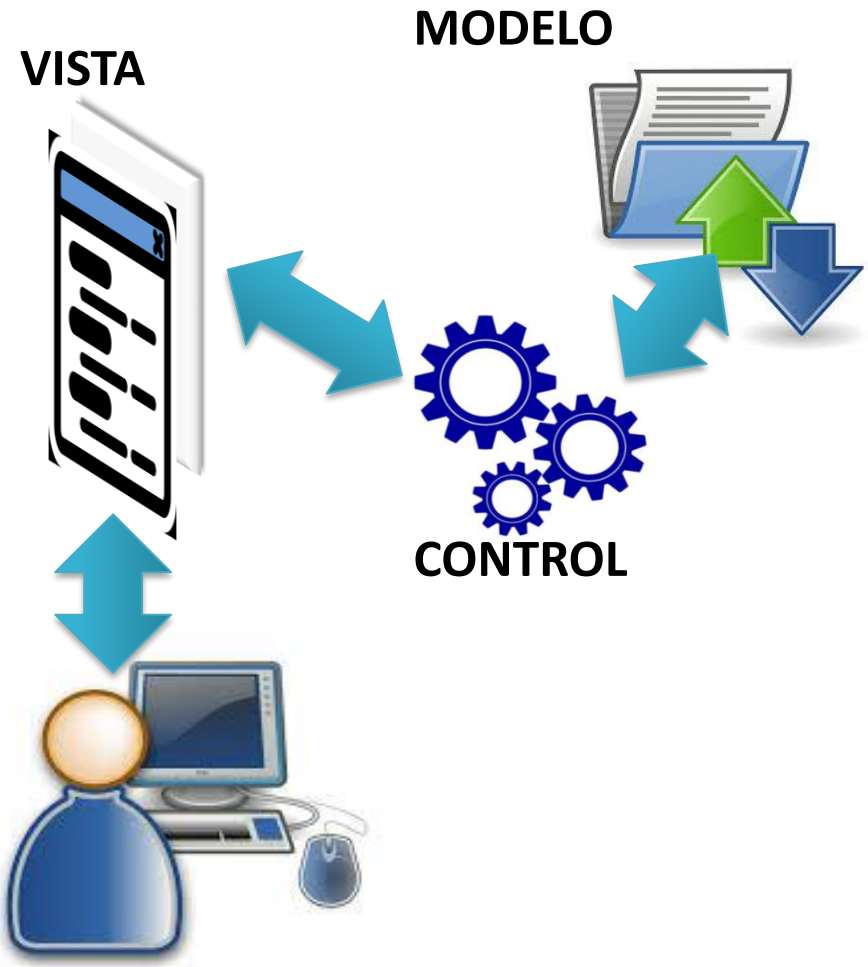
Universidad Distrital Francisco José de caldas
Tecnología en Sistematización de Datos



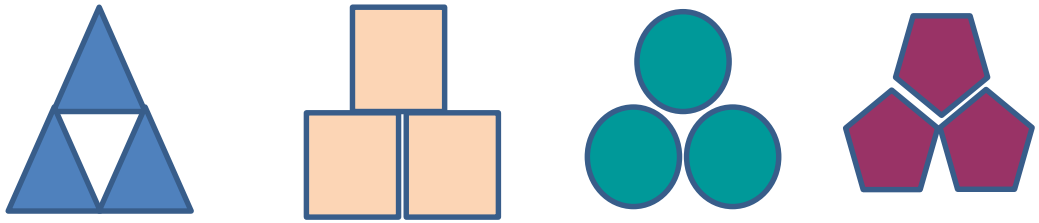
Modelo Vista Controlador

Programación Multinivel
Sonia Alexandra Pinzón Nuñez

Contenido

- Qué es un Patrón
- Categorías de Patrones
- Concepto MVC
- Metodología Desarrollo Ejercicios
- Ejemplo

Qué es un Patrón de Diseño ?



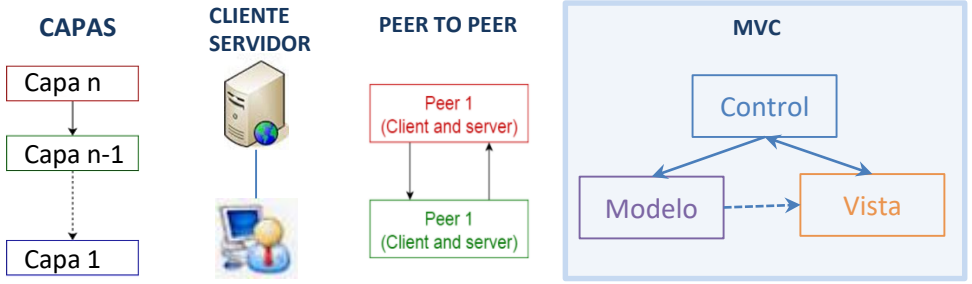
Son pautas o estrategias que se utilizan en el desarrollo de aplicaciones de software que permiten estandarizar el código y dan solución a problemas que suelen presentarse al codificar.

- Facilitan el desarrollo y el mantenimiento.
- Permiten reutilizar código con casos de éxito.
- Reducen tiempo en el desarrollo

Categorías de patrones

Patrones de Arquitectura

- Representan la estructura y organización de los sistemas de software



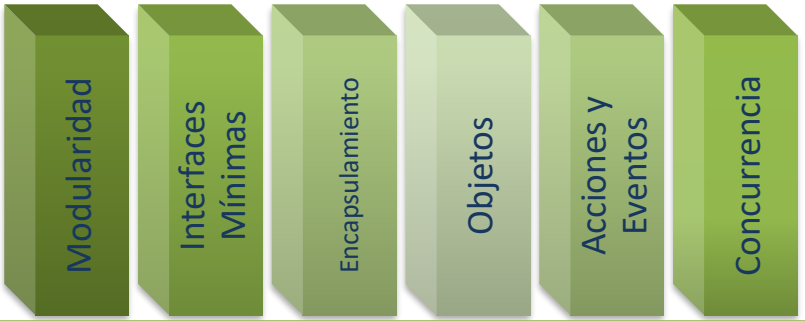
Patrones de Diseño

- Definen estructuras de diseño (o las relaciones) para la implementación de sistemas de software.

		Clase	Objeto
Creación	Instanciar o Crear Objetos	Factory Method	Abstract Factory Builder Prototype Singleton
Estructurales	Composición de Clases y Objetos	Adapter	Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy
Comportamiento	Comunicación y cooperación de responsabilidad	Interpreter Template Method	Chain of Responsability, Command, Iterator, Mediator, Memeto, **Observer**, State, Strategy, Visitor

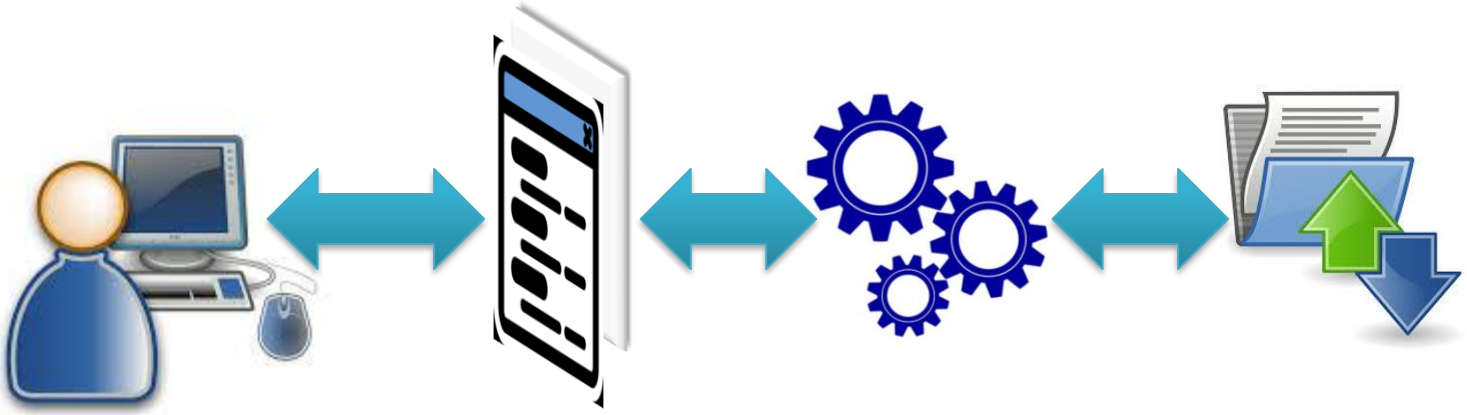
Patrones Elementales

- Son específicos de los lenguajes de programación o de los entornos de desarrollo

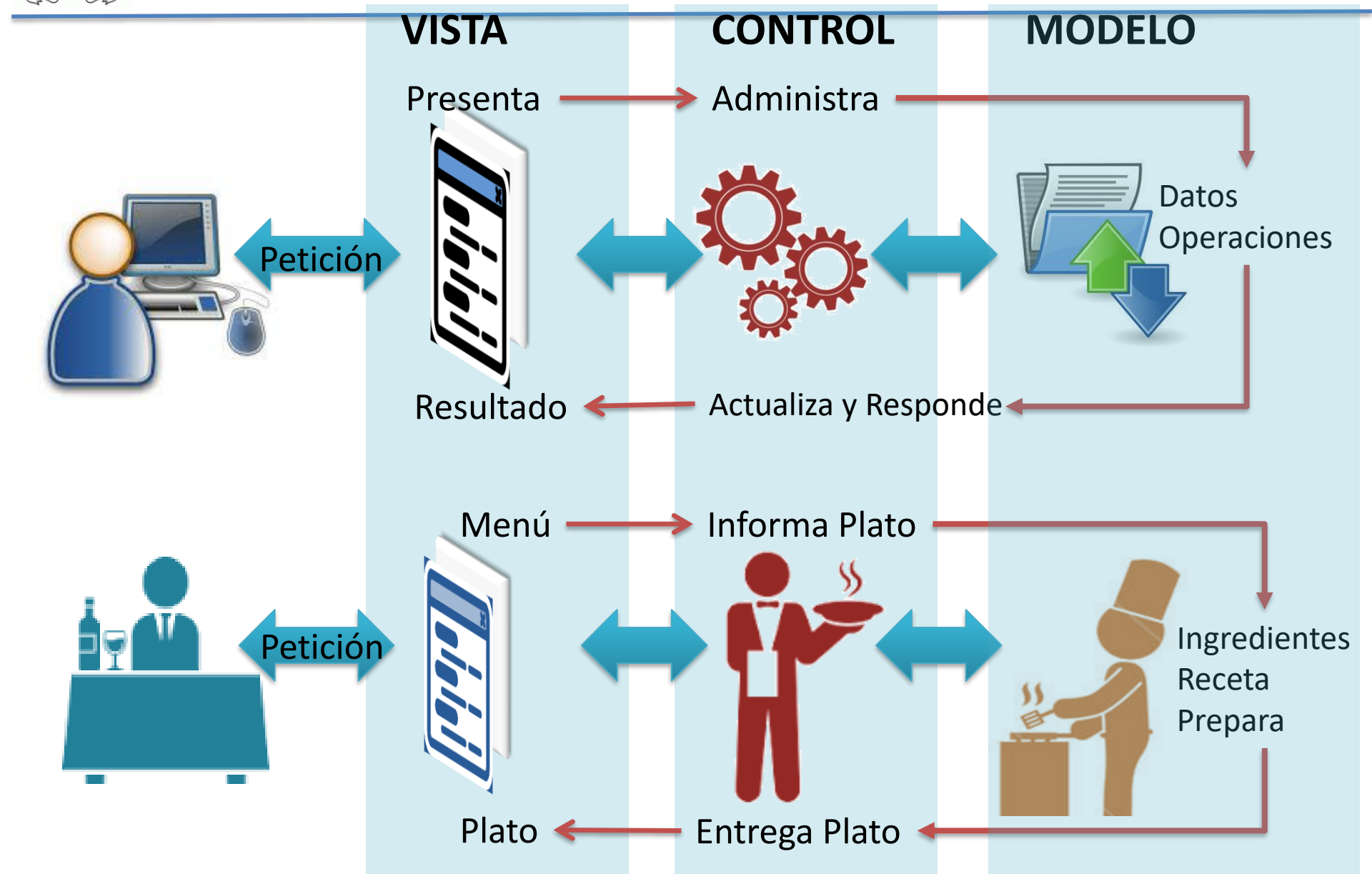


Antecedentes

- Desarrollado por el noruego Trygve Reenskaug que junto a un grupo de ingenieros de Smalltalk propuso un patrón de diseño de aplicaciones en los años 70.
- Es una forma de organizar (separar) el código de una aplicación según la función que este realiza.



CAPAS DEL MVC



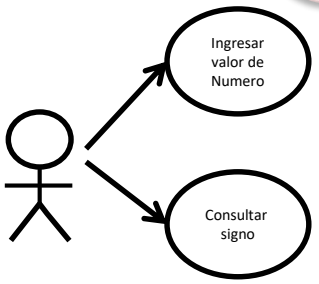
METODOLOGÍA

APLICACIÓN MVC

UNIFIED MODELING LANGUAGE™

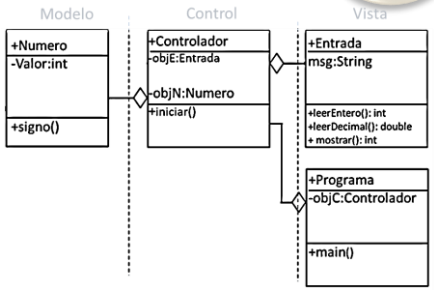
Análisis

- Diagrama Casos de uso



Diseño

- Diagrama de Clases
- Diseño de Interfaz



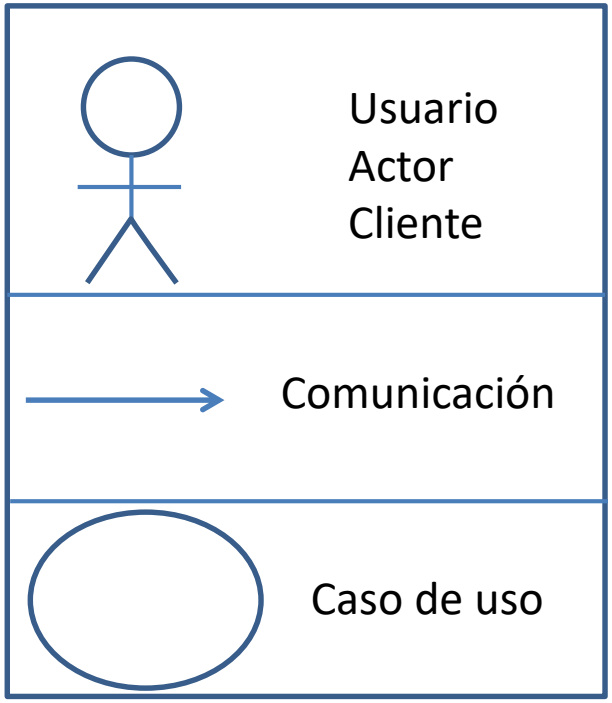
Implementación

- Codificar Clases del **Modelo**
- Codificar Clases de **Vista**
- Codificar Clases de **Control**

Diagrama Casos de Uso

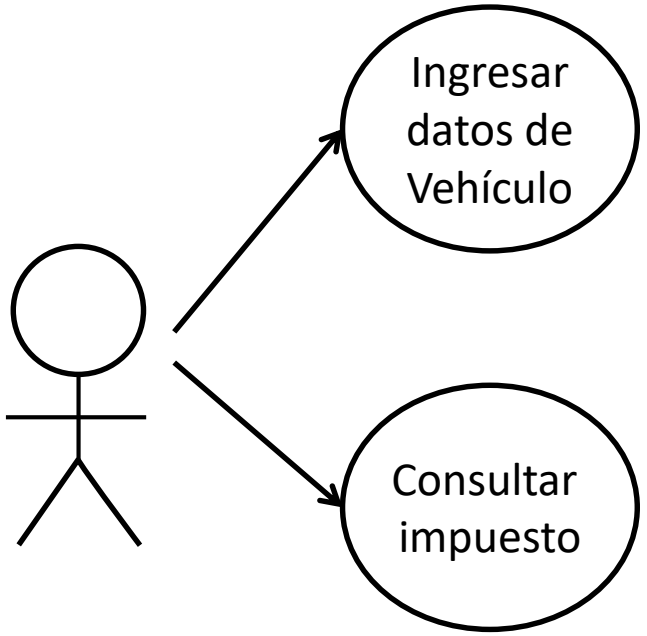
Representa las acciones o funciones que la aplicación debe facilitar al usuario.

Estereotipos



Ejemplo

Ejercicio persona

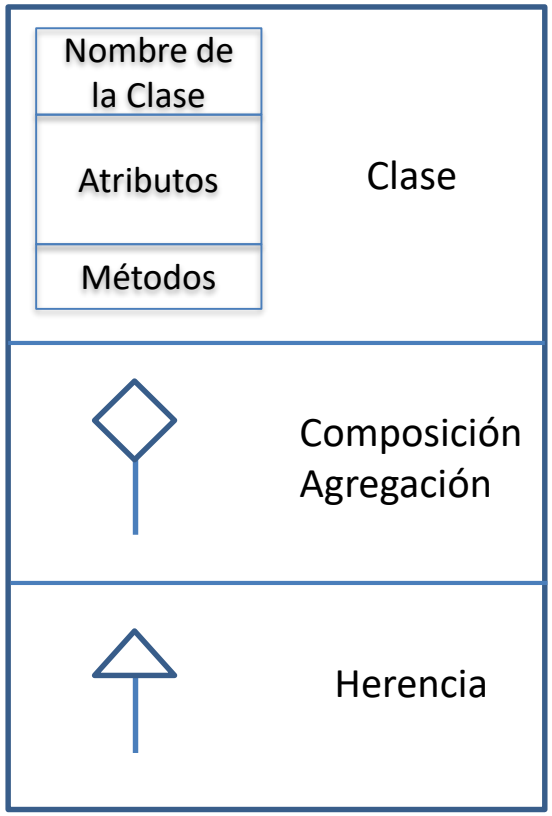


CLASE	ATRIBUTOS	COMPORTAMIENTO	OBJETO
Vehículo	placa, marca, modelo, valor	impuesto	objV("XYZ246","mazda",2001,20000000)

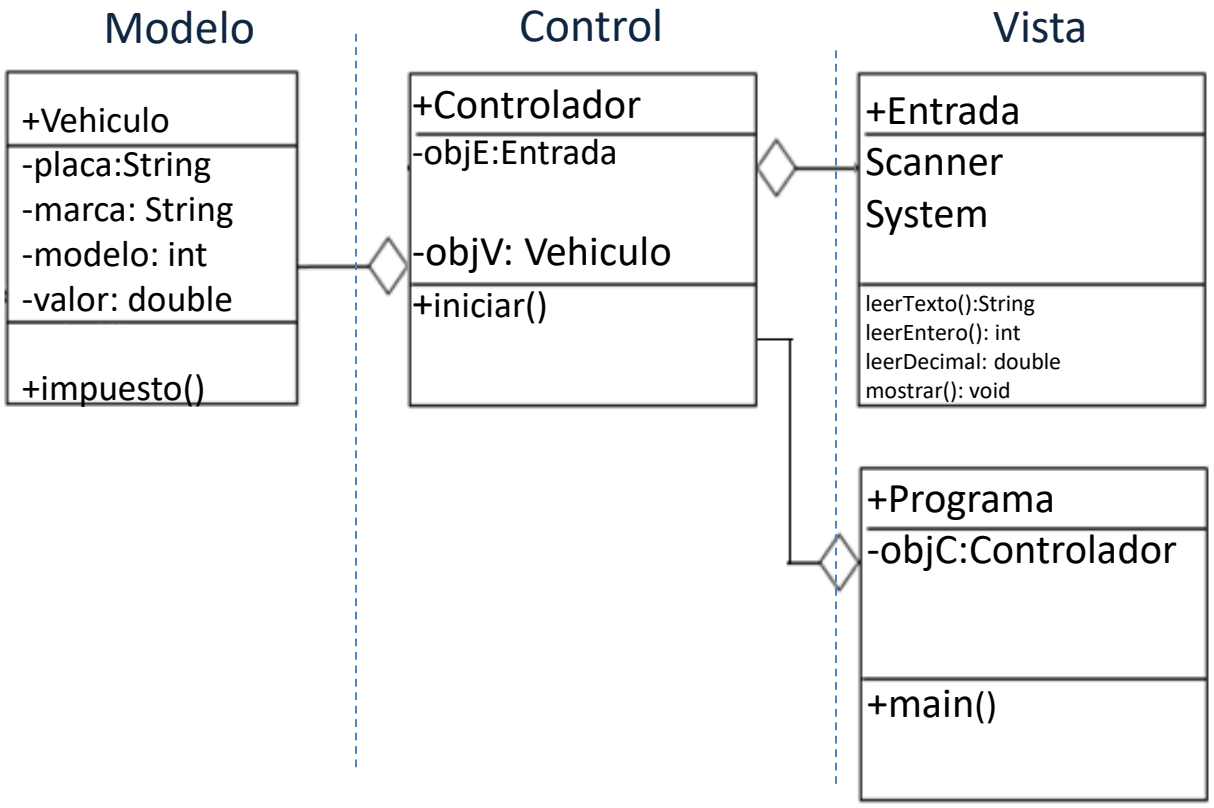
Diagrama de Clases

Representa las relaciones entre las clases definidas para el sistema (lógica del negocio)

Estereotipos



Ejemplo



Implementación:

Codificar las clases definidas en el diagrama de clases

Modelo Clase Vehiculo

```
public class Vehiculo {
    private String placa,marca;
    private int modelo;
    private double valor;
    //Constructor paramétrico
    public Vehiculo(String placa, String marca, int modelo, double valor) {
        this.placa = placa; this.marca = marca;
        this.modelo = modelo; this.valor = valor;
    }
    //Constructor básico
     public Vehiculo() {
        this.placa = ""; this.marca = "";
        this.modelo = 0; this.valor = 0;
    }
    public String getPlaca() {...3 lines }
    public void setPlaca(String placa) {...3 lines }
    public String getMarca() {...3 lines }
    public void setMarca(String marca) {...3 lines }
    public int getModelo() {...3 lines }
    public void setModelo(int modelo) {...3 lines }
    public double getValor() {...3 lines }
    public void setValor(double valor) {...3 lines }

    @Override
    public String toString() {
        return  "\n placa=" + placa + "\n marca=" + marca +
                "\n modelo=" + modelo + "\n valor=" + valor ;
    }
    public double impuesto() {...6 lines }
}
```

Implementación:

Vista Clase Entrada

Se encarga de realizar las operaciones que permiten la interacción con el usuario, procesos de entrada y salida

```
package vista;
import java.util.Scanner;
public class Entrada {
     private String titulo;
     private Scanner entrada;
    public Entrada(String titulo) {
        entrada= new Scanner(System.in);
        this.titulo = titulo;
    }
    public Entrada() {
        entrada= new Scanner(System.in);
        this.titulo = "";
    }
    public int leerEntero(String msj){
        System.out.println(msj);
        return entrada.nextInt();
    }
    public double leerDecimal(String msj){
        System.out.println(msj);
        return entrada.nextDouble();
    }
    public String leerTexto(String msj){
        System.out.println(msj);
        return entrada.next();
    }
    public void mostrar(String msj){
        System.out.println(titulo+"\n"+msj);
    }
    @Override
    public String toString() {
        return "titulo=" + titulo ;
    }
    public String getTitulo() {...3 lines }
    public void setTitulo(String titulo) {...3 lines }
}
```

Implementación:

Control

```
package control;
import modelo.Vehiculo;
import vista.Entrada;
/**
 * @author Sonia Pinzón Nuñez
 */
public class Controlador {
    Entrada objE;//objeto de la vista
    Vehiculo objV;//objeto del modelo

    public Controlador(Entrada objE, Vehiculo objV) {
        this.objE = objE;
        this.objV = objV;
    }
    public Controlador() {
        this.objE = new Entrada();
        this.objV = new Vehiculo();
    }
    public void iniciar(){
     //instrucción manejar la vista
     objE.mostrar("Ejercicio Vehículo");
     //Pasar de la vista al modelo
     objV.setPlaca(objE.leerTexto("Placa: "));
     objV.setMarca(objE.leerTexto("Marca: "));
     objV.setModelo(objE.leerEntero("Modelo: "));
     objV.setValor(objE.leerDecimal("Valor: "));
     //pasar del modelo a la vista
     objE.mostrar("Datos registrados\n"+ objV.toString()+
                  "\n Valor Impuesto: "+objV.impuesto());
    }
}
```

Método que desarrolla o inicia la acción

Implementación:

Vista Programa Autónomo

```
package proyvehiculos;
import control.Controlador;
/**
 *
 * @author Sonia Pinzón Nuñez
 */
public class ProyVehiculos {
    public static void main(String[] args) {
        //Crea un objeto de tipo controlador
        control.Controlador objC= new Controlador();
        objC.iniciar();
    }
}
```

El programa hace el llamado al controlador e inicia el proceso

Bibliografía

- 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
- LADRÓN, Jorge Martínez. Fundamentos de programación en Java - 4ed. Ed. EME.Universidad Complutense de Madrid. Madrid(España), formato Digital
- Deitel y Deitel. Programación Java. Editorial Mc Graw Hill.