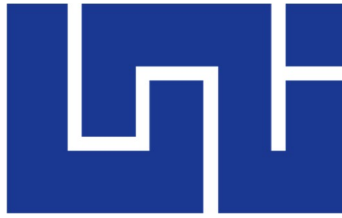


Universidad Nacional de Ingeniería



Introducción a la Programación

Nombre: Steven Alejandro Alvarez Avendaño

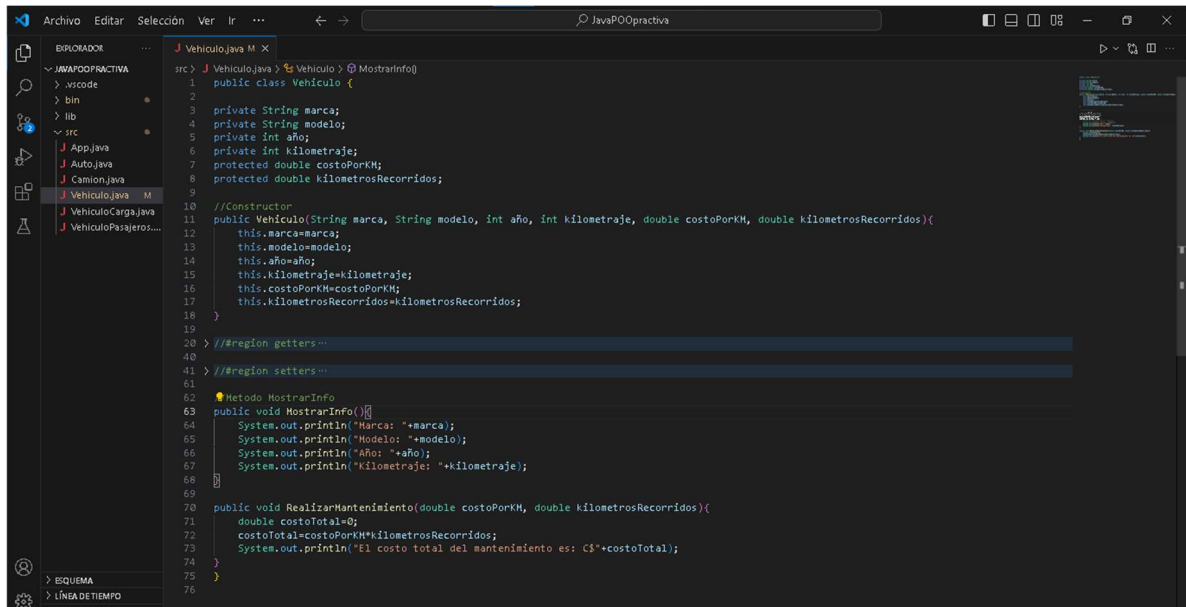
N.º Carnet: 2024-1675U

Grupo: 1M7-S

Fecha: 25/05/2024

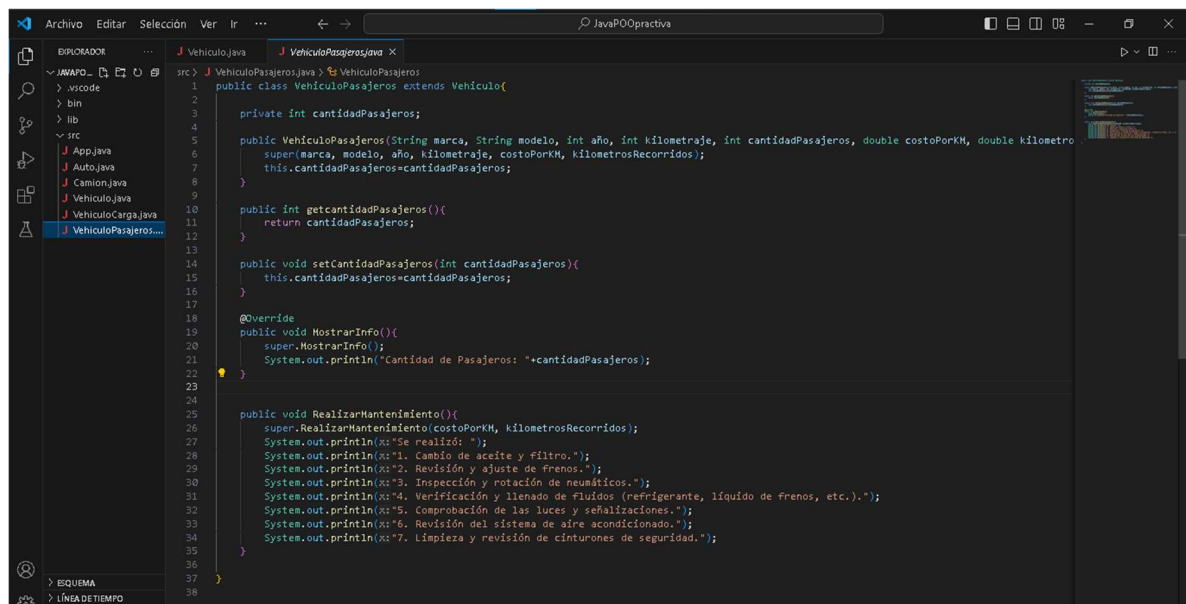
Elaboración de Guía sobre Herencia y Encapsulamiento

- 1- Creación de la Clase Vehículo junto con el método de realizarMantenimiento que calcule el costo del mantenimiento:



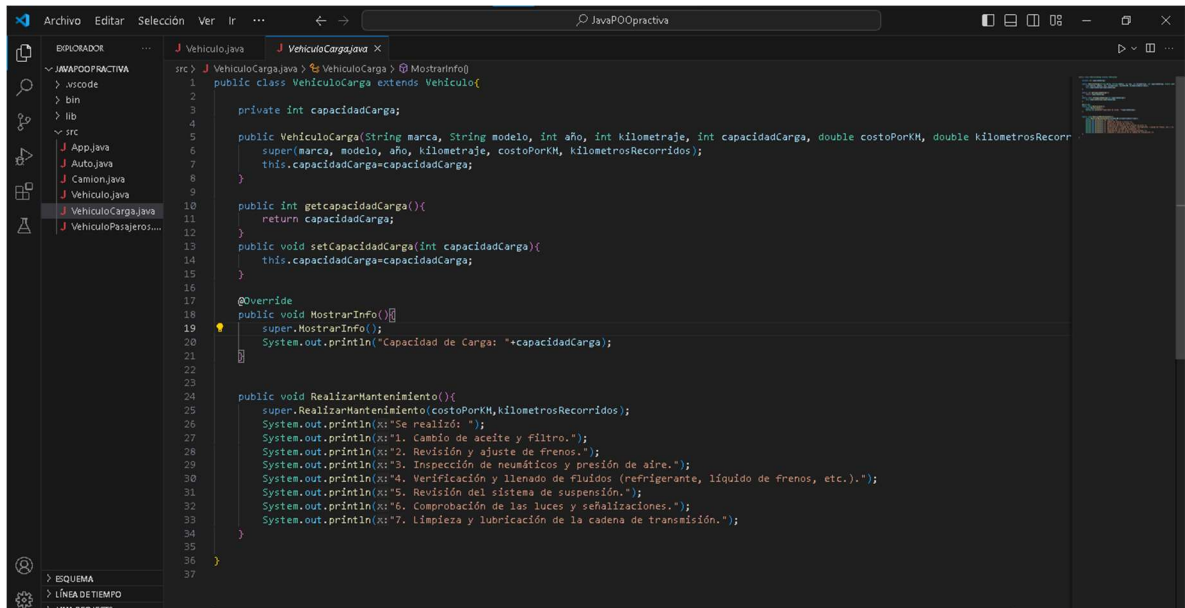
```
1 public class Vehiculo {
2
3     private String marca;
4     private String modelo;
5     private int año;
6     private int kilometraje;
7     protected double costoPorKM;
8     protected double kilometrosRecorridos;
9
10    //Constructor
11    public Vehiculo(String marca, String modelo, int año, int kilometraje, double costoPorKM, double kilometrosRecorridos){
12        this.marca=marca;
13        this.modelo=modelo;
14        this.año=año;
15        this.kilometraje=kilometraje;
16        this.costoPorKM=costoPorKM;
17        this.kilometrosRecorridos=kilometrosRecorridos;
18    }
19
20    > //region getters...
40
41    > //region setters...
61
62    //Metodo MostrarInfo
63    public void MostrarInfo(){
64        System.out.println("Marca: "+marca);
65        System.out.println("Modelo: "+modelo);
66        System.out.println("Año: "+año);
67        System.out.println("Kilometraje: "+kilometraje);
68    }
69
70    public void RealizarMantenimiento(double costoPorKM, double kilometrosRecorridos){
71        double costoTotal=0;
72        costoTotal=costoPorKM*kilometrosRecorridos;
73        System.out.println("El costo total del mantenimiento es: C$"+costoTotal);
74    }
75
76 }
```

- 2- Creación de la clase Vehículo Pasajeros heredando de la clase Vehículo y la sobre escritura del método:



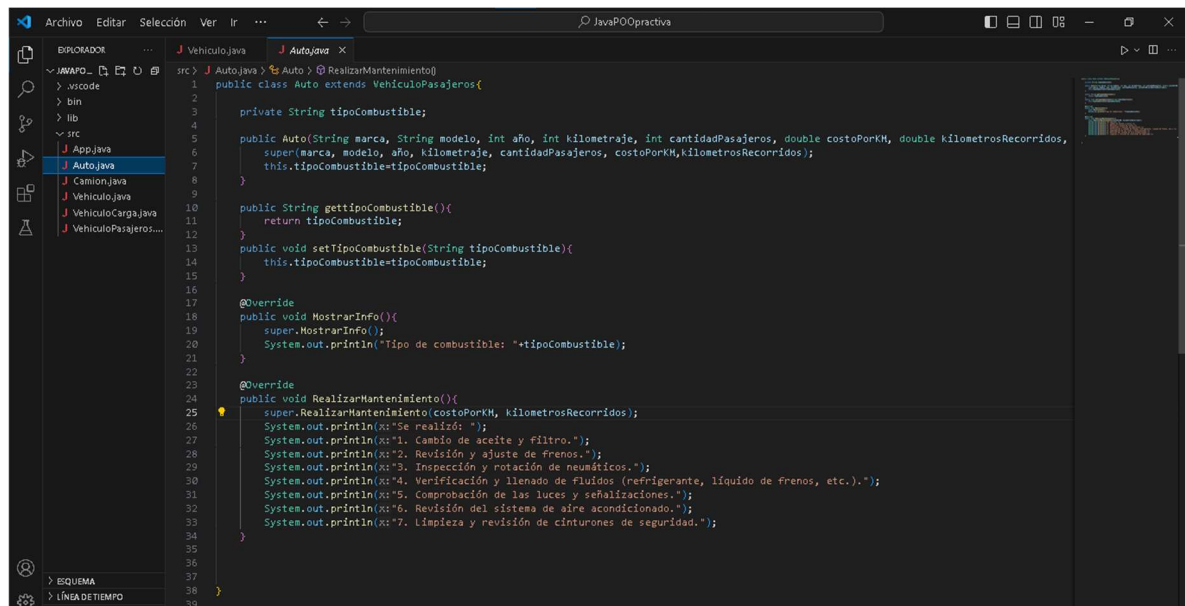
```
1 public class VehiculoPasajeros extends Vehiculo{
2
3     private int cantidadPasajeros;
4
5     public VehiculoPasajeros(String marca, String modelo, int año, int kilometraje, int cantidadPasajeros, double costoPorKM, double kilometro
6     {
7         super(marca, modelo, año, kilometraje, costoPorKM, kilometrosRecorridos);
8         this.cantidadPasajeros=cantidadPasajeros;
9     }
10
11     public int getCantidadPasajeros(){
12         return cantidadPasajeros;
13     }
14
15     public void setCantidadPasajeros(int cantidadPasajeros){
16         this.cantidadPasajeros=cantidadPasajeros;
17     }
18
19     @Override
20     public void MostrarInfo(){
21         super.MostrarInfo();
22         System.out.println("Cantidad de Pasajeros: "+cantidadPasajeros);
23     }
24
25     public void RealizarMantenimiento(){
26         super.RealizarMantenimiento(costoPorKM, kilometrosRecorridos);
27         System.out.println("Se realizó: ");
28         System.out.println("1. Cambio de aceite y filtro.");
29         System.out.println("2. Revisión y ajuste de frenos.");
30         System.out.println("3. Inspección y rotación de neumáticos.");
31         System.out.println("4. Verificación y llenado de fluidos (refrigerante, líquido de frenos, etc.).");
32         System.out.println("5. Comprobación de las luces y señalizaciones.");
33         System.out.println("6. Revisión del sistema de aire acondicionado.");
34         System.out.println("7. Limpieza y revisión de cinturones de seguridad.");
35     }
36
37 }
38 }
```

3- Creación de la clase Vehículo Carga heredando de la clase Vehículo:

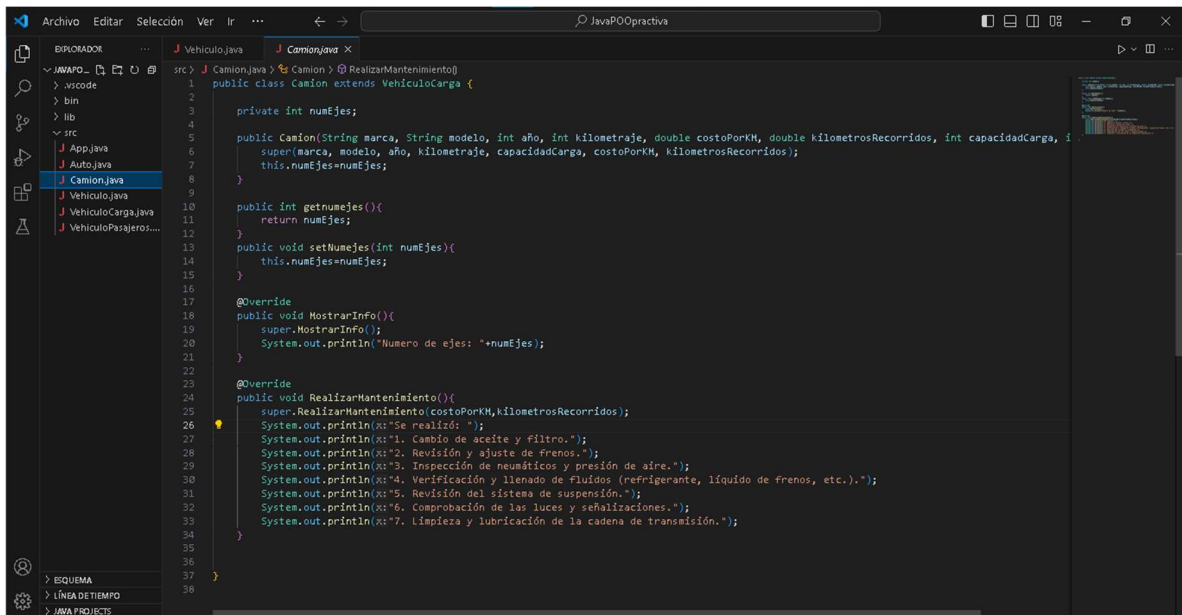


```
1 public class VehiculoCarga extends Vehiculo{
2
3     private int capacidadCarga;
4
5     public VehiculoCarga(String marca, String modelo, int año, int kilometraje, int capacidadCarga, double costoPorKM, double kilometrosRecorr
6     {
7         super(marca, modelo, año, kilometraje, costoPorKM, kilometrosRecorridos);
8         this.capacidadCarga=capacidadCarga;
9     }
10
11     public int getCapacidadCarga(){
12         return capacidadCarga;
13     }
14     public void setCapacidadCarga(int capacidadCarga){
15         this.capacidadCarga=capacidadCarga;
16     }
17
18     @Override
19     public void MostrarInfo(){
20         super.MostrarInfo();
21         System.out.println("Capacidad de Carga: "+capacidadCarga);
22     }
23
24     public void RealizarMantenimiento(){
25         super.RealizarMantenimiento(costoPorKM,kilometrosRecorridos);
26         System.out.println("Se realizó: ");
27         System.out.println("1. Cambio de aceite y filtro.");
28         System.out.println("2. Revisión y ajuste de frenos.");
29         System.out.println("3. Inspección de neumáticos y presión de aire.");
30         System.out.println("4. Verificación y llenado de fluidos (refrigerante, líquido de frenos, etc.).");
31         System.out.println("5. Revisión del sistema de suspensión.");
32         System.out.println("6. Comprobación de las luces y señalizaciones.");
33         System.out.println("7. Limpieza y lubricación de la cadena de transmisión.");
34     }
35 }
36
37 }
```

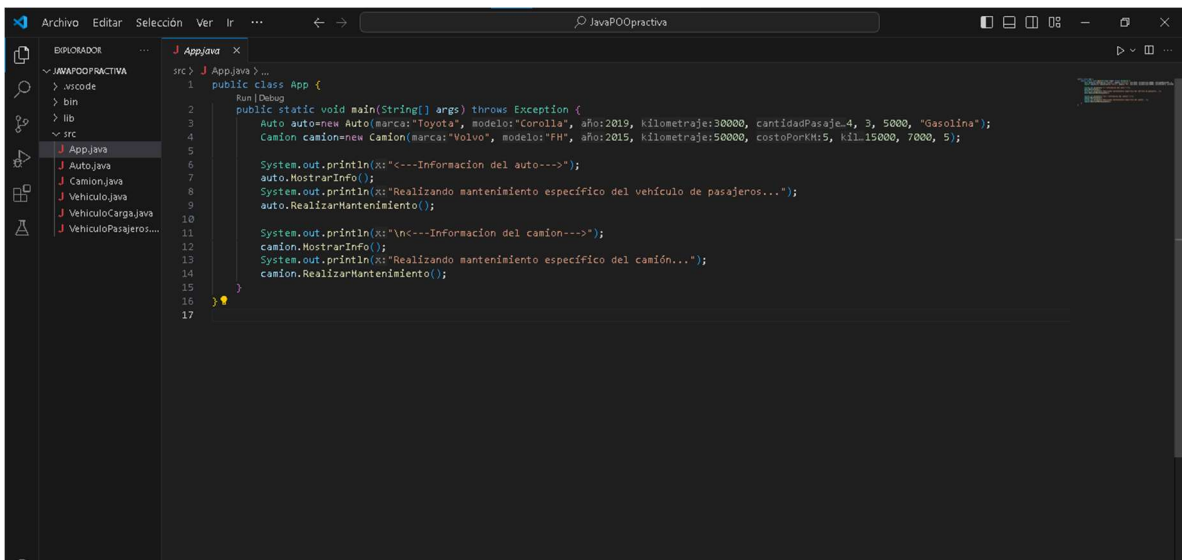
4- Creación de la clase Auto y Camión heredando de sus respectivas clases de VehiculoPasajero y VehiculoCarga:



```
1 public class Auto extends VehiculoPasajeros{
2
3     private String tipoCombustible;
4
5     public Auto(String marca, String modelo, int año, int kilometraje, int cantidadPasajeros, double costoPorKM, double kilometrosRecorridos,
6     {
7         super(marca, modelo, año, kilometraje, cantidadPasajeros, costoPorKM,kilometrosRecorridos);
8         this.tipoCombustible=tipoCombustible;
9     }
10
11     public String getTipoCombustible(){
12         return tipoCombustible;
13     }
14     public void setTipoCombustible(String tipoCombustible){
15         this.tipoCombustible=tipoCombustible;
16     }
17
18     @Override
19     public void MostrarInfo(){
20         super.MostrarInfo();
21         System.out.println("Tipo de combustible: "+tipoCombustible);
22     }
23
24     @Override
25     public void RealizarMantenimiento(){
26         super.RealizarMantenimiento(costoPorKM, kilometrosRecorridos);
27         System.out.println("Se realizó: ");
28         System.out.println("1. Cambio de aceite y filtro.");
29         System.out.println("2. Revisión y ajuste de frenos.");
30         System.out.println("3. Inspección y rotación de neumáticos.");
31         System.out.println("4. Verificación y llenado de fluidos (refrigerante, líquido de frenos, etc.).");
32         System.out.println("5. Comprobación de las luces y señalizaciones.");
33         System.out.println("6. Revisión del sistema de aire acondicionado.");
34         System.out.println("7. Limpieza y revisión de cinturones de seguridad.");
35     }
36 }
37
38 }
```



- 5- Llamamiento en la clase main y pasándole los argumentos a cada objeto, para luego mandar a imprimir los datos con el método MostrarInformacion y mostrar los detalles del mantenimiento con el método RealizarMantenimiento:



- 6- Con la guía ya elaborada pasamos a subirlo a un repositorio:

The screenshot shows the Visual Studio Code interface with a project named 'JavaPOOpractiva'. The Explorer sidebar on the left shows the project structure: 'src' containing 'App.java', 'Auto.java', 'Camion.java', 'Vehiculo.java', 'VehiculoCarga.java', and 'VehiculoPasajeros.java'. The main editor displays the 'App.java' file with the following code:

```
src > J App.java > App > main(String[] args) {
1 public class App {
```

The integrated terminal at the bottom shows the execution of Git commands to initialize a repository, add files, and push to GitHub:

```
PS D:\Universidad\UNI\Programacion\Tareas\JavaPOOpractiva> git init
Initialized empty Git repository in D:\Universidad\UNI\Programacion\Tareas\JavaPOOpractiva\.git\
PS D:\Universidad\UNI\Programacion\Tareas\JavaPOOpractiva> git remote add origin https://github.com/Stevenalv123/GuiaHerenciaEncapsulamientoPOO.git
PS D:\Universidad\UNI\Programacion\Tareas\JavaPOOpractiva> git add .
PS D:\Universidad\UNI\Programacion\Tareas\JavaPOOpractiva> git commit -m "first commit"
[master (root-commit) e5135a5] first commit
13 files changed, 246 insertions(+)
 create mode 100644 .vscode/settings.json
 create mode 100644 bin/App.class
 create mode 100644 bin/Vehiculo.class
 create mode 100644 bin/VehiculoCarga.class
 create mode 100644 bin/VehiculoPasajeros.class
 create mode 100644 src/App.java
 create mode 100644 src/Auto.java
 create mode 100644 src/Camion.java
 create mode 100644 src/Vehiculo.java
 create mode 100644 src/VehiculoCarga.java
 create mode 100644 src/VehiculoPasajeros.java
PS D:\Universidad\UNI\Programacion\Tareas\JavaPOOpractiva> git branch -M "main"
PS D:\Universidad\UNI\Programacion\Tareas\JavaPOOpractiva> git push -u origin main
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 4 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (18/18), 7.55 KiB | 552.00 KiB/s, done.
Total 18 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), done.
To https://github.com/Stevenalv123/GuiaHerenciaEncapsulamientoPOO.git
 * [new branch]    main -> main
 branch 'main' set up to track 'origin/main'.
PS D:\Universidad\UNI\Programacion\Tareas\JavaPOOpractiva>
```

7- Proyecto Subido:

