

Herencia, clases abstractas, interfaces, polimorfismo, ...

Sistema de gestión de viajes de vehículos compartidos:

El sistema tiene como objetivo poner en contacto a personas que desean compartir los gastos de viajes utilizando sus propios vehículos.

Esas personas pueden ser de dos tipos:

- ✓ **Los propietarios** que aportan sus vehículos
- ✓ **Los viajeros**, que no disponen de vehículo propio y están interesados en viajar en los vehículos proporcionados por los propietarios.

El sistema debe ser capaz de:

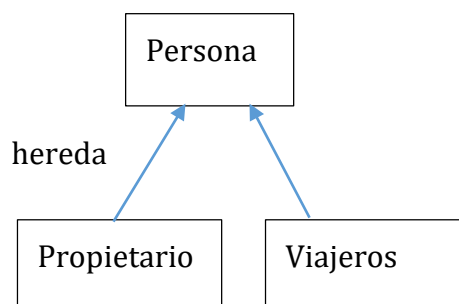
- ✓ **Anotar los viajes** realizados por cada vehículo y cada persona.
- ✓ **Generar facturas** con el gasto e ingresos de cada persona
- ✓ **Generar informes** con el kilometraje realizado por cada vehículo.

A continuación, se muestra la estructura de clases y paquetes que debes implementar:

PAQUETE USUARIOS:

El sistema admite dos clases de personas:

- ✓ **Propietarios:** Las personas que aportan los vehículos. Cuando hagan un viaje como propietario (conducen el coche), no les supondrá gasto. Además, habrá que anotar los ingresos recibidos (la suma de lo que aporten los viajeros en el viaje, que se deberán apuntar el gasto).
- ✓ **Viajeros:** Personas sin vehículos, que viajan en los vehículos proporcionados por los propietarios. En cada viaje se le cobra una cantidad de dinero (gasto).



Clase Persona – Paquete usuarios:

```

usuarios
└─ Persona
    ├── nombre : String
    ├── localidad : String
    ├── telefono : String
    ├── gasto : double
    ├── ingreso : double
    ├── Persona(String, String, String, double, double)
    ├── Persona()
    ├── getNombre() : String
    ├── setNombre(String) : void
    ├── getLocalidad() : String
    ├── setLocalidad(String) : void
    ├── getTelefono() : String
    ├── setTelefono(String) : void
    ├── getGasto() : double
    ├── setGasto(double) : void
    ├── getIngreso() : double
    ├── setIngreso(double) : void
    ├── incrementarGasto(double) : double
    ├── disminuirGasto(double) : double
    ├── incrementarIngreso(double) : double
    ├── disminuirIngreso(double) : double
    ├── toString() : String
    └── valido(String) : boolean

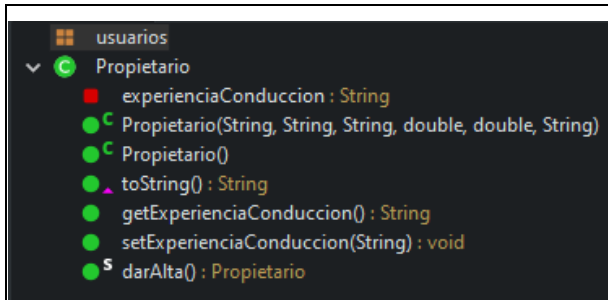
```

Atributos:

- Nombre: (String)
- Localidad: (String)
- Teléfono: (String)
- Gasto (double)
- Ingreso(double)

Métodos:

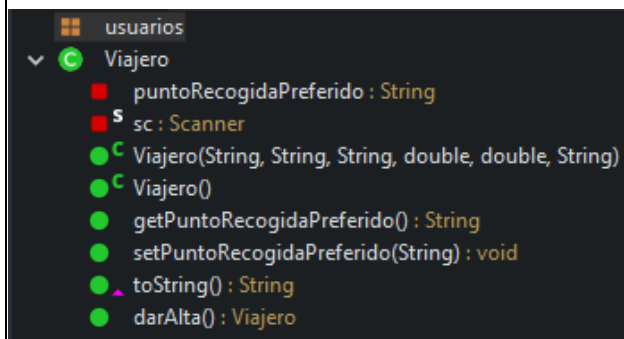
- **Constructor:** Los parámetros de entrada son el **nombre, localidad teléfono**. El **gasto** inicial y los ingresos es **cero**.
- **Métodos getters y setters**
- **Método incrementarGasto(double cantidad)** : Incrementa el gasto con la cantidad recibida como parámetro de entrada.
- **Método disminuirGasto(double cantidad)** : Disminuye el gasto con la cantidad recibida como parámetro de entrada.
- **Método incrementarIngreso(double cantidad)** : Incrementa el ingreso con la cantidad recibida como parámetro de entrada.
- **Método disminuirIngreso(double cantidad)** : Disminuye el ingreso con la cantidad recibida como parámetro de entrada.
- **toString() → mostrarPersona()** -> muestra todos los datos de la persona (nombre, localidad, teléfono y gasto e ingresos).

Clase Propietario – paquete usuarios:**Atributos:**

- ExperienciaConduccion (String)

Métodos:

- **Constructor**: Los parámetros de entrada son el **nombre, localidad teléfono y experiencia conduciendo**. El **gasto** inicial y el **ingreso** inicial es **cero**.
- **Métodos getters y setters**
- **toString() → mostrarPersona()** -> muestra todos los datos del propietario (nombre, localidad, teléfono, experiencia, gasto e ingreso).

Clase Viajero – paquete usuarios:**Atributos:**

- puntoRecogidaPreferido (String)

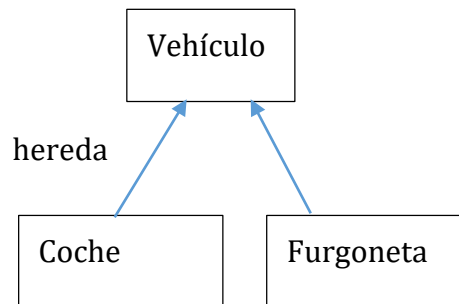
Métodos:

- **Constructor**: Los parámetros de entrada son el **nombre, localidad teléfono y el puntoRecogidaPreferido**. El **gasto** inicial es **cero**.
- **Métodos getters y setters**
- **toString() → mostrarPersona()** -> muestra todos los datos del viajero (nombre, localidad, teléfono, punto de recogida preferido y gasto).

PAQUETE MEDIOSDETRANSPORTE:

El sistema admite dos clases de vehículos:

- ✓ **Coche:** Medio de transporte más utilizado para viajar.
- ✓ **Furgoneta:** Medio de transporte más utilizado para cargar objetos de gran volumen.



Clase Vehículo: (clase abstracta) – paquete mediosdetransporte

```

mediosDeTransporte
└─ Vehículo
   ├── matricula : String
   ├── kmsRecorridos : double
   ├── Titular : Propietario
   ├── sc : Scanner
   ├── Vehículo(String, double, Propietario)
   ├── Vehículo()
   ├── getMatricula() : String
   ├── setMatricula(String) : void
   ├── getKmsRecorridos() : double
   ├── setKmsRecorridos(double) : void
   ├── getTitular() : Propietario
   ├── setTitular(Propietario) : void
   ├── toString() : String
   ├── incrementarkms(double) : double
   ├── disminuirkms(double) : double
   ├── darAlta(int) : Vehículo
   ├── valido(String) : boolean
   └─ valido2(String) : boolean
  
```

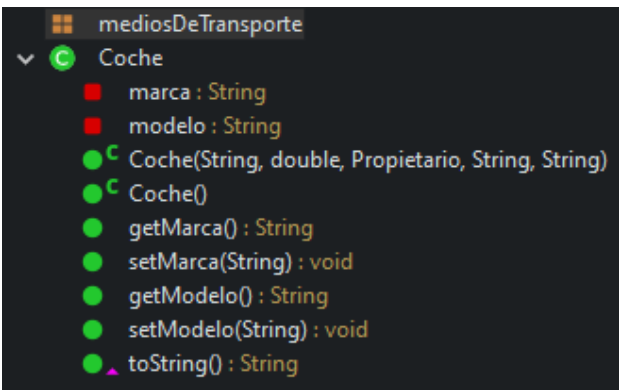
Atributos:

- matricula (String)
- kmsRecorridos (double)
- Titular (Propietario)

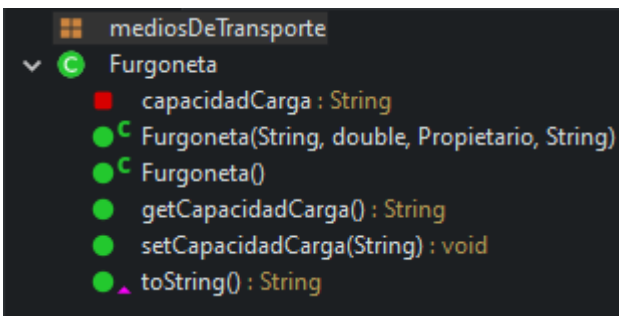
Métodos:

- **Constructor:** Los parámetros de entrada son la **matrícula y el titular**. Los **kmsRecorridos** inicialmente valdrá cero y en cada viaje se incrementará.
- **Métodos getters y setters**
- **Método incrementarkms(double cantidad)** : Incrementa los kilómetros recorridos por el vehículo, en el viaje.
- **Método disminuirkms(double cantidad)** : Disminuye los kilómetros recorridos por el vehículo.
- **toString() → mostrarVehículo()** -> método abstracto que mostrará los datos del vehículo.

Clase Coche – paquete mediosdetransporte:

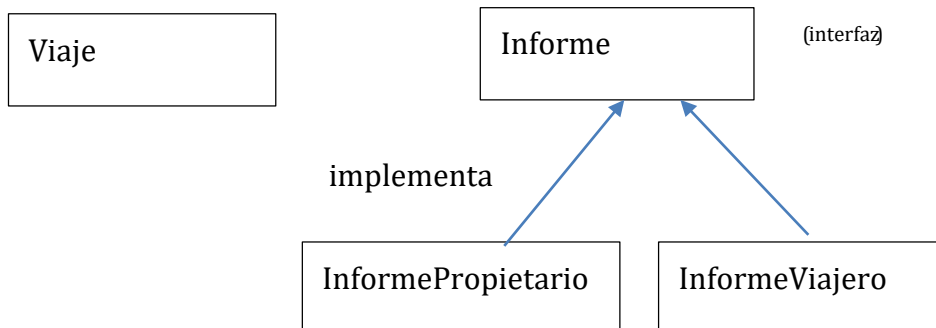
	<p>Atributos:</p> <ul style="list-style-type: none"> • marca (String) • modelo (String) <p>Métodos:</p> <ul style="list-style-type: none"> • <u>Constructor:</u> Los parámetros de entrada son la matrícula, marca, modelo y el titular. Los kmsRecorridos inicialmente valdrá cero y en cada viaje se incrementará. • <u>Métodos getters y setters</u> • toString() → mostrarVehiculo() -> método que es implementado con todos los datos del vehículo.
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Clase Furgoneta – paquete mediosdetransporte:

	<p>Atributos:</p> <ul style="list-style-type: none"> • capacidadCarga (String) <p>Métodos:</p> <ul style="list-style-type: none"> • <u>Constructor:</u> Los parámetros de entrada son la matrícula, capacidad de carga y el titular. Los kmsRecorridos inicialmente valdrá cero y en cada viaje se incrementará. • <u>Métodos getters y setters</u> • toString() → mostrarVehiculo() -> método que es implementado con todos los datos del vehículo.
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

PAQUETE RECORRIDOS:

Este paquete se encarga de gestionar los viajes y los informes ligados a esos viajes

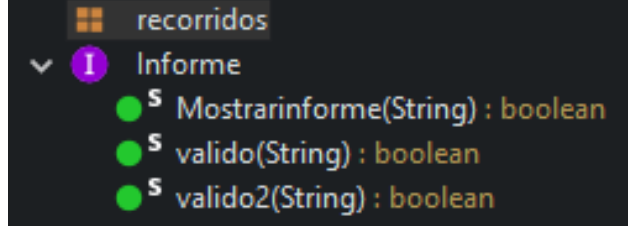
**Clase Viaje – paquete recorridos:**

<pre> package recorridos; import java.util.ArrayList; import java.util.List; public class Viaje { String origenViaje; String destinoViaje; double kmsViaje; double gastoPorPasajero; ArrayList<Persona> pasajeros; Vehiculo transporte; Viaje(String, String, double, double, ArrayList<Persona>, Vehiculo) { // Constructor } Viaje() { // Constructor vacío } String getOrigenViaje() { return origenViaje; } void setOrigenViaje(String) { // Setter } String getDestinoViaje() { return destinoViaje; } void setDestinoViaje(String) { // Setter } double getKmsViaje() { return kmsViaje; } void setKmsViaje(double) { // Setter } double getGastoPorPasajero() { return gastoPorPasajero; } void setGastoPorPasajero(double) { // Setter } ArrayList<Persona> getPasajeros() { return pasajeros; } void setPasajeros(ArrayList<Persona>) { // Setter } Vehiculo getTransporte() { return transporte; } void setTransporte(Vehiculo) { // Setter } String toString() { // Método toString } } </pre>	<p>Atributos:</p> <ul style="list-style-type: none"> • origenViaje (String) • destinoViaje (String) • kmsViaje (double) • gastoPorPasajero(double) • pasajeros(ArrayList<Persona>) • transporte (Vehiculo) <p>Métodos:</p> <ul style="list-style-type: none"> • <u>Constructor:</u> Los parámetros de entrada son el origen, el destino y los kms del viaje, el gasto por pasajero, los pasajeros y el transporte utilizado. • <u>Métodos getters y setters</u> • <i>toString()</i> → <i>mostrarViaje()</i> -> método que muestra todos los datos del viaje.
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

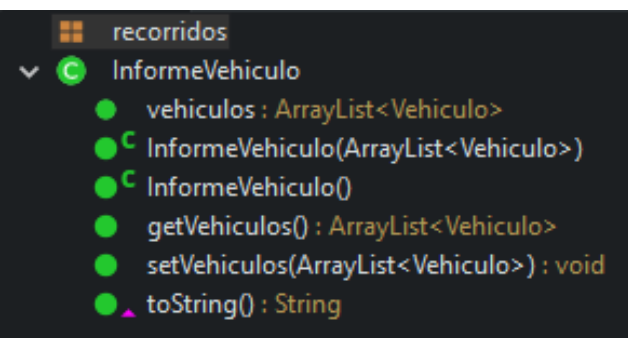
En el constructor debes realizar los siguientes pasos:

- Incrementa el gasto de todos los pasajeros de acuerdo al atributo gasto por pasajero.
- Incrementa el ingreso del propietario del coche (calcula el número de pasajeros y multiplícalo por el gasto por pasajero).
- Incrementa el número de kms realizados por el Vehiculo.

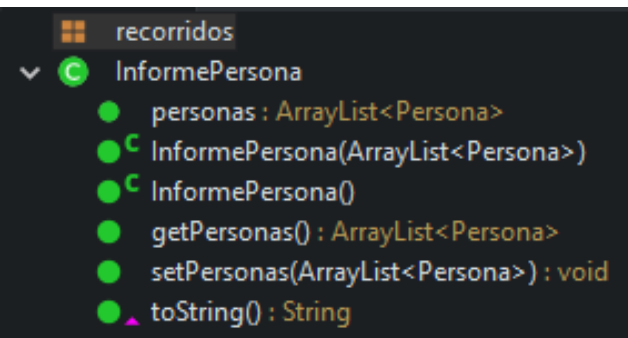
Informe: (interfaz) – paquete recorridos

	<p>Mostrarinforme() mostrará el informe de los vehículos o de las personas (aún no se sabe)</p>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------

Clase InformeVehiculo – paquete recorridos:

	<p>Atributos:</p> <ul style="list-style-type: none"> vehiculos (ArrayList<Vehiculo>) <p>Métodos:</p> <ul style="list-style-type: none"> <u>Constructor:</u> El parámetro de entrada es un ArrayList de Vehiculos. toString() → <u>Mostrarinforme()</u> mostrará el informe de los vehículos.
------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Clase InformePersona – paquete recorridos:

	<p>Atributos:</p> <ul style="list-style-type: none"> personas (ArrayList<Persona>) <p>Métodos:</p> <ul style="list-style-type: none"> <u>Constructor:</u> El parámetro de entrada es un ArrayList de personas. <p>toString() → <u>Mostrarinforme()</u> mostrará el informe de las personas.</p>
-------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------