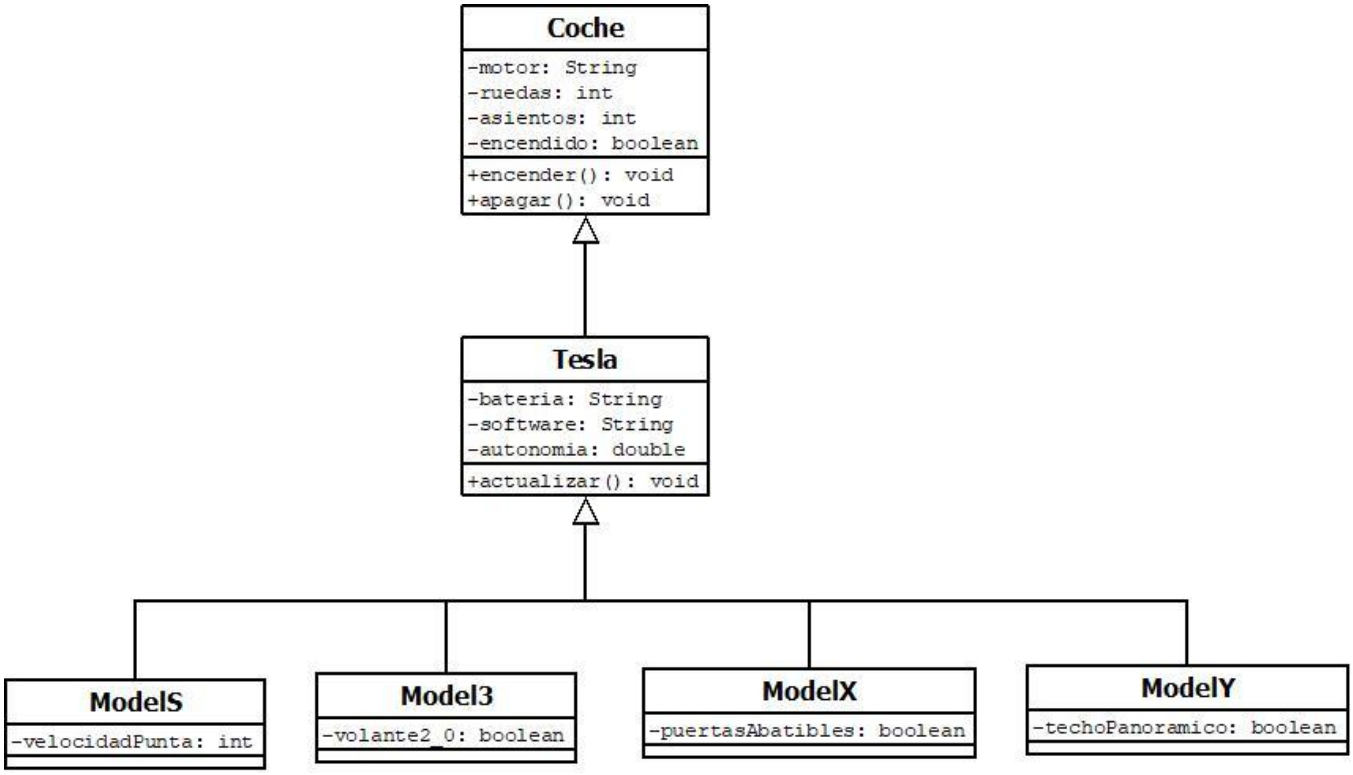
He creado una herencia simple de un Coche padre con un hijo "Tesla" del cual heredan 4 Clases: "ModelS","Model3","ModelX" y "ModelY":



En este caso la clase "Tesla" hereda tanto los atributos como los métodos de "Coche", y se pueden acceder a ellos desde un objeto Tesla.

La Clase "Tesla" añade 3 nuevos atributos y un método al cual podrán también acceder los hijos si se crean objetos de ellos.

La herencia múltiple en java no es posible, si se puede hacer una sucesión de herencias como en el ejemplo.
Para poder tener clases que sean hijas de más de una clase se puede usar las interfaces, por ejemplo, "Tesla" puede heredar de "Coche" y a parte se puede crear una interfaz de "esElectrico", el cual pueda tener métodos abstractos que haya que implementar en Tesla y sus hijos.

Diferencia entre sobrecarga y sobrescritura

Sobrecarga: Podemos crear varios métodos con el mismo nombre pero con diferentes parámetros, por ejemplo, podemos crear un constructor de "Tesla" que solo pida los atributos que herede de "Coche" y deja los demás atributos como nulo hasta que se le de un valor.

```
public Tesla(String motor, int ruedas, int asientos, boolean encendido) {
    super(motor, ruedas, asientos, encendido);
}
```

Pudiendo así crear dos objetos "Tesla" con diferentes parámetros iniciales:

```
Tesla miTesla = new Tesla("electrico", 4, 5, false);
Tesla miTesla2 = new Tesla("Electrico", 6, 7, true, "JHIE-3", "6.2", 500);
```

Sobreescritura: en el ejemplo de Coche podemos hacer que el "Model3" sobreescriba el método actualizar de su padre "Tesla":

```
@Override
public void actualizar() {
    System.out.println("Model 3 actualizado");
}
```

En este caso, cuando llamemos al método hará lo que le hemos indicado en la propia clase.

```
miModel3.actualizar();
```

```
Model 3 actualizado
```

Almacenamiento de Datos

- **Lista:** Son una secuencia de elementos que ocupan una posición determinada.

- **Pila:** Son como las listas pero pueden definirse como una sucesión de varios elementos del mismo tipo, solo pudiendo acceder a los datos desde la cima, siendo así el primer elemento que entra es el último en salir.

- **Colas:** Son como las pilas, pero en este caso el primer elemento que entra es el primero en salir.

- **Vector:** Un vector es como un ArrayList pero con un mayor número de métodos.

Ejemplos en "PruebaTesla".