Practica,11,12,13

Rodríguez Alvarez María José

November 2025

1 Marco Teórico

En esta práctica vamos a aplicar tres conceptos muy importantes para el desarrollo de aplicaciones en Java: el manejo de archivos para guardar datos, el uso de hilos para hacer tareas al mismo tiempo y los patrones de diseño para organizar mejor nuestro código.

1.1 Archivos (Persistencia)

Normalmente, cuando cerramos un programa, todas las variables y datos que estaban en la memoria RAM se borran. Para evitar esto y que la información se quede guardada ("persistencia"), usamos archivos.

En Java, esto se maneja principalmente con **Streams** (flujos de datos):

- **Archivos de Texto:** Son los que podemos leer nosotros, como un `.txt`. Usamos clases como `FileReader` o `BufferedReader` para leer línea por línea.
- **Archivos Binarios:** Guardan la información en bytes (unos y ceros). Son útiles para guardar objetos completos o imágenes.
- **Serialización:** Es el proceso de convertir un objeto de Java en una secuencia de bytes para poder guardarlo en un archivo y recuperarlo después tal cual estaba.

1.2 Hilos

Un hilo o *thread* es la unidad básica de ejecución de un programa. Usar hilos nos permite hacer **multitarea**, es decir, que nuestro programa haga dos o más cosas a la vez (por ejemplo, mostrar una interfaz gráfica mientras descarga un archivo de fondo).

Hay dos formas principales de crear hilos en Java:

1. Heredando de la clase `Thread`: Creamos una clase que extienda a Thread y sobrescribimos el método `run()`.

2. Implementando la interfaz `Runnable`: Es la forma más recomendada. Nuestra clase implementa `Runnable` y definimos qué debe hacer el hilo en el método `run()`.

Los hilos tienen un ciclo de vida: nacen, se ejecutan, a veces se pausan (esperan) y finalmente mueren cuando terminan su tarea.

1.3 Patrones de Diseño

Los patrones de diseño son como "recetas" o soluciones estándar para problemas que ocurren muy seguido al programar. Nos ayudan a que el código sea más ordenado y fácil de modificar. Se dividen en tres tipos:

- **Creacionales:** Nos dicen cómo crear objetos de forma eficiente (Ejemplo: *Singleton*, que asegura que solo exista una instancia de una clase).
- **Estructurales:** Nos ayudan a conectar clases y objetos para formar estructuras más grandes (Ejemplo: *Adapter*).
- **De Comportamiento:** Se encargan de la comunicación entre objetos (Ejemplo: *Observer*, que avisa a otros objetos cuando algo cambia).