Documentación Cafetería con Threads

por Santi Martínez

October 29, 2025

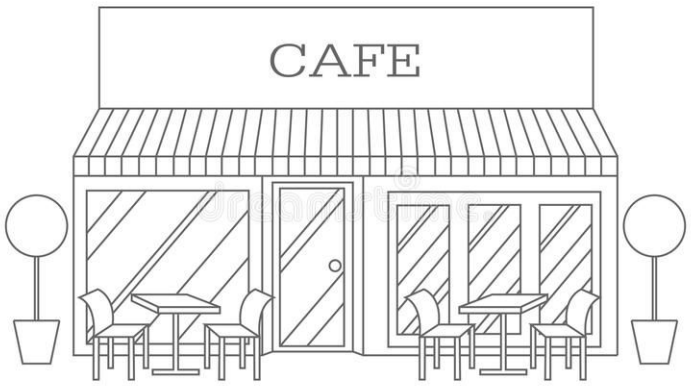
Índice

1 Introducción

Este es un programa en JavaFX en el que la principal tarea es jugar con los hilos (Threads).

Ela programa permite iniciar un turno de atención de clientes, mostrar en tiempo real el estado de cada cliente, actualizar un contador de clientes satisfechos y manejar la interacción con los botones `Iniciar turno` y `Salir`.

Los hilos se utilizan para simular la llegada de los clientes y la preparación de los cafés por los camareros, garantizando que la interfaz siga respondiendo mientras se ejecutan estas tareas en segundo plano.



2 Stack

2.1 Lenguaje de programación

El lenguaje en el que está desarrollado este programa es Java, en concreto la versión 21.0.8. El SDK es el openjdk-25.

2.2 Framework

El framework utilizado por el programa es `JavaFX`, el cual permite crear interfaces atractivas en Java.

Facilita el manejo de eventos y da una vista moderna al programa, y además integra la posibilidad de utilizar CSS, lo que genera una UI más agradable.

2.3 IDE

Elegí `IntelliJ IDEA` para hacer el programa con hilos en Java porque permite ver el comportamiento de múltiples threads en tiempo real, como es el caso del programa, en el que se verá más adelante como se va ejecutando en directo.

3 Casos de uso

1. `Abrir la cafetería`. El usuario inicia el turno con el botón `Iniciar turno`. La aplicación muestra que la cafetería está abierta y lista para recibir clientes.

2. `Atender clientes`. Cada cliente llega y espera su café. Los camareros preparan el café de forma concurrente usando hilos.

3. `Actualizar estado de los clientes`. La interfaz muestra si un cliente se va enfadado o si recibe su café. Se actualiza el contador de `clientes satisfechos` en tiempo real.

4. `Salir de la aplicación`. El usuario puede cerrar la aplicación con el botón `Salir`, que usa por detrás `Platform.exit()`.

5. `Registrar tiempos de espera y servicio`. La app compara el `tiempo de espera del cliente` con el `tiempo de preparación del camarero` y muestra mensajes depende del resultado.

4 Requisitos

Los requisitos son las condiciones que el sistema debe cumplir para satisfacer a los usuarios y funcionar correctamente.

4.1 Requisitos funcionales

Definen lo que el sistema debe de hacer. Se centran en funciones, servicios y comportamientos del programa.

En este caso, ejemplos de casos de uso funcionales serían:

- Iniciar servicio
- Atender a clientes
- Mostrar si el cliente se fue con el caféa tiempo o se fue enfadado
- Contar y mostrar los clientes que se fueron satisfechos
- Salir del programa en el momento que el usuario quiera

4.2 Requisitos no funcionales

Dicen como debe comportarse el sistema ante el programa. Se centra en varios matices, pero por gusto personal los fundamentales son seguridad, rendimiento y portabilidad.

En este caso, ejemplos de casos de uso no funcionales serían:

- La interfaz se actualiza en directo
- La app es estable mientras se procesan clientes
- Funciona enc ualquier plataforma que soporte Java y JavaFX

5 Backend

- Clases principales
- App y Launcher
- Controlador

5.1 Clases principales

- Camarero
- Cliente

5.2 App y Launcher

Sirven para ejecutar toda la App

5.3 Controlador (Flujo del programa)

Es la clase de Java donde se produce todo el backend. A continuación se muestra un breve flujo del mismo:

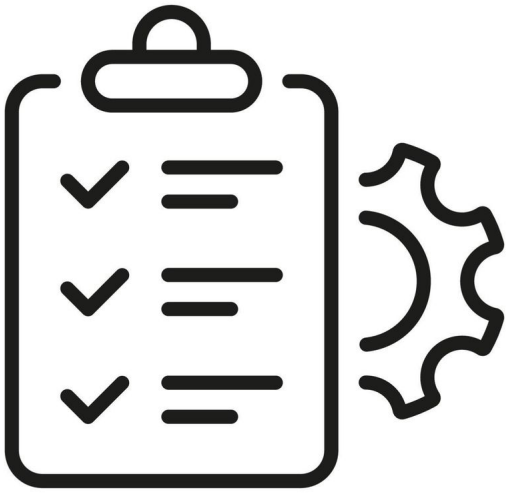
1. Se crean los objetos botones y textarea
2. se crea un método para salir del programa
3. Se crea el método principal, en el cual se realiza resto del programa.
4. Se crea un nuevo hilo, el cual es una función invocada inmediatamente, para que en el momento en que aparece se ejecuta.

5. Dentro del hilo, se crean 5 clientes, se les añade a una lista CLIENTES. Se crean también 2 camareros y se les añade a la lista CAMAREROS.

6. Bucle en el que en cada pasada se inicia el hilo de un cliente y se crea un nuevo Hilo que ejecuta a un camarero, los cuales se saltean entre pasadas entre uno y otro.

7. Antes de finalizar el bucle, se toman los tiempos de espera de camarero y cliente, se comparan, y según resultado se determina un final u otro.

8. Para terminal el flujo se añade al TextArea "Servicio finalizado"



6 Frontend

- Vistas del programa
- main.fxml
- styles.css

6.1 Vistas del programa

Esta es la `vista principal`, la que se observa al iniciar el programa. Tiene dos posibilidades: Iniciar el programa o salir de este.

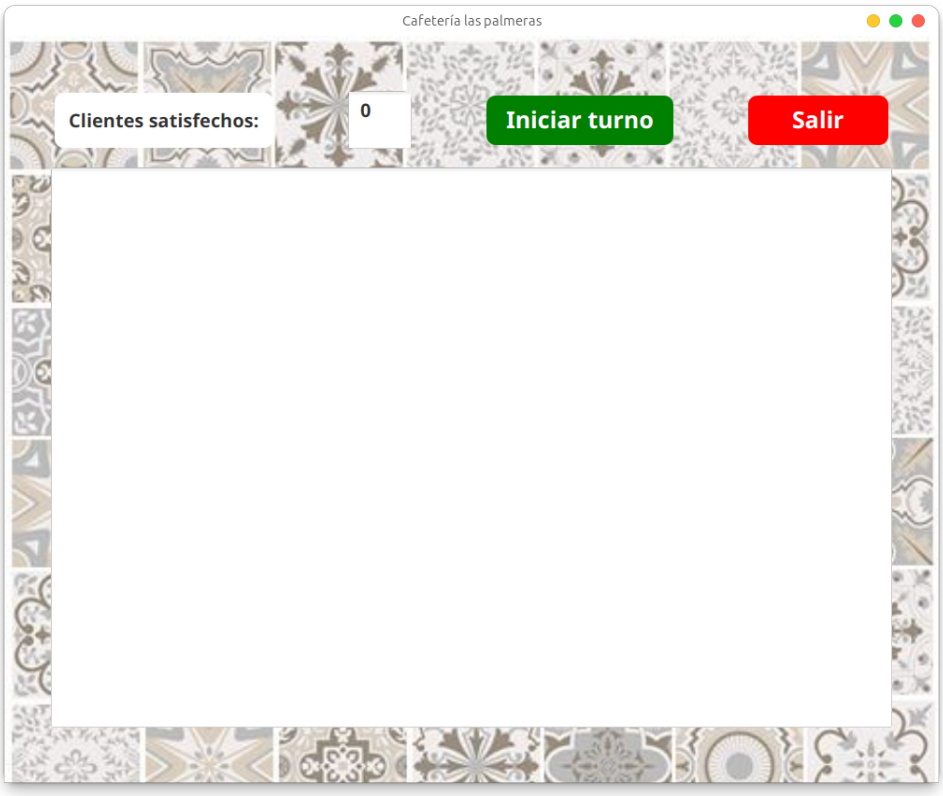


Figure 1: Vista principal del programa

Al clicar en el botón de `Iniciar programa`, ese botón se oculta y aparece la frase que da apertura a la cafetería.

Figure 2: Apertura de la cafetería

Una vez el programa concluye, aparece el texto final `Servicio finalizado`.

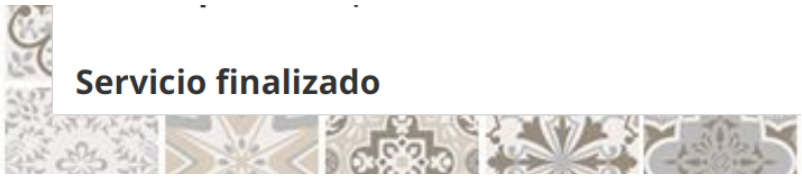


Figure 3: Servicio finalizado

Durante el transcurso del programa, cada vez que un cliente se va satisfecho con el servicio del camarero (tiempos en positivo: espera mayor que preparación), `se suma uno al contador`.

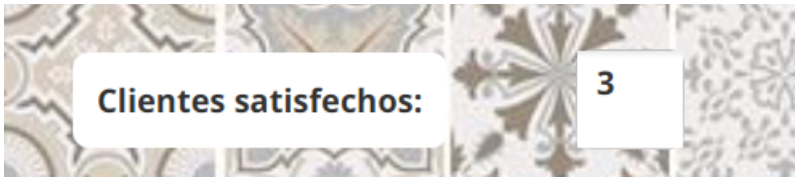


Figure 4: Contador de clientes satisfechos

6.2 main.fxml

Es la parte visible de la app. La conforma una etiqueta VBox, en la cual su interio alberga:

- `Contador` (Clientes satisfechos): Cada vez que un cliente recibe su café a tiempo, este suma uno.
- `Button` (Iniciar turno): Sirve para iniciar el método que se encuentra en el controlador, el cual ejecuta todo el funcionamiento de la app.
- `Button` (Salir): Solo aparece cuando accionas el anterior, sirve para salir de la app.
- `TextArea`: Donde se muestra todo el flujo del programa a tiempo real; Abrir cafetería, camareros preparando el café, cliente esperando, el resultado del servicio y el aviso de servicio finalizado.

6.3 styles.css

El archivo CSS cumple la función de `definir la apariencia visual de la interfaz de usuario (UI)`. En este caso, su contenido es relativamente breve, ya que el programa en cuestión no posee una gran cantidad de elementos o componentes visuales.

A través de este archivo se establecen propiedades como los colores, los tamaños de fuente, los márgenes, los rellenos, la alineación del texto y otros aspectos relacionados con la presentación de los elementos. De esta manera, se logra que la interfaz mantenga una estética uniforme y facilite la lectura y la interacción del usuario con el sistema.