Fco Javier Fernández Rumí

Herencia y Polimorfismo en Java

Introducción

El presente informe corresponde a la tarea solicitada por el profesor Javier Luque, de la asignatura programación, correspondiente al primer año del ciclo DAM. En este informe se detalla el diseño, implementación y prueba de un proyecto en Java basado en los conceptos de herencia y polimorfismo. Si bien en su día ya se realizó la entrega, consideramos que el proyecto entregado en su momento no cumplía ni siquiera el 30% de las implementaciones demandadas, por lo que se optó por la necesidad de repetirlo.

El objetivo principal del proyecto es demostrar cómo estos conceptos pueden ser utilizados para crear una jerarquía de clases que comparten comportamiento común y, al mismo tiempo, permiten la extensión y la especialización de ese comportamiento a través de subclases, así mismo se solicita aplicar todos los datos y métodos aprendidos hasta el momento.

Diseño del Proyecto

El proyecto consta de tres clases principales y una clase para realizar pruebas:

Superclase: Esta clase, que he definido como abstracta, define el comportamiento común que será compartido por todas las subclases. La he nombrado `DispositivoDigital` y asignado las propiedades privadas `peso`, `precio`, `marca`, `consumo`, así como `id` necesaria para asignar valor a la propiedad estática `nextId`.

Subclases: Son clases que extienden la funcionalidad de la superclase al proporcionar implementaciones específicas de los métodos definidos en ella. En nuestro proyecto, tenemos dos subclases: `Television` y `Lavadora`, que representan dispositivos digitales concretos y proporcionan implementaciones específicas de los métodos `toString()` y `selloEficiencia()`, el cual clasifica con un dato ‘char’ según la eficiencia de cada instancia creada.

Clase `Main`: Esta clase la he creado para probar el funcionamiento de la jerarquía de clases definida.

Implementación

A continuación, se presenta un resumen de la implementación de cada clase:

`DispositivoDigital`:

- Contiene su correspondiente constructor, el método `toString()` y el método abstracto `selloEficiencia()`.
- Se proporcionan métodos getter para los atributos comunes de los dispositivos digitales.

`Television`:

- Extiende `DispositivoDigital` y proporciona implementaciones concretas de los métodos `toString()` y `selloEficiencia()` con operaciones lógicas adaptadas a su clase para calcular la eficiencia y en base a ello, asignarle la categoría correspondiente.
- Incluye las propiedades de clase `pulgadas`, que indica el valor correspondiente al tamaño de la diagonal de la pantalla de la televisión, `altoPix` y `anchoPix`, que refieren a las medidas en pixeles del alto y ancho de la pantalla. Estas propiedades son la base de los cálculos necesarios para el método heredado `selloEficiencia()`.
- Incluye su correspondiente constructor, el método estático `damePixelesTotalesEstaticos(int altoPix, int anchoPix)`, que indica la cantidad total de píxeles del televisor por medio de un String.
- Se ha creado el método `dameResolucion()` que, en función de la cantidad de píxeles de pantalla, se encarga de clasificar el estándar de formato de resolución a través de String.

`Lavadora`:

- También extiende `DispositivoDigital` y proporciona implementaciones concretas de los métodos `toString()` y `selloEficiencia()` con operaciones lógicas adaptadas a su clase para calcular la eficiencia y en base a ello, asignarle la categoría correspondiente.
- Incluye las propiedades de clase `rpm`, que indica el valor correspondiente a las revoluciones por minuto del tambor de la lavadora, `capacidad` que corresponde al valor del peso soportado por el tambor de la lavadora. Estas propiedades son la base de los cálculos necesarios para el método heredado `selloEficiencia()`.
- Incluye su correspondiente constructor y he añadido otro constructor únicamente con los valores `precio` y `marca`, de forma que pueda tomar el resto de valores del otro constructor.
- Incluye el método `funcionSecado(long rpm)` que, en función de la cantidad de rpm que soporte la lavadora y por medio de un operador ternario, nos indica por medio de un String si solo tiene función de lavado o incluye además de secado mediante centrifugado.

`Main`:

- Esta clase se utiliza para probar la funcionalidad de la jerarquía de clases creada. Dispone de método main donde se crean las instancias de subclases y ejecutamos `toString()` en cada instancia para comprobar que funciona correctamente.
- He creado un array y lo hemos completado con las distintas instancias creadas para poder ejecutar sobre dicho array un bucle **for each** que nos devuelva la `marca` y `precio` de cada instancia contenida en el array.
- Por último he generado una lista que, si bien no corresponde a las demandas originales del proyecto, sirve de utilidad para implementar los bucles **while** y **do-while**.

Conclusiones

Siendo consciente de haber podido cumplir con las demandas del proyecto en el momento que se encargó valiéndome de la asistencia de compañeros, considero que en ese momento no manejaba con el mismo nivel de soltura e independencia todos los conceptos que se me demandaba. Cabe señalar que en este nuevo desarrollo me ha asistido mi compañero Roberto Ortiz, orientandome a la hora de encontrarme algunas dudas en determinados conceptos.

El proyecto, realizado en el IDE **Eclipse**, demuestra cómo utilizar los conceptos de herencia y polimorfismo en Java para crear una jerarquía de clases que comparten comportamiento común y permiten la extensión y la especialización de ese comportamiento a través de subclases. Este enfoque facilita la reutilización del código y promueve una estructura modular y mantenible para el desarrollo de software en Java.

Muestra también el funcionamiento de los distintos tipos de **datos primitivos** posibles, incluye métodos **públicos** y **privados**, emplea **this**, **this()**, **super** y **super()**, contiene propiedades y métodos tanto de **instancia** como **estáticos** con operaciones lógicas, contiene los bucles **for-each**, **while** y **do-while**, operadores **ternarios**, de **autoincremento** y **autodecremento**, condicionales **if-else** y **switch-case**.