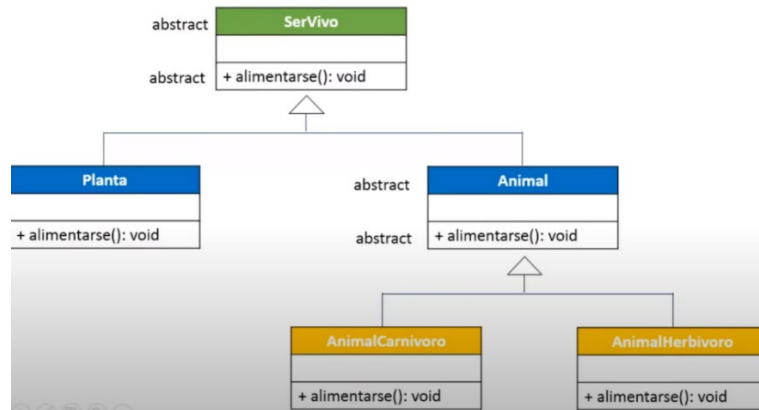


## APARTADO1 – CLASE ABSTRACTA.

1. Realiza la siguiente implementación. Atendiendo al uso de las clases/métodos abstractos en “SerVivo” y “Animal” como afecta eso cuando construimos dicha estructura.



2. Una empresa de ropa a mayoreo, ofrece ciertos descuentos a sus clientes. Los clasifica en tres grupos. Cada grupo fue clasificado dependiendo de la antigüedad que llevan comprando a la empresa. Mayor antigüedad mayores beneficios.

- GrupoA: 10%
- GrupoB: 5%
- GrupoC: 2%

Utilizando la clase abstracta:

```
public abstract class Cliente {

    public static double DESCUENTO_GRUPOA = 0.1; // 10%
    public static double DESCUENTO_GRUPOB = 0.05; // 5%
    public static double DESCUENTO_GRUPOC = 0.02; // 2%

    private String nombre;

    public Cliente(String nombre) {
        super();
        this.nombre = nombre;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

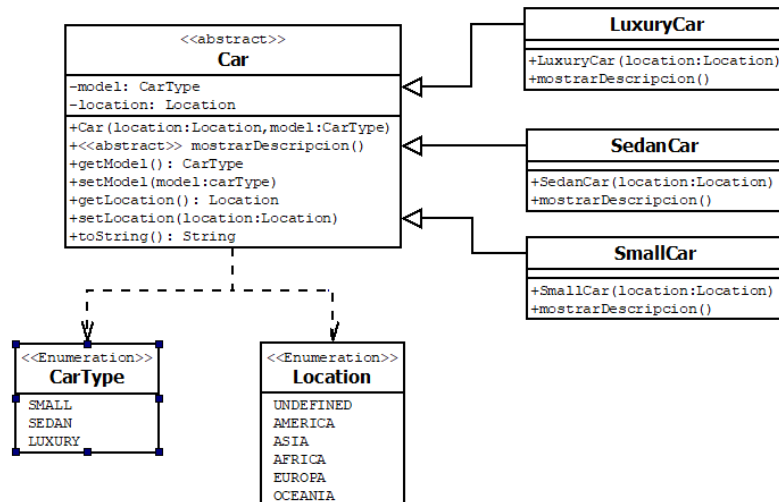
    public abstract double aPagar(double cuantia);
}
```

Crear 3 clases: “ClienteGrupoA”, “ClienteGrupoB”, “ClienteGrupoC” que utilice y complete la clase Cliente.

3. En el siguiente diagrama correspondiente a una factoría de coches de diferente calidad.

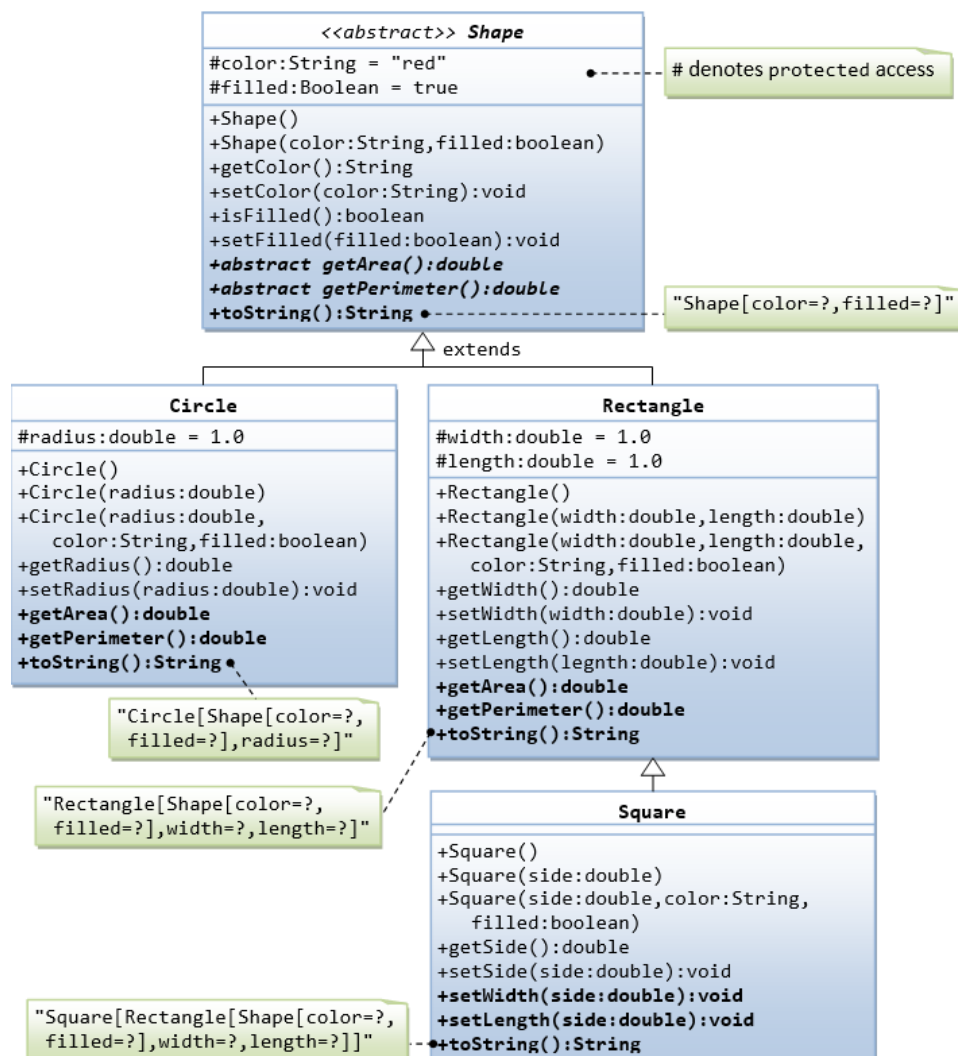
Cómo se puede apreciar uno de los métodos es abstract poniendo toda la clase en dicho estado (abstracto) y forzando que todas las clases que heredan de esta deban desarrollar/codificar

dicho método. Codifique el siguiente diagrama de clases introduciendo cualquier descripción para cada coche.



4. Declara una clase abstracta Legislador que herede de la clase Persona, con un atributo provinciaQueRepresenta (tipo String) y al menos 2 atributos más a su elección. Declara un método abstracto getCamaraEnQueTrabaja. Crea dos clases concretas que hereden de Legislador: la clase Diputado y la clase Senador que sobrescriban los métodos abstractos necesarios. Crea una lista de legisladores y muestra por pantalla la cámara en que trabajan haciendo uso del polimorfismo.

5. Implementa el siguiente diagrama de clases:



## APARTADO2 – USO DE INTERFACES.

1. Crea una interfaz llamada “Saludos” con los siguientes métodos:

- saludoFormal() : void.
- saludoInformal() : void.
- despedidaFormal() : void.
- despedidaInformal() : void.

A continuación, crea 3 clases “Español”, “Portugues” e “Ingles”. Y debemos hacer que ambas clases utilicen la interfaz anterior y posteriormente codifica cada una de ellas.

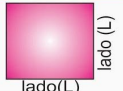
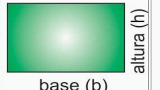
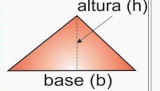
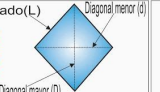
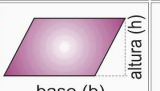
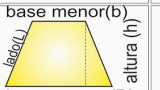
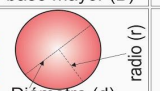
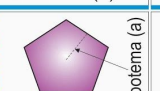
2. Define la siguiente interfaz llamada “Figura”

```
public interface Figura {

    public double calcularArea();
    public double calcularPerimetro();

}
```

A continuación, codifica una clase por cada una de las figuras geométricas que verás en la siguiente figura implementado todas ellas la interfaz anterior.

FORMULARIO DE ÁREAS Y PERÍMETROS		
<b>CUADRADO</b> 	<b>ÁREA</b> $A = L \times L$	<b>PERÍMETRO</b> $P = L + L + L + L$
<b>RECTÁNGULO</b> 	<b>ÁREA</b> $A = b \times h$	<b>PERÍMETRO</b> $P = b + b + h + h$
<b>TRIÁNGULO</b> 	<b>ÁREA</b> $A = \frac{b \times h}{2}$	<b>PERÍMETRO</b> $P = L + L + L$
<b>ROMBO</b> 	<b>ÁREA</b> $A = D \times d$	<b>PERÍMETRO</b> $P = L + L + L + L$
<b>ROMBOIDE</b> 	<b>ÁREA</b> $A = b \times h$	<b>PERÍMETRO</b> $P = b + b + h + h$
<b>TRAPECIO</b> 	<b>ÁREA</b> $A = \frac{h(B + b)}{2}$	<b>PERÍMETRO</b> $P = B + b + L + L$
<b>CÍRCULO</b> 	<b>ÁREA</b> $A = \pi \times r^2$	<b>CIRCUNFERENCIA</b> $C = \pi \times d$
<b>POLIGONO + 5</b> 	<b>ÁREA</b> $A = \frac{p \times a}{2}$	<b>PERÍMETRO</b> $P = L \times \# \text{ lados}$

3. Partiendo de la siguiente interfaz que recrear las acciones de una bicicleta:

1. **aumentarMarcha**: Si es posible se aumentará una marcha.
2. **decrementarMarcha**: Si es posible se bajará una marcha.
3. **acelerar**: Si es posible se aumentará la velocidad según lo indicado. (km/s)
4. **decelerar**: Si es posible se reducirá la velocidad según lo indicado. (km/s)
5. **frenar**: Bajar la velocidad a 0, indicándolo por pantalla cuando se ha frenado del todo, pero esperará un tiempo proporcional hasta alcanzar dicha reducción:

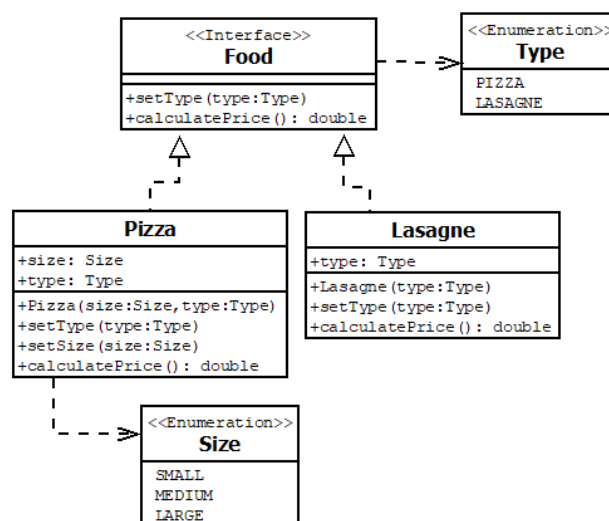
```
public interface Bicicleta {

    public void aumentarMarcha();
    public void decrementarMarcha();
    public void acelerar(int incremento);
    public void decelerar(int decremento);
    public void frenar();
}
```

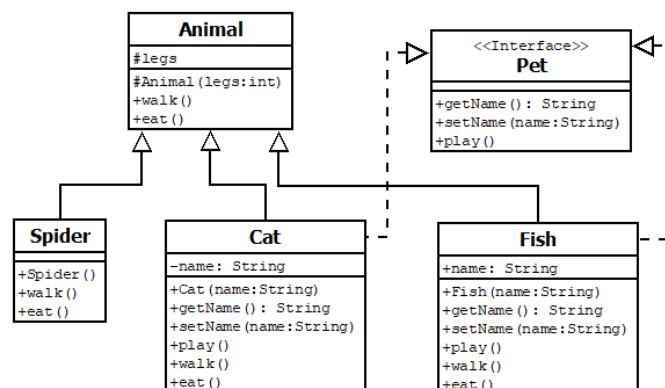
Crea 2 clases de bicicletas:

- BicicletaDeCarreras: Tendrá 5 marchas y tarda en frenar 1 segundo por cada km/s
- BicicletaUrbana: No tiene marchas y tarda en frenar 2 segundos por cada km/s

4. Implemente el siguiente diagrama UML que utiliza una interfaz. Creando tanto la interfaz “Food” como las clases “Pizza” y “Lasagne”.



5. Implemente el siguiente diagrama UML que combina herencia y uso de interfaces.



6. Implemente en Java el siguiente diagrama UML
- No es necesario codificar cada uno de los métodos, solo se pide que se implemente el siguiente esquema UML.

