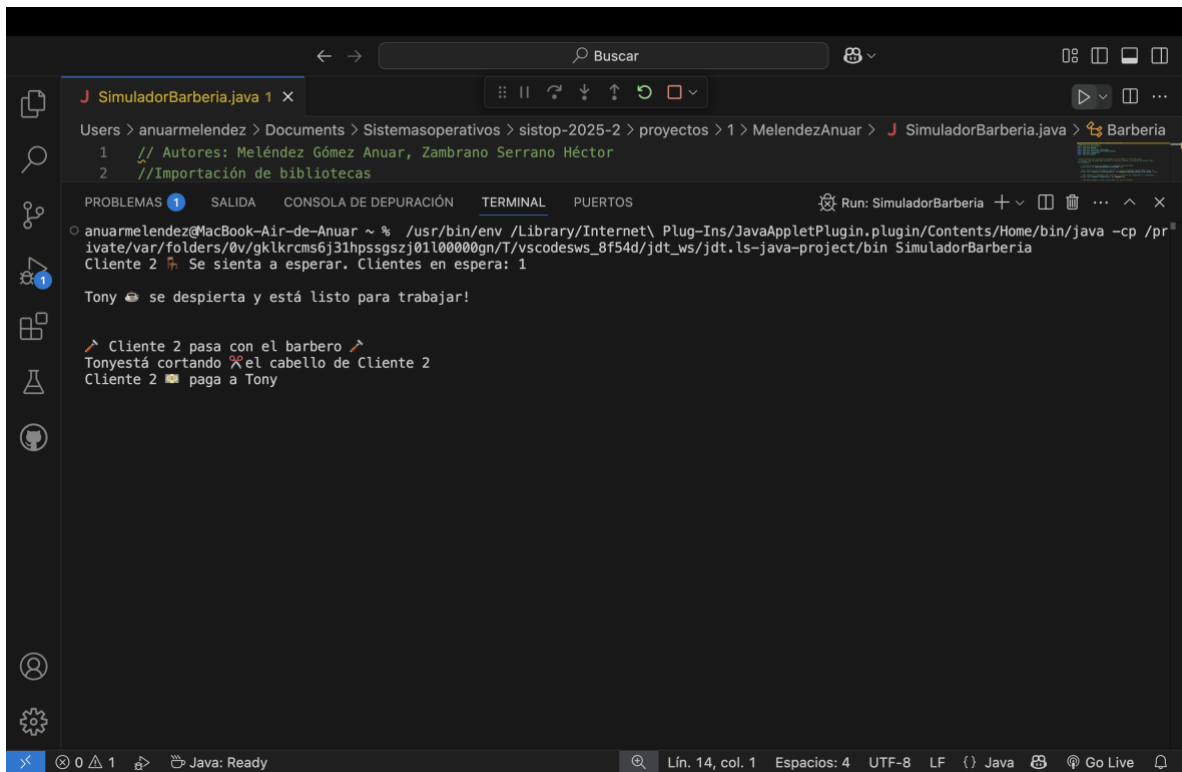
Semáforo barberoLibre: Evita que varios clientes sean atendidos simultáneamente por el mismo barbero.

Lock en clientesEnEspera: Evita condiciones de carrera al modificar la cola de clientes.

Lock en procesarPago: Asegura que el pago sea procesado de manera exclusiva por cada cliente.

Estado barberoEstaDurmiendo con synchronized: Evita que el barbero sea despertado más de una vez innecesariamente.

Capuras de ejecución del programa



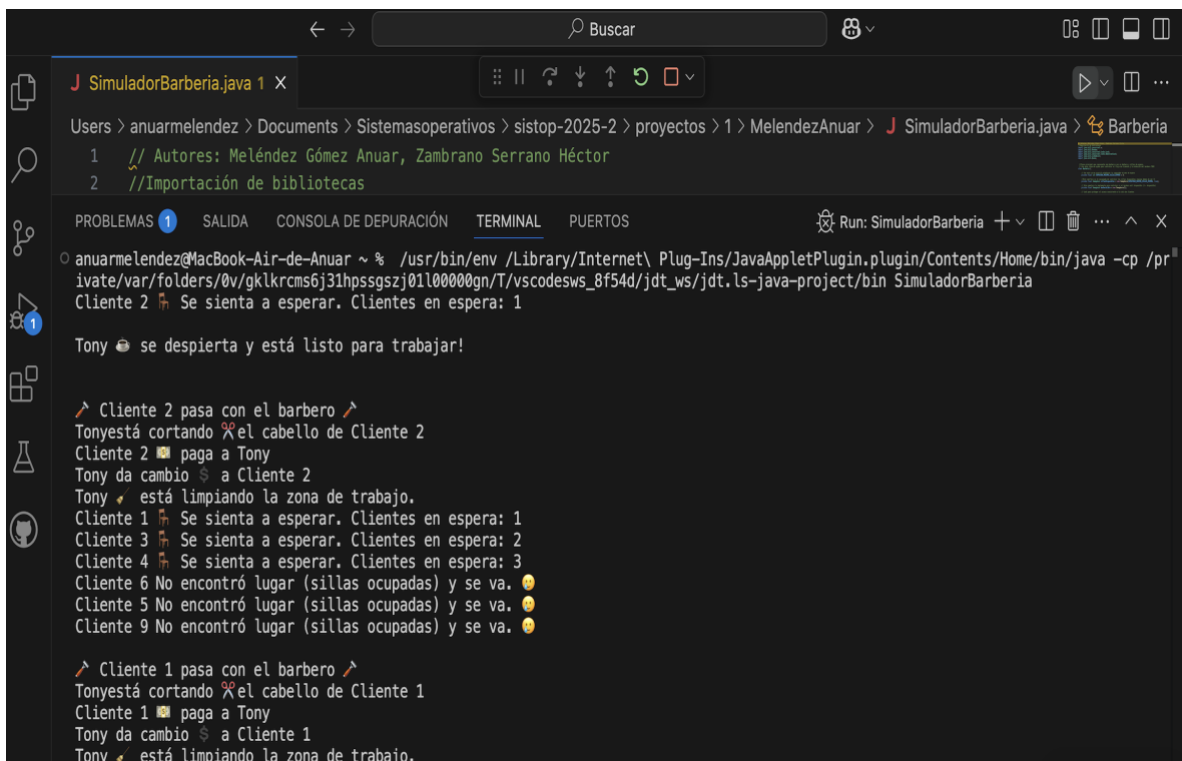
```
Users > anuarmelendez > Documents > Sistemasoperativos > sistop-2025-2 > proyectos > 1 > MelendezAnuar > J SimuladorBarberia.java > Barberia
1 // Autores: Meléndez Gómez Anuar, Zambrano Serrano Héctor
2 //Importación de bibliotecas

PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS Run: SimuladorBarberia
anuarmelendez@MacBook-Air-de-Anuar ~ % /usr/bin/env /Library/Internet\ Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/bin/java -cp /private/var/folders/0v/gklkrms6j3lhpssgszj01l00000gn/T/vscodesws_8f54d/jdt_ws/jdt.ls-java-project/bin SimuladorBarberia
Cliente 2 🕒 Se sienta a esperar. Clientes en espera: 1

Tony 🧑 se despierta y está listo para trabajar!

➡ Cliente 2 pasa con el barbero ➡
Tony está cortando ✂️ el cabello de Cliente 2
Cliente 2 💰 paga a Tony
```

En esta captura el programa inicia su ejecución



```
anuarmelendez@MacBook-Air-de-Anuar ~ % /usr/bin/env /Library/Internet\ Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/bin/java -cp /private/var/folders/0v/gklkrms6j3lhpssgszj01l00000gn/T/vscodesws_8f54d/jdt_ws/jdt.ls-java-project/bin SimuladorBarberia
Cliente 2 🕒 Se sienta a esperar. Clientes en espera: 1

Tony 🧑 se despierta y está listo para trabajar!

➡ Cliente 2 pasa con el barbero ➡
Tony está cortando ✂️ el cabello de Cliente 2
Cliente 2 💰 paga a Tony
Tony da cambio 💵 a Cliente 2
Tony 🧹 está limpiando la zona de trabajo.
Cliente 1 🕒 Se sienta a esperar. Clientes en espera: 1
Cliente 3 🕒 Se sienta a esperar. Clientes en espera: 2
Cliente 4 🕒 Se sienta a esperar. Clientes en espera: 3
Cliente 6 No encontró lugar (sillas ocupadas) y se va. 😞
Cliente 5 No encontró lugar (sillas ocupadas) y se va. 😞
Cliente 9 No encontró lugar (sillas ocupadas) y se va. 😞

➡ Cliente 1 pasa con el barbero ➡
Tony está cortando ✂️ el cabello de Cliente 1
Cliente 1 💰 paga a Tony
Tony da cambio 💵 a Cliente 1
Tony 🧹 está limpiando la zona de trabajo.
```

```
← → 🔍 Buscar ☰ ☷ 📄 🖨️  
PROBLEMAS ❶ SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS ⚙️ Run: SimuladorBarberia + ▢ 🗑️ ... ▾ ✕  
  
Tony 🌟 está limpiando la zona de trabajo.  
Cliente 7 🛋️ Se sienta a esperar. Clientes en espera: 3  
Cliente 8 No encontró lugar (sillas ocupadas) y se va. 😞  
Cliente 10 No encontró lugar (sillas ocupadas) y se va. 😞  
Cliente 9 No encontró lugar (sillas ocupadas) y se va. 😞  
Cliente 12 No encontró lugar (sillas ocupadas) y se va. 😞  
Cliente 11 No encontró lugar (sillas ocupadas) y se va. 😞  
Cliente 13 No encontró lugar (sillas ocupadas) y se va. 😞  
  
🔪 Cliente 2 pasa con el barbero 🔪  
Tony está cortando ✂️ el cabello de Cliente 2  
Cliente 2 💰 paga a Tony  
Tony da cambio 💵 a Cliente 2  
Tony 🌟 está limpiando la zona de trabajo.  
Cliente 14 🛋️ Se sienta a esperar. Clientes en espera: 3  
Cliente 15 No encontró lugar (sillas ocupadas) y se va. 😞  
  
🔪 Cliente 4 pasa con el barbero 🔪  
Tony está cortando ✂️ el cabello de Cliente 4  
Cliente 4 💰 paga a Tony  
Tony da cambio 💵 a Cliente 4  
Tony 🌟 está limpiando la zona de trabajo.  
  
🔪 Cliente 7 pasa con el barbero 🔪  
Tony está cortando ✂️ el cabello de Cliente 7  
Cliente 7 💰 paga a Tony  
Tony da cambio 💵 a Cliente 7  
Tony 🌟 está limpiando la zona de trabajo.  
  
🔪 Cliente 14 pasa con el barbero 🔪  
Tony está cortando ✂️ el cabello de Cliente 14  
Cliente 14 💰 paga a Tony  
Tony da cambio 💵 a Cliente 14  
Tony 🌟 está limpiando la zona de trabajo.  
  
Tony 🌴 está durmiendo... zZzZz
```