Taller Integrado - POO en Java: Empresa de Maletas

# 1. Contexto del Proyecto

En una empresa de ventas de maletas, se desea crear un sistema que permita gestionar la información de empleados y productos, así como realizar consultas y ventas.  
Se deben aplicar los conceptos fundamentales de la Programación Orientada a Objetos: herencia, polimorfismo, encapsulamiento, sobrecarga y sobrescritura.

# 2. Requerimientos Funcionales

- Existen empleados con diferentes roles: Cajero, Vendedor, Administrador, Gerente y Personal de Limpieza.  
- Se venden productos como Mochilas, Bolsos, Maletas de Viaje y Loncheras.  
- El cliente puede consultar el precio de un producto.  
- El vendedor y cajero pueden consultar el detalle completo del producto.  
- Solo el cajero puede realizar ventas.  
- Las mochilas tienen 10% de descuento y las maletas de viaje 20%.  
- A todo subtotal se le aplica un IVA del 15%.

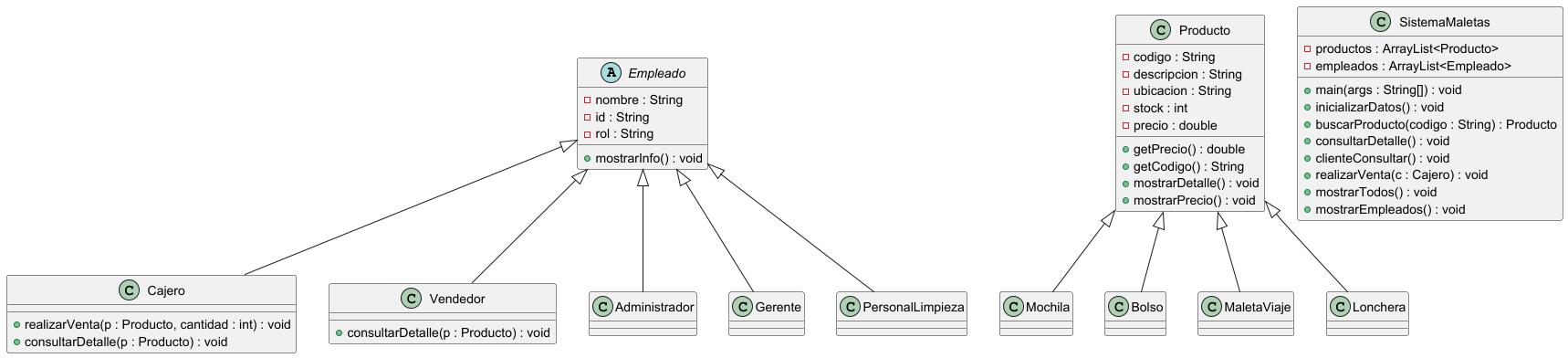
# 3. Clases e Identificación de POO

- Herencia: Las clases como Cajero, Vendedor, etc., heredan de la superclase Empleado.  
- Encapsulamiento: Los atributos de Producto están privados y se accede mediante getters.  
- Polimorfismo: Se usan listas de tipo padre (Producto o Empleado) que contienen objetos de clases hijas.  
- Sobrecarga: Puede implementarse agregando métodos con el mismo nombre y distintos parámetros.  
- Sobrescritura: El método realizarVenta() puede redefinirse en una subclase si cambia su comportamiento.

# 4. Menú del Sistema

1. Consultar detalle del producto (Cajero/Vendedor)  
2. Consultar precio (Cliente)  
3. Realizar venta (Cajero)  
4. Ver todos los productos  
5. Ver empleados registrados  
0. Salir

# 5. Diagrama UML - completar



# 6. Código Java Comentado

A continuación se presenta el código fuente completo del sistema.

// EMPRESA DE VENTA DE MALETAS - POO EN JAVA (ACTUALIZADO)

import java.util.\*;

// SUPERCLASE: Empleado

abstract class Empleado {

protected String nombre;

protected String id;

protected String rol;

public Empleado(String nombre, String id, String rol) {

this.nombre = nombre;

this.id = id;

this.rol = rol;

}

public void mostrarInfo() {

System.out.println("Empleado: " + nombre + " | Rol: " + rol);

}

}

// SUBCLASES

class Cajero extends Empleado {

public Cajero(String nombre, String id) {

super(nombre, id, "Cajero");

}

public void realizarVenta(Producto p, int cantidad) {

double subtotal = p.getPrecio() \* cantidad;

if (p instanceof Mochila) subtotal \*= 0.90;

if (p instanceof MaletaViaje) subtotal \*= 0.80;

double total = subtotal \* 1.15;

System.out.println("Venta realizada. Total a pagar: $" + total);

}

public void consultarDetalle(Producto p) {

p.mostrarDetalle();

}

}

class Vendedor extends Empleado {

public Vendedor(String nombre, String id) {

super(nombre, id, "Vendedor");

}

public void consultarDetalle(Producto p) {

p.mostrarDetalle();

}

}

class Administrador extends Empleado {

public Administrador(String nombre, String id) {

super(nombre, id, "Administrador");

}

}

class Gerente extends Empleado {

public Gerente(String nombre, String id) {

super(nombre, id, "Gerente");

}

}

class PersonalLimpieza extends Empleado {

public PersonalLimpieza(String nombre, String id) {

super(nombre, id, "Limpieza");

}

}

// SUPERCLASE PRODUCTO

class Producto {

private String codigo;

private String descripcion;

private String ubicacion;

private int stock;

private double precio;

public Producto(String codigo, String descripcion, String ubicacion, int stock, double precio) {

this.codigo = codigo;

this.descripcion = descripcion;

this.ubicacion = ubicacion;

this.stock = stock;

this.precio = precio;

}

public double getPrecio() {

return precio;

}

public String getCodigo() {

return codigo;

}

public void mostrarDetalle() {

System.out.println("\nCódigo: " + codigo);

System.out.println("Descripción: " + descripcion);

System.out.println("Ubicación: " + ubicacion);

System.out.println("Stock: " + stock);

System.out.println("Precio: $" + precio);

}

public void mostrarPrecio() {

System.out.println("\nPrecio del producto: $" + precio);

}

}

// SUBCLASES DE PRODUCTO

class Mochila extends Producto {

public Mochila(String codigo, String descripcion, String ubicacion, int stock, double precio) {

super(codigo, descripcion, ubicacion, stock, precio);

}

}

class Bolso extends Producto {

public Bolso(String codigo, String descripcion, String ubicacion, int stock, double precio) {

super(codigo, descripcion, ubicacion, stock, precio);

}

}

class MaletaViaje extends Producto {

public MaletaViaje(String codigo, String descripcion, String ubicacion, int stock, double precio) {

super(codigo, descripcion, ubicacion, stock, precio);

}

}

class Lonchera extends Producto {

public Lonchera(String codigo, String descripcion, String ubicacion, int stock, double precio) {

super(codigo, descripcion, ubicacion, stock, precio);

}

}

// CLASE PRINCIPAL CON ARRAY Y MENU

public class SistemaMaletas {

static ArrayList<Producto> productos = new ArrayList<>();

static ArrayList<Empleado> empleados = new ArrayList<>();

static Scanner sc = new Scanner(System.in);

public static void main(String[] args) {

inicializarDatos();

Cajero cajero1 = (Cajero) empleados.get(0);

Cajero cajero2 = (Cajero) empleados.get(1);

Vendedor vendedor1 = (Vendedor) empleados.get(2);

int opcion;

do {

System.out.println("\n--- MENÚ PRINCIPAL ---");

System.out.println("1. Consultar detalle del producto (Cajero/Vendedor)");

System.out.println("2. Consultar precio (Cliente)");

System.out.println("3. Realizar venta (Cajero)");

System.out.println("4. Ver todos los productos");

System.out.println("5. Ver empleados registrados");

System.out.println("0. Salir");

System.out.print("Seleccione una opción: ");

opcion = sc.nextInt();

switch(opcion) {

case 1 -> consultarDetalle();

case 2 -> clienteConsultar();

case 3 -> realizarVenta(cajero1);

case 4 -> mostrarTodos();

case 5 -> mostrarEmpleados();

}

} while(opcion != 0);

}

static void inicializarDatos() {

// Productos

productos.add(new Mochila("M01", "Mochila Escolar", "Estante A", 10, 20.5));

productos.add(new Bolso("B01", "Bolso Mujer", "Estante B", 5, 35.0));

productos.add(new MaletaViaje("MV01", "Maleta de Viaje", "Estante C", 3, 60.0));

productos.add(new Lonchera("L01", "Lonchera Niños", "Estante D", 12, 15.0));

// Empleados

empleados.add(new Cajero("Carlos", "C01"));

empleados.add(new Cajero("Laura", "C02"));

empleados.add(new Vendedor("Luis", "V01"));

empleados.add(new Vendedor("Ana", "V02"));

empleados.add(new Vendedor("Mario", "V03"));

empleados.add(new Administrador("Lucia", "A01"));

empleados.add(new Gerente("Raul", "G01"));

empleados.add(new PersonalLimpieza("Pablo", "L01"));

empleados.add(new PersonalLimpieza("Sofia", "L02"));

}

static Producto buscarProducto(String codigo) {

for (Producto p : productos)

if (p.getCodigo().equalsIgnoreCase(codigo))

return p;

return null;

}

static void consultarDetalle() {

System.out.print("Ingrese el código del producto: ");

String cod = sc.next();

Producto p = buscarProducto(cod);

if (p != null) p.mostrarDetalle();

else System.out.println("Producto no encontrado.");

}

static void clienteConsultar() {

System.out.print("Ingrese el código del producto: ");

String cod = sc.next();

Producto p = buscarProducto(cod);

if (p != null) p.mostrarPrecio();

else System.out.println("Producto no encontrado.");

}

static void realizarVenta(Cajero c) {

System.out.print("Ingrese el código del producto: ");

String cod = sc.next();

Producto p = buscarProducto(cod);

if (p != null) {

System.out.print("Cantidad a comprar: ");

int cant = sc.nextInt();

c.realizarVenta(p, cant);

} else {

System.out.println("Producto no encontrado.");

}

}

static void mostrarTodos() {

for (Producto p : productos) p.mostrarDetalle();

}

static void mostrarEmpleados() {

for (Empleado e : empleados) e.mostrarInfo();

}

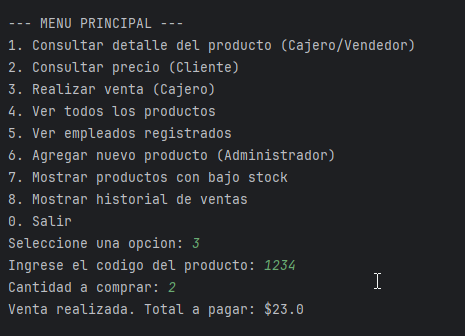
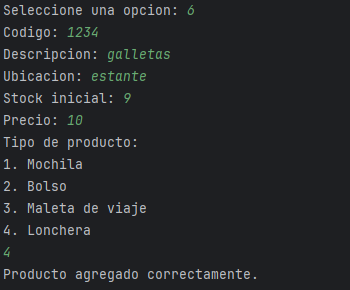
}

# 7. Actividades Adicionales

1. Implementar control de stock al realizar la venta (disminuir cantidad).  
2. Permitir al administrador agregar productos nuevos desde consola.  
3. Mostrar productos con stock menor a 3 unidades.  
4. Agregar opción de registrar una venta con fecha y mostrar historial de ventas.  
5. Aplicar sobrecarga en algún método (ej. consultar con más parámetros).

# 8. Instrucciones para la entrega

- El estudiante debe entregar el archivo .java funcionando.  
- Capturas de pantalla mostrando el funcionamiento del sistema.

  
- Documento Word con respuestas a:

¿Qué es herencia?

Herencia es un principio de la programación orientada a objetos que permite a una clase (subclase) heredar atributos y métodos de otra clase (superclase)

¿Dónde hay polimorfismo?

Hay polimorfismo de inclusión porque se usan referencias de tipo Empleado que apuntan a objetos de distintas clases hijas:

static ArrayList<Empleado> empleados = new ArrayList<>();

¿Qué aprendiste?  
Cómo organizar un sistema con jerarquía de clases, usando herencia para evitar repetir código común.

Cómo encapsular lógica relacionada a productos y empleados en clases distintas.

Cómo aplicar descuentos y reglas específicas según el tipo de producto.

Cómo usar colecciones con polimorfismo: manejar diferentes tipos de empleados en una sola lista.