Documentacíon Proyecto Cafetería

Birhan Fdez Fdez

October 29, 2025

1. Índice

2. Introducción

Este proyecto tiene como objetivo simular el funcionamiento básico de una cafetería utilizando el lenguaje de programación Java y la programación concurrente mediante *threads*. La simulación modela la interacción entre clientes y camareros, permitiendo estudiar conceptos de concurrencia, sincronización y gestión de recursos compartidos en un entorno realista

3. Requisitos funcionales

A continuación se listan los requisitos funcionales del sistema:

RF-1: El sistema debe permitir crear clientes con un nombre y un tiempo máximo de espera para recibir su café.

RF-2: Cada cliente debe llegar a la cafetería como un hilo independiente que espera ser atendido.

RF-3: El sistema debe permitir crear camareros como hilos independientes que atienden a los clientes en orden de llegada.

RF-4: Cada camarero debe preparar el café de un cliente simulando el tiempo de preparación mediante `Thread.sleep()`.

RF-5: Si el tiempo de preparación excede el tiempo de espera del cliente, el cliente se debe ir sin recibir su café.

RF-6: El sistema debe mantener un registro de los clientes que han recibido su café.

RF-7: Al finalizar la simulación, el sistema debe mostrar un listado de todos los clientes atendidos y un mensaje indicando que todos los clientes han sido procesados.

4. Requisitos no funcionales

RNF-1: El sistema debe ejecutar múltiples clientes y camareros de manera concurrente sin bloqueos innecesarios.

RNF-2: Las listas compartidas (`list_clientes` y `list_atendidos`) deben ser gestionadas de forma segura para hilos concurrentes.

RNF-3: Los mensajes de la consola deben ser claros y comprensibles, indicando qué camarero atiende a qué cliente, el tiempo de preparación y el resultado del pedido.

RNF-4: El sistema debe ser mantenible, con clases separadas para clientes y camareros, permitiendo futuras ampliaciones.

RNF-5: La simulación debe ser portable y ejecutable en cualquier entorno que soporte Java 8 o superior.

RNF-6: El tiempo de respuesta del sistema ante la llegada de un cliente debe ser inmediato, simulando la concurrencia real.

5. Casos de Uso

A continuacion se describe el diagrama general de casos de uso del sistema..

Caso de uso	**Descripción**
Llegada de cliente	Un cliente llega a la cafetería y se agrega a la lista de espera, permaneciendo allí hasta ser atendido o hasta que expire su tiempo de espera.
Atención de cliente	Un camarero atiende al primer cliente en la lista de espera, prepara su café y entrega el pedido si el cliente aún espera.
Preparación del café	El camarero simula la preparación del café usando un tiempo aleatorio; si el café se termina antes de que el cliente se vaya, se entrega el café; de lo contrario, el cliente se va sin él.
Finalización de la simulación	Una vez que todos los clientes han sido atendidos o se han ido, el sistema muestra un mensaje indicando que la simulación ha terminado y lista los clientes que recibieron su café.

Table 1: Casos de uso del sistema de cafetería

6. Historias de usuario

ID	**Como...**	**Quiero...**	**Para...**
HU-01	Cliente	Esperar mi café	Recibir mi pedido a tiempo
HU-02	Camarero	Atender clientes	Entregar cafés eficientemente
HU-03	Cliente	Irme si demora mucho	No perder tiempo
HU-04	Sistema	Registrar cafés entregados	Llevar historial de clientes
HU-05	Sistema	Mostrar resumen al final	Saber qué clientes fueron atendidos

Table 2: Historias de usuario del sistema de cafetería

7. Clases principales

- **cliente**
  Representa a un cliente de la cafetería.
  - **Atributos:**
    - `nombre`: nombre del cliente.
    - `tiempo_espera`: tiempo máximo que el cliente espera su café.
  - Funciona como un objeto simple para almacenar información de cada cliente.

- **camarero**
  Representa a un camarero que atiende a los clientes.
  - **Atributos:**
    - `nombrecama`: nombre del camarero.
    - `list_clientes`: lista de clientes en espera.
    - `list_atendidos`: lista de clientes que recibieron su café.
  - Extiende `Thread` y ejecuta cada camarero como hilo independiente.
  - **Método principal:** `run()`, que retira clientes de la lista, simula la preparación del café con `Thread.sleep()`, y registra si el cliente recibe el café o se va.
  - Gestiona la concurrencia usando `synchronized` sobre las listas compartidas.
- **Main**
  Clase principal que inicializa y ejecuta la simulación.
  - Crea listas de clientes y camareros.
  - Inicia los hilos de los camareros (`start()`).
  - Espera que todos los camareros terminen (`join()`).
  - Muestra al final los clientes que fueron atendidos correctamente.

8. Interfaz

- **Interfaz gráfica con JavaFX:** Muestra secciones para cada camarero con TextAreas que registran la actividad en tiempo real, un título destacado y un botón para iniciar la simulación.
- **Simulación concurrente:** Cada camarero y cliente funciona como un hilo independiente, respetando tiempos de espera y orden de atención.
- **Actualización dinámica de la UI:** Los TextAreas se actualizan en tiempo real mediante `Platform.runLater()`, mostrando mensajes sobre atención, preparación del café y clientes que se van.
- **Gestión de clientes y pedidos:** Listas centralizadas para clientes en espera y atendidos, con un mensaje final indicando qué clientes recibieron su café.
- **Preparación de café con tiempos aleatorios:** Simula variabilidad en el servicio y permite que clientes que esperan demasiado se vayan sin café.
- **Código organizado y escalable:** Separación en clases (`cliente`, `camarero`, `HelloController`) y fácil ampliación de clientes o camareros.