Parcial de Paradigmas de Programación V

Nos solicitan modelar un "***e-commerce***" utilizando nuestros conocimientos en objetos. Nuestro sistema contará con:

Productos

De los productos sabemos su nombre y su precio base. El precio de venta de cada producto está dado por su precio base más los montos que agreguen los modificadores. En este momento solo hay 3 tipos de productos en la aplicación pero en un futuro seguro se agreguen más:

- **Muebles**: tienen un recargo fijo de $1000 en su precio de venta por la complejidad logística que representan.
- **Indumentaria**: no tienen recargos adicionales en su precio.
- **Bebidas Alcohólicas**: solo pueden ser compradas por usuarios mayores de 18 años.

Además de su tipo, los productos pueden tener modificadores especiales que afectan tanto su precio como el costo de envío:

- **Promoción**: reduce el precio de venta del producto en un porcentaje determinado.
- **Pesado**: por su dificultad de manipulación, agrega a su precio de venta un extra por envío en $3000.
- **Tax-Free**: disponible solo para usuarios extranjeros, elimina el agregado de IVA al precio de venta del producto. Todos los productos que no tengan este modificador se agregara el IVA del 21% sobre el precio base.

Un producto puede tener múltiples modificadores simultáneamente. Por ejemplo, podríamos tener un mueble pesado en promoción del 30% que un usuario extranjero compra tax-free: sería un mueble (con su recargo de $1000), pesado (sumando $3000 al envío), en promoción (reduciendo el precio un 30%) y sin IVA.

Usuarios

Los usuarios son los principales actores de nuestro sistema ya que son quienes realizan las compras. Ellos cargan dinero en su cuenta antes de comprar productos. Las compras son premiadas con puntos de lealtad. Conocemos su nombre, edad, su saldo disponible, sus puntos acumulados, su nivel actual, si es extranjero o nacional, y su carrito de productos.

El sistema de niveles funciona de la siguiente manera según los puntos acumulados:

- **Bronce**: es el nivel inicial de todo usuario nuevo. Solo permite tener un producto en el carrito a la vez y no puede tener saldo negativo.
- **Plata**: Se alcanza al acumular 5.000 puntos. Permite tener hasta 5 productos en el carrito y un saldo negativo de $5000.
- **Oro**: Se alcanza con 15.000 puntos. No tiene restricciones en la cantidad de productos en el carrito y un saldo negativo de $20000

El nivel se actualiza automáticamente cuando el usuario alcanza los puntos necesarios, pero también puede descender si pierde puntos por penalizaciones.

Se pide que los usuarios puedan:

- **Agregar un producto a su carrito**, respetando las restricciones de su nivel actual y verificando que cumple con los requisitos del producto (por ejemplo, la edad para bebidas alcohólicas).
- **Cargar saldo en su cuenta** para poder realizar compras.
- **Realizar la compra** de todos los productos de su carrito, lo cual implica:
    - calcular el precio total de los productos (aplicando tax-free si corresponde) más el costo de envío;
    - verificar que tiene saldo suficiente;
    - debitar el monto de su saldo;
    - acreditar puntos equivalentes al 10% del valor pagado;
    - vaciar el carrito.
- **Actualizar nivel:** Luego de cada compra se actualiza el nivel del usuario.

Tienda Virtual

Nuestra aplicación conoce a todos los usuarios registrados y todos los productos disponibles. La tienda debe poder:

- **Gestionar venta:** dado un usuario con productos en su carrito, permite la compra. Si el usuario es menor y lleva bebidas alcohólicas se las retira del carrito y las pone nuevamente en los productos disponibles.
- **Gestionar la morosidad:** aplica descuento de 100 puntos a los usuarios con saldo negativo.

Entregables

1. Diagrama de diseño de clases tipo UML.
    a. Clases que representen todas las entidades pedidas.
    b. Métodos principales que relacionan las clases
    c. Relaciones entre clases y cardinalidad.
    d. Nota: Esquema con detalle mínimo necesario para comprender la solución a codificar.
2. Código: Realizar la implementación del sistema pedido, que resuelva los requerimientos aplicando los patrones de diseño de objetos.
3. Testing Unitario: Aplique el framework de testing automático visto en clase para los requerimientos que lo soliciten.

Requerimientos

1) Probar que un mueble pesado, tax-free y de promoción muestre etiqueta correcta.
2) Probar que que un mueble pesado de $60.000 con un 30% de descuento tenga recargo por ser un mueble (con su recargo de $1000), recargo por envío ($3000),
3) Probar que usuario Bronce con 1 y $5000 de saldo cuando compra una mochila sumaria 1000 puntos y quedaría con saldo en $0.

4) Probar que el negocio aplique correctamente la penalización de morosidad.
5) Probar que usuario Bronce con 1 punto pase a Plata al darle 5000 puntos.
6) Probar que usuario Bronce con saldo de $4999 no puede comprar una mochila de $5000
7) Probar que usuario menor de 18 años no puede comprar una botella de cerveza por más que le alcance su saldo.
8) Probar que un usuario extranjero puede aprovechar el beneficio tax-free (sin IVA).
9) Realizar **un test automático** que valide el siguiente escenario: usuario Bronce con 4.900 puntos realiza una compra por $1.000. Al completar la compra gana 100 puntos, alcanzando exactamente los 5.000 puntos necesarios. Verificar que su nivel se actualiza automáticamente a Plata.
10) Realizar **un test automático** que valide el siguiente escenario: un usuario Bronce **no puede** al mismo tiempo agregar mochila y cartuchera al carrito provocando una Excepción.

Aspectos a considerar

- El programa no debe fallar de forma abrupta (control de excepciones): Si al ejecutar el programa lanza excepción se descuentan 2 puntos.
- Respeto de los principios y estilo de la Programación Orientada a Objetos.
- Debe tener más del 50% de los puntos totales para aprobar.