```java
public class Camion extends Vehiculocarga{
 private int numEjes;
    public Camion(String marca, String modelo, int año, int kilometraje, int capacidadcarga, int numEjes) {
        super(marca, modelo, año, kilometraje, capacidadcarga);
        this.numEjes = numEjes;
    }
  //getter y setter para Numejes
  public int getnumEjes() {
  return numEjes;
  }


  public void setnumEjes(int numEjes) {
  this.numEjes = numEjes;
  }
  //sobreescribir el metodo mostrarInfo()
  @Override
  public void mostrarInfo() {
  super.mostrarInfo();
  System.out.println("Numero de Ejes:" + numEjes);
    }
    //sobreescribo el realizar mantenimiento
    @Override
    public void realizarMantenimiento(double costoPorkm, double kilometrosRecorridos) {
     System.out.println(x:"El mantenimiento del frenado en el camion.");
      super.realizarMantenimiento(costoPorkm, kilometrosRecorridos);
    }
}
```

```java
public class vehiculosPasajeros extends Vehiculo{

    private int numPasajeros;

    //constructor
    public vehiculosPasajeros(String marca ,String modelo, int año, int kilometraje, int numPasajeros) {
        super(marca, modelo, año, kilometraje);
        this.numPasajeros = numPasajeros;
    }

    //getter y Setter para numPasajeros
public int getNumPasajeros() {
return numPasajeros;
}

public void setNumPasajeros(int numPasajeros) {
    this.numPasajeros = numPasajeros;
}
//Sobreescribir el metodo mostrar Info()
@Override
public void mostrarInfo() {
 super.mostrarInfo();
 System.out.println("Numero de Pasajeros: " + numPasajeros);

}

 }

```

```java
public class Vehiculocarga extends Vehiculo {

private int capacidadcarga;

//constructor
public Vehiculocarga(String marca, String modelo, int año, int kilometraje, int capacidadcarga) {
    super(marca, modelo, año, kilometraje);
    this.capacidadcarga = capacidadcarga;
}
//Getter y Setter para capacidad de carga
public int getCapacidadcarga() {
    return capacidadcarga;
}

public void setCapacidadCarga(int capacidadcarga) {
    this.capacidadcarga = capacidadcarga;
}

//sobreescribir el metodo MostrarInfo
@Override
public void mostrarInfo(){
    super.mostrarInfo();
    System.out.println("capacidad de carga:" +capacidadcarga + " kg" );
}
```

```java
public class Vehiculo {

//metodo para mostrar la info del vehiculo
public void mostrarInfo() {
    System.out.println("Marca:" +marca);
    System.out.println("Modelo:" +modelo);
    System.out.println("Año:" +año);
    System.out.println("kilometraje:" +kilometraje);

}

//Mectodo para realizar mantenimiento basico
public void realizarMantenimiento(double costoPorkm, double kilometrosRecorridos) {
    double costoTotal = costoPorkm * kilometrosRecorridos;
    System.out.println("Costo total del mantenimiento: $" +costoTotal);
}
Run | Debug
public static void main(String[] args) {
    Auto elAuto = new Auto();
    Camion elcamion = new Camion();

}

}
```

```java
public class Auto extends vehiculosPasajeros {
    private String tipoCombustible;

    public Auto(String marca, String modelo, int año, int kilometraje, int numPasajeros, String tipoCombustible) {
        super(marca, modelo, año, kilometraje, numPasajeros);
        this.tipoCombustible = tipoCombustible;
    }
  //getter and setter
  public String getTipoCombustible() {
  return tipoCombustible;
  }

  public void settipoCombustible(String tipoCombustible) {
    this.tipoCombustible = tipoCombustible;
  }
  //sobreescribir el  metodo mostrarInfo()
@Override
public void mostrarInfo() {
super.mostrarInfo();
System.out.println("Tipo de combustible:" + tipoCombustible);
}
//Sobreescribo el mostrar mantenimiento
 @Override
 public void realizarMantenimiento(double costoPorkm, double kilometrosRecorridos) {
  super.realizarMantenimiento(costoPorkm, kilometrosRecorridos);
  System.out.println(x:"Verificar la alineacion de las ruedas.");
  System.out.println(x:"Inspeccionar la direccion asistida.");
  System.out.println(x:"Lubricar componentes del sistema de direccion.");
 }
   }

```

C: > Users > 50585 > OneDrive > Desktop > clase 20 de mayo 2024 > src > App.java > App > main(String[])

```java
public class App {
    Run | Debug
    public static void main(String[] args) throws Exception {
        System.out.println(x:"Acerca del carro");

        Auto auto= new Auto(marca:"Toyota", modelo:"Corrola", año:2020, kilometraje:15000, numPasajeros:5, t…"Gasolina");

        Camion camion = new Camion(marca:"Volvo", modelo:"FH", año:2019, kilometraje:75000, capacidadcarga…20000, 4);

        //Mostrar la informacion de cada Vehiculo
        System.out.println(x:"Informacion del Auto:");
        auto.mostrarInfo();
        System.out.println(x:"Mantenimiento del sistema de direccion en el Auto.");
        auto.realizarMantenimiento(costoPorkm:4, kilometrosRecorridos:14);

        System.out.println(x:"informacion del camion:");
        camion.mostrarInfo();
        camion.realizarMantenimiento(costoPorkm:12, kilometrosRecorridos:30);
    }
}

```

CÓDIGO 1

- El código en Java crea instancias de las clases Auto y Camión, muestra su información y realiza mantenimiento en ambos vehículos. Utiliza métodos para imprimir detalles y realizar operaciones específicas de mantenimiento.

CÓDIGO 2

- El código define la clase Auto, que hereda de vehiculosPasajeros. Añade el atributo tipoCombustible, métodos getter y setter, y sobrescribe los métodos mostrarInfo y realizarMantenimiento para incluir detalles específicos del auto, como el tipo de combustible y pasos adicionales en el mantenimiento del sistema de dirección.

CÓDIGO 3

- El código define la clase Camión, que extiende de Vehiculocarga. Su funcionalidad es la siguiente: Añade un mensaje específico para el mantenimiento del sistema de frenado del camión antes de llamar al método de la superclase para realizar el mantenimiento estandar. Este código permite crear objetos Camión con características específicas, mostrar su información detallada y realizar tareas de mantenimiento especializadas para el camión.

CÓDIGO 4

- Propósito: La clase VehÍculo sirve como una plantilla básica para crear y gestionar diferentes tipos de vehículos en un sistema, permitiendo la reutilización de código y la extensión para incluir características específicas de cada tipo de vehículo.

CÓDIGO 5

- Propósito: La clase VehÍculocarga se utiliza para representar vehículos que tienen la capacidad de transportar carga, como camiones o furgonetas. Permite gestionar y mostrar información específica sobre la capacidad de carga de estos vehículos, lo que facilita su manejo y seguimiento en un sistema de gestión de flotas o similar.

CÓDIGO 6

- Propósito: La clase vehÍculosPasajeros se utiliza para representar vehículos diseñados para transportar pasajeros, como automóviles o autobuses. Permite gestionar y mostrar información específica sobre la capacidad de pasajeros de estos vehículos, lo que facilita su manejo y seguimiento en un sistema de gestión de flotas o similar.