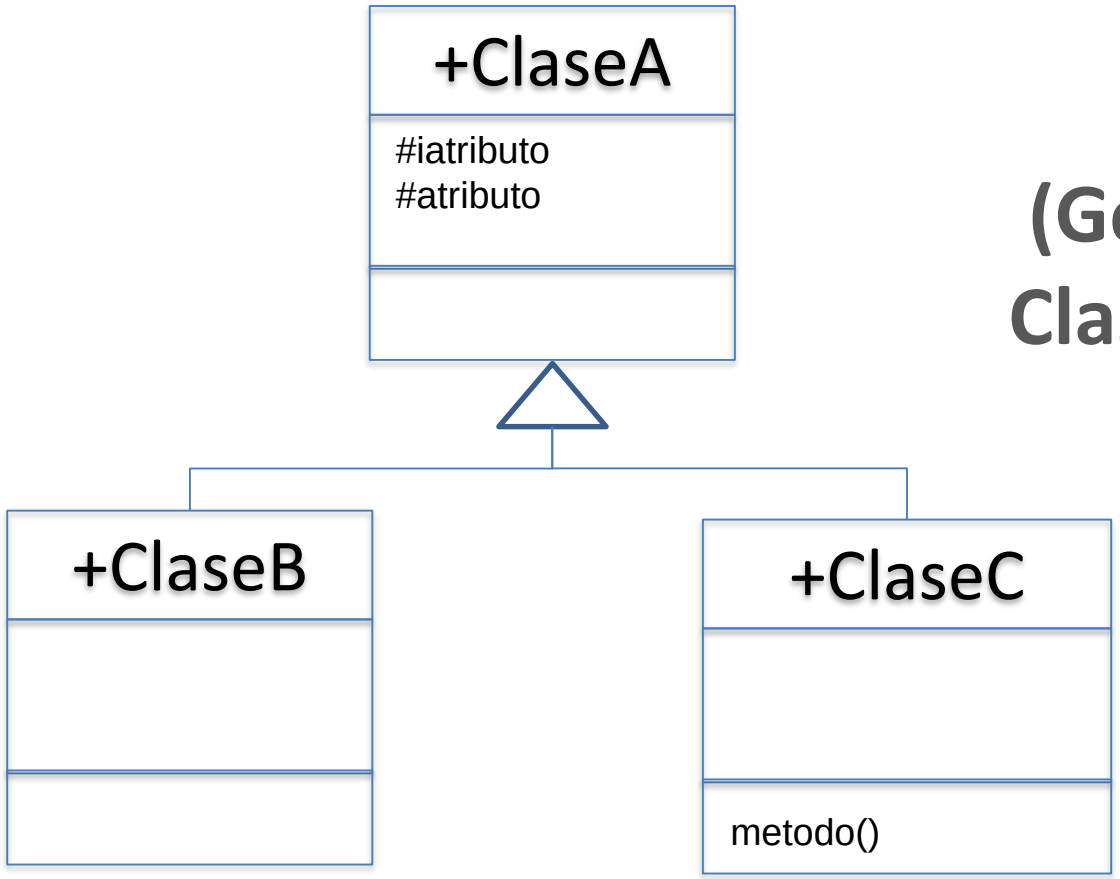
Universidad Distrital Francisco José de caldas
Tecnología en Sistematización de Datos



Herencia (Generalización) Clases y Métodos Abstractos

Programación Multinivel
Sonia Alexandra Pinzón Nuñez

Contenido

- Ejercicio Vehículos
- Análisis
- Concepto Herencia
- Herencia - Diseño
- Concepto Clases Abstractas
- Métodos abstractos
- Ejemplo Vehículos

Ejercicio Vehículos

Registrar vehículos(Motos, Autos), calcular impuesto

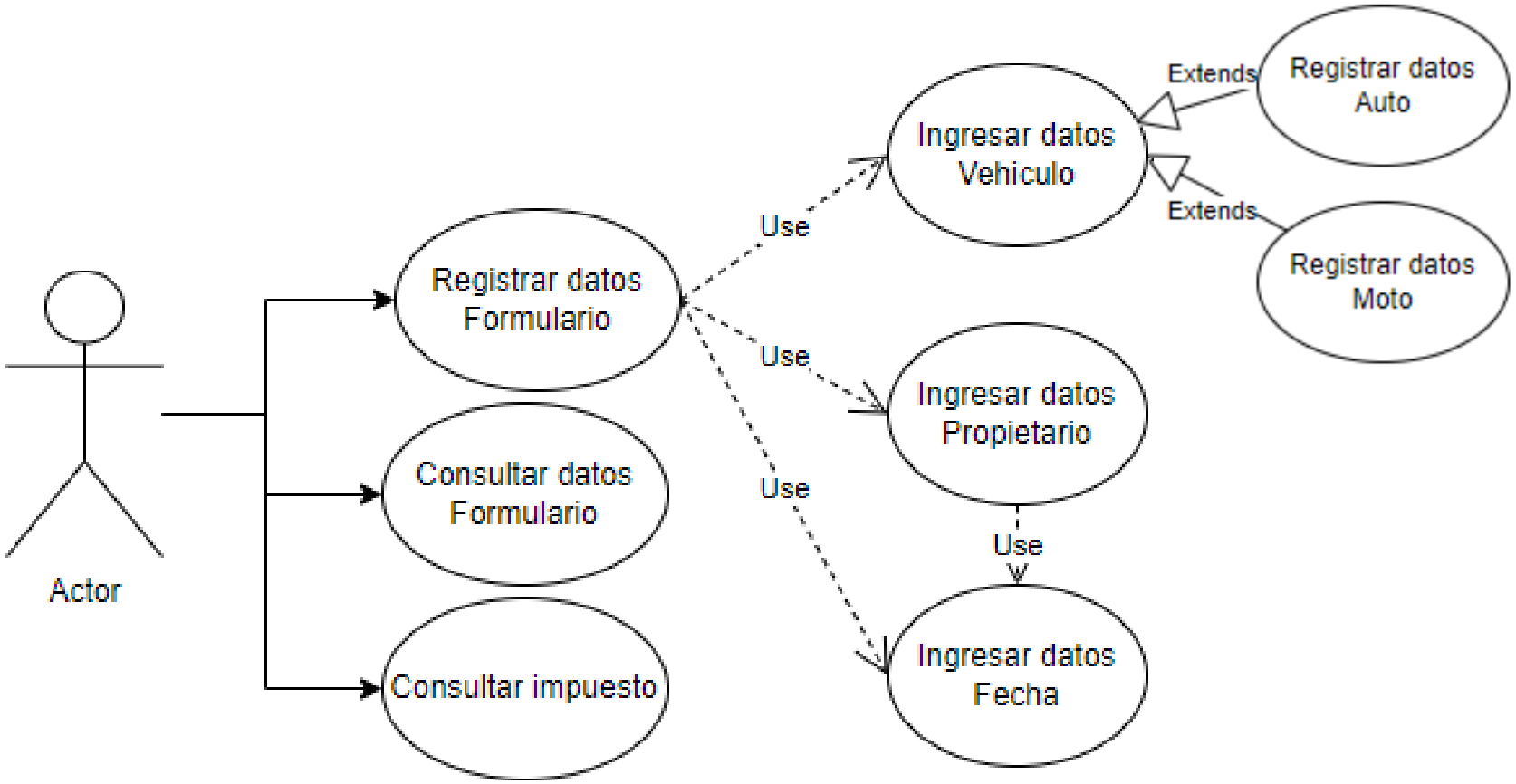
Criterio	impuesto
Cilindraje >125	10% del valor
Otros	0

Criterio	impuesto
Modelo<2000	5% del valor
Otros	10% del valor

Se debe generar el Formulario con los siguientes datos(NroFormulario, Fecha, Datos Propietario (id, nombre, teléfono, fecha Nacimiento), Datos vehículo(placa, marca, modelo, valor))

Análisis

Registrar vehículos(Motos, Autos), calcular impuesto



Análisis

Registrar vehículos(Motos, Autos), calcular impuesto

Criterio	impuesto
Cilindraje >125	10% del valor
Otros	0

+Moto
- placa: String - marca: String - modelo: int - valor: double - cilindraje: double
+impuesto(): double

Criterio	impuesto
Modelo<2000	5% del valor
Otros	10% del valor

+Auto
- placa: String - marca: String - modelo: int - valor: double
+impuesto(): double

Herencia - Concepto

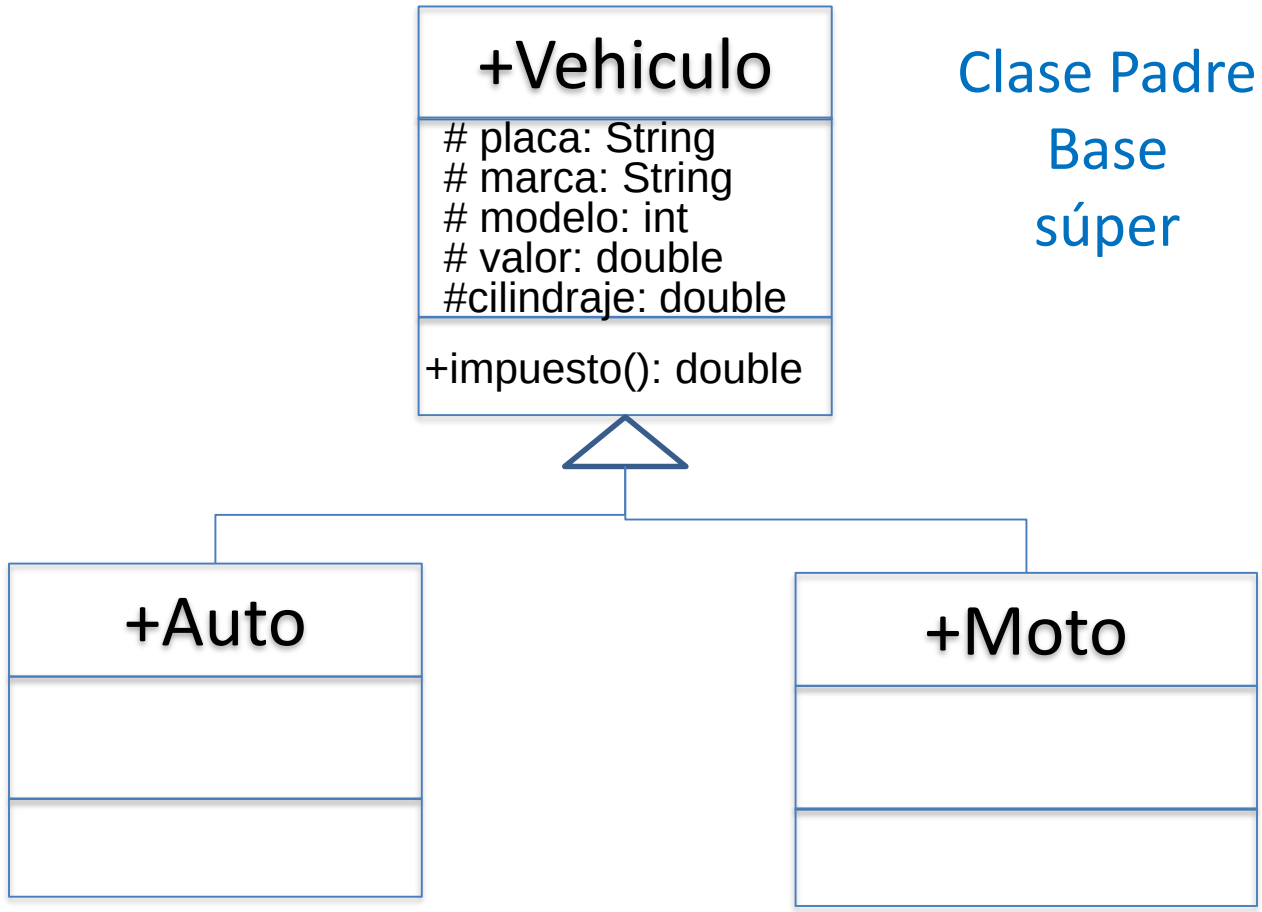
Cuando varios objetos tienen datos comunes (atributos), se puede agrupar esta información en otra clase para que pueda ser reutilizada.

Atributos Comunes

+Moto	+Auto
- placa: String - marca: String - modelo: int - valor: double - cilindraje: double	- placa: String - marca: String - modelo: int - valor: double
+ impuesto(): double	+ impuesto(): double

Herencia - Diseño

La Herencia es el mecanismo por el que se pueden crear nuevas clases a partir de clases definidas previamente, por medio de esta se hace reutilización de código



Clase Padre
Base
súper

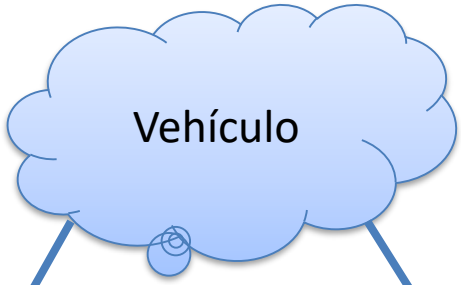
Clase hija
derivada
subclase

Clases Abstractas

Las clases abstractas son clases que permiten la generalización de estado (atributos) y comportamiento (métodos) pero sin la posibilidad de crear instancias (crear objetos), ya que no se necesitan. Generalmente las clases **Super** tienen a ser abstractas.

Clase Abstracta,
No existe un objeto específico que se pueda definir en esta clase

Clase Concreta,
Se pueden definir objetos tipo Auto y Moto, porque se conocen sus características específicas, ejemplo: placa, marca, modelo.



Objeto 1:BYP213, Mazda, 1998.

Objeto 2: AZY12B, Yamaha, 2000.

Clase abstracta: Sintaxis

Para definir una clase como abstracta solo se debe agregar la palabra reservada abstract antes de la clausula class y el nombre de la clase.

```
[Modo de acceso] abstract class nombre_Clase{

}
```

```
public abstract class Vehiculo{

}
```

Métodos abstractos:

Estos métodos corresponden al comportamiento definido en la clase abstracta, aunque no contienen implementación para facilitar a las clases concretas que asuman dicha implementación de acuerdo a sus necesidades.

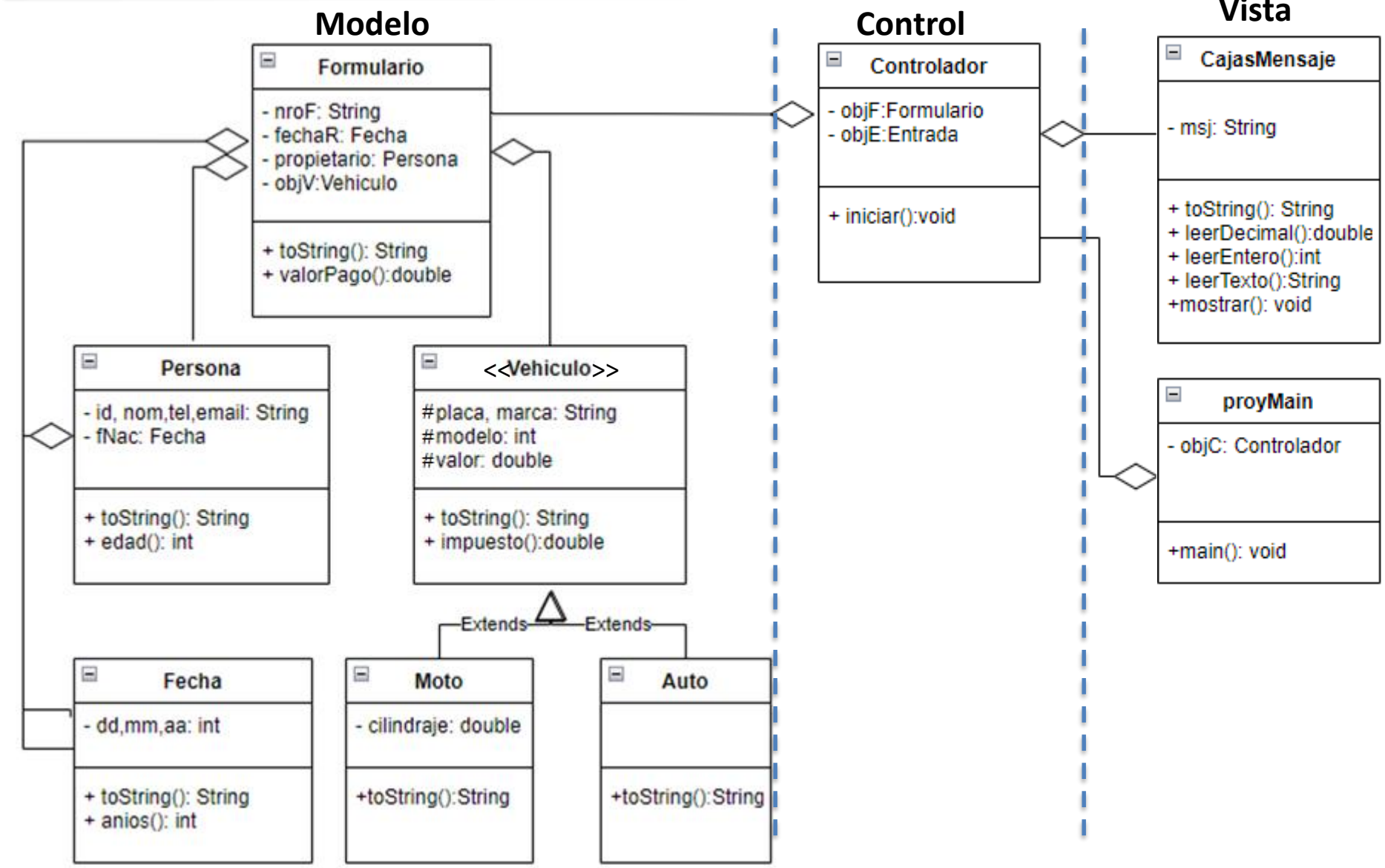
Sintaxis:

```
[Modo de acceso] abstract [tipo_retorno] nombre_Método();
```

Ejemplos:

```
public abstract void registrar();
public abstract double impuesto();
```

Herencia - Diseño



Herencia y Abstractas : Comparativo C++ y Java

C++

```cpp
#include <iostream>
#include <sstream>
#include <string>
using namespace std;
class Vehiculo {
protected:
  string placa,marca;
  int modelo;
  double valor;
public:
  Vehiculo(){placa=""; marca=""; modelo=0;valor=0;}
  Vehiculo(string p, string m, int model, double v){
      placa=p; marca=m; modelo= model; valor=v;}
  ~Vehiculo(){}
  string getPlaca(){ return placa; }
  void setPlaca(string p) { placa = p; }
  string getMarca(){ return marca; }
  void setMarca(string m) { marca = m; }
  int getModelo(){ return modelo; }
  void setModelo(double m) { modelo = m; }
  double getValor(){ return valor; }
  void setValor(double v) { valor = v; }
  string toString() {
   stringstream datos;
   datos.precision(2);datos<<fixed;
   datos<<"Placa: "<<placa<<"\nMarca: "<<marca;
   datos<<"\n Modelo: "<<modelo<<"\n Avaluo: "<<avaluo;
   return datos.str();
  }
  virtual double impuesto()=0;
};
```

Java

```java
package modelo;
public abstract class Vehiculo {
    protected String placa,marca;
    protected int modelo;
    protected double valor;
    public Vehiculo(String placa, String marca, int modelo, double valor) {
        this.placa = placa;   this.marca = marca;
        this.modelo = modelo; this.valor = valor;
    }
    public Vehiculo() {
        this.placa = "";  this.marca = "";
        this.modelo = 0;  this.valor = 0;
    }
    public String getPlaca() {return placa;    }
    public void setPlaca(String placa) { this.placa = placa;}
    public String getMarca() {return marca;   }
    public void setMarca(String marca) { this.marca = marca;}
    public int getModelo() {  return modelo;  }
    public void setModelo(int modelo) {  this.modelo = modelo;}
    public double getValor() {return valor;   }
    public void setValor(double valor) { this.valor = valor; }
    @Override
    public String toString() {
        return  "placa: " + placa +
                "\n marca=" + marca +
                "\n modelo=" + modelo +
                "\n valor=" + valor ;
    }
    public abstract double impuesto();
}
```

Herencia /Generalización

C++

Java

RELACIÓN DE HERENCIA

Operador de ámbito

Palabra reservada

```
class Auto: public Vehiculo {
public:
  Auto(): Vehiculo(){}
  Auto(string p, string m, , int modelo, double v): Vehiculo(p,m,model,v){}
  ~Auto(){}
 double impuesto() {
   if(modelo<2000)
        return valor*0.05;
      else
        return valor*0.1;
 }
};
```

```
public class Auto extends Vehiculo{
    public Auto(String placa, String marca, int modelo, double valor) {
       super(placa,marca,modelo, valor);
    }
    public Auto() {
        super();
    }
    @Override
    public double impuesto() {
       if(modelo<2000)
         return valor*0.05;
      else
          return valor*0.1;
    }
}
```

La palabra reservada **super** referencia el uso de la clase padre **Vehiculo**

Clase CajasMensaje (Vista)

```
public class cajasMensaje {
    String titulo;
    public cajasMensaje(String titulo) {...3 lines }
    public cajasMensaje() {...3 lines }
    public int leerEntero(String msg){
     int val;
     val=Integer.parseInt(JOptionPane.showInputDialog(null,
             msg, titulo, JOptionPane.INFORMATION_MESSAGE));
     return val;
    }
    public double leerDecimal(String msg){
     double val;
     val=Double.parseDouble(JOptionPane.showInputDialog(null,
             msg, titulo, 1));
     return val;
    }
    public String leerTexto(String msg){
     return JOptionPane.showInputDialog(null, msg, titulo,1);
    }
    public void mostrar(String msg){
     JOptionPane.showMessageDialog(null, msg,titulo,1);
    }
    public boolean confirmar(String msg){
      int resp= JOptionPane.showConfirmDialog(null, msg,titulo,
              JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
      if(resp== JOptionPane.YES_OPTION)
        return true;
      else
        return false;
    }
    public String getTitulo() {...3 lines }
    public void setTitulo(String titulo) {...3 lines }
}
```

Clase Vehiculo (Modelo)

```
package modelo;
public abstract class Vehiculo {
    protected String placa,marca;
    protected int modelo;
    protected double valor;
    public Vehiculo(String placa, String marca, int modelo, double valor) {
        this.placa = placa;   this.marca = marca;
        this.modelo = modelo; this.valor = valor;
    }
    public Vehiculo() {
        this.placa = "";  this.marca = "";
        this.modelo = 0;  this.valor = 0;
    }
    public String getPlaca() {return placa;    }
    public void setPlaca(String placa) { this.placa = placa;}
    public String getMarca() {return marca;   }
    public void setMarca(String marca) { this.marca = marca;}
    public int getModelo() {  return modelo;  }
    public void setModelo(int modelo) {  this.modelo = modelo;}
    public double getValor() {return valor;   }
    public void setValor(double valor) { this.valor = valor; }
    @Override
    public String toString() {
        return  "placa: " + placa +
                "\n marca=" + marca +
                "\n modelo=" + modelo +
                "\n valor=" + valor ;
    }
    public abstract double impuesto();
}
```

Clase Auto (Modelo)

```
public class Auto extends Vehiculo{
    public Auto(String placa, String marca, int modelo, double valor) {
       super(placa,marca,modelo, valor);
    }
    public Auto() {
        super();
    }
    @Override
    public double impuesto() {
       if(modelo< 1990 )
         return valor*0.05;
      else
          return valor*0.1;
    }
}
```

Clase Persona (Modelo)

```java
public class Persona {
    private String id, nom,tel,email;
    private Fecha fNac;

    public Persona(String id, String nom, String tel, String email, Fecha fNac)
     public Persona() {
        this.id = "";
        this.nom = "";
        this.tel = "";
        this.email = "";
        this.fNac = new Fecha();//Agregación
    }
    @Override
    public String toString() {
        return "Propietario: " +
                "\n Identicación: " + id +
                "\n Nombre: " + nom +
                "\n Teléfono: " + tel +
                "\n Email: " + email +
                "\n fecha Nacimiento: " + fNac.toString();
    }
    public int edad() {...2 lines }
    public String getId() {...3 lines }
    public void setId(String id) {...3 lines }
    public String getNom() {...3 lines }
    public void setNom(String nom) {...3 lines }
    public String getTel() {...3 lines }
    public void setTel(String tel) {...3 lines }
    public String getEmail() {...3 lines }
    public void setEmail(String email) {...3 lines }
    public Fecha getfNac() {...3 lines }
    public void setfNac(Fecha fNac) {...3 lines }
}
```

Clase Formulario (Modelo)

```java
public class Formulario {
    private String nroF;
    private Fecha fRegistro;
    private Persona propietario;
    private Vehiculo objV;
    public Formulario() {
        int cod= (int)(Math.random()*999 + 100);
        this.nroF = "FRM-"+cod;
        this.fRegistro = new Fecha();
        this.propietario = new Persona();
        this.objV = null;
    }
    @Override
    public String toString() {
        return "Formulario Impuesto\n" +
                " nroF:" + nroF +
                "\n fRegistro: " + fRegistro.toString() +
                "\n Propietario: " + propietario.toString() +
                "\n Vehículo: \n" + objV.toString();
    }
    public double valorPago(){...3 lines }
    public String getNroF() {...3 lines }
    public void setNroF(String nroF) {...3 lines }
    public Fecha getfRegistro() {...3 lines }
    public void setfRegistro(Fecha fRegistro) {...3 lines }
    public Persona getPropietario() {...3 lines }
    public void setPropietario(Persona propietario) {...3 lines }
    public Vehiculo getObjV() {...3 lines }
    public void setObjV(Vehiculo objV) {...3 lines }
}
```

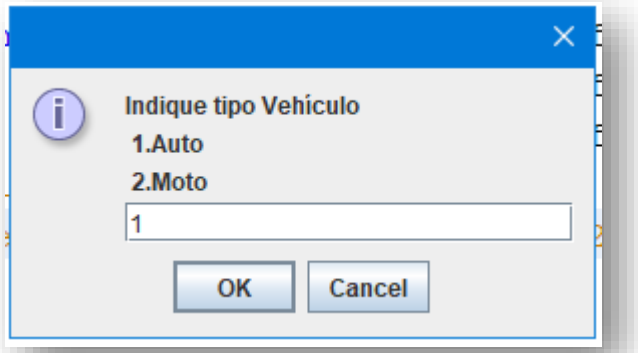
Clase Controlador (Control)

```
public class Controlador {
    Entrada objE;//objeto de la vista
    Formulario objF;//objeto del modelo
    public Controlador(Entrada objE, Formulario objF) {...4 lines }
     public Controlador() {
        this.objE = new Entrada();
        this.objF = new Formulario();
    }
     public void iniciar(){
     objE.mostrar("Ejercicio Impuesto Vehículo");
     objE.mostrar("Datos del propietario");
     objF.getPropietario().setId(objE.pedirTexto("Identificaciòn: "));
     objF.getPropietario().setNom(objE.pedirTexto("Nombre: "));
     objF.getPropietario().setTel(objE.pedirTexto("Telèfono: "));
     objF.getPropietario().setEmail(objE.pedirTexto("Email: "));
     objE.mostrar("Datos del Vehículo");
     int op=objE.pedirEntero("Indique tipo Vehículo\n 1.Auto\n 2.Moto\n");
     switch (op) {
         case 1:
             Auto objA= new Auto();
             objA.setPlaca(objE.pedirTexto("Placa: "));
             objA.setMarca(objE.pedirTexto("Marca: "));
             objA.setModelo(objE.pedirEntero("Modelo: "));
             objA.setValor(objE.pedirDecimal("Valor: "));
             objF.setObjV(objA);//Polimorfismo
             break;
         case 2:// implementar objeto moto
             break;
         default:
             objE.mostrar("Error: Tipo no existe...");
     }
     objE.mostrar("Datos registrados\n"+ objF.toString()+
                  "\n Valor Impuesto: "+objF.valorPago());
    }
}
```

Clase Principal (autónomo)

```
public class Principal {
    public static void main(String[] args) {
        Controlador objC= new Controlador();
        objC.iniciar();
    }

}
```



Bibliografía

- Documentación Java Oracle paquete Swing. Disponible en: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html.
- Deitel y Deitel. Programación Java. Editorial Mc Graw Hill.
- Pinzón, Sonia Alexandra. Material de Clase Moodle y Drive.
- Pinzón, Sonia Alexandra. Rodríguez Guerrero, Rocío. Vanegas, Carlos Alberto. Java y el patrón Modelo- Vista – Controlador (MVC).Editorial Universidad Distrital F.J.D.C. 2021