REPORTE – PROYECTO PRIMER PARCIAL-PAR #1

PROGRAMACIÓN ORIENTADA A OBJETOS

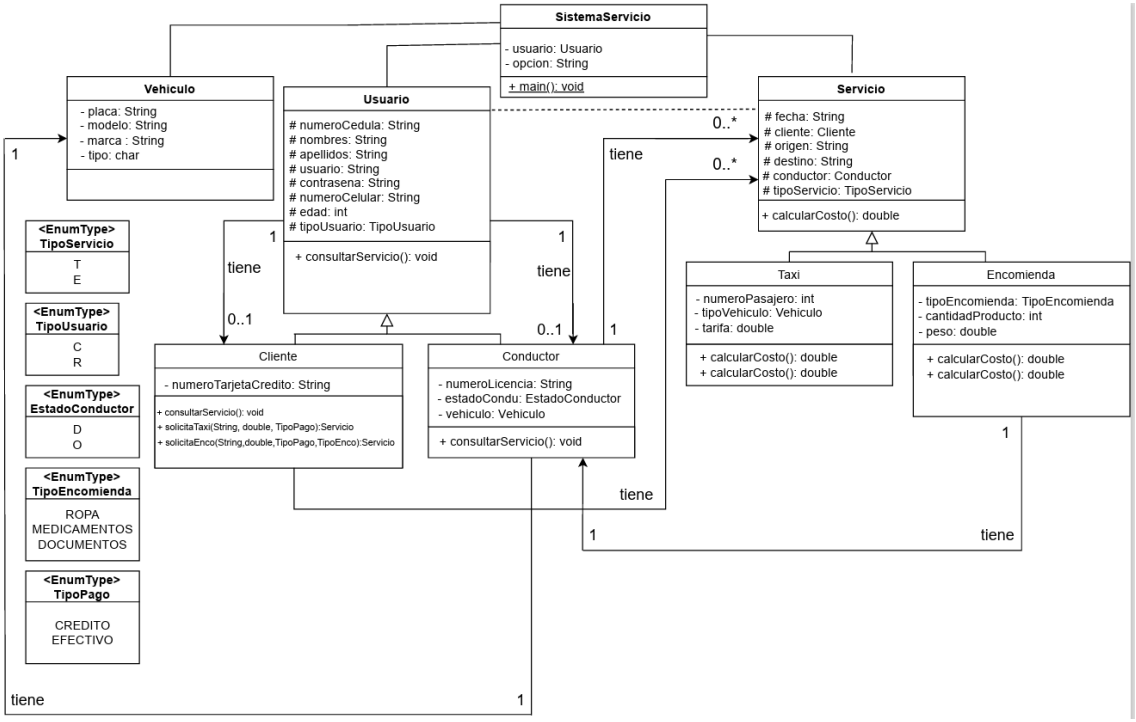
II PAO 2023-2024

Integrantes:
Paula Bastidas
Guillermo Mero
Pedro Vinueza

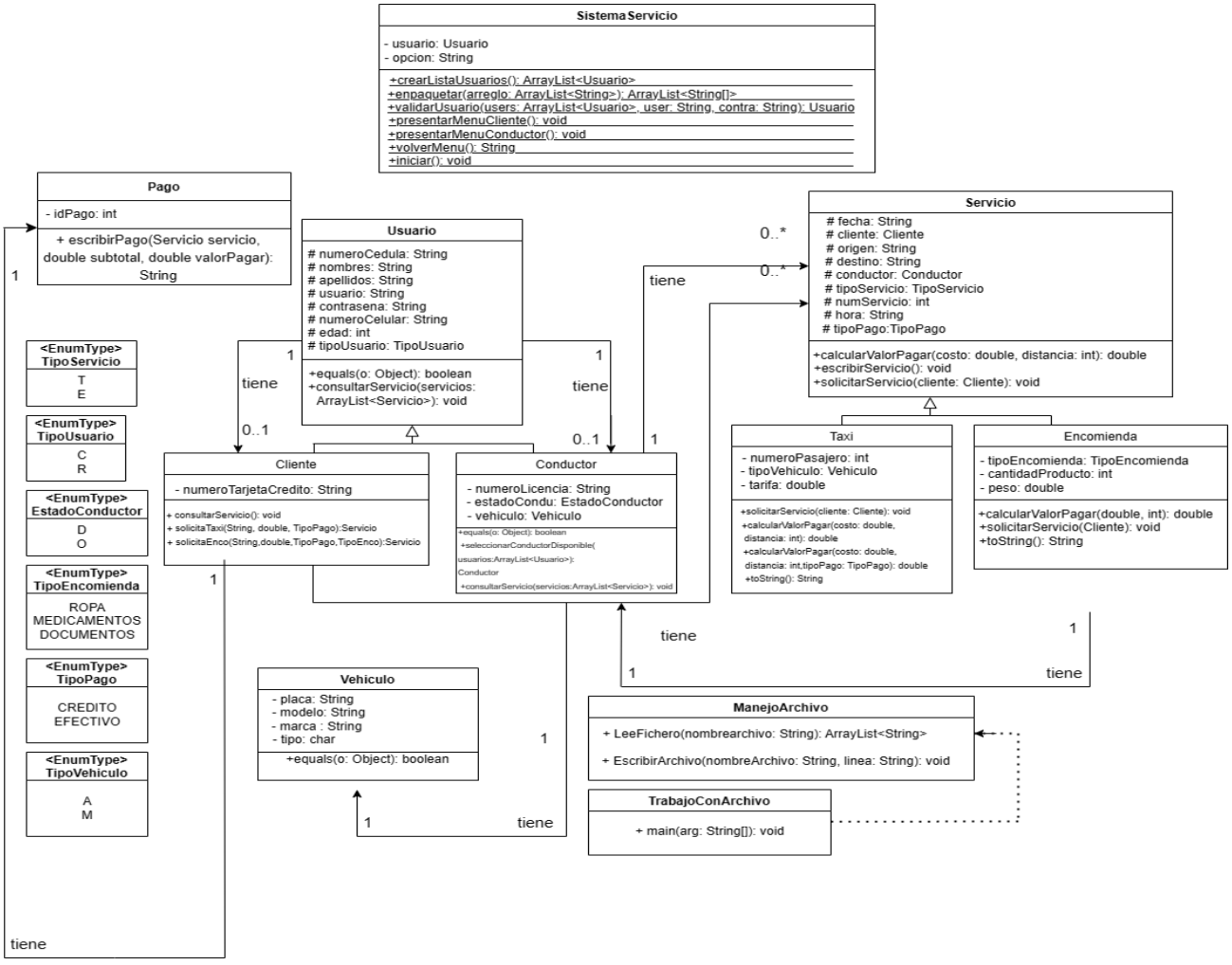
URL Repositorio:
https://github.com/GuillermoMero/POO5_1P_BASTIDAS_MERO_VINUEZA

Fecha:

1. Diagrama de clases versión 1



2. Diagrama de clases versión 2



3. Tareas

En esta sección incluirán el detalle de las tareas que se asignó a cada estudiante.

Estudiante (Paula Bastidas):

1. Realizar reporte y UML (diseño y aporte)
2. Crear los nombres de las clases y verificar herencia
3. Escribir atributos y métodos get & set en java (comentarlos para el java doc)

Estudiante (Guillermo Mero):

1. Realizar UML (listas y archivos)
2. Desarrollar los métodos de clases y polimorfismo (crear la estructura de los métodos)
3. Verificar las clases y la sobreescritura

Estudiante (Pedro Vinueza):

1. Verificar el proceso (clases, atributos, implementación de herencia y sobreescritura)
2. Desarrollar los métodos de clases (crear contenido de los métodos e interfaz)
3. Polimorfismo en las clases

4. Evidencias de Tareas

Estudiante (Paula Bastidas):

Commit 1

comentar los metodos

main

polipopi committed 23 minutes ago

Showing **11 changed files** with **155 additions** and **25 deletions**.

22 ■■■■ Proyecto1P_Grupo2/src/main/java/Servicio/Encomienda.java

```
@@ -43,21 +43,37 @@ public void setpeso(double peso){
43  43      public double getpeso(){
44  44          return peso;
45  45      }
    46  +   /**
    47  +    * Calcula el costo del servicio de encomienda.
    48  +    */
46  49      public void calcularcosto(){
47  50
48  51      }
```

Commit 2

metodos get y set — Browse files

main

polipopi committed 4 days ago — 1 parent b465075 commit fbe1de2

Showing **7 changed files** with **183 additions** and **65 deletions**. Split | Unified

41 ■■■■■ Proyecto1P_Grupo2/src/main/java/Servicio/Encomienda.java

```
@@ -0,0 +1,41 @@
1  + /*
2  +  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  +  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  +  */
5  + package Servicio;
6  + import Usuario.Cliente;
7  + import Usuario.Conductor;
8  + /**
9  +  *
10 +  * @author Paula
11 +  */
12 + public class Encomienda extends Servicio{
```

Commit 3

Herencia de clasea

main

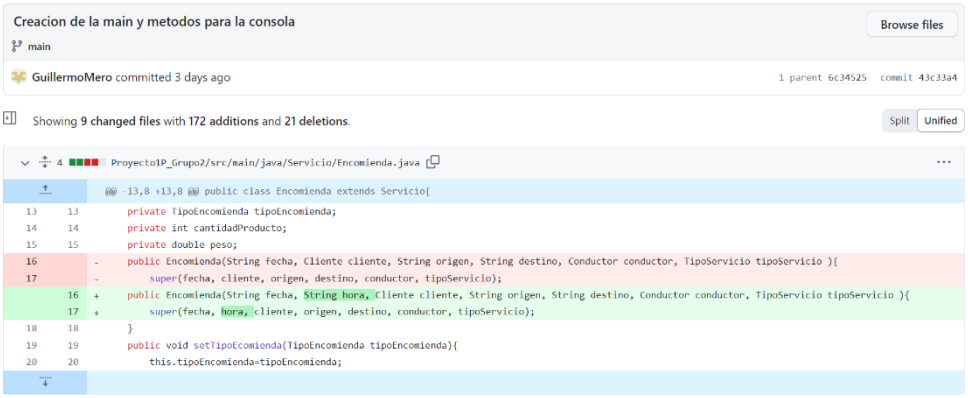
polipopi committed 4 days ago

Showing **7 changed files** with **183 additions** and **65 deletions**.

41 ■■■■■ Proyecto1P_Grupo2/src/main/java/Servicio/Encomienda.java

```
@@ -0,0 +1,41 @@
1  + /*
2  +  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to chan
3  +  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this temp
4  +  */
5  + package Servicio;
6  + import Usuario.Cliente;
7  + import Usuario.Conductor;
8  + /**
9  +  *
10 +  * @author Paula
11 +  */
12 + public class Encomienda extends Servicio{
13 +     private TipoEncomienda tipoEncomienda;
14 +     private int cantidadProducto;
15 +     private double peso;
16 +     public Encomienda(String fecha, Cliente cliente, String origen, String destino, Co
```

Commit 1



Creacion de la main y metodos para la consola

Browse files

main

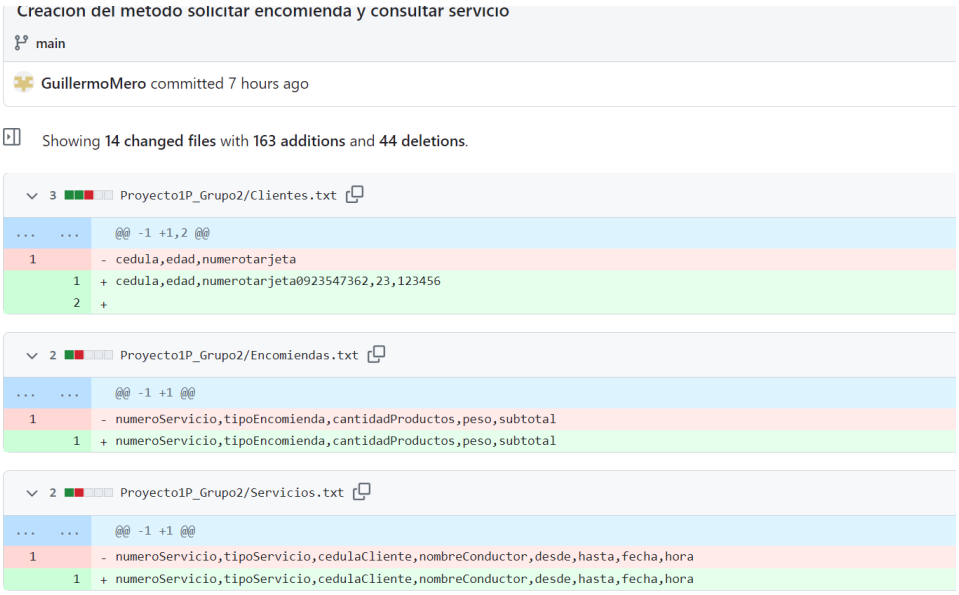
GuillermoMero committed 3 days ago — 1 parent 6c34525 commit 43c33a4

Showing 9 changed files with 172 additions and 21 deletions. Split Unified

4 Proyecto1P_Grupo2/src/main/java/Servicio/Encomienda.java

```
@@ -13,8 +13,8 @@ public class Encomienda extends Servicio{
    private TipoEncomienda tipoEncomienda;
    private int cantidadProducto;
    private double peso;
-   public Encomienda(String fecha, Cliente cliente, String origen, String destino, Conductor conductor, TipoServicio tipoServicio ){
-       super(fecha, cliente, origen, destino, conductor, tipoServicio);
+   public Encomienda(String fecha, String hora, Cliente cliente, String origen, String destino, Conductor conductor, TipoServicio tipoServicio ){
+       super(fecha, hora, cliente, origen, destino, conductor, tipoServicio);
    }
    public void setTipoEcomienda(TipoEncomienda tipoEncomienda){
        this.tipoEncomienda=tipoEncomienda;
```

Commit 2



Creacion del metodo solicitar encomienda y consultar servicio

main

GuillermoMero committed 7 hours ago

Showing 14 changed files with 163 additions and 44 deletions.

3 Proyecto1P_Grupo2/Clientes.txt

```
@@ -1 +1,2 @@
- cedula,edad,numerotarjeta
+ cedula,edad,numerotarjeta0923547362,23,123456
+
```

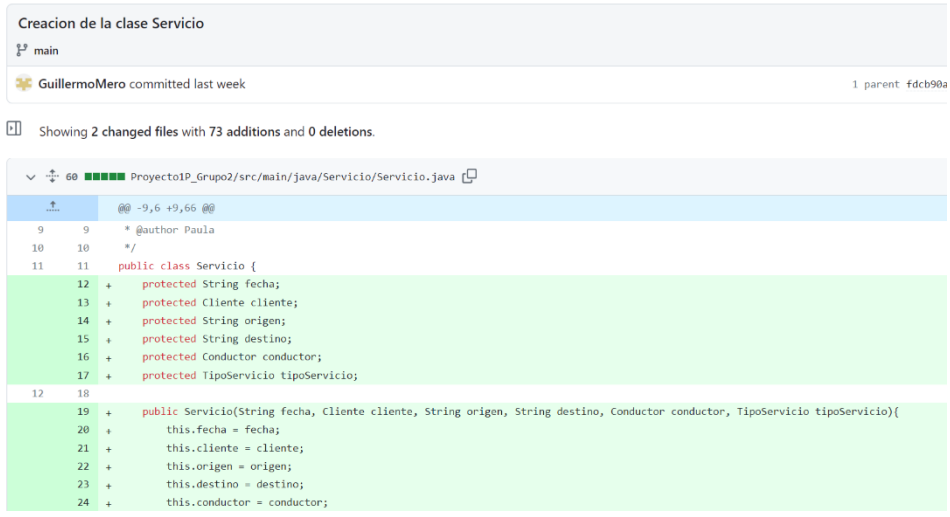
2 Proyecto1P_Grupo2/Encomiendas.txt

```
@@ -1 +1 @@
- numeroServicio,tipoEncomienda,cantidadProductos,peso,subtotal
+ numeroServicio,tipoEncomienda,cantidadProductos,peso,subtotal
```

2 Proyecto1P_Grupo2/Servicios.txt

```
@@ -1 +1 @@
- numeroServicio,tipoServicio,cedulaCliente,nombreConductor,desde,hasta,fecha,hora
+ numeroServicio,tipoServicio,cedulaCliente,nombreConductor,desde,hasta,fecha,hora
```

Commit 3



Creacion de la clase Servicio

main

GuillermoMero committed last week — 1 parent fdcb90a

Showing 2 changed files with 73 additions and 0 deletions.

60 Proyecto1P_Grupo2/src/main/java/Servicio/Servicio.java

```
@@ -9,6 +9,66 @@
 * @author Paula
 */
public class Servicio {
+    protected String fecha;
+    protected Cliente cliente;
+    protected String origen;
+    protected String destino;
+    protected Conductor conductor;
+    protected TipoServicio tipoServicio;

+    public Servicio(String fecha, Cliente cliente, String origen, String destino, Conductor conductor, TipoServicio tipoServicio){
+        this.fecha = fecha;
+        this.cliente = cliente;
+        this.origen = origen;
+        this.destino = destino;
+        this.conductor = conductor;
```

Commit 1

Creacion y finalizacion del metodo servicio Taxi

main

PedroVinueza committed 15 hours ago

Showing **19 changed files** with **307 additions** and **224 deletions**.

1 Proyecto1P_Grupo2/Clientes.txt

```
@@ -0,0 +1 @@
+ cedula,edad,numerotarjeta
```

1 Proyecto1P_Grupo2/Encomiendas.txt

```
@@ -0,0 +1 @@
+ numeroServicio,tipoEncomienda,cantidadProductos,peso,subtotal
```

1 Proyecto1P_Grupo2/Pagos.txt

```
@@ -0,0 +1 @@
+ idPago,FechaPago,numeroServicio,formaPago,cedulaCliente,subtotal,valorPagar
```

Commit 2

Creacion de arrayList de Usuarios. PedroVinueza

main

PedroVinueza committed 3 weeks ago

Showing **2 changed files** with **82 additions** and **21 deletions**.

37 Proyecto1P_Grupo2/src/main/java/SistemaServicio/sistemaServicio.java

```
@@ -4,6 +4,7 @@
 */
package SistemaServicio;

+ import Usuario.TipoUsuario;
import Usuario.Usuario;
import manejoArchivos.*;
import java.util.ArrayList;
@@ -12,35 +13,29 @@
 *
 * @author Paula
 */
```

Commit 3

37 Proyecto1P_Grupo2/src/main/java/Usuario/Conductor.java

```
@@ -0,0 +1,37 @@
+ /*
+  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
+  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
+  */
+ package Usuario;
+ import Vehiculo.Vehiculo;
+ /**
+  *
+  * @author PEDRO VINUEZA
+  */
+ public class Conductor extends Usuario{
+     private String numeroLicencia;
+     private EstadoConductor estadoConductor;
+     private Vehiculo vehiculo;
+
+     public Conductor(String numeroCedula, String nombre, String apellidos, String usuario, String contraseña, String numeroCel
  tipoUsuario) {
+         super(numeroCedula, nombre, apellidos, usuario, contraseña, numeroCelular, edad, tipoUsuario);
```

5. Identificación de pilares de la programación orientada a objetos.

Abstracción

1)

```java
public class Encomienda extends Servicio{
    private TipoEncomienda tipoEncomienda;
    private int cantidadProducto;
    private double peso;
```

```java
public class Pago {
    private int numeroPago;
    private String fechaPago;
    private int numeroServicio;
    private String formaPago;
    private String cedulaCliente;
    private double valorPago;
    private TipoPago tipoPago;
```

```java
public class Servicio {
    protected int numServicio;
    protected TipoServicio tipoServicio;
    protected String cedulaCliente;
    protected String nombreConductor;
    protected String origen;
    protected String destino;
    protected String fecha;
    protected String hora;
    protected TipoPago tipoPago;
```

```java
*/
public class Taxi extends Servicio{
    private int numeroPasajero;
    private Scanner sc;

    public Taxi(){
        sc = new Scanner(System.in);
```

```java
public class Vehiculo {
    private String placa;
    private String modelo;
    private String marca;
    private char tipo;
```

```java
public class SistemaServicio{
    private Scanner sc;
    private static ArrayList<Usuario> usuarios;
```

Explicación

La abstracción se refleja en la creación de clases como Encomienda, Pago, Servicio, Taxi, Vehiculo, SistemaServicio, Cliente, Conductor, y Usuario, donde cada una representa entidades y sus propiedades y métodos esenciales dentro de del sistema de servicios de transporte.

2)

```java
public static ArrayList<String> LeeFichero(String nombrearchivo) {
```

Estos métodos (LeeFichero y EscribirArchivo) permiten a otros desarrolladores o partes del código utilizar operaciones de lectura y escritura de archivos sin conocer los detalles internos de cómo se implementan estas operaciones. Proporcionan una interfaz simple y clara para interactuar con archivos.

Encapsulamiento

1)

```java
public void setTipoEcomienda(TipoEncomienda tipoEncomienda){
    this.tipoEncomienda=tipoEncomienda;
}
public TipoEncomienda gettipoEncomienda(){
    return tipoEncomienda;
}
public void setcantidadProducto(int cantidadProducto){
    this.cantidadProducto=cantidadProducto;
}
public int getcantidadProducto(){
    return cantidadProducto;
}
```

```java
public int getNumeroPago() {
    return numeroPago;
}

public void setNumeroPago(int numeroPago) {
    this.numeroPago = numeroPago;
}

public String getFechaPago() {
    return fechaPago;
}

public void setFechaPago(String fechaPago) {
    this.fechaPago = fechaPago;
`
```

Explicación

El encapsulamiento se aplica mediante el uso de modificadores de acceso (private, protected) para los atributos de las clases (numeroServicio, tipoServicio, cedulaCliente, …), lo que controla su acceso directo y requiere el uso de métodos (getters y setters) para modificar o acceder a estos atributos.

2)

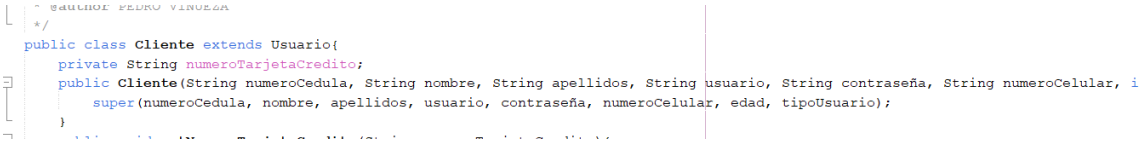
```java
public static ArrayList<String> LeeFichero(String nombrearchivo) {
```

Explicación

Ambos métodos encapsulan la lógica de lectura y escritura de archivos en funciones específicas.

Herencia

1)

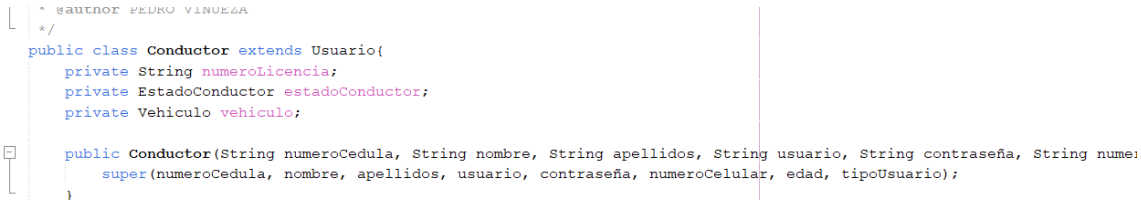


```
 */
public class Cliente extends Usuario{
    private String numeroTarjetaCredito;
    public Cliente(String numeroCedula, String nombre, String apellidos, String usuario, String contraseña, String numeroCelular, i
        super(numeroCedula, nombre, apellidos, usuario, contraseña, numeroCelular, edad, tipoUsuario);
    }
```

Explicación

En este caso, la clase base es Usuario, que contiene propiedades generales compartidas por cliente, como la cédula, nombres, apellidos, usuario, contraseña y número de celular. Al utilizar herencia, evitamos duplicar código, ya que las subclases heredan automáticamente estas propiedades y métodos comunes. Esta herencia se reconoce en el código con un extends de la clase padre adquiriendo un super con los atributos de la clase padre.

2)

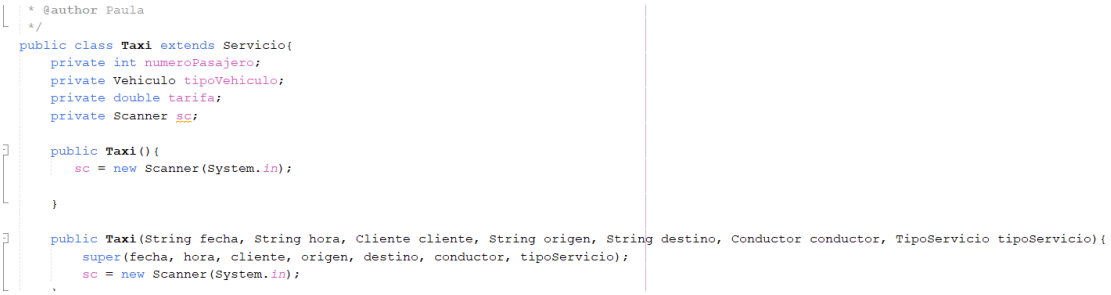


```
 */
public class Conductor extends Usuario{
    private String numeroLicencia;
    private EstadoConductor estadoConductor;
    private Vehiculo vehiculo;

    public Conductor(String numeroCedula, String nombre, String apellidos, String usuario, String contraseña, String nume
        super(numeroCedula, nombre, apellidos, usuario, contraseña, numeroCelular, edad, tipoUsuario);
    }
```

Explicación

La clase base es Usuario, que contiene propiedades generales compartidas por Conductor, como la cédula, nombres, apellidos, usuario, contraseña y número de celular. Al utilizar herencia, evitamos duplicar código, ya que las subclases heredan automáticamente estas propiedades y métodos comunes. Esta herencia se reconoce en el código con un extends de la clase padre adquiriendo un super con los atributos de la clase padre.

3)

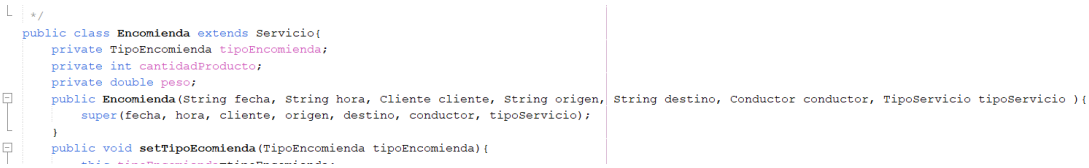


```
 * @author Paula
 */
public class Taxi extends Servicio{
    private int numeroPasajero;
    private Vehiculo tipoVehiculo;
    private double tarifa;
    private Scanner sc;

    public Taxi(){
        sc = new Scanner(System.in);

    }

    public Taxi(String fecha, String hora, Cliente cliente, String origen, String destino, Conductor conductor, TipoServicio tipoServicio){
        super(fecha, hora, cliente, origen, destino, conductor, tipoServicio);
        sc = new Scanner(System.in);
```

Explicación

Cuando creamos la clase derivada Taxi, podemos heredar de la clase padre Servicio y agregar propiedades y métodos específicos para el servicio de taxi, como el número de pasajeros que viajarán y todos los demás atributos.

4)



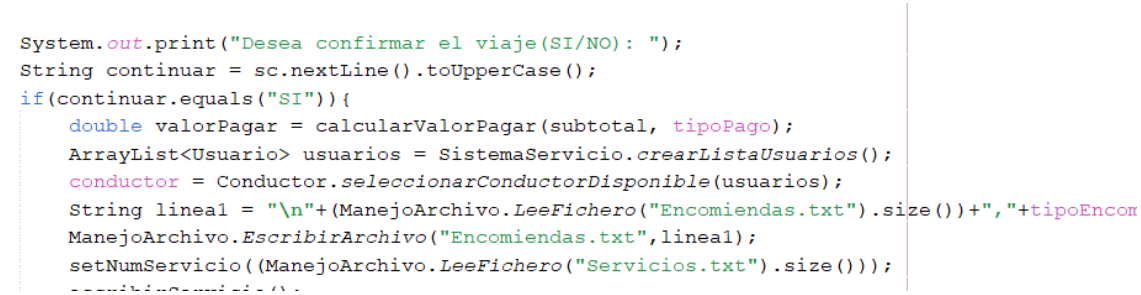
```
*/
public class Encomienda extends Servicio{
    private TipoEncomienda tipoEncomienda;
    private int cantidadProducto;
    private double peso;
    public Encomienda(String fecha, String hora, Cliente cliente, String origen, String destino, Conductor conductor, TipoServicio tipoServicio ){
        super(fecha, hora, cliente, origen, destino, conductor, tipoServicio);
    }
    public void setTipoEcomienda(TipoEncomienda tipoEncomienda){
```

Explicación

Al utilizar herencia de esta manera, se logra una estructura jerárquica donde Taxi y Encomienda comparten características comunes definidas en Servicio, pero también tienen la capacidad de extender y especializarse según sus propias necesidades.

Polimorfismo

1)



```
System.out.print("Desea confirmar el viaje(SI/NO): ");
String continuar = sc.nextLine().toUpperCase();
if(continuar.equals("SI")){
    double valorPagar = calcularValorPagar(subtotal, tipoPago);
    ArrayList<Usuario> usuarios = SistemaServicio.crearListaUsuarios();
    conductor = Conductor.seleccionarConductorDisponible(usuarios);
    String linea1 = "\n"+(ManejoArchivo.LeeFichero("Encomiendas.txt").size())+","+tipoEncom
    ManejoArchivo.EscribirArchivo("Encomiendas.txt",linea1);
    setNumServicio((ManejoArchivo.LeeFichero("Servicios.txt").size()));
```

Explicación

El método solicitarServicio está diseñado para ser llamado desde la clase base Servicio, permitiendo a las clases derivadas como Encomienda tener su propia implementación del método calcularValorPagar. Esto es un ejemplo de polimorfismo, donde un método común (solicitarServicio en la clase base) se comporta de manera diferente según la clase concreta de objeto que lo llama.

6. Identificación de otros conceptos

ArrayLists

1)

```
public class ManejoArchivo {
    public static ArrayList<String> LeeFichero(String nombrearchivo) {
        ArrayList<String> lineas = new ArrayList<>();
        File archivo = null;
        FileReader fr = null;
        BufferedReader br = null;

        try {
            // Apertura del fichero y creacion de BufferedReader para poder
            // hacer una lectura comoda (disponer del metodo readLine()).
            archivo = new File(nombrearchivo);
            fr = new FileReader(archivo,StandardCharsets.UTF_8);
            br = new BufferedReader(fr);
```

Explicación

ArrayList<String> en este caso lo hicimos para brindar flexibilidad, facilidad de manipulación y compatibilidad con otras características de Java, haciéndolo más práctico para almacenar líneas leídas de un archivo de texto.

2)

```
public SistemaServicio(){
    sc = new Scanner(System.in);
    usuarios = new ArrayList<>();
}

public static ArrayList crearListaUsuarios(){
    ArrayList<Usuario> arreglo = new ArrayList<>();
    ArrayList<String[]> listasUsuarios = new ArrayList<>();
    ArrayList<String[]> listasConductores = new ArrayList<>();
    ArrayList<String[]> listasVehiculos = new ArrayList<>();
    listasUsuarios= enpaquetar(ManejoArchivo.LeeFichero("usuarios.txt"));
    for(int i=0; i<listasUsuarios.size(); i++){
        if("C".equals(listasUsuarios.get(i)[6])){
            Usuario usuario = new Cliente(listasUsuarios.get(i)[0],listasUsuar
            arreglo.add(usuario);
        }else{
            listasConductores =
```

Explicación

Se usan ArrayList en varias ocasiones para almacenar información de usuarios, conductores y vehículos (ArrayList<Usuario> usuarios, ArrayList<String[]> listasUsuarios, ArrayList<String[]> listasConductores, ArrayList<String[]> listasVehiculos).

3)

```
public static Conductor seleccionarConductorDisponible(ArrayList<String> arregloLineas){
    ArrayList<Usuario> usuarios = SistemaServicio.listaUsuarios();
```

Explicación

En la clase Conductor, se utiliza un ArrayList para almacenar información sobre los usuarios y obtener una lista de conductores disponibles.

4)

```
ArrayList<String> lineas = new ArrayList<>();
```

Explicación

Esto declara una nueva instancia de ArrayList para almacenar las líneas leídas del archivo.

Creación de objetos a partir de datos de archivo

```
public static ArrayList crearListaUsuarios(){
```

Explicación
El método crearListaUsuarios() utiliza datos de archivos para crear objetos Cliente o Conductor a partir de la información leída de archivos de texto (usuarios.txt, conductores.txt, vehiculos.txt).

```
ArrayList<String> linea = ManejoArchivo.LeeFichero("conductor.txt");
```

Explicación
Este archivo se lee para identificar conductores y crear objetos Conductor, basados en la información obtenida del archivo. La información leída del archivo se utiliza para inicializar objetos de la clase Conductor con los datos recuperados.

Sobreescritura

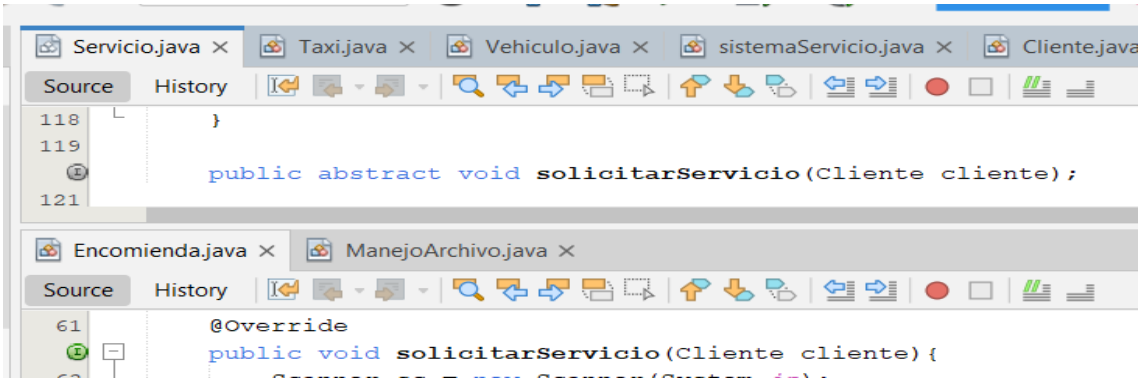
1)

```
@Override
public double calcularValorPagar(double costo, int distancia) {
    return 0.0;
}
```

Explicación

En el Método CalcularValorapagar se lo considera una sobreescritura ya que este método le pertenece a la clase padre y está siendo utilizada por la clase hija, por lo tanto, debe de ser sobreescrita.

2)



Explicación

El método solicitarServicio le pertenece a la clase padre y está siendo sobreescrito en la clase Encomienda para proporcionar una implementación específica y personalizada de ese método en esta subclase.

3)

```
@Override
public String toString(){
    return "/*****************************************/ \nTipo: Encomien
          +"\nHora: "+hora+"\nDesde: "+origen+"\nHasta: "+destino+"\n";
}
```

Explicación

El método toString() es un método de la clase Object que se puede sobrescribir en clases hijas y en este caso se lo está sobreescribiendo en una clase hija.

4)

```
    @Override
    public void consultarServicio(ArrayList<Servicio> servicios) {
        for(int i=0; i<servicios.size();i++){
            if(servicios.get(i).getConductor().getNombre().equals(getNombre())
                System.out.println(servicios.get(i));
            }else{
                System.out.println("No tiene asignado algun servicio");
            }
```

Explicación

En este caso se considera una sobreescritura, porque el método pertenece a la clase padre de Usuario y lo esta utilizando la clase hija, por lo tanto, debe ser sobreescrito.

5)

```
    @Override
    public boolean equals(Object o){
      if (o == this){
        return true;
      }
      if (o != null && this.getClass() == o.getClass()){
        Usuario otro = (Usuario)o;
        return otro.numeroCedula.equalsIgnoreCase(this.numeroCedula) &&
      }else{
        return false;
      }
```

Explicación

el método equals se está sobrescribiendo para cambiar su comportamiento predeterminado de comparación de objetos.

6)

```
public static void main(String[] arg) {
    ManejoArchivo.EscribirArchivo("archivosalida.txt","Hola!");
    ManejoArchivo.EscribirArchivo("archivosalida.txt","Afios!");
    ManejoArchivo.EscribirArchivo("archivosalida.txt","NUevo!");
    ManejoArchivo.LeeFichero("archivo.txt");
```

Explicación

En TrabajoConArchivos, se llama repetidamente al método EscribirArchivo de la clase ManejoArchivo para escribir diferentes líneas en el mismo archivo.

Sobrecarga

1)

```
@Override
public double calcularValorPagar(double costo, int distancia) {
    return 0.0;
}

public double calcularValorPagar(double subtotal, TipoPago tipoPago){
    if(tipoPago == tipoPago.TC)
        return subtotal*1.15;
    return subtotal;
}
```

Explicación

La sobrecarga ocurre cuando una clase tiene múltiples métodos con el mismo nombre pero con diferentes listas de parámetros, en este caso ambos métodos se llaman calcularValorPagar, pero tienen diferentes tipos y cantidad de parámetros

7. Programa en ejecución

```
++++++++++++++++++++++++++++++++++++++++++++++++
          BIENVENIDO AL SISTEMA
++++++++++++++++++++++++++++++++++++++++++++++++
USUARIO: lmancero
CONTRASENA: qwerty
/******************MENU*********************/
/*                                          */
/*******************************************/
1. Solicitar servicio de taxi
2. Solicitar entrega encomienda
3. Consultar servicios
4. Cerrar sesion

Elija una opcion: 1



/*************DETALLES DE LA RUTA*************/
Origen: xxxx
Destino: yyyy
Fecha (dd/mm/aa): 12/12/23
Hora (hh:mm) 2:30
Forma de Pago (TC/E): TC
Numero de pasajeros: 4
El subtotal del valor a pagar por el servicio de taxi es de: 12.5
Desea confirmar el viaje(SI/NO): SI


/*******************************************/
Tipo: Viaje
Cantidad pasajeros: 4
Fecha: 12/12/23
Hora: 2:30
Desde: xxxx
Hasta: yyyy

¿Volver al menú? (SI/NO): SI
```

```
/******************MENU*********************/
/*                                          */
/*******************************************/
1. Solicitar servicio de taxi
2. Solicitar entrega encomienda
3. Consultar servicios
4. Cerrar sesion

Elija una opcion: 2


/*************DETALLES DE LA RUTA*************/
Origen: TTTT
Destino: RRRR
Fecha (dd/mm/aa): 13/1/23
Hora (hh:mm): 2:50
Forma de Pago (TC/E): E
Tipo de encomienda: (MEDICAMENTOS/DOCUMENTOS): DOCUM
Cantidad de producto: 3
Peso del producto [Kg]: 3
El subtotal del valor a pagar por el servicio de enc
Desea confirmar el viaje(SI/NO): SI


/*******************************************/
Tipo: Encomienda
Tipo encomienda: DOCUMENTOS
Cantidad: 3
Fecha: 13/1/23
Hora: 2:50
Desde: TTTT
Hasta: RRRR

¿Volver al menú? (SI/NO): SI
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++
          BIENVENIDO AL SISTEMA
++++++++++++++++++++++++++++++++++++++++++++++++++
USUARIO: jgome
CONTRASENA: 38373
/***************MENU CONDUCTOR******************/
/*                                              */
/************************************************/
1. Consultar Servicio Asignado
2. Datos de su vehiculo
3. Cerrar sesion

Elija una opcion: 1


¿Volver al menú? (SI/NO): si
```

8. 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