FACULTAD DE INGENIERÍA

Carrera de Ciencia de Datos

TITULO

PC2

CURSO

Programación Orientada a Objetos I

DOCENTE

Roberto Josué Rodríguez Urquiaga

INTEGRANTE

Zúñiga Benites Camila Michelle, 2312494

2024-2

Empresa de Transporte

Usted está desarrollando un sistema para una empresa de transporte que gestiona diferentes tipos de vehículos: camiones y furgonetas. Cada tipo de vehículo tiene características específicas, como su capacidad de carga, pero todos comparten algunas propiedades comunes, como el modelo y el precio de alquiler por día.

Además, la empresa necesita almacenar la información sobre los alquileres en una lista de objetos en memoria. Cada alquiler incluirá información sobre el vehículo, la duración del alquiler y el costo total. El sistema debe ser capaz de calcular el costo total del alquiler teniendo en cuenta posibles descuentos y las características de los vehículos.

a) ¿Cómo se podría implementar el polimorfismo en este sistema de transporte? (5 puntos)

(Responde la pregunta copiando la porción de código respectiva y su explicación breve)

- El polimorfismo se implementa utilizando una clase abstracta Vehículo, que es extendida por las clases Camion y Furgoneta. También, cada una de estas clases proporciona su propia implementación del método calcularAlquiler().

```java
public abstract class Vehiculo {  2 usages  2 inheritors
    //atributos
    public String modeloVehiculo;  2 usages
    public double precioAlquiler;  2 usages

    public Vehiculo(String modeloVehiculo, double precioAlquiler){  2 usages
        this.modeloVehiculo = modeloVehiculo;
        this.precioAlquiler = precioAlquiler;
    }

    //metodos
    public String getModeloVehiculo(){  no usages
        return modeloVehiculo;
    }
    public double getPrecioAlquiler(){  2 usages
        return precioAlquiler;
    }

    public abstract double calcularAlquiler(int dia);  no usages  2 implementations
}
```

```java
public class Camion extends Vehiculo{  no usages
    //atributo
    public double capicidadCarga;  1 usage
    public Camion(String modeloVehiculo, double precioAlquiler, double capicidadCarga){
        super(modeloVehiculo, precioAlquiler); //conector a la clase principal
        this.capicidadCarga = capicidadCarga;
    }
    //metodos
    public double calcularAlquiler(int dia){  no usages
        //double costo = dia * getPrecioAlquiler();
        return getPrecioAlquiler()*dia;
    }
}
```

```java
public class Furgoneta extends Vehiculo{  1 usage
    public double capacidadCarga;  1 usage

    public Furgoneta(String modeloVehiculo, double precioAlquiler, double capacidadCarga){
        super(modeloVehiculo, precioAlquiler); //conector a la clase principal
        this.capacidadCarga = capacidadCarga;

    }
    @Override  1 usage
    public double calcularAlquiler(int dia) {
        return getPrecioAlquiler()*dia;
    }
}
```

b) ¿Cómo se podría almacenar la información de los alquileres en una lista? (5 puntos)

(Responde la pregunta copiando la porción de código respectiva y su explicación breve)

- En la clase EmpresaTransporte se almacena la información en una lista utilizando ArrayList.

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

public class EmpresaTransporte {  2 usages
    public List<Alquiler> listaAlquileres;  3 usages
    public EmpresaTransporte(){  1 usage
        listaAlquileres = new ArrayList<>();
    }
    public void agregarAlquiler(Alquiler alquiler){  1 usage
        listaAlquileres.add(alquiler);
    }
    public List<Alquiler> getListaAlquileres(){  1 usage
        return listaAlquileres;
    }
}
```

c) ¿Cómo usarías una interfaz para permitir la gestión común de los vehículos en el sistema? (5 puntos)

(Responde la pregunta copiando la porción de código respectiva y su explicación breve)

- Se utiliza una interfaz GestionVehiculos para permitir la gestión común de los vehículos.

```java
public interface GestionVehiculos {  no usages
    double calcularAlquiler(int dia);  no usages
    String getModeloVehiculo();  no usages
    double getPrecioAlquiler();  no usages
}
```

d) Realizar el diagrama de clases para todo el caso descrito. (5 puntos)

(Copia y pega el diagrama en este informe)

Código

```java
public abstract class Vehiculo {  2 usages  2 inheritors
    //atributos
    public String modeloVehiculo;  2 usages
    public double precioAlquiler;  2 usages

    public Vehiculo(String modeloVehiculo, double precioAlquiler){  2 usages
        this.modeloVehiculo = modeloVehiculo;
        this.precioAlquiler = precioAlquiler;
    }

    //metodos
    public String getModeloVehiculo(){  no usages
        return modeloVehiculo;
    }
    public double getPrecioAlquiler(){  2 usages
        return precioAlquiler;
    }

    public abstract double calcularAlquiler(int dia);  no usages  2 implementations
}
```

```java
public class Camion extends Vehiculo{  no usages
    //atributo
    public double capicidadCarga;  1 usage
    public Camion(String modeloVehiculo, double precioAlquiler, double capicidadCarga){
        super(modeloVehiculo, precioAlquiler); //conector a la clase principal
        this.capicidadCarga = capicidadCarga;
    }
    //metodos
    public double calcularAlquiler(int dia){  no usages
        //double costo = dia * getPrecioAlquiler();
        return getPrecioAlquiler()*dia;
    }
}
```

```java
public class Furgoneta extends Vehiculo{  1 usage
    public double capacidadCarga;  1 usage

    public Furgoneta(String modeloVehiculo, double precioAlquiler, double capacidadCarga){
        super(modeloVehiculo, precioAlquiler); //conector a la clase principal
        this.capacidadCarga = capacidadCarga;

    }
    @Override  1 usage
    public double calcularAlquiler(int dia) {
        return getPrecioAlquiler()*dia;
    }
}
```

```java
public class Alquiler {  5 usages
    public Vehiculo vehiculo;  3 usages
    //public int dia;
    public int duracionAlquiler;  3 usages
    public double costoTotal;  3 usages

    public Alquiler(Vehiculo vehiculo, int duracionAlquiler){  1 usage
        this.vehiculo = vehiculo;
        this.duracionAlquiler = duracionAlquiler;
        this.costoTotal = vehiculo.calcularAlquiler(duracionAlquiler);
    }
    //metodos
    public Vehiculo getVehiculo(){  no usages
        return vehiculo;
    }
    public int getDuracionAlquiler(){  no usages
        return duracionAlquiler;
    }
    public double getCostoTotal(){  no usages
        return costoTotal;
    }
    public String detallesSistemaTransporte(){  no usages
        return "Modelo: "+ vehiculo.getModeloVehiculo() + ", Dias: " + duracionAlquiler + ", Costo: " + costoTota
    }
}
```

```java
public interface GestionVehiculos {  no usages
    double calcularAlquiler(int dia);  no usages
    String getModeloVehiculo();  no usages
    double getPrecioAlquiler();  no usages
}
```

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

public class EmpresaTransporte {  2 usages
    public List<Alquiler> listaAlquileres;  3 usages
    public EmpresaTransporte(){  1 usage
        listaAlquileres = new ArrayList<>();
    }
    public void agregarAlquiler(Alquiler alquiler){  1 usage
        listaAlquileres.add(alquiler);
    }
    public List<Alquiler> getListaAlquileres(){  1 usage
        return listaAlquileres;
    }
}
```

```java
public class Main {
    public static void main(String[] arg){
        EmpresaTransporte empresa = new EmpresaTransporte();
        Vehiculo camion = new Camion( modeloVehiculo: "Honda",  precioAlquiler: 250, capicidadCarga: 12000);
        Vehiculo Furgoneta= new Furgoneta( modeloVehiculo: "Honda",  precioAlquiler: 250,  capacidadCarga: 10000);

        Alquiler alquilerCamion1 = new Alquiler(camion,  duracionAlquiler: 5);
        empresa.agregarAlquiler(alquilerCamion1);

        for(Alquiler alquiler : empresa.getListaAlquileres()){
            System.out.println("Vehiculo: "+ alquiler.getVehiculo().getModeloVehiculo()+ ", Costo Total: " + alqu
        }
    }
}
```