

Informe

Nombre: Robinson Josué Romero Nolivos

NRC: 1322

Se declara la clase “Cliente”, que es una plantilla para crear objetos que representan a un cliente en el sistema.

```
1 class Cliente {
```

Variables de Instancia

```
private int idCliente;  
public String nombre;  
public String email;
```

“private int idCliente”: Esta es una variable de instancia privada que almacena el identificador único del cliente. Al ser privada, no se puede acceder directamente desde fuera de la clase, lo que ayuda a mantener la encapsulación.

“public String nombre”: Esta es una variable de instancia pública que almacena el nombre del cliente. Al ser pública, puede ser accedida y modificada directamente desde fuera de la clase.

“public String email”: Similar a “nombre”, esta variable pública almacena el correo electrónico del cliente.

Constructor de la Clase

```
public Cliente(int idCliente, String nombre, String email) {  
    this.idCliente = idCliente;  
    this.nombre = nombre;  
    this.email = email;  
}
```

Este es el constructor de la clase “Cliente”. Se utiliza para inicializar las variables de instancia cuando se crea un nuevo objeto de tipo “Cliente”.

Métodos de la Clase

```
public void registrarse() {  
    System.out.println(nombre + " se ha registrado.");  
}  
  
public void iniciarSesion() {  
    System.out.println(nombre + " ha iniciado sesión.");  
}  
  
public void cerrarSesion() {  
    System.out.println(nombre + " ha cerrado sesión.");  
}  
  
public int getIdCliente() {  
    return idCliente;  
}
```

Se declara la clase “Producto”, que representa un producto en un sistema de gestión de inventarios o ventas.

```
1 class Producto {  
    :  
    :
```

Variables de Instancia

```
private int idProducto;  
public String nombre;  
public double precio;  
public int stock;
```

idProducto (int): Un identificador único para cada producto. Este atributo es privado, lo que significa que no puede ser accedido directamente desde fuera de la clase.

nombre (String): El nombre del producto. Este atributo es público, lo que permite su acceso y modificación desde otras clases.

precio (double): El precio del producto. También es un atributo público y puede ser accedido y modificado externamente.

stock (int): Cantidad de unidades disponibles del producto. Este atributo es público y puede ser modificado desde otras partes del programa.

Constructor de la Clase

```
public Producto(int idProducto, String nombre, double precio, int stock) {  
    this.idProducto = idProducto;  
    this.nombre = nombre;  
    this.precio = precio;  
    this.stock = stock;  
}
```

Métodos de la Clase

```
public void agregarStock(int cantidad) {  
    stock += cantidad;  
    System.out.println("Se han agregado " + cantidad + " unidades de " + nombre + ". Stock actual: " + stock);  
}  
  
public void reducirStock(int cantidad) {  
    if (stock >= cantidad) {  
        stock -= cantidad;  
        System.out.println("Se han reducido " + cantidad + " unidades de " + nombre + ". Stock actual: " + stock);  
    } else {  
        System.out.println("No hay suficiente stock de " + nombre);  
    }  
}  
  
public double getPrecio() {  
    return precio;  
}  
  
public String getnombre() {  
    return nombre;  
}
```

Se declara la clase “Pedido”, que representa un pedido de productos en un sistema de compras.

```
import java.util.ArrayList;  
import java.util.List;
```

```
class Pedido {
```

Variables de Instancia

```
-----  
    private int idPedido;  
    private List<Producto> listaProductos;  
    private double total;
```

idPedido: un entero que representa el identificador único del pedido.

listaProductos: una lista que almacena objetos de tipo Producto, representando los productos que se han agregado al pedido.

total: un valor de tipo double que almacena el total acumulado del precio de los productos en el pedido.

Constructor de la clase

```
public Pedido(int idPedido) {  
    this.idPedido = idPedido;  
    this.listaProductos = new ArrayList<>();  
    this.total = 0.0;  
}
```

Métodos de la clase:

```
public void agregarProducto(Producto producto) {  
    listaProductos.add(producto);  
    total += producto.getPrecio();  
    System.out.println("Producto " + producto.getnombre() + " agregado al carrito. Total: " + total);  
}  
  
public void eliminarProducto(Producto producto) {  
    if (listaProductos.remove(producto)) {  
        total -= producto.getPrecio();  
        System.out.println("Producto " + producto.getnombre() + " eliminado del carrito. Total: " + total);  
    } else {  
        System.out.println("El producto no está en el carrito.");  
    }  
}  
  
public double calcularTotal() {  
    return total;  
}  
  
public List<Producto> getListaProductos() {  
    return listaProductos;  
}
```

Se declara la clase “Pago”, que es una representación simple de un sistema de pago.

```
1 class Pago {  
    .  
    .  
    .
```

Variables de Instancia

```
private int idPago;  
private String metodoPago;  
private double monto;
```

idPago: un entero que representa el identificador único del pago. Este atributo es privado, lo que significa que no puede ser accedido directamente desde fuera de la clase.

metodoPago: una cadena que indica el método utilizado para realizar el pago (por ejemplo, tarjeta de crédito, PayPal, etc.).

monto: un valor de tipo doble que representa la cantidad de dinero a pagar.

Constructor de la clase

```
public Pago(int idPago, String metodoPago, double monto) {  
    this.idPago = idPago;  
    this.metodoPago = metodoPago;  
    this.monto = monto;  
}
```

Métodos de la clase

```
public void procesarPago() {  
    System.out.println("Procesando pago #" + idPago + "de " + monto + " dólares con el método " + metodoPago + ".");  
}  
  
public void confirmarPago() {  
    System.out.println("Pago de " + monto + " dólares confirmado.");  
}
```

Se declara la clase “Factura”, que representa un modelo que gestiona la información y el estado de una factura en un sistema de ventas.

```
1 class Factura {
```

Variables de Instancia

```
    private int idFactura;  
    private Cliente cliente;  
    private Pedido pedido;  
    private Pago pago;  
    private String estado;
```

idFactura: Un entero que identifica de manera única cada factura.

cliente: Un objeto de tipo Cliente, que representa al cliente asociado a la factura.

pedido: Un objeto de tipo Pedido, que contiene información sobre el pedido relacionado.

pago: Un objeto de tipo Pago, que representa los detalles del pago realizado.

estado: Una cadena que indica el estado actual de la factura (por ejemplo, "Creada", "Anulada", "Pagada").

Constructor de la clase

```
public Factura(int idFactura, Cliente cliente, Pedido pedido, Pago pago) {  
    this.idFactura = idFactura;  
    this.cliente = cliente;  
    this.pedido = pedido;  
    this.pago = pago;  
    this.estado = "Creada";  
}
```

Métodos de la clase

```
public void crearFactura() {  
    System.out.println("Factura " + idFactura + " creada para el cliente " + cliente.getIdCliente() + ".");  
}  
  
public void anularFactura() {  
    estado = "Anulada";  
    System.out.println("Factura " + idFactura + " ha sido anulada.");  
}  
  
public void pagarFactura() {  
    estado = "Pagada";  
    System.out.println("Factura " + idFactura + " ha sido pagada.");  
}  
  
public String getEstado() {  
    return estado;  
}
```

Se define la clase Main, que contiene el método main, el punto de entrada de la aplicación. Dentro de este método se ejecutará el flujo principal del programa.

```
public class Main {  
    public static void main(String[] args) {
```

Se crea un objeto Cliente con un ID, nombre y correo electrónico.

Se llama al método registrarse() que imprime un mensaje indicando que el cliente se ha registrado.

Se llama al método iniciarSesion() que imprime un mensaje indicando que el cliente ha iniciado sesión.

```
Cliente cliente = new Cliente(1, "Juan Pérez", "juan@example.com");  
cliente.registrarse();  
cliente.iniciarSesion();
```

Se crean dos objetos Producto, cada uno con un ID, nombre, precio y cantidad en stock.

```
Producto productol = new Producto(101, "Laptop", 800.00, 10);
Producto producto2 = new Producto(102, "Smartphone", 300.00, 20);
```

Se crea un objeto Pedido con un ID.

Se agregan los productos creados al pedido utilizando el método agregarProducto(), que también actualiza el total del pedido e imprime un mensaje sobre la adición de productos.

```
Pedido pedido = new Pedido(1);
pedido.agregarProducto(productol);
pedido.agregarProducto(producto2);
```

Se crea un objeto Pago con un ID, el método de pago y el monto total (calculado a partir del pedido).

Se llama al método procesarPago(), que imprime un mensaje sobre el procesamiento del pago.

Luego, se llama al método confirmarPago(), que imprime un mensaje indicando que el pago ha sido confirmado.

```
Pago pago = new Pago(1, "Tarjeta de Crédito", pedido.calcularTotal());
pago.procesarPago();
pago.confirmarPago();
```

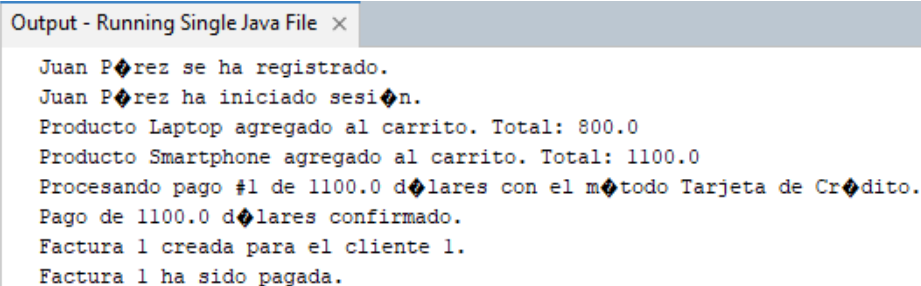
Se crea un objeto Factura con un ID, el cliente, el pedido y el pago asociados.

Se llama al método crearFactura(), que imprime un mensaje indicando que la factura ha sido creada.

Finalmente, se llama al método pagarFactura(), que cambia el estado de la factura a "Pagada" e imprime un mensaje indicando que la factura ha sido pagada.

```
Factura factura = new Factura(1, cliente, pedido, pago);
factura.crearFactura();
factura.pagarFactura();
```

Salida:



```
Output - Running Single Java File x
Juan Pérez se ha registrado.
Juan Pérez ha iniciado sesión.
Producto Laptop agregado al carrito. Total: 800.0
Producto Smartphone agregado al carrito. Total: 1100.0
Procesando pago #1 de 1100.0 dólares con el método Tarjeta de Crédito.
Pago de 1100.0 dólares confirmado.
Factura 1 creada para el cliente 1.
Factura 1 ha sido pagada.
```