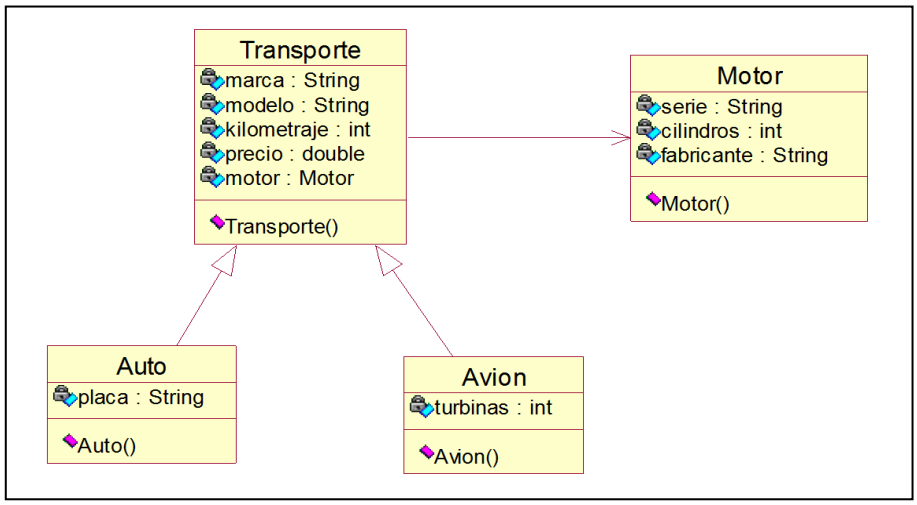
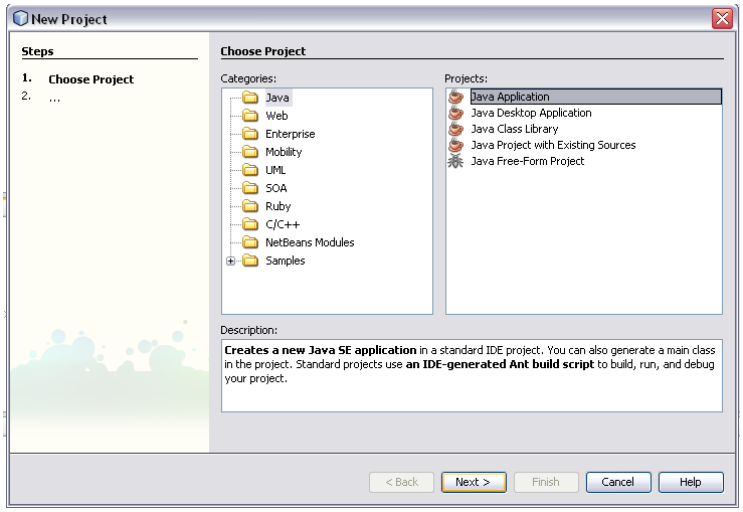
GUÍA DE LABORATORIO 04

…analicemos este caso:



Ahora lo implementamos en NetBeans:

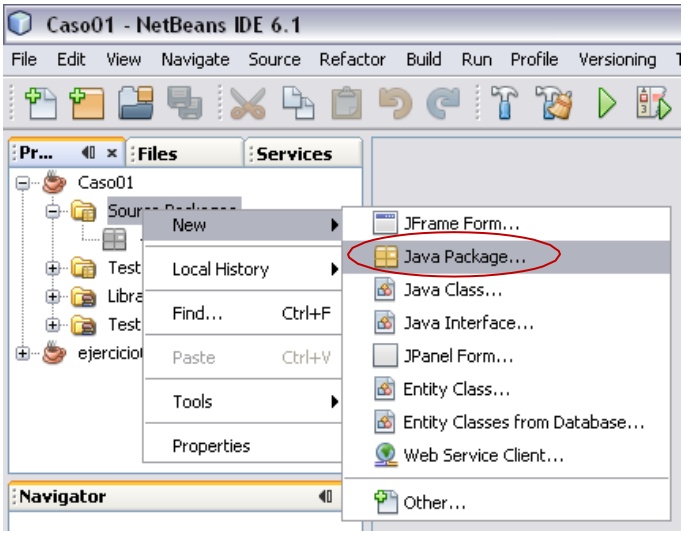
Creamos una aplicación Java con el nombre Caso01.

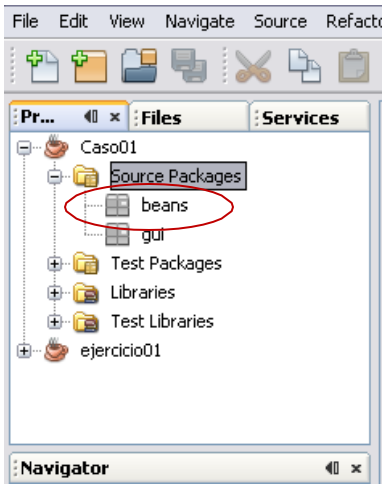


Ahora creamos nuestros paquetes donde se alojara la lógica del negocio, y le colocamos beans porque debe cumplir con lo siguiente:

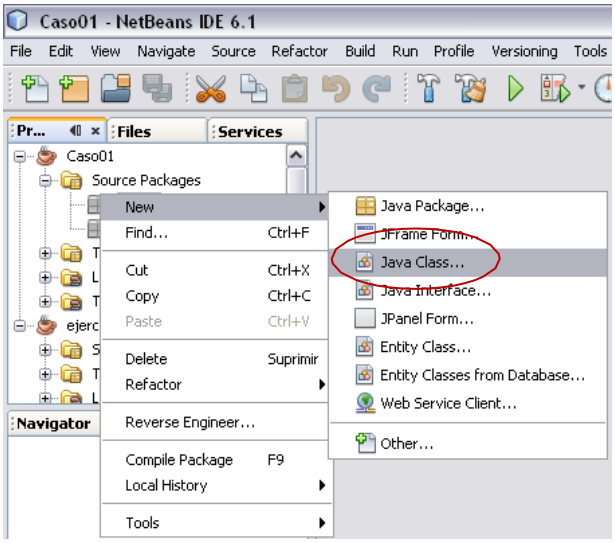
- Implementación en serie.
- Tener todos sus atributos privados (private).
- Tener métodos set () y get () públicos de los atributos privados que nos interese.
- Tener un constructor público por defecto.

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

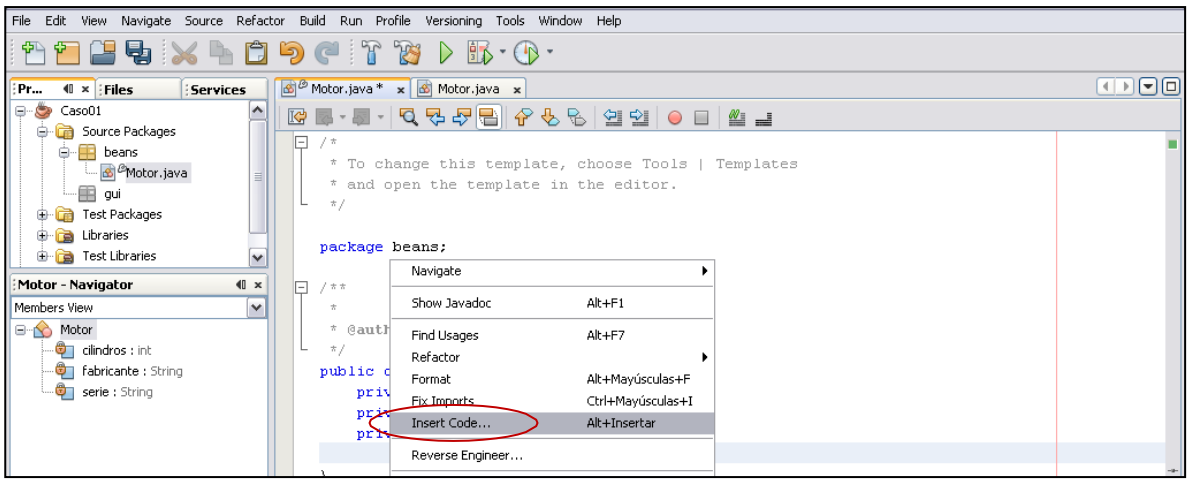




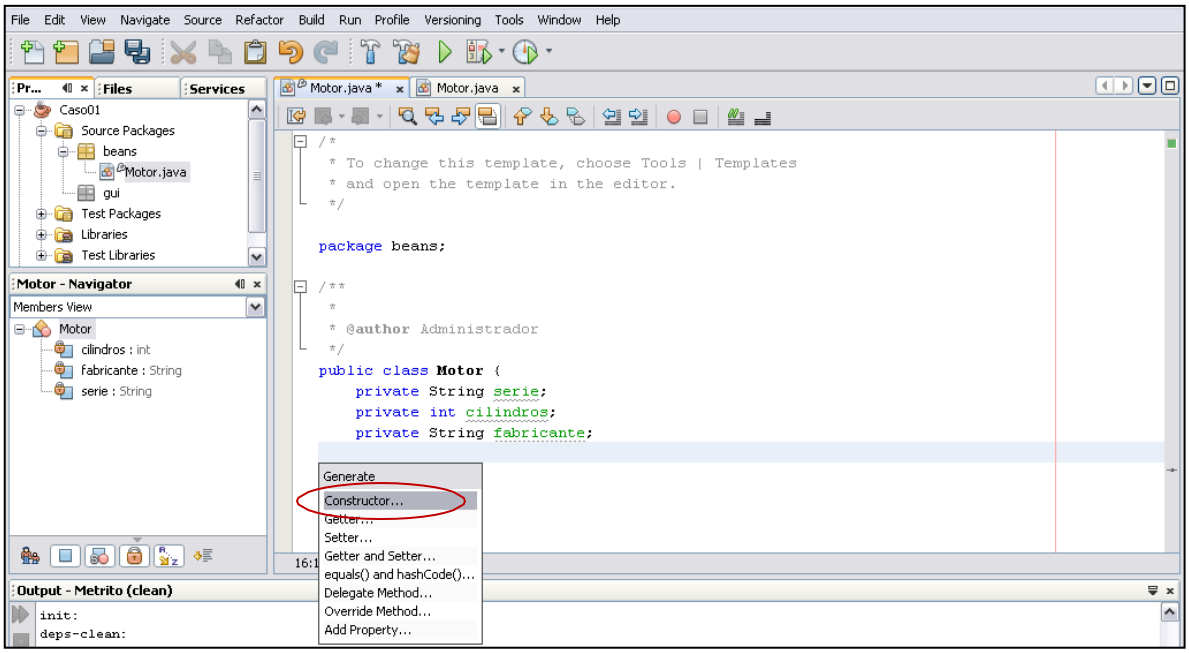
Según en concepto anterior empezamos a crear nuestras clases *Transporte, Motor, Auto y Avión* en el paquete **bean**.



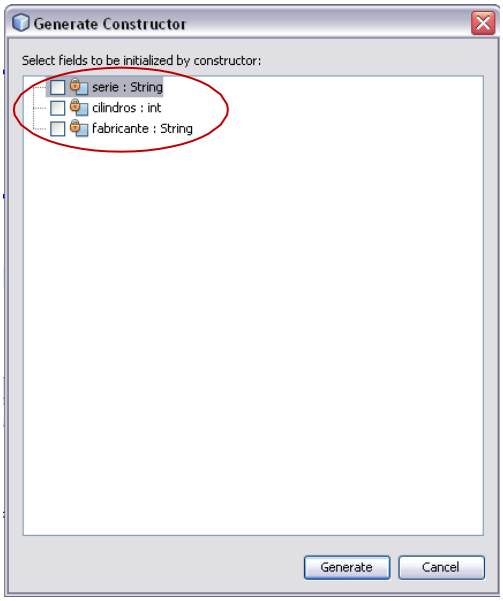
Java nos facilita el trabajo a los desarrolladores cuando se trata de crear los constructores…como se muestra en la siguiente figura:

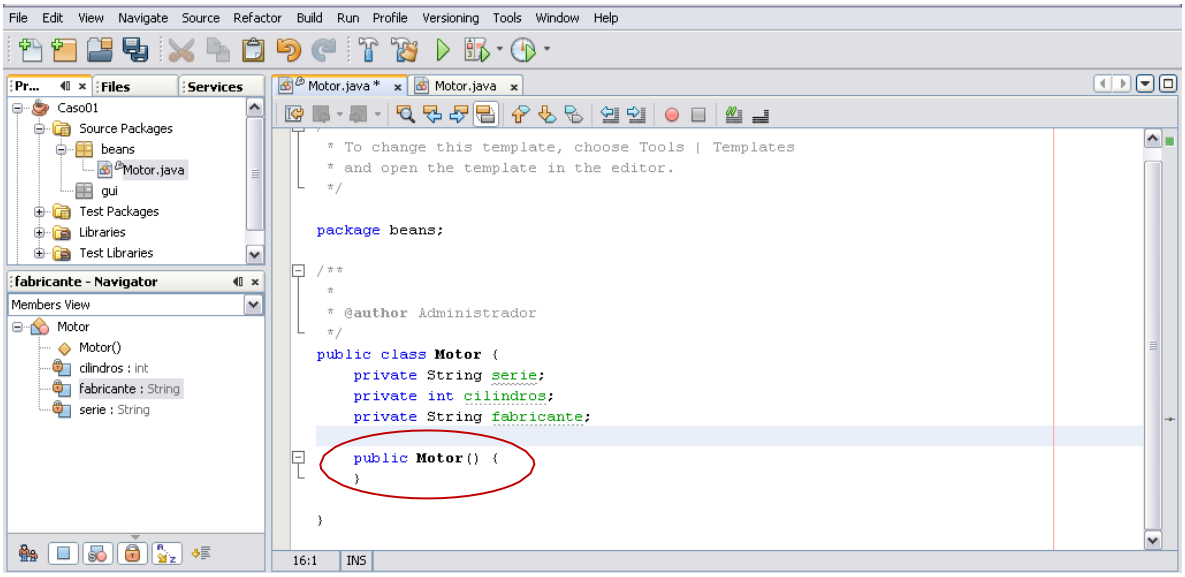


Ahora insertaremos dos constructores a nuestra clase.

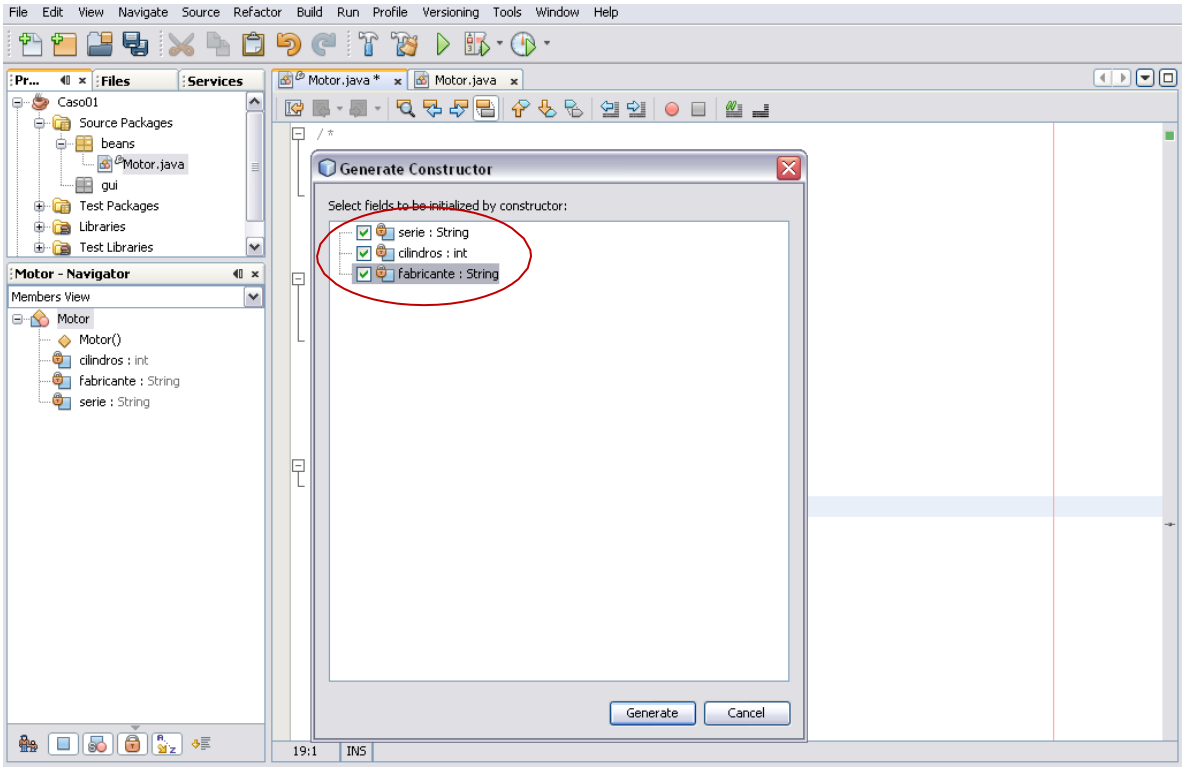


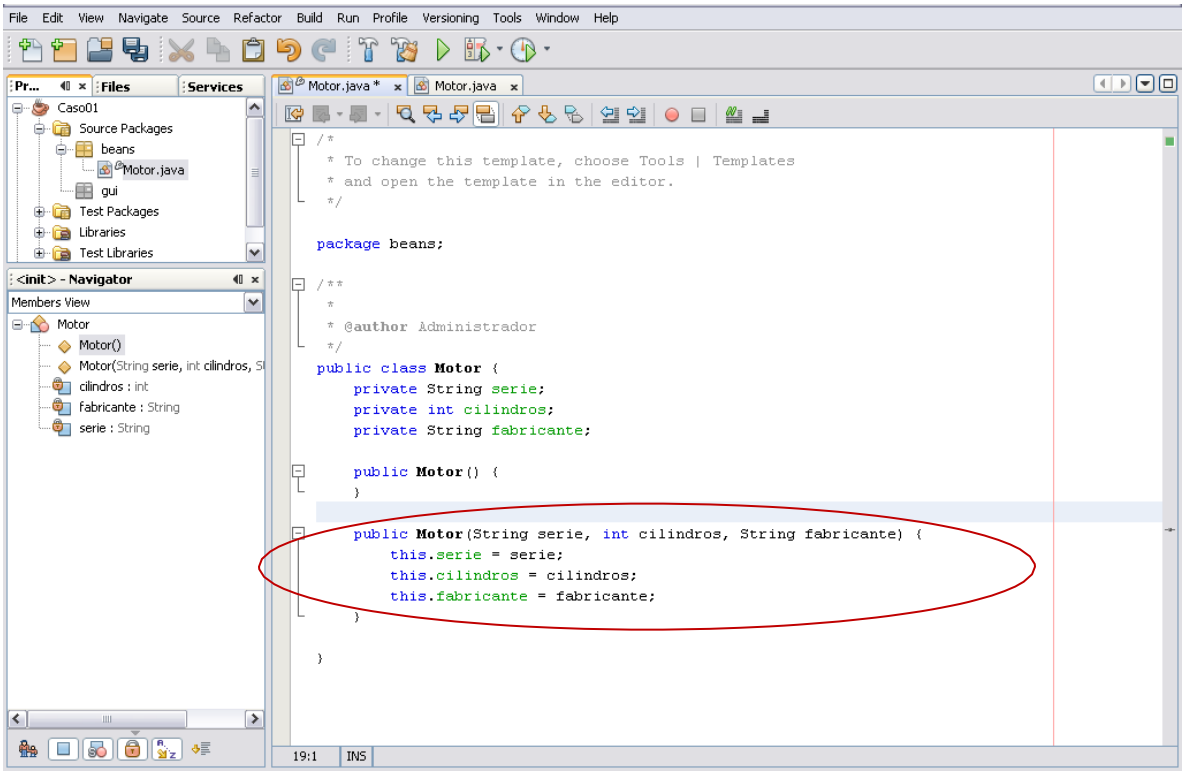
Cuando queramos crear un constructor sin parámetros no le debemos dar clic en ninguno de los campos que se nos aparece, *además debemos tener en cuenta que de no tener un constructor sin parámetros java le implementa uno por defecto.*



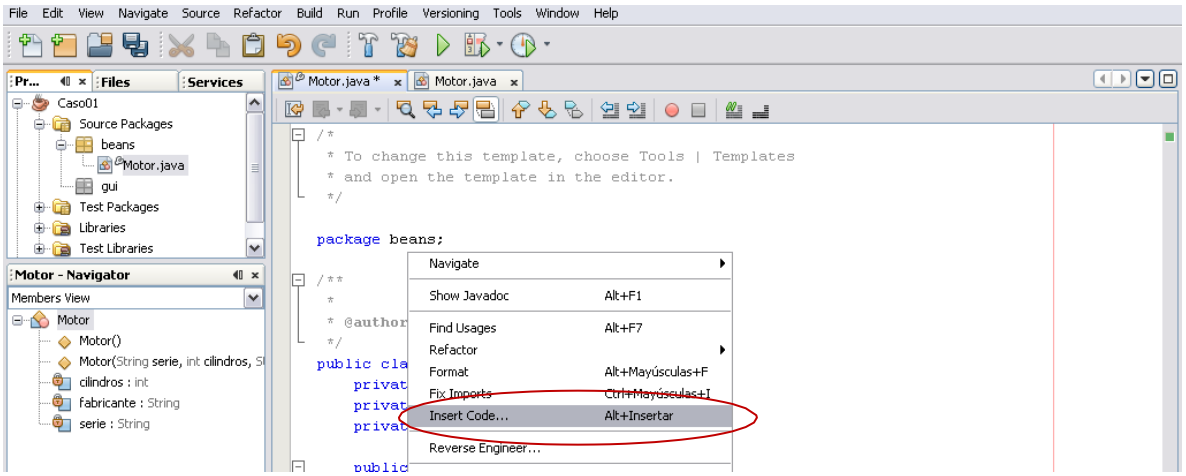


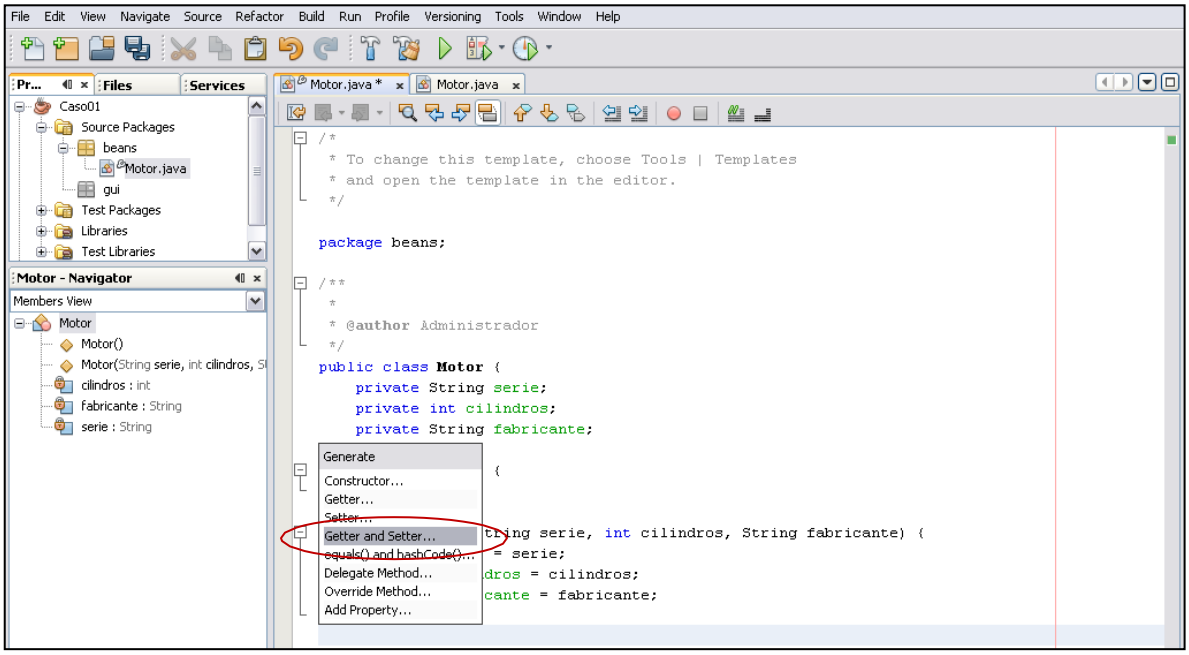
Ahora creamos un constructor con parámetros, recordemos que la idea de los constructores es establecer valores iniciales a los campos. Nótese algo interesante cuando implementemos este constructor, en realidad estamos provocando una **sobrecarga** de este método.



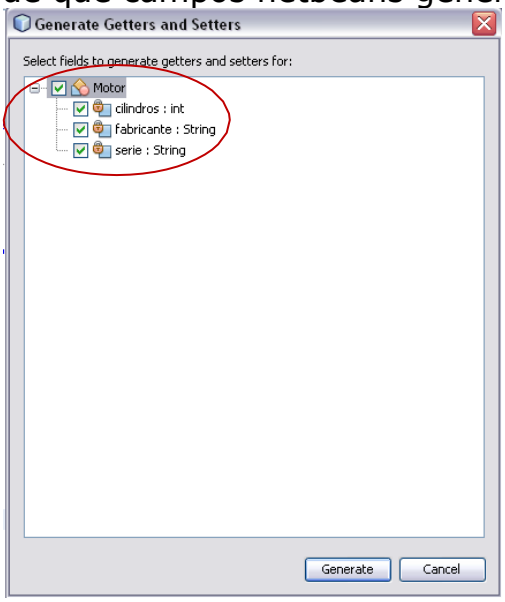


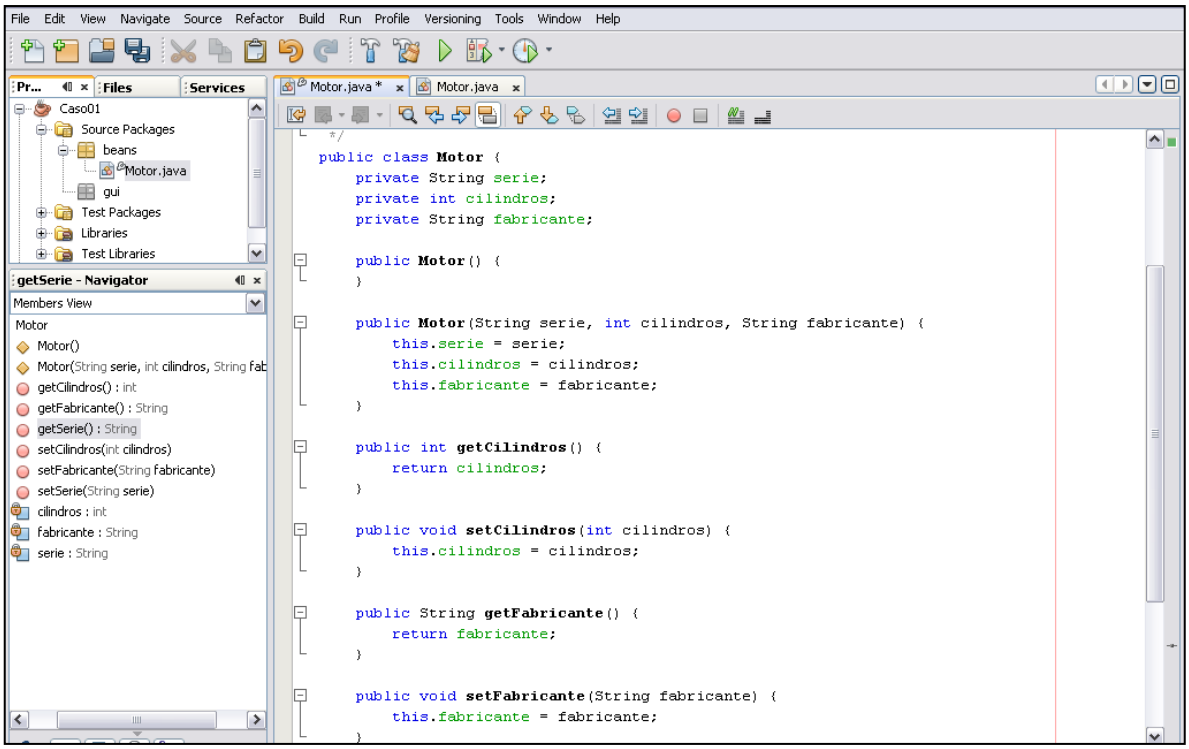
Volvemos a insertar código, pero esta será para los métodos y funciones que permitirán acceder a nuestros campos privados



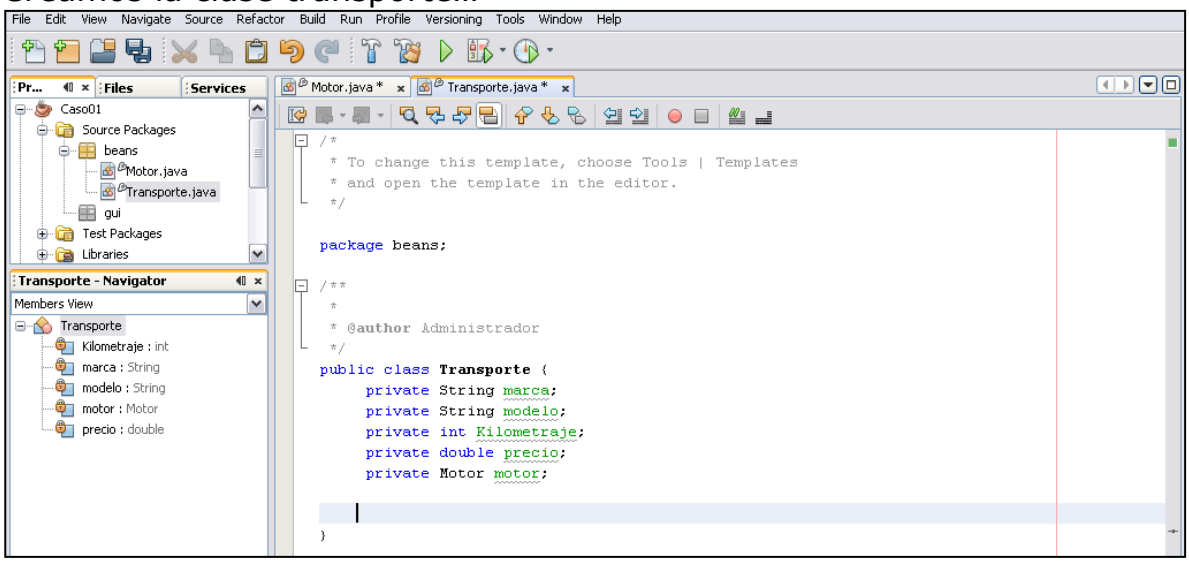


Debemos especificar de qué campos netbeans generara los get y set.

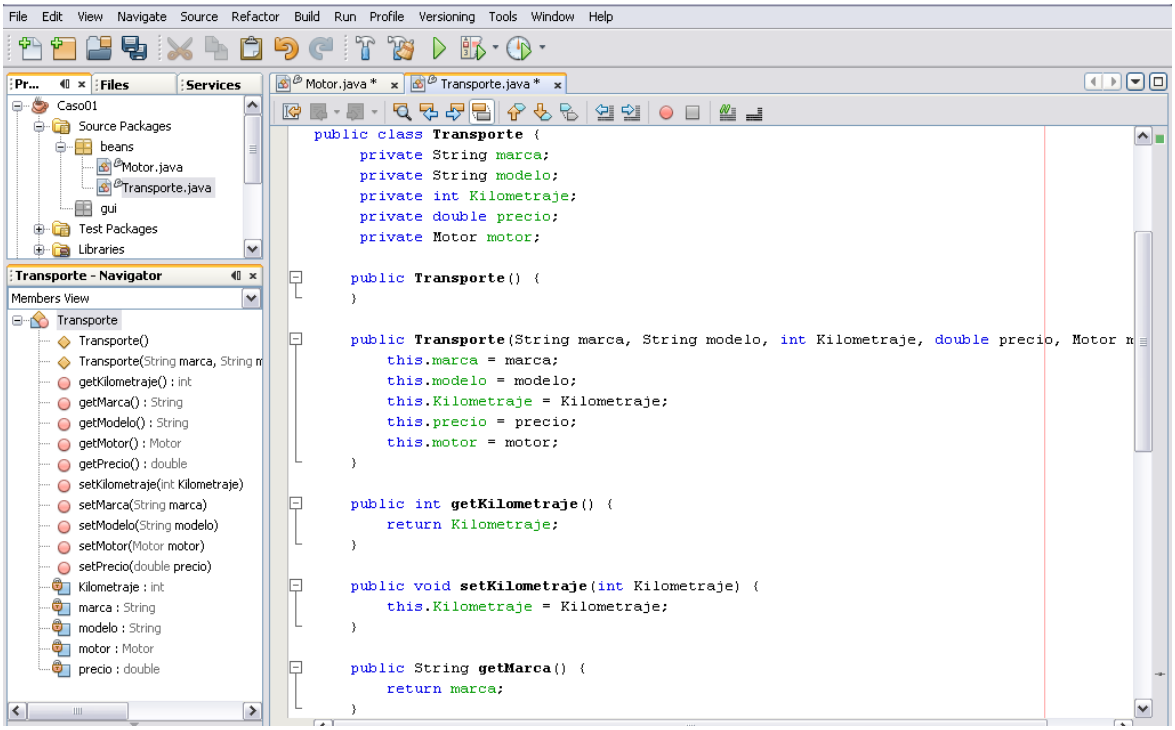




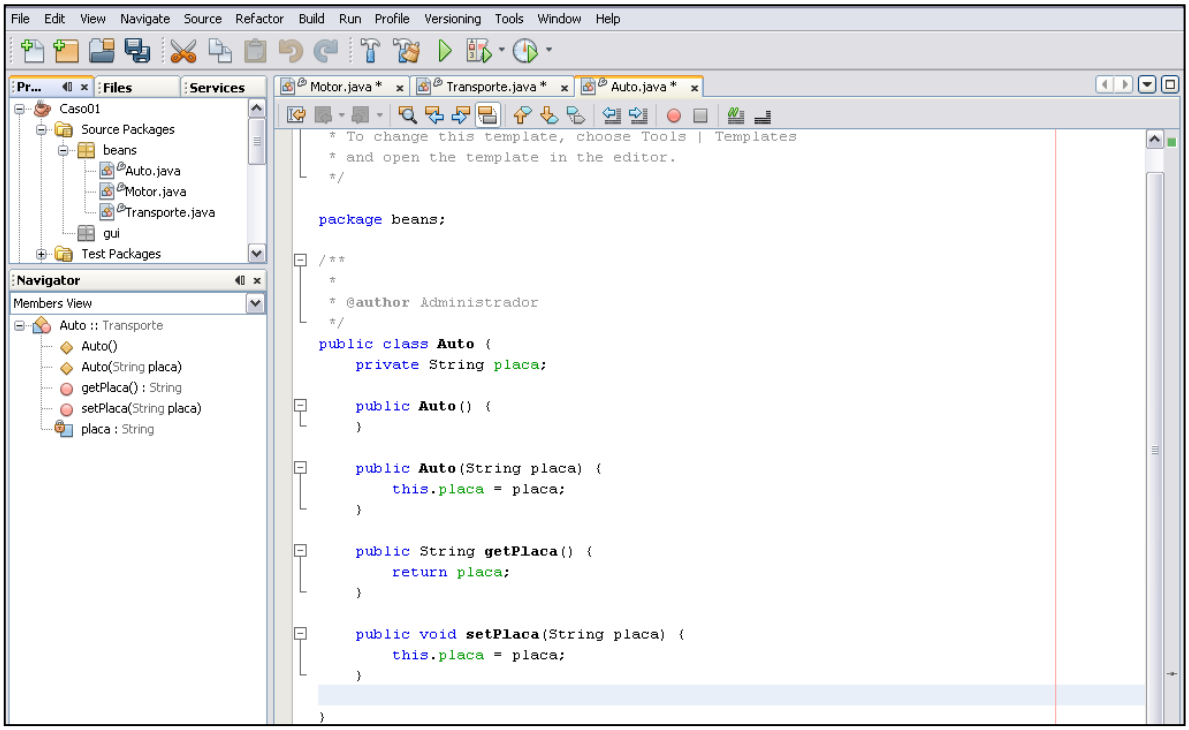
Creamos la clase transporte…



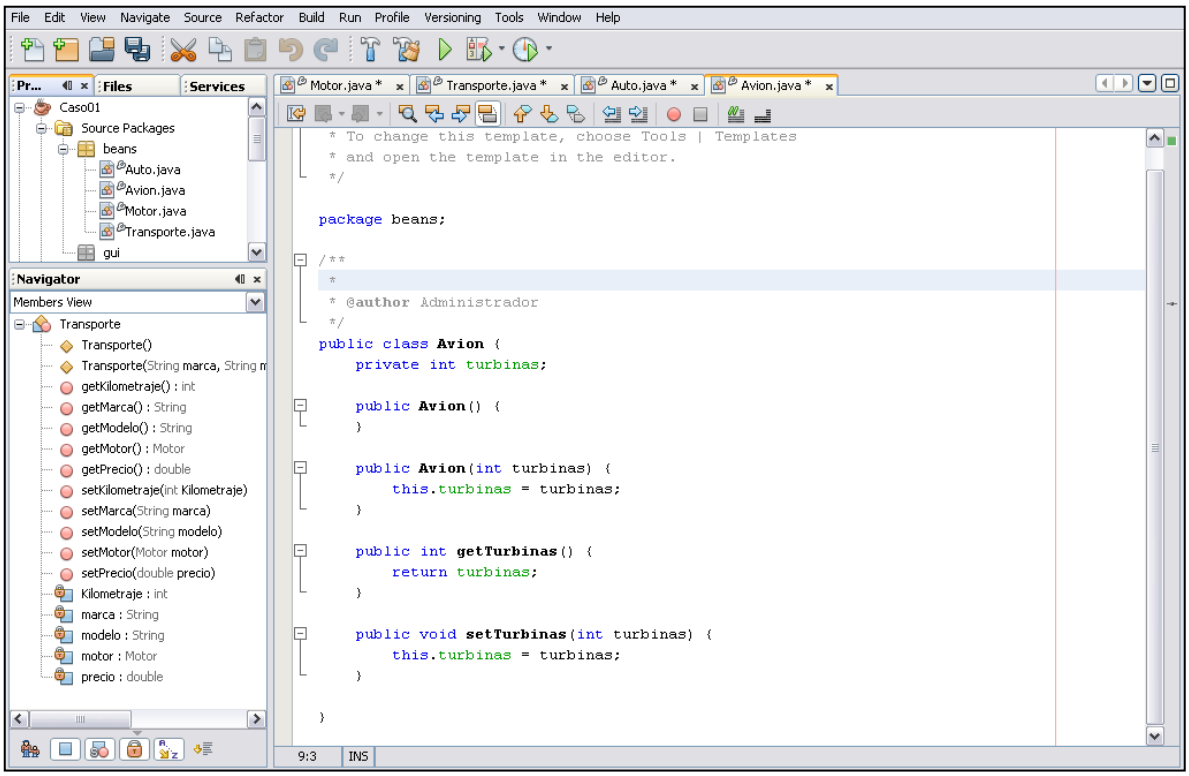
Y de la misma manera que en la clase Motor, generamos una sobrecarga de los constructores y los get y set de los campos.



Creamos la clase Auto…



Creamos la clase Avión…

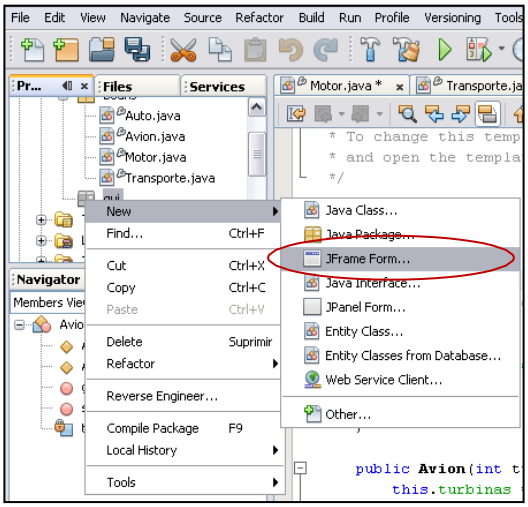


IMPORTANTE:

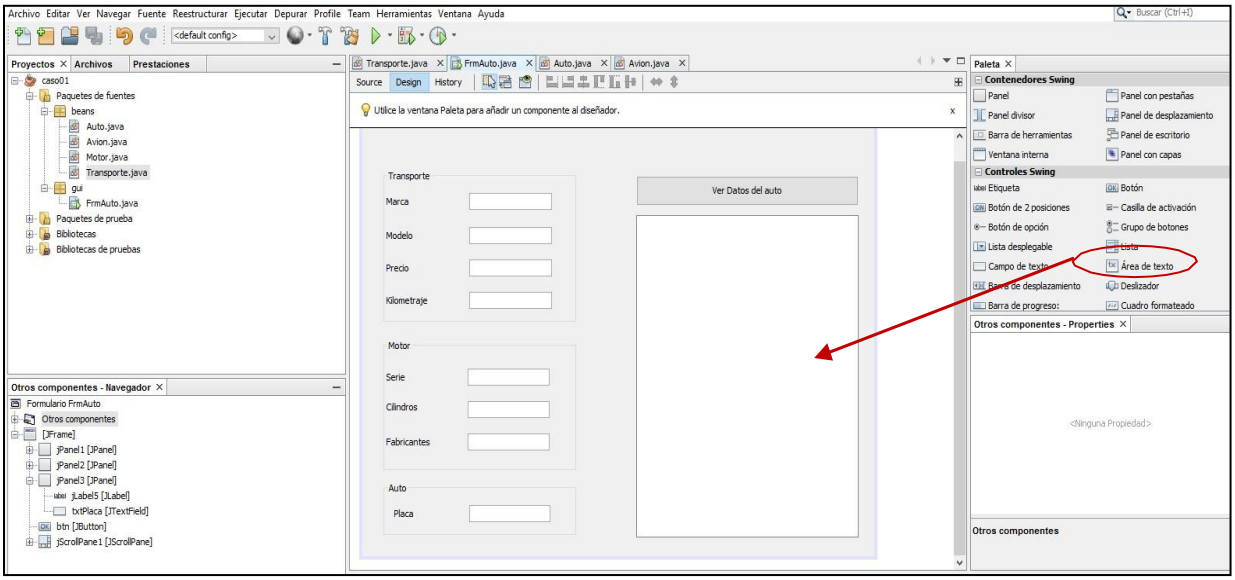
Recuerde que debemos realizar la herencia de las clases "Avion" y "Auto" con la clase "Transporte":

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

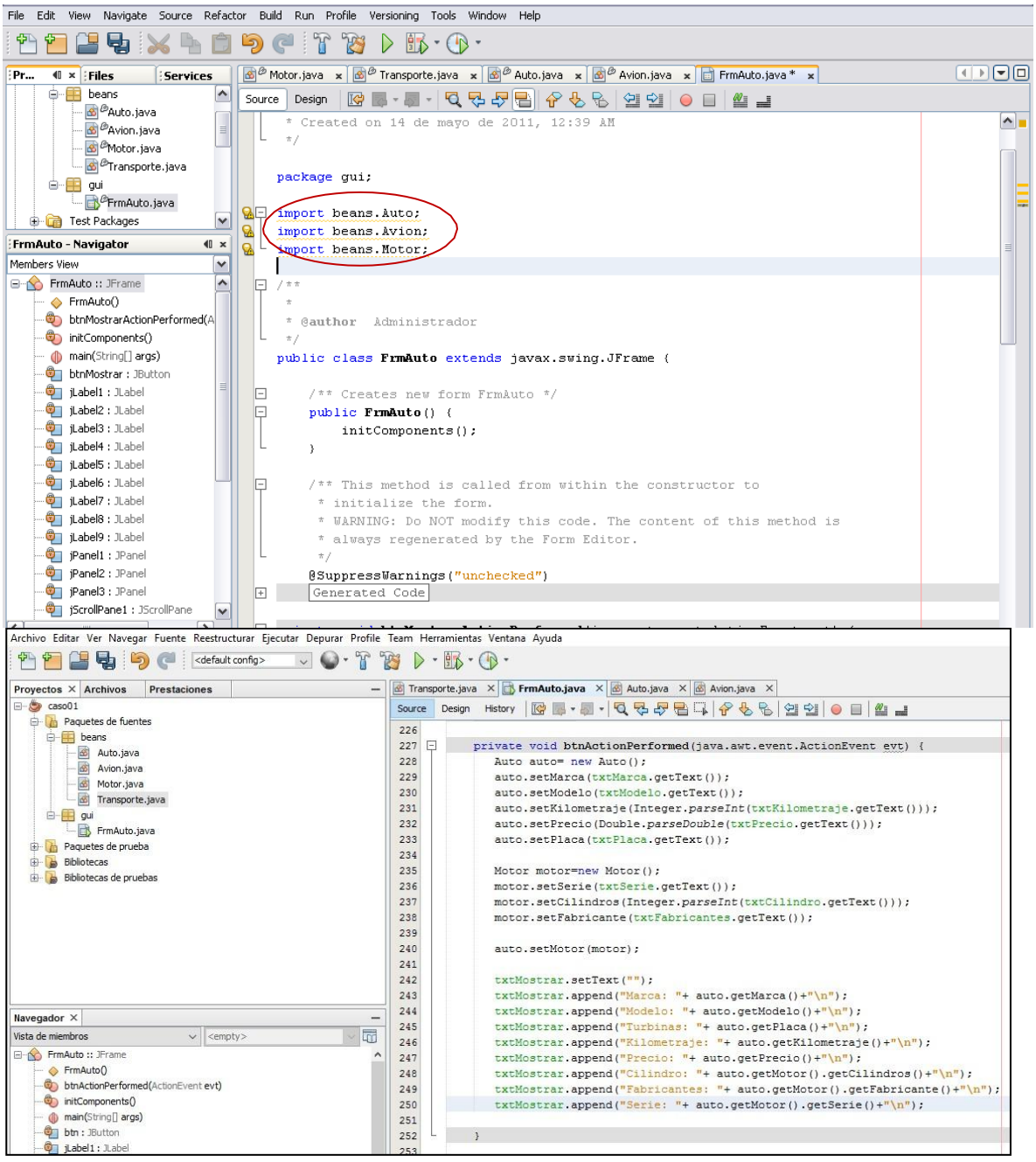
Ahora implementamos nuestro paquete **GUI,** donde alojaremos nuestro formulario que servirá para administrar los datos a nuestras clases.



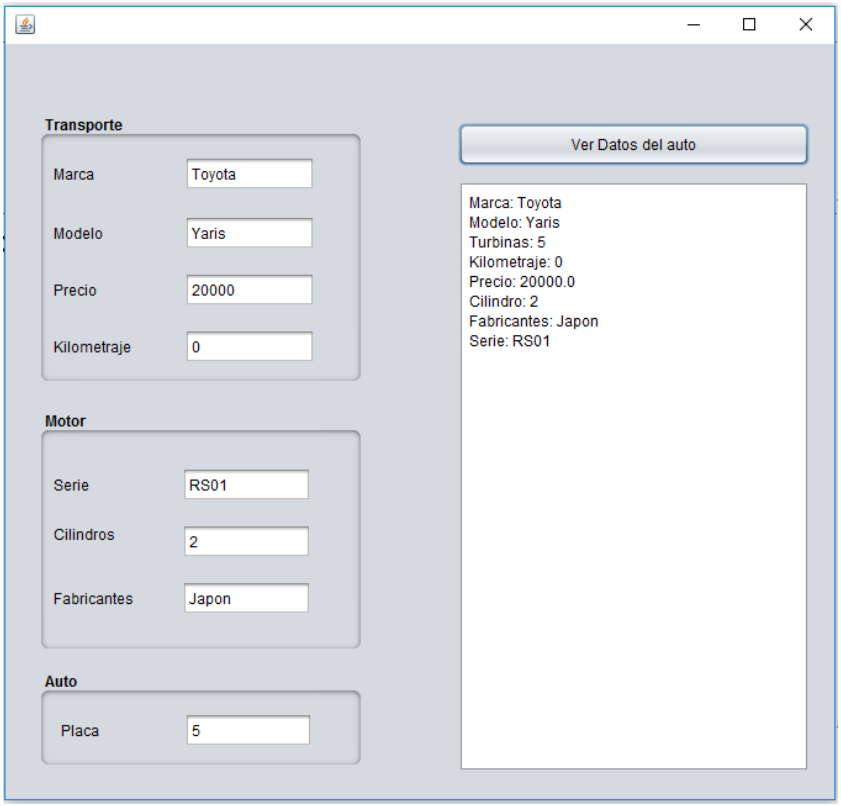
Realizamos el diseño adecuado de nuestro formulario**.**



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 manejaran los datos de Transporte.

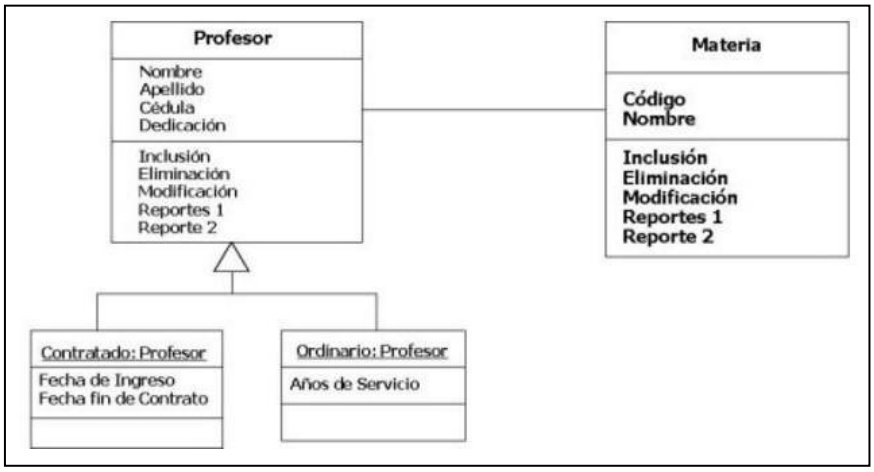

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Files Services
beans
Auto.java
Avion.java
Motor.java
Transporte.java
gui
FrmAuto.java
Test Packages
FrmAuto - Navigator
Members View
FrmAuto :: JFrame
FrmAuto()
initComponents()
main(String[] args)
btnMostrar : JButton
Motor.java Transporte.java Auto.java Avion.java FrmAuto.java *
Source Design
* Created on 14 de mayo de 2011, 12:39 AM
package gui;
import beans.Auto;
import beans.Avion;
import beans.Motor;
/**
* @author Administrador
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
Paquetes de prueba
Bibliotecas
Bibliotecas de pruebas
Navegador
Vista de miembros
Transporte.java FrmAuto.java Auto.java Avion.java
Source Design History
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.