Biblioteca Digital - Práctica 2

Equipo: Chavalines

- Guzman Ramirez Diego Arturo 321182324
- Castro Cazares Hector Alonso 118002682
- Marcial Ahedo Andrick Iorel 321304074

Descripción del Proyecto

Este proyecto consiste en la implementación de un sistema de biblioteca digital, gestiona préstamos de libros electrónicos, revistas y audiolibros, así como la administración de usuarios y la exploración del catálogo. El sistema utiliza tres patrones de diseño: **State**, **Template** e **Iterator**.

Casos de Uso

- **Caso 1: Estado Moroso**
  Descripción:
    - Un usuario (por ejemplo, "Carlos") inicia en estado **Activo** y puede solicitar préstamos.
    - Si se retrasa en la devolución (simulado mediante la opción "Simular retraso"), el sistema cambia su estado a **Moroso** y deniega nuevas solicitudes hasta que se devuelva el material.
    - Al devolver el material, el usuario vuelve a estar **Activo**.

    Veredicto: Caso 1: **Correcto**.
- **Caso 2: Modalidades de Préstamo y Renovación**
  Descripción:
    - Un usuario (por ejemplo, "Mariana") puede solicitar el préstamo de "Cálculo Avanzado".
    - Si selecciona **Préstamo Normal**, el préstamo tiene una duración de 15 días y ofrece la opción de renovar (de forma interactiva, cada renovación extiende el préstamo en 5 días, hasta 3 veces).
    - Si selecciona **Préstamo Express**, el préstamo dura 7 días sin opción a renovación.

    Veredicto: Caso 2: **Correcto**.
- **Caso 3: Exploración de Materiales con Iterator**
  Descripción:
    - Un usuario (por ejemplo, "Jorge") puede explorar los materiales (libros, revistas, audiolibros) a través de un menú interactivo.

    - El sistema utiliza el patrón **Iterator** para recorrer las colecciones sin importar su estructura interna.
      Veredicto: Caso 3: **Correcto**.
- **Caso 4: Reserva y Notificación**
  Descripción:

    - Un usuario (por ejemplo, "Ana") intenta solicitar "Cien años de soledad", pero el material ya está en préstamo.
    - Entonces, se ofrece la opción de reservar el material.
    - Cuando se devuelve, el sistema notifica automáticamente a Ana que "Cien años de soledad" está disponible.
      Veredicto: Caso 4: **Correcto**.

Patrones Implementados

Patrón State

- **Objetivo:**
  Gestionar el estado del usuario (Activo o Moroso) para controlar la posibilidad de solicitar nuevos préstamos.

- **Funcionamiento:**
  La interfaz `EstadoUsuario` define métodos como `solicitarPrestamo()` y `devolverPrestamo()`, y se implementa en:

    - **Activo.java:** Permite realizar solicitudes de préstamo.
    - **Moroso.java:** Deniega nuevas solicitudes y, al devolver un préstamo, cambia el estado del usuario a Activo.
- **Clase Principal:**

    - **Usuario.java:** Contiene el estado actual del usuario y delega las operaciones a la instancia de `EstadoUsuario`.

Patrón Template

- **Objetivo:**
  Definir una plantilla para el proceso de préstamo, permitiendo personalizar ciertos pasos según el tipo de préstamo.

- **Funcionamiento:**
  La clase abstracta `Prestamo` implementa el método `procesarPrestamo()` que sigue estos pasos:

    - Asigna la fecha de inicio.
    - Calcula la fecha de vencimiento (según el tipo de préstamo).
    - Notifica al usuario.

    - Pregunta de forma interactiva si desea renovar (en el caso del préstamo normal).
- **Clases:**
    - **Prestamo.java:** Clase abstracta que define el flujo de proceso.
    - **PrestamoNormal.java:** Implementa un préstamo normal (15 días, hasta 3 renovaciones de 5 días cada una).
    - **PrestamoExpress.java:** Implementa un préstamo express (7 días sin renovación).

Patrón Iterator

- **Objetivo:**
  Permitir recorrer colecciones de materiales (libros, revistas, audiolibros) sin importar la estructura de almacenamiento.

- **Funcionamiento:**
  Se define una interfaz `IteratorInterface` con métodos básicos `hasNext()` y `next()`.
  Se implementan iteradores específicos:
    - **LibrosIterator.java:** Para recorrer una lista de libros.
    - **RevistasIterator.java:** Para recorrer un arreglo de revistas.
    - **AudiolibrosIterator.java:** Para recorrer una tabla hash de audiolibros (convertida a lista para facilitar la iteración).
- **Clase Agregadora:**
    - **ColeccionMaterial.java:** Contiene las colecciones y ofrece métodos para obtener el iterador correspondiente para cada tipo de material.

Cómo Correr la Práctica

1. Requisitos:

Tener instalado el JDK. Si no lo tienes, instálalo con:

```
sudo apt install default-jdk
```

2. Compilación:

Desde la raíz del proyecto (donde se encuentra la carpeta `src`), ejecuta:

```
javac src/*.java src/state/*.java src/template/*.java
src/iterator/*.java
```

Este comando utiliza el compilador de Java (`javac`) para **compilar** todos los archivos con extensión `.java` que se encuentren en esas rutas.

3. Ejecución:
Ejecuta el programa especificando el directorio `src` en el classpath:

```
java -cp src Main
```

4. Uso del Menú Interactivo:

- Sigue las instrucciones en pantalla para explorar materiales, solicitar préstamos, devolver, reservar y simular retrasos.
- Por ejemplo, para simular un usuario en estado moroso, utiliza la opción "Simular retraso".

¿POR QUE SE IMPLEMENTARON DE ESTA MANERA?

Patrón State (gestión del estado del usuario):

- **Razón de uso:** Se necesitaba manejar el comportamiento del usuario según su estado (Activo o Moroso). Con el patrón State, se encapsulan estos comportamientos en clases específicas (Activo y Moroso) y se evita el uso de condicionales dispersos.
- **Ventaja:** Facilita la extensión (por ejemplo, añadir un nuevo estado en el futuro) y mantiene el código limpio y coherente al delegar las acciones (solicitar o devolver préstamos) directamente a la clase de estado correspondiente.

Patrón Template (diferentes tipos de préstamo):

- **Razón de uso:** Existen distintas modalidades de préstamo (Normal y Express) que comparten un proceso general (asignar fecha, calcular vencimiento, notificar al usuario), pero difieren en algunos pasos (días de vencimiento, renovaciones).
- **Ventaja:** El método plantilla `procesarPrestamo` define un flujo único y deja que las subclases (PrestamoNormal y PrestamoExpress) personalicen los detalles específicos (permitir o no la renovación, calcular la fecha de vencimiento, etc.). Esto evita duplicar código y hace más clara la diferencia entre tipos de préstamo.

Patrón Iterator (recorrido de distintas estructuras de datos):

- **Razón de uso:** El catálogo de la biblioteca almacena libros en listas, revistas en arreglos y audiolibros en tablas hash, y se necesitaba una forma uniforme de recorrer estos materiales sin que el código dependiera de la estructura subyacente.
- **Ventaja:** Cada colección (lista, arreglo, tabla hash) tiene su propio iterador concreto, pero se presenta al usuario mediante una interfaz común (IteratorInterface). De esta forma, se puede recorrer cualquier tipo de colección de materiales con la misma lógica, mejorando la flexibilidad y la mantenibilidad del código.