Laboratorio 3: Herencia

Competencias por desarrollar

- Utilizar los conceptos de herencia para modelar la situación planteada.
- Diseñar la jerarquía de clases utilizando lenguaje UML.
- Identifica los requerimientos funcionales del sistema.
- Identifica los componentes del problema y la solución siguiendo el paradigma de orientación a objetos.
- Elabora el análisis y diseño del programa que debe crear a partir de la situación planteada, describiendo lo requerido para facilitar el desarrollo de un programa que implemente lo analizado y diseñado.
- Codifica en Java lo diseñado.

Ejercicio: Tu outfit ideal

Recientemente un emprendimiento llamado “Tu outfit ideal” ha solicitado apoyo para manejar el control de inventario de prendas dentro del negocio.

En una reunión con los dueños le han pedido desarrollar un sistema que llevara el control prendas que ingresan y están disponibles para la venta, para esto usted debe de tomar en cuenta las siguientes características:

1. Cada **prenda** debe de poseer los siguientes datos **(clase padre):**
    a. Id (un numero aleatorio que no se puede repetir)
    b. Cantidad Disponible
    c. Cantidad Vendidos
    d. Estado (disponible, reservado, vendido)
    e. Precio
2. Dentro del inventario se manejan los siguientes tipos **de prenda (clases hijas):**
    a. **Pantalones**
        i. Talla **(28,30,32,34,36… etc)**
        ii. Tipo (**lona, jogger, pants...etc**)
    b. **Blusa/Camisa**
        i. Talla **(xs, s ,m ,l xl)**
        ii. Color (rojo, azul, morado … etc)
    c. **Collares**
        i. Talla **(xs, s ,m ,l xl)**
        ii. Material (oro, bronce, plata)

Requisitos funcionales:

- Su programa deberá permitir cargar por medio de un archivo csv los productos que posee en inventario utilizando la siguiente estructura:
    - **<id_producto > | <nombre> | <cantidad_disponible> | <cantidad_vendidos> | <estado> | <precio> | <categoria> | <talla> |<tipo> | <color> | <material>**

- Recuerde que no es necesario tener todos los datos dado que cada categoría pose sus propios elementos
- Deberá buscar un producto por medio del id.
- Deberá listar los productos de una categoría
- Deberá ser capaz de mostrar las ventas actuales de la tienda.

Recuerde que estamos aplicando el concepto de herencia por lo cual, su programa tendrá solamente **un ArrayList de la clase padre donde almacenarán las categorías.**

Resultados esperados

Después de cargar el programa mostrará la opción de ver el informe, en donde desplegará:

1. Listado de **categorías** con el total de productos
    a. Pantalones – 100
    b. Blusas – 50
    c. Collares - 25
2. Listado de productos por categoría (ejemplo : pantalones)
    a. Jogger
    b. Lona
    c. Pants
3. Total, de ventas por categoría
    a. Pantalones: Q3000
    b. Blusas: Q1000
    c. Collares: Q500

Material para entregar en canvas:

- Documento .pdf con el análisis.
- Archivo de imagen (.png, .jpg) con el diagrama de clases UML.
- Código fuente de la aplicación desarrollada (todos los archivos **.java**).
- Archivos de ejemplo utilizados para probar el funcionamiento correcto de la aplicación.
- Video explicativo del uso de la función (subido a youtube o similar).

Notas:

- Si el código no coincide suficientemente con el análisis y diseño, a criterio del calificador o calificadora, no se calificará.
- Si el programa no se ejecuta correctamente no se puede calificar. Asegúrese de que el programa corre (sin dependencias de IDE's u otro *software*) antes de enviarlo.

Evaluación:

- **(30 puntos)** Análisis:
  - **(5 puntos)** Se listan correctamente todos los requisitos funcionales del sistema a desarrollar, punto por punto.
  - **(5 puntos)** Se describe el propósito de cada una de las clases identificadas. La descripción es lógica y coherente con lo que hace la clase en el sistema que modela y resuelve el problema presentado.
  - **(10 puntos)** Se describe el propósito de cada atributo de cada clase. La descripción es suficiente para comprender la razón de la creación del atributo y su importancia como propiedad de las instancias (o clase).
  - **(10 puntos)** Se describe el propósito de la creación de cada método de cada clase. La descripción es suficiente para comprender qué acciones realizará cada método, qué información requerirá para funcionar y que resultados producirá.
- **(30 puntos)** Diagrama de Clases UML:
  - **(8 puntos)** La representación de las clases coincide con el estándar del lenguaje UML.
  - **(12 puntos)** Están representados cada uno de los componentes de la clase de manera correcta y coinciden con lo descrito en el análisis.
  - **(10 puntos)** Las relaciones entre las clases están correctas.

- **(40 puntos)** Solución al problema y código fuente:
  - **(16 puntos)** La cantidad de clases identificadas es suficiente para resolver el problema planteado.
  - **(17 puntos)** El problema se resolvió empleando programación orientada a objetos. Se almacenó la información necesaria creando y llenando objetos de las clases identificadas.
  - **(7 puntos)** El código fuente está suficientemente comentado para comprender que se hace, incluyendo el encabezado con la información del creador, descripción del programa y crédito a toda fuente de información que haya aportado al desarrollo del programa. Preferiblemente, que también incluya fecha de creación y fecha de última modificación.