

# **REPORTE – PROYECTO PRIMER PARCIAL-PAR #5 PROGRAMACIÓN ORIENTADA A OBJETOS I PAO 2025**

## **Grupo #3-Integrantes:**

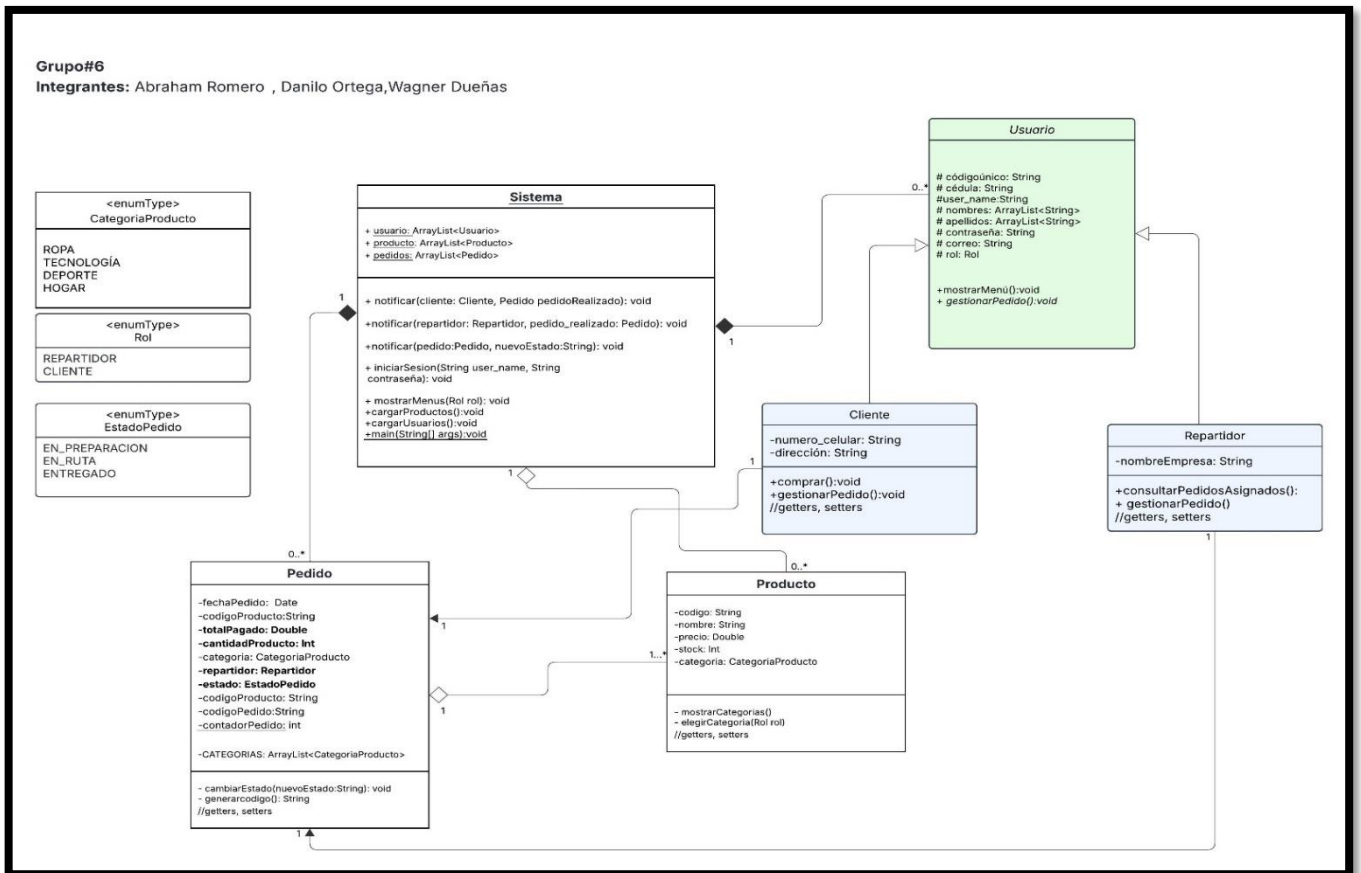
- Abraham Romero
- Danilo Ortega
- Wagner Dueñas

## **URL Repositorio:**

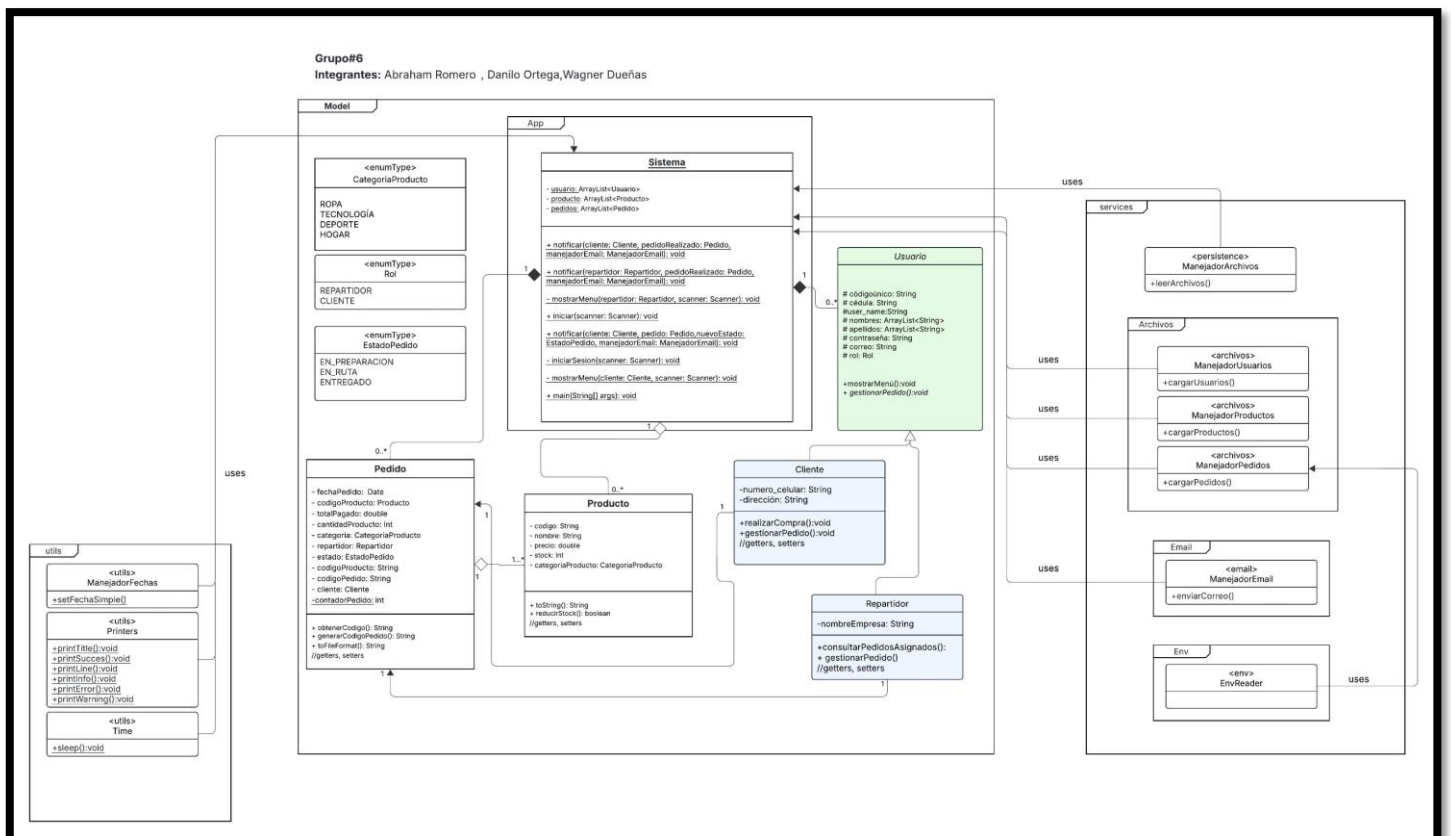
[https://github.com/WSmithDR/POO4\\_1P\\_DUENAS\\_ROMERO\\_ORTEGA.git](https://github.com/WSmithDR/POO4_1P_DUENAS_ROMERO_ORTEGA.git)

**Fecha:** 06/07/2025

# 1. Diagrama de clases versión 1



# 2. Diagrama de clases versión 2



### 3. Tareas

Tareas asignadas a cada estudiante:

#### Estudiante (Abraham Romero Smith Rondón):

1. Crear la clase Producto y los enums CategoriaProducto y EstadoPedido.
2. Cargar la lista de usuarios al sistema con la clase ManejadorUsuario y cargar lista de productos con la clase ManejadorProductos.
3. Crear el método de iniciarSesion() y mostrarMenus() a la clase Sistema.
4. Crear el método de consultarPedidosAsignados() a la clase Repartidor.
5. Implementar el método comprar() de la clase Cliente.

#### Estudiante (Danilo Ortega):












1. Crear la clase Pedido.
2. Corregir variables de instancia en la clase Pedido.
3. Sobrescribir el método gestionarPedido() para la clase Cliente.
4. Crear la clase ManejadorPedido y de la clase ManejadorFecha

#### Estudiante (Wagner Dueñas):

1. Creación de la clase User y sus subclases heredadas.
2. Organización de la primera etapa y segunda etapa del proyecto.
3. Crear el método notificar() a la clase Sistema.
4. Arreglos en algunos métodos y algunas cosas en la lógica

### 5. Evidencias de Tareas

Aquí podemos observar las tareas que fueron asignadas a cada estudiante y se mostrara como evidencia los commit de esas tareas por cada estudiante.

<input type="checkbox"/>	Tipo	Clave	Resumen	Estado	Comentarios	Persona asignada
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-7	Creacion de la clase User y sus herencias	FINALIZADA	 Añadir comentario	 Wagner Smith Duen...
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-21	feat: Organizacion inicial del entorno para la segunda etapa ...	FINALIZADA	 Añadir comentario	 Wagner Smith Duen...
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-13	fix: Corregir variables de instancia de la clase Pedido	FINALIZADA	 Añadir comentario	 Danilo Javier Ortega ...
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-8	Creación de la clase Producto	FINALIZADA	 Añadir comentario	 Abraham Romero
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-9	Creacion de clase Pedido	FINALIZADA	 Añadir comentario	 Danilo Javier Ortega ...
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-10	Poner documentacion inicial	FINALIZADA	 Añadir comentario	 Wagner Smith Duen...
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-11	Incializacion de proyecto de java	FINALIZADA	 Añadir comentario	 Wagner Smith Duen...
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-12	feat: Construir el metodo comprar() de la clase Cliente	FINALIZADA	 Añadir comentario	 Abraham Romero

<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-14	feat: Sobreescibir el metodo gestionarPedido() para la clase ...	FINALIZADA	Añadir comentario	DC Danilo Javier Ortega ...
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-15	feat: Sobreescibir el metodo gestionarPedido() para la clase ...	FINALIZADA	Añadir comentario	DC Danilo Javier Ortega ...
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-16	feat: Crear el metodo de consultarPedidosAsginados() a la cl...	FINALIZADA	Añadir comentario	AR Abraham Romero
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-17	feat: Crear el metodo notificar() a la clase Sistema	EN CURSO	1 comentario	WR Wagner Smith Duen...
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-18	feat: Cargar lista de usuarios al sistema, crear el metodo de i...	FINALIZADA	1 comentario	AR Abraham Romero
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-20	doc: Asegurarnos de que todos los metodos en los que haya...	TAREAS POR HA...	Añadir comentario	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-22	fix: Mejorar logica en no presentar como categorias disponib...	TAREAS POR HA...	1 comentario	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OPS-23	fix: Arreglar logica de gestionar pedido en la clase Cliente	TAREAS POR HA...	Añadir comentario	

# Estudiante (Abraham Romero ):

## Commit 1

Commit 80894d7

RomeroAbraham authored 2 weeks ago (Verified)

Merge pull request #5 from WSmithDR/OPS-8-Creación-de-la-clase-Producto  
Clase productos y enums de Cateo

main (#5) 2 parents 72d4704 + f07b7a1 commit 80894d7

Filter files...

src/main/java  
Enums  
CategoriaProducto.java  
Producto.java

2 files changed +72 -0 lines changed

src/main/java/Enums/CategoriaProducto.java +5  
1 + package Enums;  
2 +  
3 + public enum CategoriaProducto {  
4 +  
5 + }  
src/main/java/Producto.java +67  
1 + import Enums.CategoriaProducto;  
2 + import Enums.Rol;  
3 + public class Producto {  
4 + private String nombre;  
5 + private double precio;

## Commit 2

Commit ec7fc94

TheWalkerr15 committed 17 hours ago

Se implemento el método realizarCompra en la clase Cliente, se actualizo el formato del enum CategoriaProducto y del enum EstadoPendiente

main (#11) 1 parent 15d1e1e commit ec7fc94

Filter files...


resources  
Pedidos.txt  
Productos.txt  
src/main/java  
app  
Sistema.java  
model  
Enums  
CategoriaProducto.java  
EstadoPedido.java

11 files changed +556 -107 lines changed


resources/Pedidos.txt +13  
1 + CódigoPedido|CédulaCliente|CédulaRepartidor|CódigoProducto|Cantidad|ValorPagado|FechaPedido|Estado PED1751767148671|0923456789|0967788990|P004|2|2400.0|  
2025-07-05 20:59:08|En Preparación  
2 + PED1751767528684|0923456789|0956677889|P007|3|105.0|2025-07-05 21:05:28|En Preparación  
3 + PED1751767656506|0923456789|0967788990|P004|4|4800.0|2025-07-05 21:07:36|En Preparación  
4 + PED1751768326114|0923456789|0956677889|P005|3|75.0|2025-07-05 21:18:46|En Preparación  
5 + PED1751768364789|0923456789|0956677889|P006|2|240.0|2025-07-05 21:19:24|En Preparación  
6 + PED1751768575353|0923456789|0956677889|P012|2|240.0|2025-07-05 21:22:55|En Preparación  
7 + PED1751768585830|0923456789|0967788990|P002|3|135.0|2025-07-05 21:23:05|En Preparación  
8 + PED1751768669235|0923456789|0967788990|P006|2|240.0|2025-07-05 21:24:29|En Preparación  
9 + PED1751769041140|0923456789|0956677889|P001|2|50.0|2025-07-05 21:30:41|En Preparación  
10 + PED1751769372003|0923456789|0956677889|P008|3|1240.0|2025-07-05 21:35:52|En Preparación

## Commit 3

Commit **c139cd2**

 TheWalkerr15 committed 15 hours ago

Implementacion del metodo de consultarPedidosAsignados

 main (#12)

1 parent [85ad292](#) commit [c139cd2](#)

Filter files...

src/main/java

app

Sistema.java

model/Roles

Repartidor.java

2 files changed +43 -3 lines changed

src/main/java/app/Sistema.java

@@ -143,7 +143,7 @@ private static void mostrarMenuRepartidor(Repartidor repartidor) {

143 143 System.out.println("Función de gestión de pedido en desarrollo...");

144 144 break;

145 145 case "2":

146 - System.out.println("Función de consulta de pedidos en desarrollo...");

146 + repartidor.consultarPedidosAsignados(pedidos);

147 147 break;

148 148 case "3":

149 149 System.out.println("Cerrando sesión...");

src/main/java/model/Roles/Repartidor.java

@@ -1,8 +1,9 @@


1 1 package model.Roles;

2 2


Estudiante (Danilo Ortega):

## Commit 1

Commit **72d4704**

 OrtegaDanilo committed 2 weeks ago

Clase pedido

 main

1 parent [07f84b9](#) commit [72d4704](#)

Filter files...

src/main/java/Pedido

Pedido.java

1 file changed +101 -0 lines changed

src/main/java/Pedido/Pedido.java

@@ -0,0 +1,101 @@

1 + package Pedido;

2 +

3 + import java.util.Date;

4 + import Enums.Rol;

5 + import Roles.Repartidor;

6 +

7 + public class Pedido {

8 + private Date fecha;

9 + private String codigoProducto;

10 + private double precio;

11 + private int stock;

12 + private CategoriaProducto categoria;

13 + private Repartidor repartidor;

14 + private String estado;

15 + private String codigoPedido;

## Commit 2

Commit **3430c5e**

OrtegaDanilo committed yesterday

Método gestionarPedido de la clase Cliente

main (#10) 1 parent f916848 commit 3430c5e

Filter files...

src/main/java/model

Pedido.java

Roles

Cliente.java

2 files changed +10 -5 lines changed

src/main/java/model/Pedido.java

+1 -1

...  
18 18 @@ -18,7 +18,7 @@ public class Pedido {  
19 19 public Pedido (Date fechaPedido, String codigoProducto, double totalPagado, int cantidadProducto, CategoriaProducto categoria,  
20 20 String codRepartidor, String estadoPedido, String codigoPedido) {  
21 - this.fechaPedido=new Date();  
21 + this.fechaPedido=fechaPedido;  
22 22 this.codigoProducto=codigoProducto;  
23 23 this.totalPagado=totalPagado;  
24 24 this.cantidadProducto=cantidadProducto;  
...  
+

src/main/java/model/Roles/Cliente.java

+9 -4

...  
... @@ -1,10 +1,8 @@  
1 1 package model.Roles;  
2 2  
...  
+

## Commit 3

Commit **0cc726b**

OrtegaDanilo committed 16 hours ago

Modificaciones a Clases: Sistema, EstadoPedido, Pedido, Cliente, Repartidor, ManejoFechas

main (#10) 1 parent bfcc11d commit 0cc726b

Filter files...

src/main/java

app

Sistema.java

model

Enums

EstadoPedido.java

Pedido.java

Roles

Cliente.java

Repartidor.java

utils

ManejoFechas.java

6 files changed +164 -70 lines changed

src/main/java/app/Sistema.java

+1 -1

...  
... @@ -99,7 +99,7 @@ private static void mostrarMenuCliente(Cliente cliente) {  
99 99 System.out.println("Función de compra en desarrollo...");  
100 100 break;  
101 101 case "2":  
102 - System.out.println("Función de gestión de pedido en desarrollo...");  
102 + cliente.gestionarPedido(scanner);  
103 103 break;  
104 104 case "3":  
105 105 System.out.println("Cerrando sesión...");  
...  
+

src/main/java/model/Enums/EstadoPedido.java

+26

...  
... @@ -0,0 +1,26 @@  
1 + package model.Enums;  
2 +  
...  
+

## Estudiante (Wagner Dueñas):

### Commit 1

Commit 2e2a47a

WSmithDR committed 3 weeks ago

feat: creadas las clase Usuario y sus herencias Cliente y Repartidor asi como la construccion de sus setters y getters

main (#4)

1 parent d2f74f5 commit 2e2a47a

Filter files...

.vscode

settings.json

src/main/java

Demo1

Demo1.java

Enums

Rol.java

Main.java

Roles

Cliente.java

Repartidor.java

Usuario.java

7 files changed +168 -0 lines changed

Search within code

.vscode/settings.json

+3

@@ -0,0 +1,3 @@

1 + {

2 + "java.configuration.updateBuildConfiguration": "interactive"

3 + }

src/Demo1/Demo1.java → src/main/java/Demo1/Demo1.java

File renamed without changes.

src/main/java/Enums/Rol.java

+5

@@ -0,0 +1,5 @@

1 + package Enums;

2 +

3 + public enum Rol {

### Commit 2

Commit 8c0727b

WSmithDR committed yesterday

Organizada la arquitectura del proyecto para la segunda etapa

main (#8)

1 parent 80894d7 commit 8c0727b

Filter files...

resources

Clientes.txt

Productos.txt

Repartidores.txt

Usuarios.txt

src/main/java

Demo1

Demo1.java

app

Sistema.java

model

Enums

CategoriaProducto.java

14 files changed +124 -24 lines changed

Search within code

resources/Clientes.txt

+3

@@ -0,0 +1,3 @@

1 + CódigoÚnico|Cédula|Nombres|Apellidos|Celular|Dirección

2 + C001|0923456789|Ana|Torres|0991234567|Av. Principal y Calle 10

3 + C002|0911122233|David|Vélez|0987654321|Calle Falsa 123

resources/Productos.txt

+9

@@ -0,0 +1,9 @@

1 + Código|Categoría|Nombre|Precio|Stock

2 + P001|Tecnología|Audifonos Bluetooth|35.99|20

3 + P002|Tecnología|Mouse Inalámbrico|18.50|15

4 + P003|Ropa|Camiseta Deportiva Hombre|22.99|30

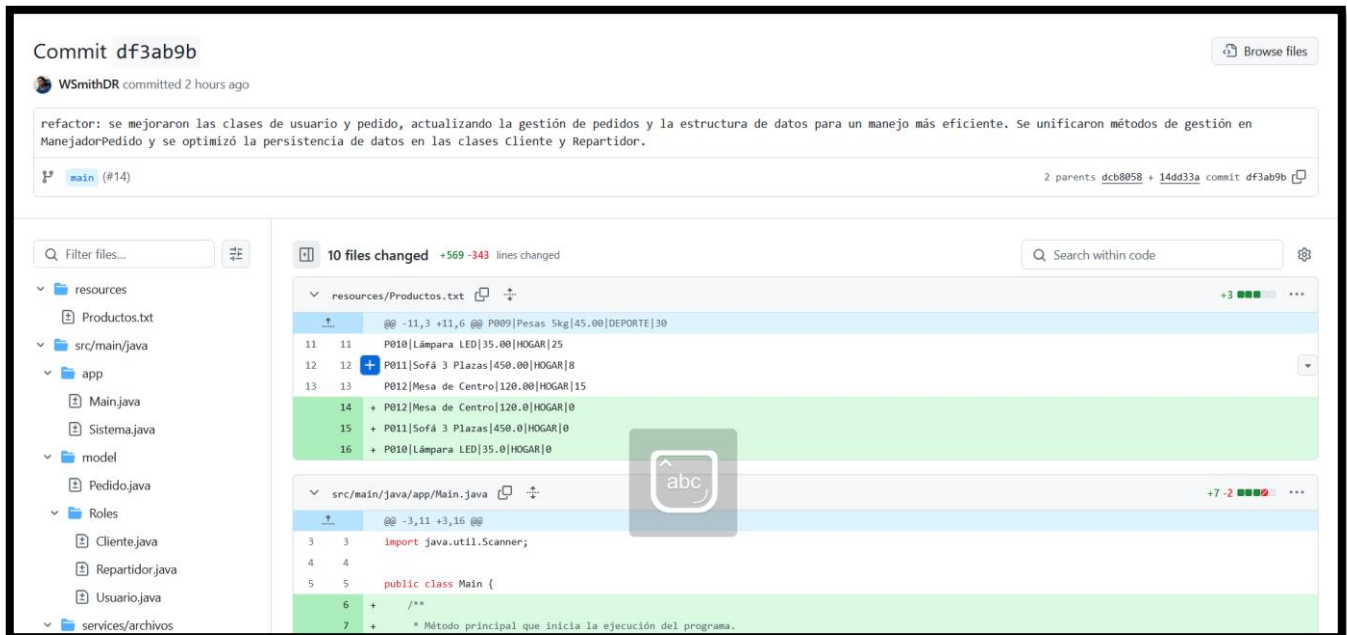
5 + P004|Ropa|Pantalón Deportivo Mujer|32.75|25

6 + P005|Hogar|Lámpara LED Recargable|12.00|40

7 + P006|Deportes|Balón de Fútbol|25.00|10

8 + P007|Hogar|Taza Térmica Acero|9.99|50

## Commit 3



### 6. Identificación de pilares de la programación orientada a objetos.

En esta sección se incluirá el detalle de los bloques de código del proyecto donde identificamos cada pilar de este paradigma: Abstracción, Encapsulamiento, Herencia y Polimorfismo. Ejemplo:

#### Abstracción

```
POO4_1P_DUEÑAS_ROMERO_ORTEGA > src > main > java > model > Roles > Usuario
6
7
8
9 public abstract class Usuario {
10     protected String codigoUnico;
11     protected String cedula;
12     protected String user_name;
13     protected ArrayList<String> nombres;
14     protected ArrayList<String> apellidos;
15     protected String contrasenia;
16     protected String correo;
17     protected Rol rol;
```

Explicación: Se utilizó la abstracción en la clase Usuario debido a que promueve la herencia, y “Usuario” es un concepto amplio que puede referirse a diferentes tipos de personas en este caso (clientes, repartidores), por lo tanto, en clientes y repartidores se heredan los atributos de Usuario y cada uno con sus atributos únicos.



## Encapsulamiento

```
import java.util.ArrayList;

import model.Enums.Rol;

public abstract class Usuario {
    protected String codigoUnico;
    protected String cedula;
    protected String user_name;
    protected ArrayList<String> nombres;
    protected ArrayList<String> apellidos;
    protected String contrasenia;
    protected String correo;
    protected Rol rol;
```

Explicación: Usamos aquí el encapsulamiento es para proteger los datos de una clase para que no se puedan acceder o modificar directamente desde fuera, usando métodos especiales como los getters y setters, ya que con estos métodos podemos modificar valores de forma controlada.

## Herencia

```
package model.Roles;

import java.util.ArrayList;
import model.Enums.Rol;
import model.Enums.EstadoPedido;
import model.Pedido;

public class Repartidor extends Usuario {
    private String nombreEmpresa;
```

Explicación: Usamos la herencia en la clase Repartidor y Cliente como ya habíamos dicho anteriormente estas dos clases heredan de Usuario para también así mismo heredar sus atributos y usarlos en las clases hijas que en este caso son Repartidos y Cliente.

## Polimorfismo

```
for (Usuario u : usuarios) {
    if (u.getUser_name().equals(userInput) && u.getContrasenia().equals(passInput)) {
        usuarioEncontrado = true;
        System.out.println(x:"Usuario autenticado correctamente.");

        if (u instanceof Cliente c) {
            System.out.println(x:"Rol detectado: CLIENTE");
            System.out.println("Bienvenido, " + c.getNombres().get(index:0) + " " + c.getApellidos().get(index:0));
            System.out.println("Celular registrado: " + c.getNumeroCelular());
            System.out.print(s:"¿Este número de celular es correcto? (S/N): ");
            String verif = scanner.nextLine();
            if (verif.equalsIgnoreCase(anotherString:"S")) {
                mostrarMenuCliente(c);
            } else {
                System.out.println(x:"Verificación fallida. Cerrando sesión.");
            }
        } else if (u instanceof Repartidor r) {
            System.out.println(x:"Rol detectado: REPARTIDOR");
            System.out.println("Bienvenido, " + r.getNombres().get(index:0) + " " + r.getApellidos().get(index:0));
            System.out.println("Empresa asignada: " + r.getNombreEmpresa());
            System.out.print(s:"¿Esta empresa es correcta? (S/N): ");
            String verif = scanner.nextLine();
            if (verif.equalsIgnoreCase(anotherString:"S")) {
                mostrarMenuRepartidor(r);
            } else {
                System.out.println(x:"Verificación fallida. Cerrando sesión.");
            }
        }
    }
}
break;
```

Explicación: Aquí podemos observar polimorfismo ya que tratamos a todos como Usuarios, sin importar si en realidad son Cliente o Repartidor.

Entonces usamos polimorfismo de sustitución: ya que Cliente y Repartidor heredan de Usuario se guardan todos en una lista o ArrayList y se tratan como usuarios hasta que necesitamos sus comportamientos específicos y bueno luego realizamos downcasting mediante instanceof para poder hacer lo que queremos realmente dependiendo de su comportamiento.

## 7. Identificación de otros conceptos

### ArrayLists

```
public class Sistema {  
    private static ArrayList<Usuario> usuarios = new ArrayList<>();  
    private static ArrayList<Producto> productos = new ArrayList<>();  
    private static ArrayList<Pedido> pedidos = new ArrayList<>();  
    /**
```

Explicación: Aquí hacemos uso de ArrayList para Usuarios, productos y pedidos aquí lo que hacemos es inicializar estas listas y así guardar los datos de usuarios registrados, productos disponibles y pedidos realizados, para posteriormente usar esas listas con la información para métodos que lo requieran.

### Creación de objetos a partir de datos de archivo

```
if (partes[7].equals(Rol.CLIENTE.toString())) {  
    Cliente cliente = cargarDatosCliente(codigoUnico, cedula, nombre, apellido, username, correo,  
        contrasenia);  
    if (cliente != null) {  
        usuarios.add(cliente);  
    }  
} else if (partes[7].equals(Rol.REPARTIDOR.toString())) {  
    Repartidor repartidor = cargarDatosRepartidor(codigoUnico, cedula, nombre, apellido, username, correo,  
        contrasenia);  
    if (repartidor != null) {  
        usuarios.add(repartidor);  
    }  
}
```

Explicación: esta parte del código podemos observar que se encarga de identificar el rol del usuario (cliente o repartidor) según la información leída de un archivo. Y según el caso, crea el objeto adecuado (Cliente o Repartidor) y lo añade a la lista general de usuarios del sistema. Se utiliza equals para comparar roles y null como verificación de seguridad antes de insertar en la lista

## Sobreescritura

```
@Override
public void gestionarPedido(ArrayList<Pedido> pedidos, Scanner scanner) {
    ManejadorPedido.gestionarPedido(this, pedidos, scanner);
}
```

```
/**
 * Implementación del método abstracto de Usuario en la clase Repartidor
 * Permite al repartidor gestionar sus pedidos asignados
 * @param pedidos Lista de todos los pedidos del sistema
 */
@Override
public void gestionarPedido(ArrayList<Pedido> pedidos, Scanner scanner) {
    ManejadorPedido.gestionarPedido(this, pedidos, scanner);
}
```

```
/**
 * Implementación del método abstracto de Usuario
 * Permite al repartidor gestionar sus pedidos asignados
 * @param pedidos Lista de todos los pedidos del sistema
 */
@Override
public void gestionarPedido(ArrayList<Pedido> pedidos, Scanner scanner) {
    ManejadorPedido.gestionarPedido(this, pedidos, scanner);
}
```

```
/**
 * Método abstracto para gestionar pedidos según el rol del usuario
 *
 * @param pedidos Lista de pedidos disponibles para gestionar
 * @param scanner scanner para poder interactuar con la consola
 */
public abstract void gestionarPedido(ArrayList<Pedido> pedidos, Scanner scanner);
```

Explicación: Podemos observar sobreescritura ya que estamos redefiniendo un método abstracto de la clase padre Usuario, podemos observar el @Override que nos indica que esta bien sobreescrito el método, y por último observamos que a diferencia de sobrecarga aquí se mantiene la misma firma o sea se reciben los mismos parámetros.

## Sobrecarga

```
public static void notificar(
    Cliente cliente,
    Pedido pedidoRealizado,
    ManejadorEmail manejadorEmail) {
    String asunto = "Pedido realizado";
    String cuerpo = String.format(
        "<html><body>"
        + "El/La cliente <strong>%s %s</strong> ha realizado un pedido con código <strong>%s</strong> el día <strong>%s</strong>."
        +
        "<br><br>" +
        "Producto: <strong>%s</strong>" +
        "<br>" +
        "Cantidad: <strong>%d</strong>" +
        "<br>" +
        "Valor pagado: <strong>$.2f</strong>" +
        "<br>" +
        "Estado inicial: <strong>%s</strong>" +
        "<br><br>" +
        "Gracias por su compra. Recibirá actualizaciones del estado de su pedido por este medio."
        + "</body></html>",
        cliente.getNombre(),
        cliente.getApellido(),
        pedidoRealizado.getCodigoPedido(),
        pedidoRealizado.getFechaPedido(),
        pedidoRealizado.getCodProducto(),
        pedidoRealizado.getCantidadProducto(),
        pedidoRealizado.getTotalPagado(),
        pedidoRealizado.getEstadoPedido());
    System.out.println("Enviando correo al " + Rol.CLIENTE + " " + cliente.getCorreo());
    manejadorEmail.enviarCorreo(cliente.getCorreo(), asunto, cuerpo);
}
```

```
public static void notificar(
    Repartidor repartidor,
    Pedido pedidoAsignado,
    ManejadorEmail manejadorEmail) {
    Cliente cliente = ManejadorUsuario.buscarClientePorCodigoUnico(pedidoAsignado.getCodCliente());
    String asunto = "Nuevo pedido asignado";
    String cuerpo = String.format(
        "<html><body>" + "Estimado/a <strong>%s %s</strong>," +
        "<br>" +
        "Se le ha asignado un nuevo pedido con los siguientes detalles:" +
        "<br><br>" +
        "Código del pedido: <strong>%s</strong>" +
        "<br>" +
        "Fecha del pedido: <strong>%s</strong>" +
        "<br>" +
        "Cliente: <strong>%s %s</strong>," +
        "<br>" +
        "Estado actual: <strong>%s</strong>" +
        "<br><br>" +
        "Por favor, prepare la logística necesaria para la entrega." +
        "Gracias por su trabajo." + "</body></html>",
        repartidor.getNombre(),
        repartidor.getApellido(),
        pedidoAsignado.getCodigoPedido(),
        pedidoAsignado.getFechaPedido(),
        cliente.getNombre(),
        cliente.getApellido(),
        pedidoAsignado.getEstadoPedido());
    System.out.println("Enviando correo al " + Rol.REPARTIDOR + " " + repartidor.getCorreo());
    manejadorEmail.enviarCorreo(repartidor.getCorreo(), asunto, cuerpo);
}
```

Explicación: Se aplica sobrecarga, porque hay métodos con el mismo nombre, el mismo modificador (static), y que no retornan ningún valor (void), pero se diferencian en los parámetros que reciben (tipo, número o ambos). Esto nos permite realizar diferentes acciones dependiendo de los datos o parámetros que les pasemos.

## 8. Programa en ejecución

Aquí incluiremos screenchots del programa siendo ejecutado:

```
=====
BIENVENIDO/A A
DUENAS ROMERO ORTEGA
DELIVERY SYSTEM
=====
```

```
=====
1. Iniciar sesión
2. Salir del sistema
Seleccione una opción: 1
=====
```

```
=====
Inicio de sesión
=====
```

```
Usuario: atorres
Contraseña: cliente123
[ÉXITO] Usuario autenticado correctamente.
=====
```

```
=====
Bienvenido cliente
=====
```

```
Nombre: Ana Torres
Celular registrado: 0991234567
```

```
¿Este número de celular es correcto? (S/N): s
[ÉXITO] ¡Identidad confirmada!
```

```
=====
Menú cliente
=====
```

```
1. Comprar
2. Gestionar pedido
3. Cerrar sesión
```

```
Seleccione una opción: 1
```

```
=====
COMPRAR PRODUCTO
=====
```

```
=====
CATEGORÍAS DISPONIBLES
=====
```

```
1. TECNOLOGIA
2. HOGAR
3. DEPORTES
4. ROPA
Elige una categoría (1-4): 1
=====
```

```
=====
Elige una categoría (1-4): 1
=====
```

```
Productos disponibles:
```

```
1. Teclado Mecánico - $45.0 (Stock: 8)
2. Mouse Inalámbrico - $18.5 (Stock: 15)
3. Audífonos Bluetooth - $35.99 (Stock: 20)
```

```
Elige un producto (1-3): 1
```

```
¿Cuántos quieres comprar? (máximo 8): 5
```

```
Total a pagar: $225.0
```

```
¿Confirmar compra? (s/n): s
```

```
Enviando correo al REPARTIDOR: camila.rios@email.com
```

```
Correo enviado a camila.rios@email.com
```

```
Enviando correo al CLIENTE ana.torres@email.com
```

```
Correo enviado a ana.torres@email.com
```

```
¡Compra exitosa!
```

```
Repartidor: Camila Ríos
```

```
Código de pedido: PED1
```

```
=====
Menú cliente
=====
```

```
1. Comprar
2. Gestionar pedido
3. Cerrar sesión
```

```
Seleccione una opción: 2
```

```
=====
CONSULTA DE ESTADO DE PEDIDO
=====
```

```
=====
TUS PEDIDOS
=====
```

```
1. Código: PED1
   Fecha: 2025-07-13
   Estado: EN_PREPARACION
```

```
-----
Ingrese el código del pedido: PED1
```

```
-----
Fecha del pedido: 2025-07-13
Producto comprado: Teclado Mecánico (Código: P008)
Cantidad: 5
Valor pagado: $225,00
Estado actual: EN_PREPARACION
Repartidor: Camila Ríos (Código: R001)
```

```
-----
[INFO] Su pedido está siendo preparado para su envío.
```

A

abraham.romero.rondon.15@gmail.com

para ana.torres ▾

El/La cliente **Ana Torres** ha realizado un pedido con código **PED1** el día **Sun Jul 13 21:08:34 ECT 2025**.

Producto: **P008**

Cantidad: **5**

Valor pagado: **\$225,00**

\*\*\*

Estado inicial: **EN\_PREPARACION**

Gracias por su compra. Recibirá actualizaciones del estado de su pedido por este medio.

...

A

abraham.romero.rondon.15@gmail.com

para camila.rios ▾

Estimado/a **Camila Ríos**,

Se le ha asignado un nuevo pedido con los siguientes detalles:

Código del pedido: **PED1**

Fecha del pedido: **Sun Jul 13 21:08:34 ECT 2025**

Cliente: **Ana Torres**,

Estado actual: **EN\_PREPARACION**

\*\*\*

Por favor, prepare la logística necesaria para la entrega. Gracias por su trabajo.

...

```
[INFO] Cerrando sesión...
```

```
BIENVENIDO/A A  
DUENAS_ROMERO_ORTEGA  
DELIVERY SYSTEM
```

```
1. Iniciar sesión  
2. Salir del sistema  
Seleccione una opción: 1
```

```
Inicio de sesión
```

```
Bienvenido repartidor
```

```
Nombre: Camila Ríos  
Empresa asignada: Envíos Express
```

```
¿Esta empresa es correcta? (S/N): s  
[ÉXITO] ¡Identidad confirmada!
```

```
Menú repartidor
```

```
1. Gestionar pedido  
2. Consultar pedidos asignados  
3. Cerrar sesión
```

```
Seleccione una opción: 2
```

```
PEDIDOS ASIGNADOS
```

```
[INFO] Buscando pedidos asignados no entregados...
```

```
[ÉXITO] Pedidos encontrados:
```

```
1. Código: PED1  
   Fecha del pedido: 2025-07-13  
   Estado actual: EN_PREPARACION
```

```
Total de pedidos pendientes: 1  
Recuerde que solo puede gestionar los pedidos que se encuentren EN_PREPARACION o EN_CAMINO.
```

```
GESTIONAR ESTADO DE PEDIDO
```

```
Ingrese el codigo de alguno de los pedidos presentados
```

```
Codigo: PED1  
[ÉXITO] Pedido encontrado:
```

```
Fecha del pedido: 2025-07-13  
Código del producto: P008  
Estado actual: EN_PREPARACION
```

```
Seleccione el nuevo estado del pedido
```

```
1. EN_CAMINO  
2. ENTREGADO  
Opción: 1  
[ÉXITO] Estado actualizado correctamente a EN_CAMINO.  
Enviando correo al CLIENTE: ana.torres@email.com  
Correo enviado a ana.torres@email.com
```

```
GESTIONAR ESTADO DE PEDIDO
```

```
Ingrese el codigo de alguno de los pedidos presentados
```

```
Codigo: PED1  
[ÉXITO] Pedido encontrado:
```

```
Fecha del pedido: 2025-07-13  
Código del producto: P008  
Estado actual: EN_CAMINO
```

```
Seleccione el nuevo estado
```

```
1. ENTREGADO  
Opción: 1  
[ÉXITO] Estado actualizado correctamente a ENTREGADO.  
Enviando correo al CLIENTE: ana.torres@email.com  
Correo enviado a ana.torres@email.com
```

```
PEDIDOS ASIGNADOS
```

```
[INFO] Buscando pedidos asignados no entregados...
```

```
[INFO] No tienes pedidos asignados pendientes.
```



abraham.romero.rondon.15@gmail.com

para ana.torres ▾

Estimado/a **Ana Torres**,

Le informamos que el estado de su pedido con código **PED1** ha cambiado a: **EN\_CAMINO**.

Fecha del pedido: **Sun Jul 13 00:00:00 ECT 2025**

Producto: **P008**

Repartidor asignado: **R001**

...

Gracias por confiar en nosotros.

...



abraham.romero.rondon.15@gmail.com

para ana.torres ▾

Estimado/a **Ana Torres**,

Le informamos que el estado de su pedido con código **PED1** ha cambiado a: **ENTREGADO**.

...

Fecha del pedido: **Sun Jul 13 00:00:00 ECT 2025**

Producto: **P008**

Repartidor asignado: **R001**

Gracias por confiar en nosotros.

...

## 9. JAVADOC

Agregar la documentación JAVADOC como una carpeta adicional a este reporte. Los métodos deben estar siempre comentados con el formato explicado en clase.

Referencia:<https://www.youtube.com/watch?v=KChdcRscFt0>