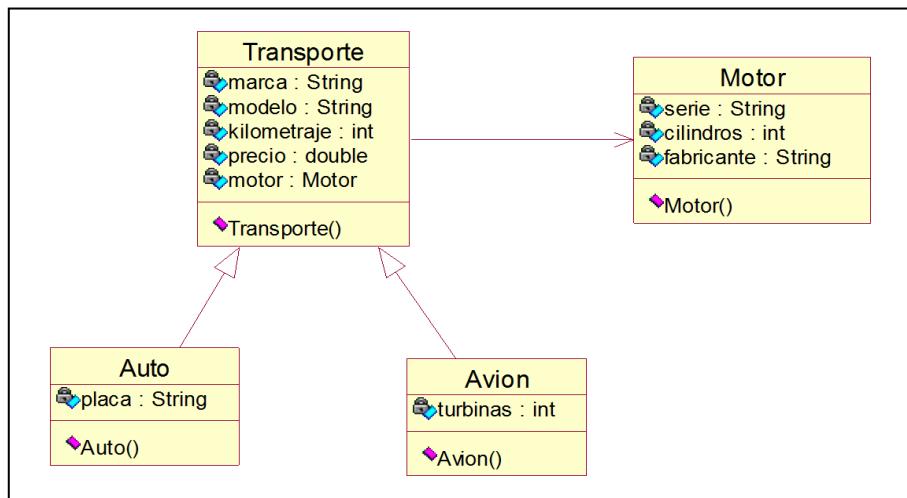


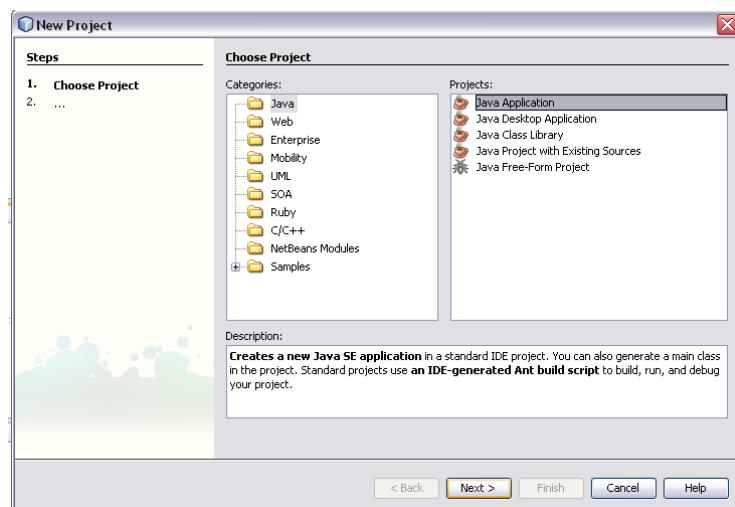
GUÍA DE LABORATORIO 04

...analicemos este caso:



Ahora lo implementamos en NetBeans:

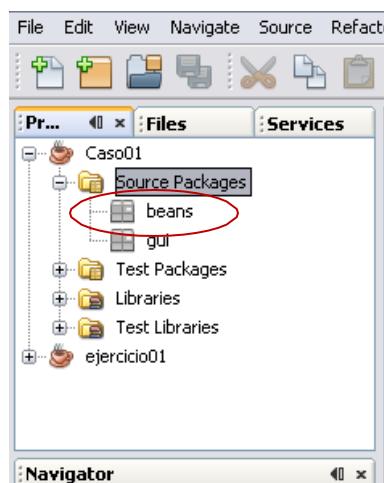
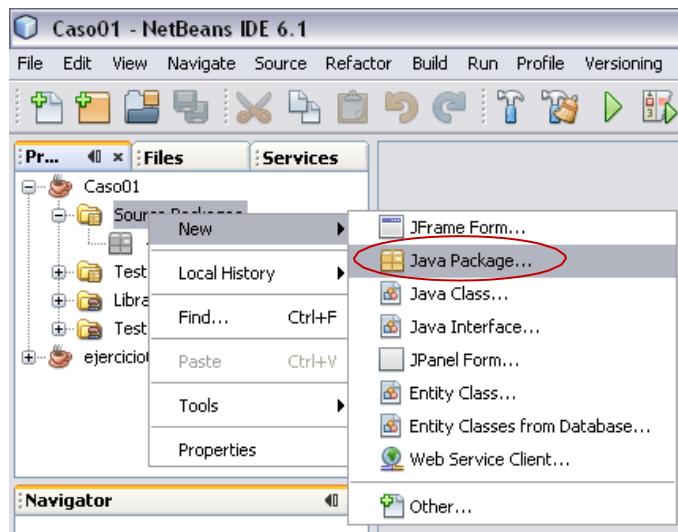
Creamos una aplicación Java con el nombre Caso01.



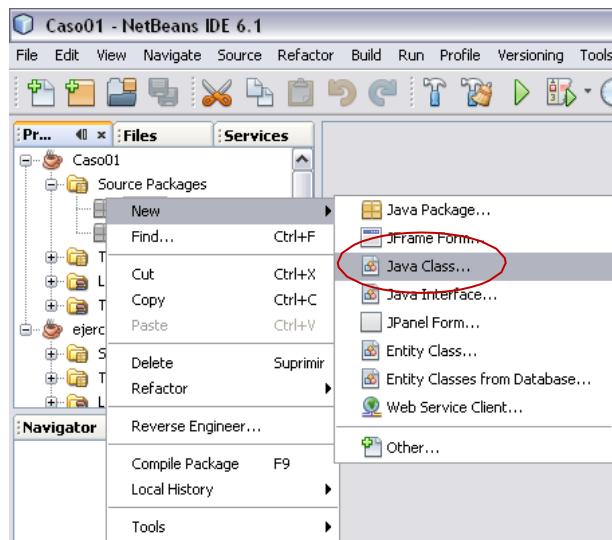
Ahora creamos nuestros paquetes donde se alojara la lógica del negocio, y le colocamos beans porque debe cumplir con lo siguiente:

- Implementación en serie.
- Tener todos sus atributos privados (private).
- Tener métodos set () y get () públicos de los atributos privados que nos interese.
- Tener un constructor público por defecto.

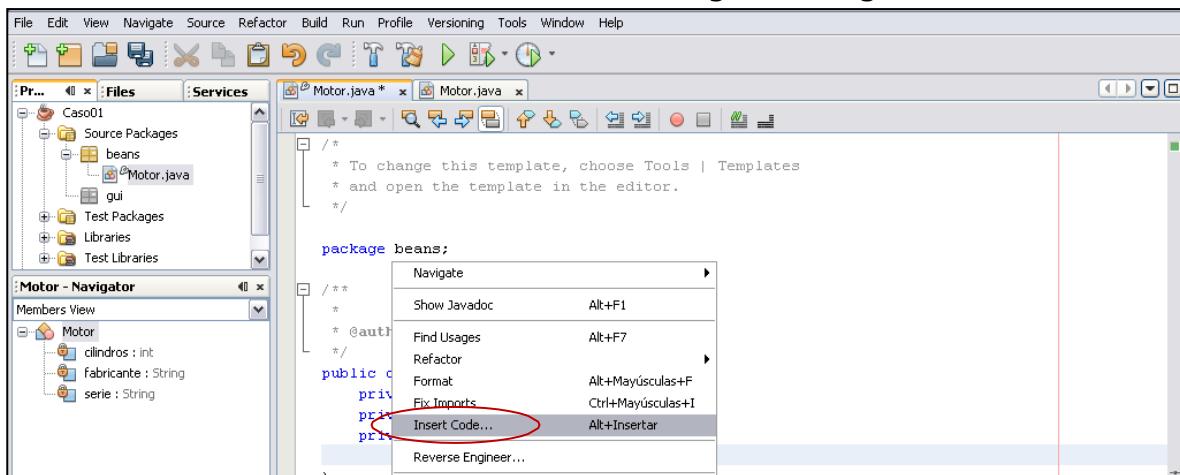
Además un paquete que contendrá la Interfaz Grafica de Usuario (GUI).



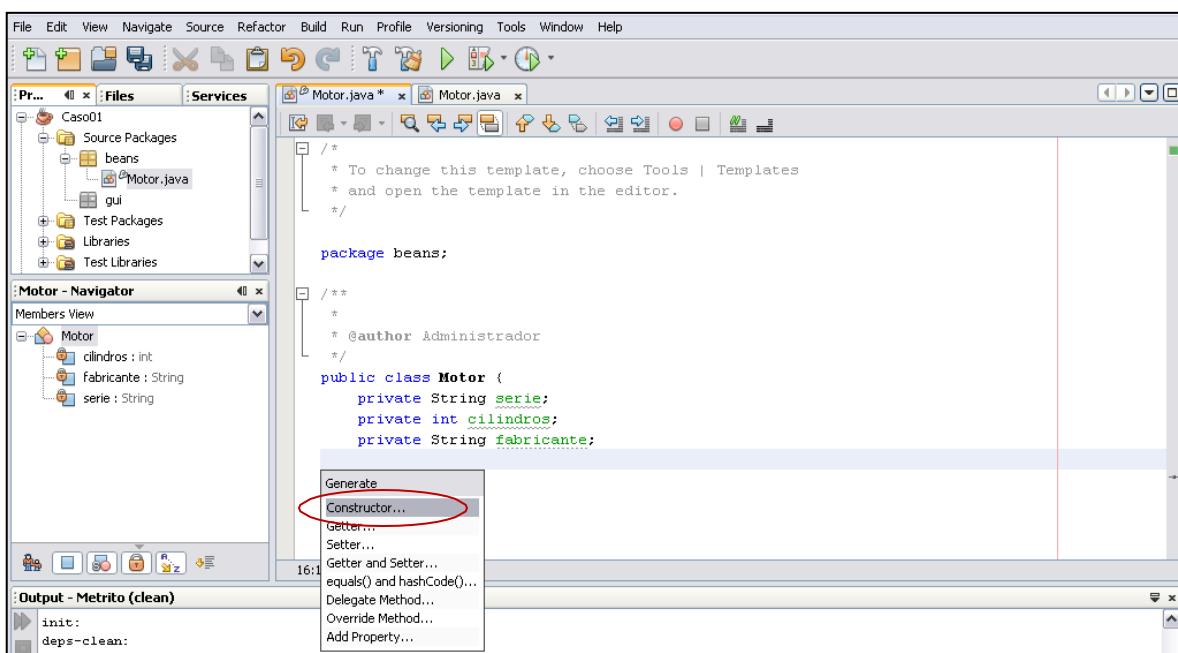
Según en concepto anterior empezamos a crear nuestras clases *Transporte, Motor, Auto y Avión* en el paquete **bean**.



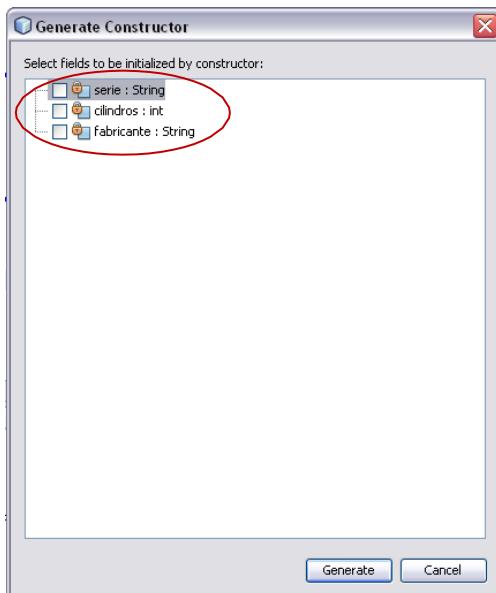
Java nos facilita el trabajo a los desarrolladores cuando se trata de crear los constructores...como se muestra en la siguiente figura:

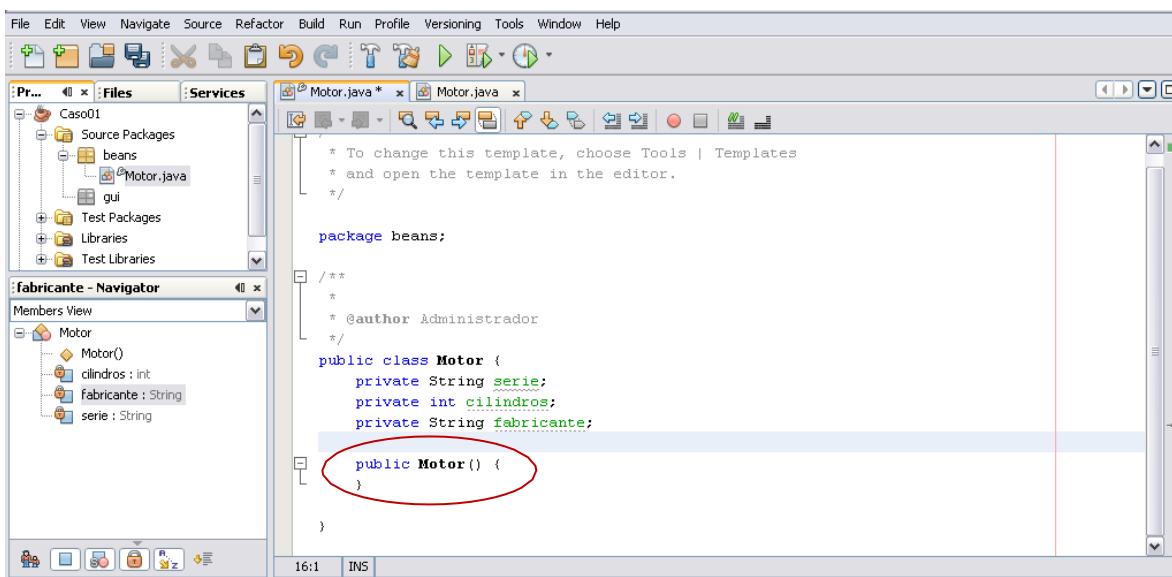


Ahora insertaremos dos constructores a nuestra clase.



Cuando queramos crear un constructor sin parámetros no le debemos dar clic en ninguno de los campos que se nos aparece, *además debemos tener en cuenta que de no tener un constructor sin parámetros java le implementa uno por defecto.*



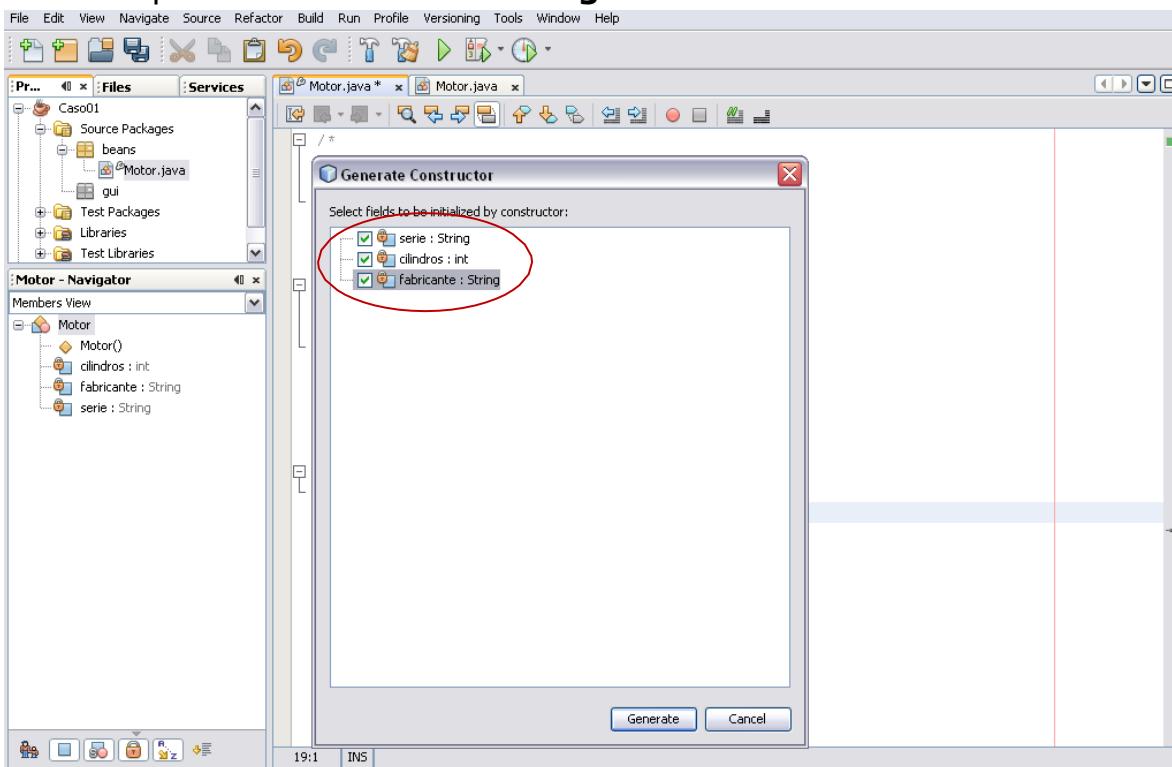


```

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Pr... Files Services
Pr... Caso01
Source Packages beans
  Motor.java
gui
Test Packages
Libraries
Test Libraries
Motor - Navigator Members View
Motor
  Motor()
  cilindros : int
  Fabricante : String
  serie : String
Motor.java * Motor.java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package beans;

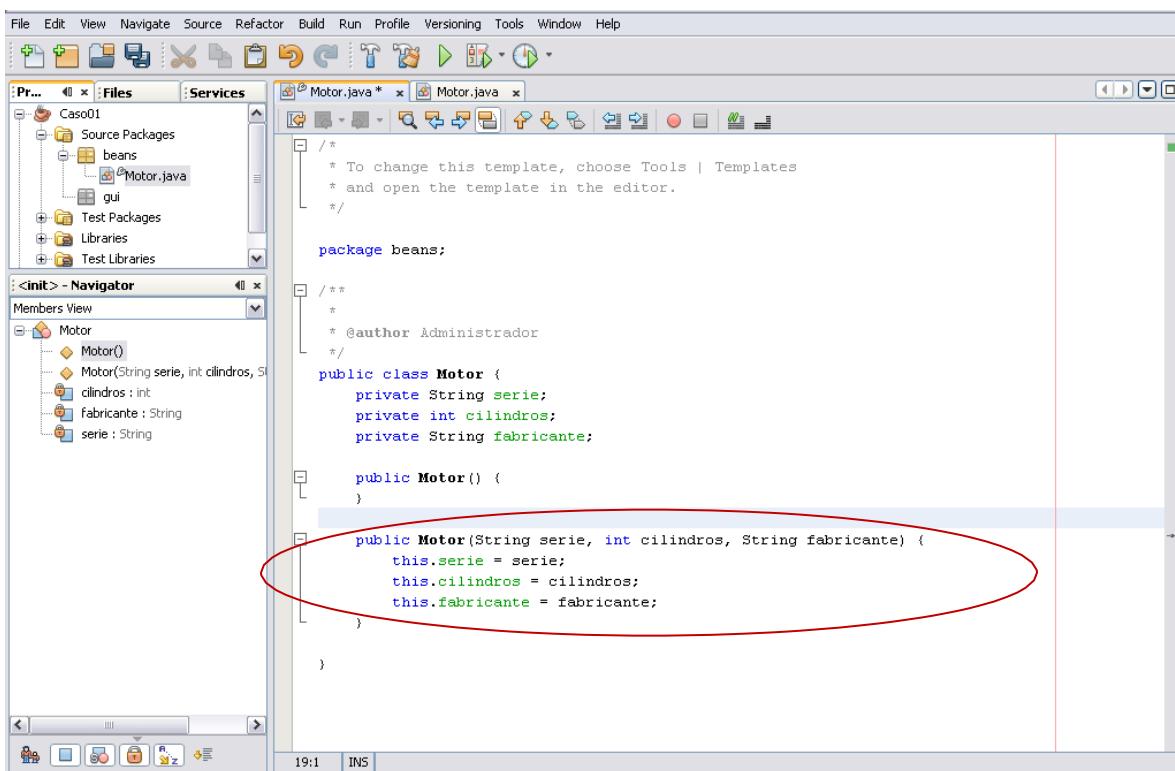
/**
 *
 */
public class Motor {
    private String serie;
    private int cilindros;
    private String fabricante;
    public Motor() {
    }
}
16:1 INS
    
```

Ahora creamos un constructor con parámetros, recordemos que la idea de los constructores es establecer valores iniciales a los campos. Nótese algo interesante cuando implementemos este constructor, en realidad estamos provocando una **sobrecarga** de este método.



```

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Pr... Files Services
Pr... Caso01
Source Packages beans
  Motor.java
gui
Test Packages
Libraries
Test Libraries
Motor - Navigator Members View
Motor
  Motor()
  cilindros : int
  Fabricante : String
  serie : String
Motor.java * Motor.java
/*
 */
Generate Constructor
Select fields to be initialized by constructor:
   serie : String
   cilindros : int
   fabricante : String
Generate Cancel
19:1 INS
    
```



```

    /*
     * To change this template, choose Tools | Templates
     * and open the template in the editor.
     */

    package beans;

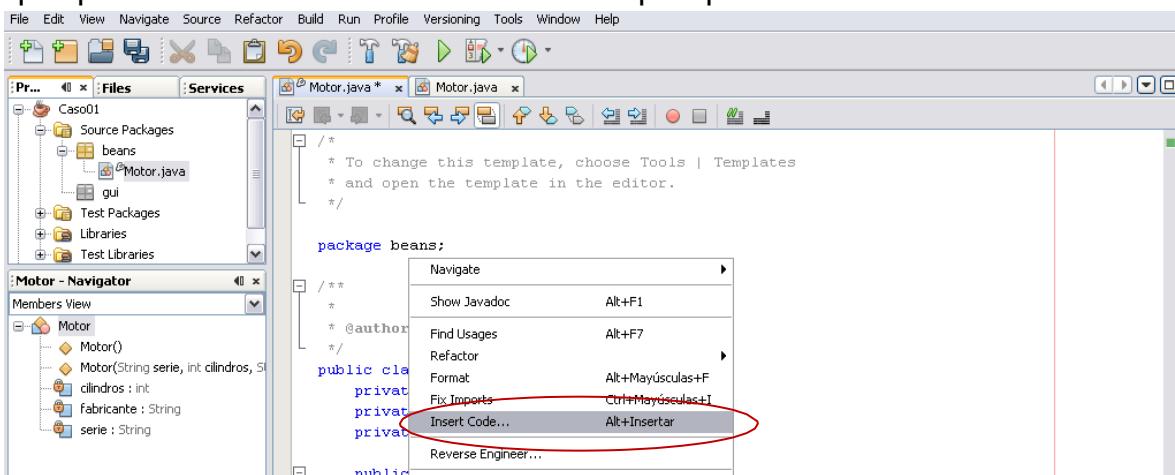
    /**
     *
     * @author Administrador
     */
    public class Motor {
        private String serie;
        private int cilindros;
        private String fabricante;

        public Motor() {
        }

        public Motor(String serie, int cilindros, String fabricante) {
            this.serie = serie;
            this.cilindros = cilindros;
            this.fabricante = fabricante;
        }
    }

```

Volvemos a insertar código, pero esta será para los métodos y funciones que permitirán acceder a nuestros campos privados



File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

Pr... Files Services

Caso01

Source Packages beans Motor.java

Test Packages

Libraries

Test Libraries

Motor - Navigator

Members View

Motor

- Motor()
- Motor(String serie, int cilindros, String fabricante)
- cilindros : int
- Fabricante : String
- serie : String

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

Pr... Files Services

Caso01

Source Packages beans Motor.java

Test Packages

Libraries

Test Libraries

Motor - Navigator

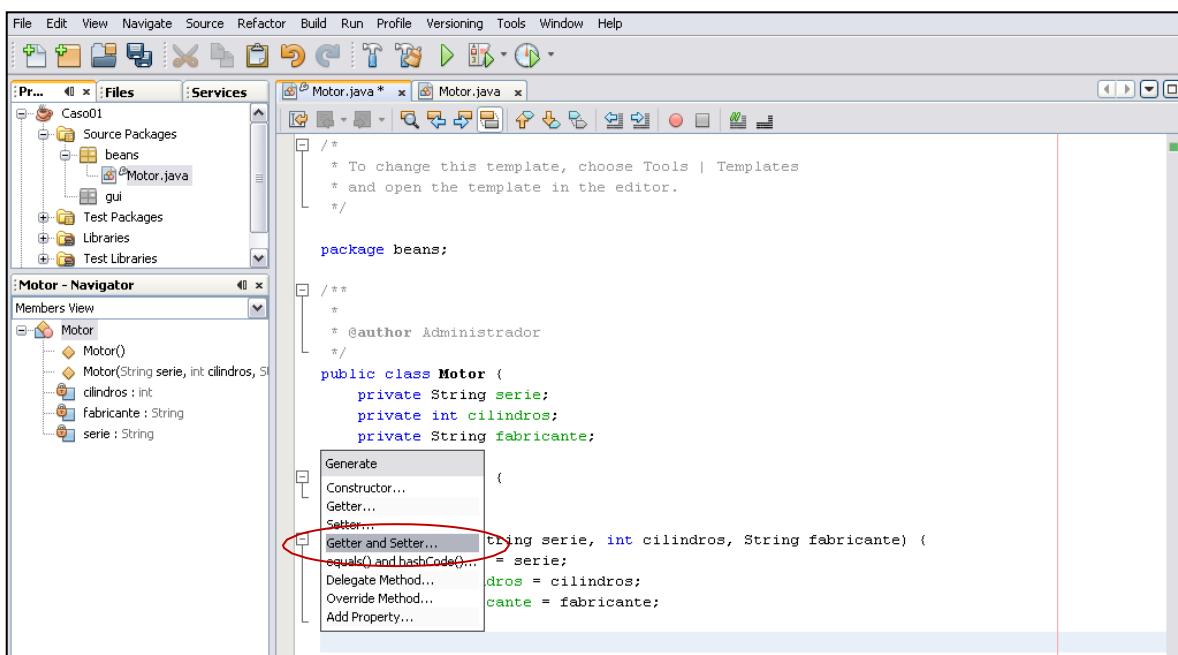
Members View

Motor

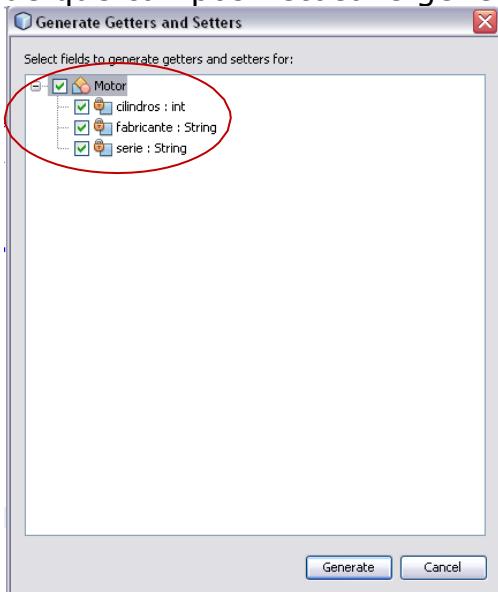
- Motor()
- Motor(String serie, int cilindros, String fabricante)
- cilindros : int
- Fabricante : String
- serie : String

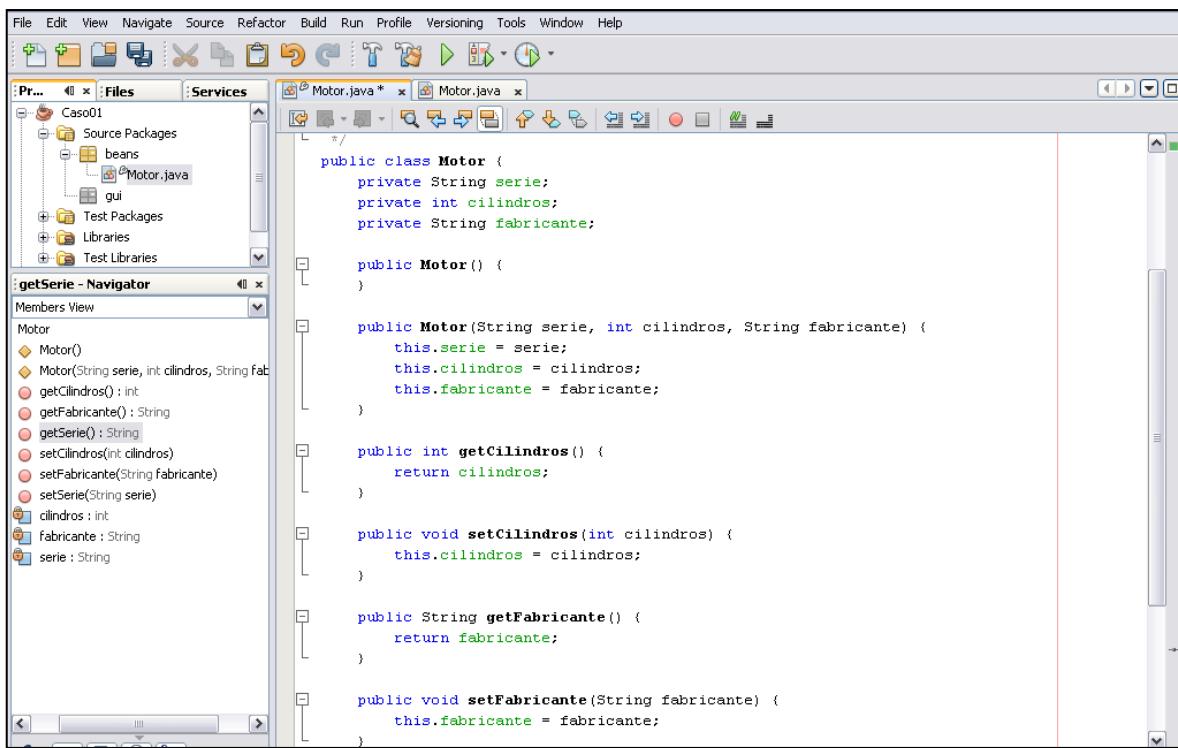
Navigation

- Show Javadoc Alt+F1
- Find Usages Alt+F7
- Refactor
- Format Alt+Mayúsculas+F
- Fix Imports Ctrl+Mayúsculas+I
- Insert Code... Alt+Insertar**
- Reverse Engineer...



Debemos especificar de qué campos netbeans generara los get y set.





```

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Pr... Files Services
Casos01
Source Packages beans
gui
Test Packages
Libraries
Test Libraries
getSerie - Navigator
Members View
Motor
Motor()
Motor(String serie, int cilindros, String fabricante)
getCilindros() : int
getFabricante() : String
getSerie() : String
setCilindros(int cilindros)
setFabricante(String fabricante)
setSerie(String serie)
cilindros : int
Fabricante : String
serie : String
    */
public class Motor {
    private String serie;
    private int cilindros;
    private String fabricante;

    public Motor() {
    }

    public Motor(String serie, int cilindros, String fabricante) {
        this.serie = serie;
        this.cilindros = cilindros;
        this.fabricante = fabricante;
    }

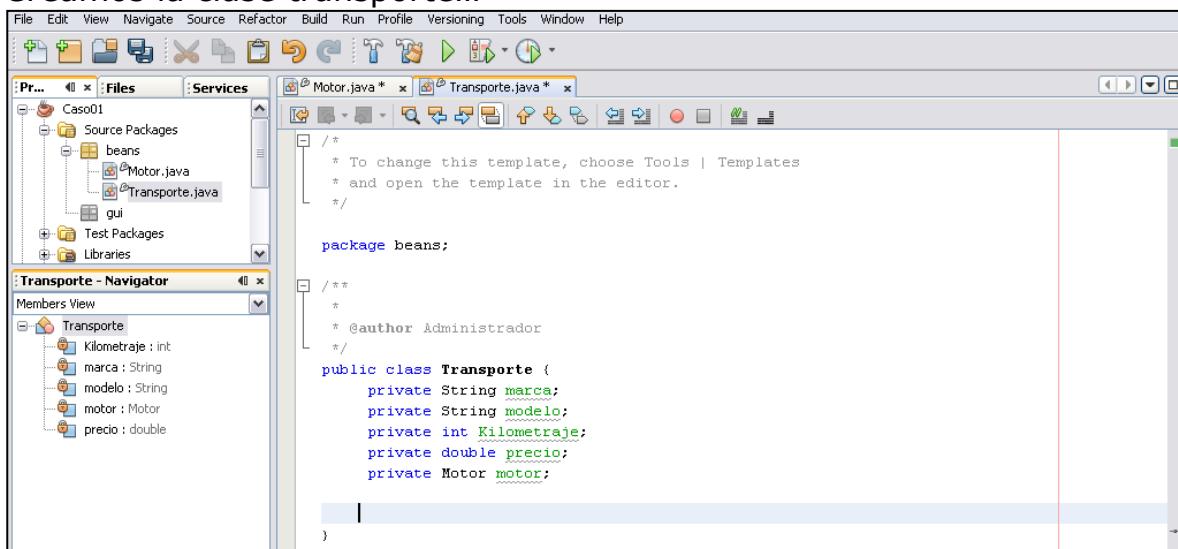
    public int getCilindros() {
        return cilindros;
    }

    public void setCilindros(int cilindros) {
        this.cilindros = cilindros;
    }

    public String getFabricante() {
        return fabricante;
    }

    public void setFabricante(String fabricante) {
        this.fabricante = fabricante;
    }
}
    
```

Creamos la clase transporte...



```

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Pr... Files Services
Casos01
Source Packages beans
gui
Test Packages
Libraries
Transporte - Navigator
Members View
Transporte
Kilometraje : int
marca : String
modelo : String
motor : Motor
precio : double
    /*
     * To change this template, choose Tools | Templates
     * and open the template in the editor.
     */

    package beans;

    /**
     * @author Administrador
     */
    public class Transporte {
        private String marca;
        private String modelo;
        private int Kilometraje;
        private double precio;
        private Motor motor;
    }
}
    
```

Y de la misma manera que en la clase Motor, generamos una sobrecarga de los constructores y los get y set de los campos.

Screenshot of an IDE showing the code editor and Navigator panel for the `Transporte` class.

```

public class Transporte {
    private String marca;
    private String modelo;
    private int Kilometraje;
    private double precio;
    private Motor motor;

    public Transporte() {
    }

    public Transporte(String marca, String modelo, int Kilometraje, double precio, Motor motor) {
        this.marca = marca;
        this.modelo = modelo;
        this.Kilometraje = Kilometraje;
        this.precio = precio;
        this.motor = motor;
    }

    public int getKilometraje() {
        return Kilometraje;
    }

    public void setKilometraje(int Kilometraje) {
        this.Kilometraje = Kilometraje;
    }

    public String getMarca() {
        return marca;
    }
}

```

The Navigator panel shows the members of the `Transporte` class, including its constructor, getters, setters, and fields.

Creamos la clase Auto...

Screenshot of an IDE showing the code editor and Navigator panel for the `Auto` class.

```

package beans;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

public class Auto {
    private String placa;

    public Auto() {
    }

    public Auto(String placa) {
        this.placa = placa;
    }

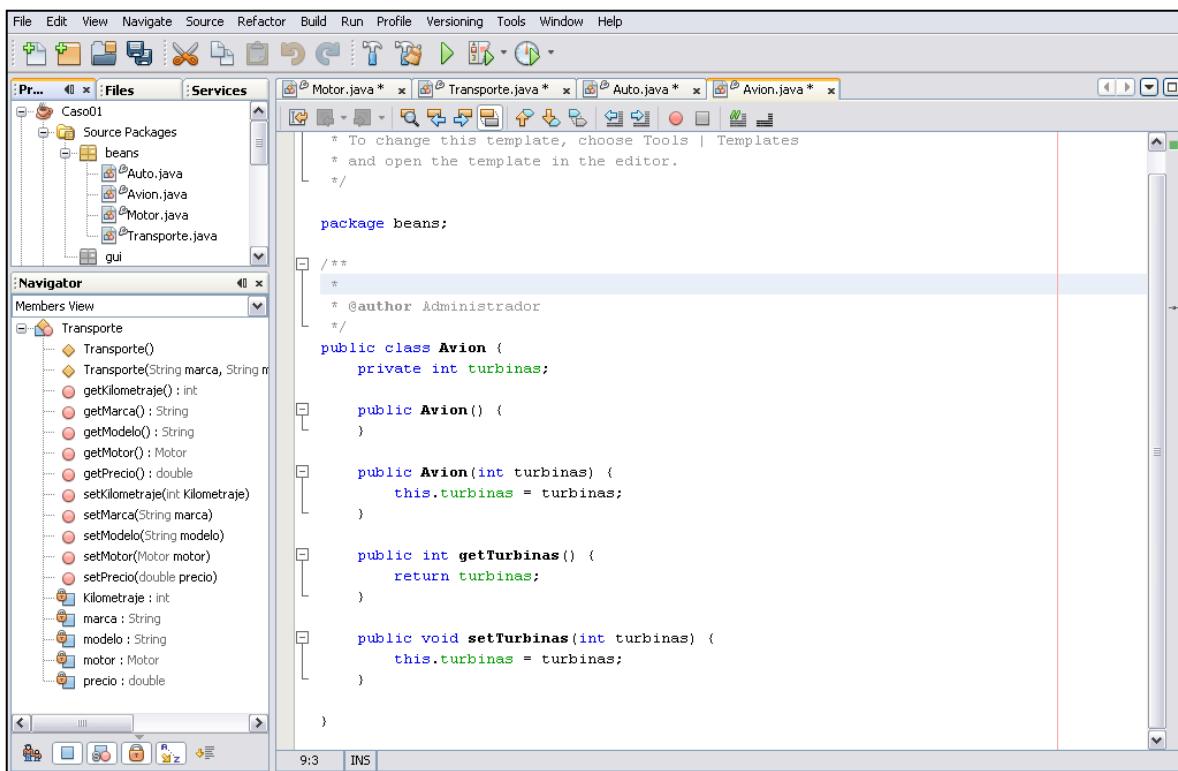
    public String getPlaca() {
        return placa;
    }

    public void setPlaca(String placa) {
        this.placa = placa;
    }
}

```

The Navigator panel shows the members of the `Auto` class, including its constructor, getters, setters, and field.

Creamos la clase Avión...



```

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Pr... Files Services
Caso01
Source Packages beans Auto.java Avion.java Motor.java Transporte.java
gui
Navigator Members View
Transporte
Transporte()
Transporte(String marca, String modelo)
getKilometraje(): int
getMarca(): String
getModelo(): String
getMotor(): Motor
getPrecio(): double
setKilometraje(int Kilometraje)
setMarca(String marca)
setModelo(String modelo)
setMotor(Motor motor)
setPrecio(double precio)
Kilometraje : int
marca : String
modelo : String
motor : Motor
precio : double
9:3 INS
package beans;

/**
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

public class Avion {
    private int turbinas;

    public Avion() {
    }

    public Avion(int turbinas) {
        this.turbinas = turbinas;
    }

    public int getTurbinas() {
        return turbinas;
    }

    public void setTurbinas(int turbinas) {
        this.turbinas = turbinas;
    }
}
    
```

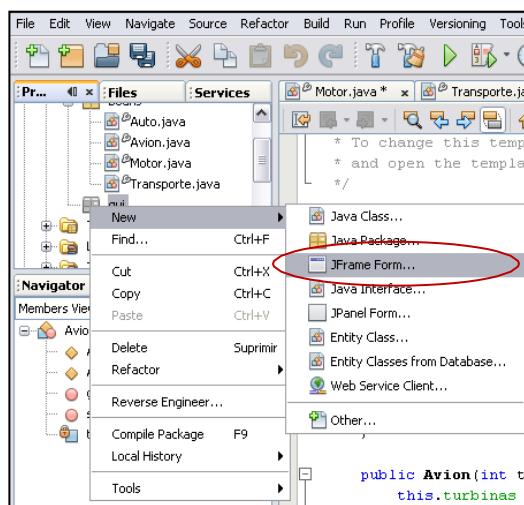
IMPORTANTE:

Recuerde que debemos realizar la herencia de las clases "Avion" y "Auto" con la clase "Transporte":

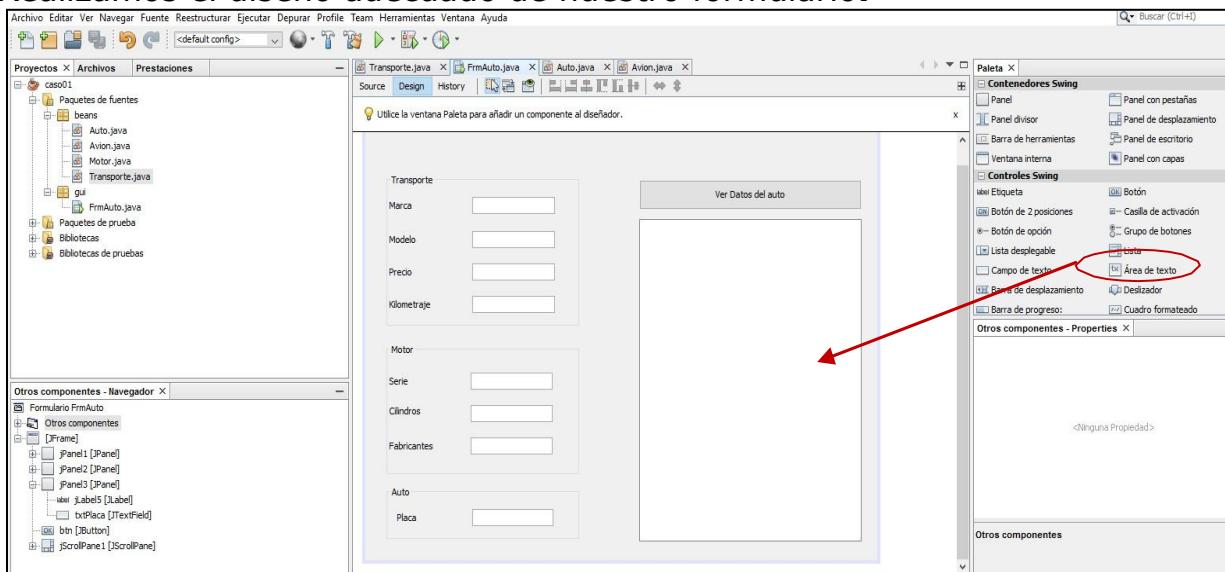
```

public class Avion extends Transporte{
public class Auto extends Transporte{
    
```

Ahora implementamos nuestro paquete **GUI**, donde alojaremos nuestro formulario que servirá para administrar los datos a nuestras clases.



Realizamos el diseño adecuado de nuestro formulario.



Ahora lo que debemos hacer es importar las clases necesarias con las que trabajaremos, nótese que no se importa la clase Transporte es por una sencilla razón; que esos datos son heredados por Avión y Auto, por tanto, ellos son los que manejarán los datos de Transporte.

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

Pr... Files Services

beans
Auto.java
Avion.java
Motor.java
Transporte.java
gui
FrmAuto.java
Test Packages

FrmAuto - Navigator Members View

```

    package gui;

    import beans.Auto;
    import beans.Avion;
    import beans.Motor;

    /**
     * Created on 14 de mayo de 2011, 12:39 AM
     */

    public class FrmAuto extends javax.swing.JFrame {

        /**
         * Creates new form FrmAuto
         */
        public FrmAuto() {
            initComponents();
        }

        /**
         * This method is called from within the constructor to
         * initialize the form.
         * WARNING: Do NOT modify this code. The content of this method is
         * always regenerated by the Form Editor.
         */
        @SuppressWarnings("unchecked")
        // Generated Code
    }

```

Archivo Editar Ver Navegar Fuente Reestructurar Ejecutar Depurar Profile Team Herramientas Ventana Ayuda

Proyectos Archivos Prestaciones

caso01
Paquetes de fuentes
beans
Auto.java
Avion.java
Motor.java
Transporte.java
gui
FrmAuto.java
Paquetes de prueba
Bibliotecas
Bibliotecas de pruebas

Navegador Vista de miembros

```

    private void btnActionPerformed(java.awt.event.ActionEvent evt) {
        Auto auto= new Auto();
        auto.setMarca(txtMarca.getText());
        auto.setModelo(txtModelo.getText());
        auto.setKilometraje(Integer.parseInt(txtKilometraje.getText()));
        auto.setPrecio(Double.parseDouble(txtPrecio.getText()));
        auto.setPlaca(txtPlaca.getText());

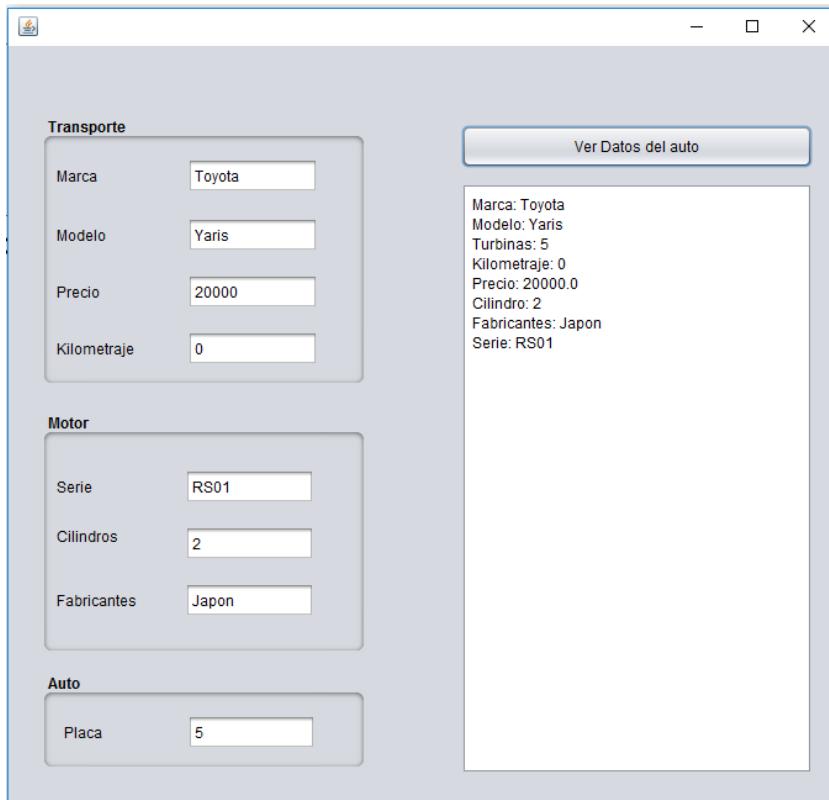
        Motor motor=new Motor();
        motor.setSerie(txtSerie.getText());
        motor.setCilindros(Integer.parseInt(txtCilindro.getText()));
        motor.setFabricante(txtFabricantes.getText());

        auto.setMotor(motor);

        txtMostrar.setText("");
        txtMostrar.append("Marca: "+ auto.getMarca()+"\n");
        txtMostrar.append("Modelo: "+ auto.getModelo()+"\n");
        txtMostrar.append("Turbinas: "+ auto.getPlaca()+"\n");
        txtMostrar.append("Kilometraje: "+ auto.getKilometraje()+"\n");
        txtMostrar.append("Precio: "+ auto.getPrecio()+"\n");
        txtMostrar.append("Cilindro: "+ auto.getMotor().getCilindros()+"\n");
        txtMostrar.append("Fabricantes: "+ auto.getMotor().getFabricante()+"\n");
        txtMostrar.append("Serie: "+ auto.getMotor().getSerie()+"\n");
    }

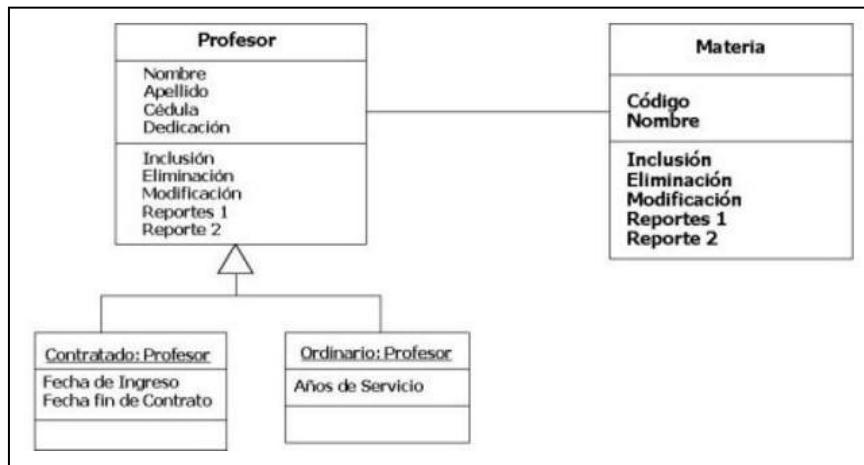
```

Compilamos...



PRACTICA

Del siguiente diagrama...



...realice lo siguiente:

1. Realice las clases en capas.
2. Cree el código necesario para representar las relaciones (asociación y herencia)
3. Realice la interfaz gráfica e impleméntelo para acceder a mediante el profesor ordinario y el profesor contratado.