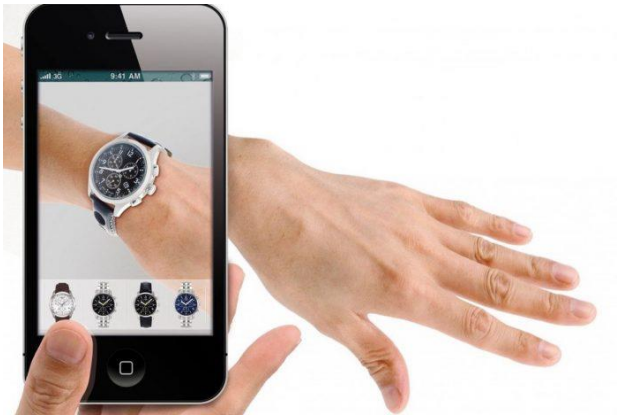
Universidad del Valle de Guatemala
CC2008 - Introducción a la Programación Orientada a Objetos
Semestre II, 2021

Ejercicio 6: Polimorfismo vía interfaces

Problema a resolver



La tienda "Electrónica Latinoamericana" promueve la venta de sus productos a través de una plataforma de AR (*Augmented Reality*). A través de esta plataforma también se pueden hacer ventas, por lo que necesitan su apoyo para modelar y simular las funcionalidades de sus productos y su plataforma.

El listado de productos se muestra a continuación:

1. Smartphones
   a. Hacen llamadas
   b. Toman fotografías
   c. Pueden navegar por internet
   d. Pueden reproducir videos
   e. Son portables
2. Teléfonos Celulares
   a. Hacen llamadas
   b. Son portables
3. Teléfonos fijos
   a. Hacen llamadas
4. Cámaras fotográficas
   a. Toman fotografías
   b. Pueden reproducir videos
   c. Son portables
5. Computador Personal (Desktop)
   a. Pueden navegar por internet
   b. Pueden reproducir videos
   c. Ejecutar Videojuegos
6. Computador Personal (Laptop)
   a. Pueden navegar por internet

   b. Ejecutar Videojuegos
   c. Pueden reproducir videos
   d. Son Portables
7. Smart TV
   a. Pueden navegar por internet
   b. Pueden reproducir videos
8. Tablets
   a. Toman fotografías
   b. Pueden navegar por internet
   c. Pueden reproducir videos
   d. Son portables
9. Smartwatch
   a. Toman fotografías
   b. Son portables
   c. Hacen llamadas

Cada producto posee también información propia como precio, serie, marca, fecha de fabricación y el marcador AR para mostrar el modelo 3D.

También se debe tomar en cuenta que “Electrónica Latinoamericana” cuenta con varias sucursales en varios países, por lo que al momento de realizar una compra a través de la plataforma se debe agregar la información de la tienda más cercana.

La información de las tiendas como dirección, código, país, ciudad, etc. así como también la información de sus productos deberá estar almacenada de manera persistente en la computadora, por lo que debe modelar también las clases involucradas en los mecanismos de lectura / escritura de sus datos.

El programa debe ser capaz de agregar a un “carrito de compras” los productos que desea comprar, se debe tener la posibilidad de probar las funcionalidades, si el cliente lo desea, previo a agregarlo al carrito de compras. Una vez dentro del carrito de compras se deberá poder ordenar los artículos de acuerdo al precio, fecha de fabricación o marca; y también se puede eliminar un elemento del carrito de compras.

Para efectuar el ordenamiento de artículos en su carrito de compras podrá usar algoritmos de ordenamiento existentes (siempre y cuando cite su fuente y adapte el código/algoritmo a las necesidades de este ejercicio). También podrá apoyarse en el uso de la interfaz *Comparable* y la manera en la que Java la usa, para desarrollar el procedimiento de ordenamiento.

Un ejemplo para poder probar la funcionalidad de llamadas de un Smartphone sería:

```
Ingrese el número de teléfono al que desea llamar:
```

31209493

```
Llamando al número 31209493 desde mi SmartPhone Huawei…
```

Y si desea Validar la funcionalidad de navegación de su computadora sería:

```
Ingrese la dirección web que desea visitar
```

www.uvg.edu.gt

```
Visitando la dirección web: www.uvg.edu.gt desde mi Laptop HP…
```

El programa tendrá la capacidad de generar una factura con el detalle de los dispositivos adquiridos mostrando el nombre del cliente, NIT, fecha, número de factura (generado de manera aleatoria) y el monto total de la compra.

Previamente la empresa "NotGoodAtOOP S. A." había iniciado con el desarrollo de la plataforma, por lo que también se le entrega el repositorio que estaban utilizando para que usted pueda clonarlo y evaluar si puede continuar a partir de este código.

https://github.com/malonso-uvg/NotGoodAtOOP.git

Debe entregar en Canvas:

- **[40 pts.] Parte 1 -** Diagrama de clases.
  - **[10 pts.]** Sintaxis y correcto uso de relaciones y modificadores de visibilidad.
  - **[05 pts.]** Identificar las fallas de implementación de la empresa anterior.
  - **[05 pts.]** Correcto uso de herencia en las relaciones entre clases.
  - **[15 pts.]** Correcto uso de polimorfismo a través de interfaces y herencia si es necesario.
  - **[10 pts.]** Correcta separación de responsabilidades y aplicación de MVC y los principios de diseño para el cumplimiento de los requisitos funcionales.
  - **[05 pts.]** Buenas prácticas de programación:
    - *Override* de `toString` (e `equals`, donde sirva).
    - Encapsulación mediante modificadores de visibilidad y *getters/setters*.
- **[50 pts.] Parte 2 -** Programa que implemente su diseño.
  - **[15 pts.]** Correcto uso de *interfaces* (se sugiere implementación de la interfaz *Comparable* para el ordenamiento).
  - **[20 pts.]** Cumplimiento de requisitos funcionales.
  - **[10 pts.]** Usabilidad; interfaz amigable con el o los usuarios.
  - **[05 pts.]** Comentarios y encabezados.
- **[10 pts.] Parte 3 –** Trabajo colaborativo.
  - **[10 pts.]** Trabajar el ejercicio en parejas donde cada integrante deberá aportar la elaboración de clases, dejando evidencia de sus *commits* en Github.