**Enunciado Tema I - Zapatería**

La zapatería "Zapatos Felices" necesita un programa para gestionar su inventario de zapatos. Se requiere implementar un sistema en Java que utilice conceptos de colecciones, polimorfismo, ordenamiento y manejo de excepciones para administrar los diferentes tipos de zapatos que ofrece la tienda.

* Crea una clase abstracta llamada **Zapato** con los siguientes atributos privados:
  + **nombre** (String): el nombre del zapato.
  + **precio** (double): el precio del zapato.
  + ***stock*** (int): la cantidad de zapatos en stock.
* Esta clase debe tener un único constructor que reciba los valores para los atributos y los inicialice. Además, implementa los siguientes métodos:
  + **getters y setters**
  + **mostrarInformacion(): void** (no es el método tostring) muestra en pantalla el nombre, precio y stock del zapato.
  + crear el método abstracto **void** **calcularPrecio()**
* Crea tres clases derivadas de **Zapato: Zapatilla, Sandalia y Bota**. Cada clase derivada debe tener atributos adicionales que sean específicos del tipo de zapato. Por ejemplo, **Zapatilla** puede tener un atributo marca, Sandalia puede tener un atributo estilo, y **Bota** puede tener un atributo talla.
* Implementar el método **calcularPrecio(),** que calcula el precio final incrementando el atributo precio, según corresponda: a la zapatilla le incrementa 20%, a la sandalia le adiciona 10% y a la bota 5%.
* Asegúrate de que cada clase derivada tenga un constructor que reciba los valores para los atributos y los inicialice correctamente. Además, cada clase derivada debe sobrescribir el método **mostrarInformacion()** para mostrar la información específica del tipo de zapato.
* Crear una clase **Zapateria** que declare una colección (ArrayList) llamada inventari**o** para almacenar los zapatos en stock. Se debe incluir el método **void** **agregarZapato(Zapato zapato),** también debe incluir un método **mostrarInventario** que muestre la colección con un forEach.
* En el método main(), crear tres objetos tipo Zapato, utilizando los constructores de las clases derivadas. Asegúrate de proporcionar valores para todos los atributos necesarios.
* Agrega un método llamado **ordenarPorPrecio()** en la clase **Zapateria** que ordena la colección inventario en base al precio de los zapatos, de forma ascendente.
* (Nota:tenga en cuenta que las clases envoltorio implementan compareTo)
* Crear una interfaz Oferta que incluya el **método void calcularDescuento(int porcentaje)** que setea el descuento en precio según el porcentaje. Hacer que las zapatillas implementen esta interfaz.
* Tenga en cuenta la siguiente clase Main

|  |
| --- |
| public class Main {  public static Zapato pedirDatos(String tipo){ *//propagar execpciones*  Zapato zapato=null;  Scanner leer=new Scanner (System.in);  System.out.println("Ingrese nombre: ");  String nombre=leer.nextLine();  System.out.println("Ingrese precio: ");  double precio=leer.nextDouble();  leer.nextLine();  System.out.println("Ingrese stock: ");  int stock=leer.nextInt();  leer.nextLine();  switch (tipo){  case "Zapatilla": System.out.println("Ingrese marca: ");  String marca=leer.nextLine();  zapato=new Zapatilla(nombre,precio,stock,marca);  break;  case "Sandalia": System.out.println("Ingrese estilo: ");  String estilo=leer.nextLine();  zapato=new Sandalia(nombre,precio,stock,estilo);  break;    case "Bota": System.out.println("Ingrese talla: ");  int talla=leer.nextInt();  leer.nextLine();  zapato=new Bota(nombre,precio,stock,talla);  break;    }  return zapato;  }  public static void main(String[] args) {  Zapateria zapateria = new Zapateria();  *//controlar excepciones como dice el enunciado*  Zapato zapatilla = pedirDatos("Zapatilla");  *//agregar zapatilla a la coleccion*  Zapato sandalia = pedirDatos("Sandalia");  *//agregar sandalia a la coleccion*  Zapato bota = pedirDatos("Bota");  *//agregar bota a la coleccion*  *//calcular 15% de descuento a zapatilla*  *//ordenar colección*  *//mostrar coleccion*  }  } |

* se pide modificar la clase Main para que:
* - Modificar el método static Zapato pedirDatos(String tipo) para que **propague excepciones** del tipo InputMismatchException y Exception.
* 1- Dentro del método main(), se debe agregar un bloque try-catch-finally al crear un nuevo zapato. en caso de que se produzca una excepción y proporcionar información adicional sobre el problema:
* -Si ocurre una excepción tipo InputMismatchException se debe mostrar el mensaje “Error de tipo de datos”, si ocurre una excepción Exception, se debe mostrar un cartel “Error general”. El bloque finally debe mostrar un cartel “En bloque finally”
* 2- agregar los zapatos creados a la colección.
* 3-calcular descuento del objeto zapatilla.
* 3- ordenar la colección.
* 4- Mostrar la información de cada zapato usando el método de la clase Zapatería .
* Enunciado Tema II - Ferretería
* La ferretería "Herramientas Útiles" necesita desarrollar un sistema para gestionar su inventario de herramientas. Se requiere implementar un programa en Java que utilice conceptos de clases, herencia, interfaces y manejo de excepciones para administrar los diferentes tipos de herramientas que ofrece la tienda.
* Crea una clase base llamada "Herramienta" con los siguientes atributos privados:
* nombre (String): el nombre de la herramienta.
* precio (double): el precio de la herramienta.
* stock (int): la cantidad de herramientas en stock.
* Esta clase debe tener un constructor que reciba los valores para los atributos y los inicialice. Además, implementa los siguientes métodos:
* getters y setters
* mostrarInformacion(): muestra en pantalla el nombre, precio y stock de la herramienta.
* Crea tres clases derivadas de "Herramienta": "Destornillador", "Sierra" y "Martillo". Cada clase derivada debe tener atributos adicionales que sean específicos del tipo de herramienta. Por ejemplo, "Destornillador" puede tener un atributo "tamaño", "Sierra" puede tener un atributo "material", y "Martillo" puede tener un atributo "peso". Asegúrate de que cada clase derivada tenga un constructor que reciba los valores para los atributos y los inicialice correctamente. Además, cada clase derivada debe sobrescribir el método "mostrarInformacion()" para mostrar la información específica del tipo de herramienta.
* Crea una clase "Ferreteria" que declare una colección (ArrayList) llamada "inventario" para almacenar las herramientas disponibles en stock. Se debe incluir el método void agregarHerramienta(Herramienta herramienta), también debe incluir un método void mostrarInventario() que muestre la información de todas las herramientas en el inventario utilizando un bucle "for each".
* Agrega un método llamado aplicarDescuento() en la clase Ferreteria que aplique un descuento del 10% al precio de todas las herramientas en el inventario.
* Crea una interfaz "Oferta" que incluya el método void calcularDescuento(int porcentaje) que setea el descuento en precio según el porcentaje. Hacer que las herramientas tipo Martillo implementen esta interfaz.
* Modifica la clase Main siguiendo estas pautas:
* Modifica el método static Herramienta pedirDatos(String tipo) para que propague excepciones del tipo InputMismatchException y Exception.
* Dentro del método main(), agrega un bloque try-catch-finally al crear un nuevo Herramienta. En caso de que se produzca una excepción, proporciona información adicional sobre el problema:
* Si ocurre una excepción tipo InputMismatchException, muestra el mensaje "Error de tipo de datos".
* Si ocurre una excepción Exception, muestra un cartel "Error general".
* El bloque finally debe mostrar un cartel "En bloque finally".
* Agrega los objetos de herramientas creados a la colección en la ferretería.
* Calcula el descuento del 10% para todas las herramientas en la colección.
* Muestra la información de cada herramienta utilizando el método de la clase Ferreteria.
* Recuerda seguir buenas prácticas de programación, como encapsulamiento y modularidad, y manejar adecuadamente las excepciones.