UNNOBA
UNIVERSIDAD NACIONAL
NOROESTE | BUENOS AIRES

IO - 2019

Departamento de Informática y Tecnología **Guía de JAVA-Like y BlueJ**

Objetivos:
El objetivo de esta mini guía es explicar el lenguaje que vamos a usar a partir del trabajo práctico número 2.

Por un lado vamos a utilizar un lenguaje de programación basado en un pseudo JAVA (JAVA-LIKE) con algunas limitaciones para facilitar los ejercicios.

Y por otro lado se brinda un mini tutorial-guía de *BlueJ*. *BlueJ* es un mini ambiente de desarrollo basado en JAVA para poder aprender Objetos de una manera sencilla y sin las complicaciones inherentes al lenguaje. La última versión de BlueJ es la **4.2.0 del 7 de Febrero del 2019**

De Wikipedia http://es.wikipedia.org/wiki/Lenguaje_de_programación_Java

Java *es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria*

BlueJ *is an Integrated Development Environment for the Java programming language, developed mainly for educational purposes, but also suitable for small-scale software development.*

JAVA-LIKE

A continuación se da una guía de cómo resolver los ejercicios de los trabajos prácticos en JAVA-LIKE.

Nos vamos a enfocar en:

1. Definición de clases
2. Declaración de atributos
3. Métodos "especiales" constructores
4. Métodos getter y setter
5. Métodos
6. Colecciones básicas
7. Estructuras de control básicas
8. Salida por consola
9. Uso de Fechas (Date)

```
1. Definicion de clases

class NombreClase [extends SuperClase]{
/**
 * Codigo de la clase
 */
}
```

Lo que esta entre [ ] es opcional. Es decir, puede ir como no. En este caso el **extends** significa que extiende de otra clase. Es decir, es subclase de…

2. Declaración de atributos

Solamente vamos a necesitar (por ahora) atributios que contengan String, Integer, Float, Booleans y Dates.

Recordar que los atributos SIEMPRE deben ir como private. Para poder acceder a ellos se lo debe hacer mediante métodos para no romper el encapsulamiento.

```
private String attr1;
private int attr2;
private float attr3;
private boolean attr4;
private Date attr5;
```

3. Metodos Constructores

Es el primer método que se llama cuando se instancia una clase. Es un método especial que sirve para "construir" el Objeto a partir de la Clase. Como todo método, se le pueden enviar parámetros. No tiene valor de retorno.

```
public NombreClase(){
/**
 * Codigo que sirve para inicializar datos
 */
}
```

4. Metodos getters y setters

Los métodos pueden ser públicos o privados. Los privados solo son conocidos por el mismo objeto y sirven para factorizar métodos en métodos más pequeños. En cambio los públicos son vistos por cualquier otro objeto. Si es un **seter** (se reconoce por el prefijo set, el valor de retorno es **VOID**, en cambios si es un **getter** (prefijo get) el valor de retorno debe ser igual al que se retorna en el **return**.

```
public|private TipoRetorno getNombreMetodo() {
            return nombre;
}

public|private void setNombre(TipoParametro parametro) {
            this.nombre = nombre;
}
```

5. Métodos

Los métodos son el código que se ejecuta cuando un Objeto recibe un mensaje. Pueden llamarse de cualquier forma siempre que comiencen con minúscula. Se deber respetar las mismas reglas que para los getter y setters en lo que respecta a los modificadores de acceso y valores de retorno.

6. Colecciones basicas

Como colecciones vamos a usar solo colecciones básicas. En este caso vamos a usar Vector:

```
private Vector<String> col = new Vector<String>();
```

A los vectores (y a otras colecciones más) hay que indicarles que objetos van a contener (entre < y >). En este caso Strings.
A una colección se le pueden enviar muchos mensajes. Pero los más importantes para nosotros son:

```
col.add(elemento);

col.remove(elemento);

col.size();

col.contains(elemento);

col.clear();

col.firstElement();

col.lastElement();

col.isEmpty();
```

7. Estructuras de Control

```
if (cond) {
      // Codigo si verdadero
      }[ else {
            // Codigo si falso
      }]
```

```
for (String elemento : getCol()) {
            /*
             * código
             */
      }
```

El **for-each** sirve para recorrer una colección. En este caso estamos recorriendo **col** que es la colección de Strings declarada anteriormente.

8. Salida por consola

Para poder imprimir valores por consola hay un método especial del sistema.

```
System.out.println(valores);
```

Por ejemplo:

```
System.out.println("Hola Mundo");
```

9. Uso de Fechas (Date)

Todos los lenguajes de programación tienen alguna representación de fechas. En el caso de JAVA-Like es el objeto Date.

Por ejemplo:

El **new Date()** retorna un objeto date que representa la fecha actual.

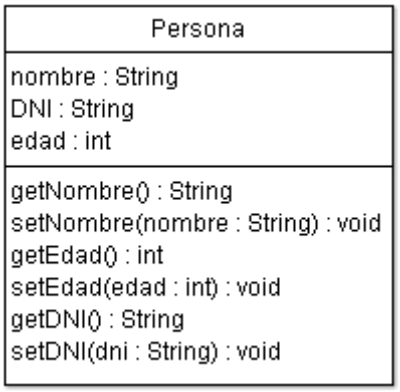
```
Date date = new Date();
System.out.println("Today is " + date);
```

Ejemplo 2:

```
String stringDate = "31/12/2010";
DateFormat formatter  formatter = new SimpleDateFormat("dd/MM/yyyy");
Date date = (Date)formatter.parse(stringDate);
```

En este caso estamos creado la fecha a partir de un String con el formato **dd/MM/yyyy**

Ejemplo con la clase Persona
Supongamos que tenemos el siguiente diagrama de clases:




```
class Persona {
	private String nombre;
	private int edad;
	private String DNI;

	public Persona(){

	}

	public String getNombre() {
		return nombre;
	}

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

	public int getEdad() {
		return edad;
	}

	public void setEdad(int edad) {
		this.edad = edad;
	}

	public String getDNI() {
		return DNI;
	}

	public void setDNI(String dni) {
		DNI = dni;
```

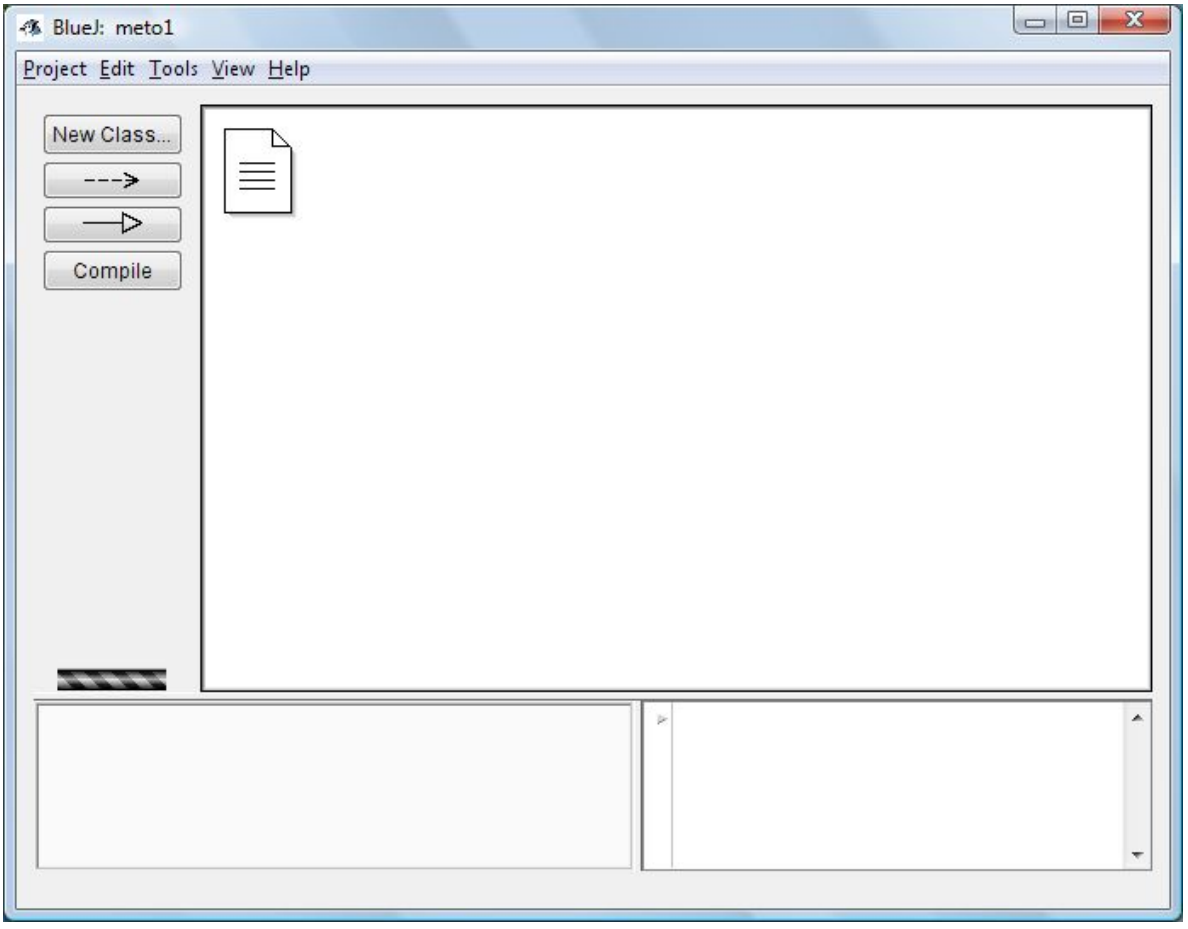
```
        }

}
```

Guia de BlueJ

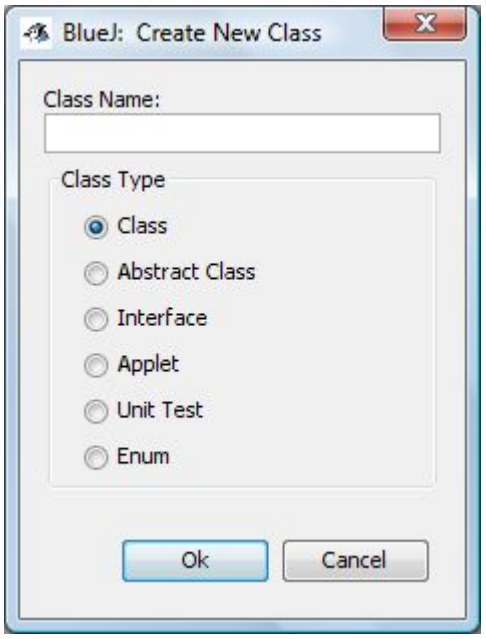
El BlueJ se puede bajar de la página: http://www.bluej.org/download/download.html Es necesario que tengan instalado el JRE 1.7.x de JAVA (http://java.sun.com/) o Superior.

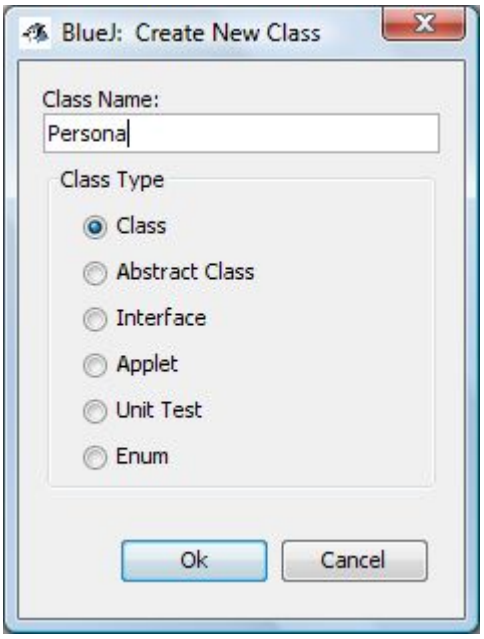
En el ejemplo vamos a ver paso a paso como crear la clase Persona y luego trabajar con ella.

Una vez que tengamos el BlueJ instalado (y el JRE de JAVA) al iniciar el BlueJ obtendremos la ventana principal del IDE.

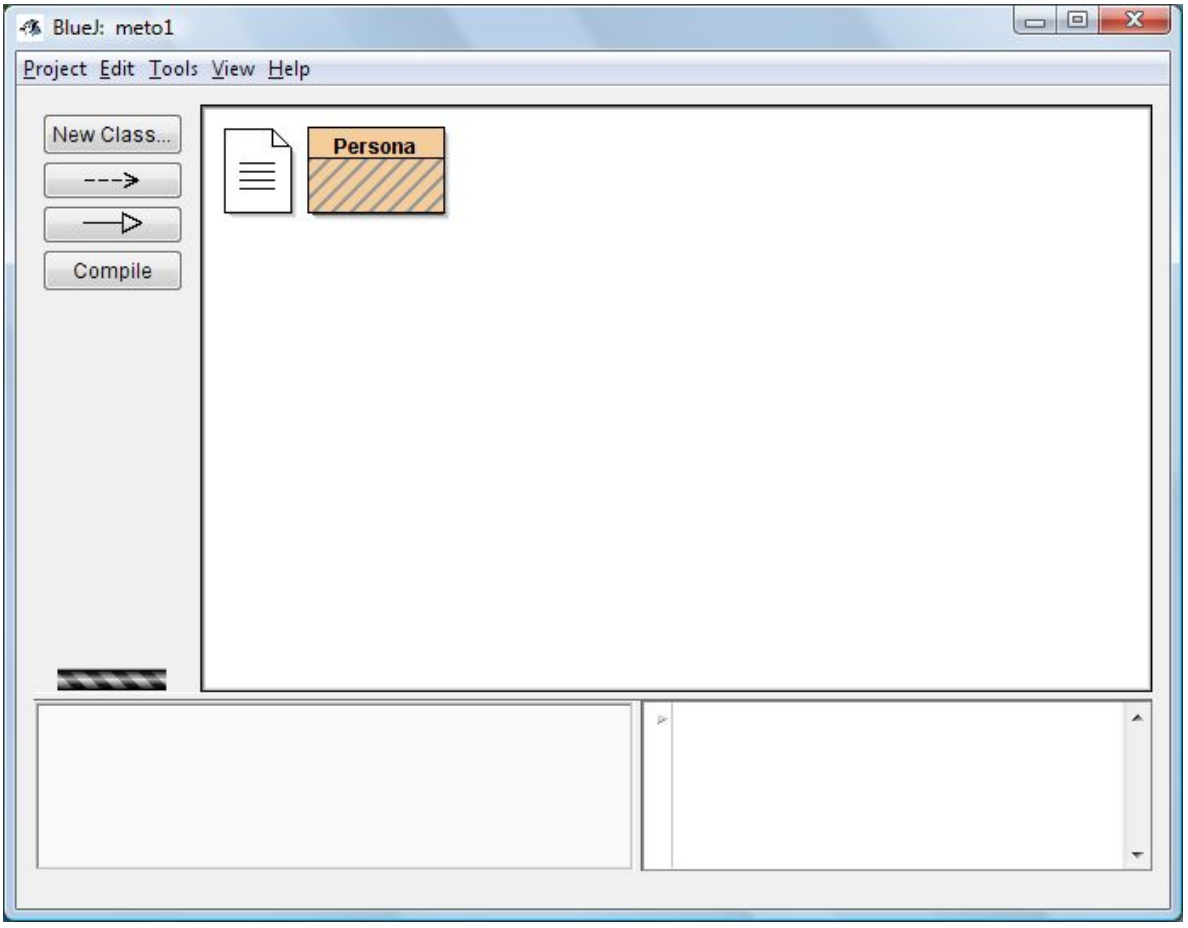


Al hacer click en New Class se abrirá un cuadro de dialogo para ingresar en nombre de la clase (recuerden que la primera letra del la clase debe ser en mayúscula).

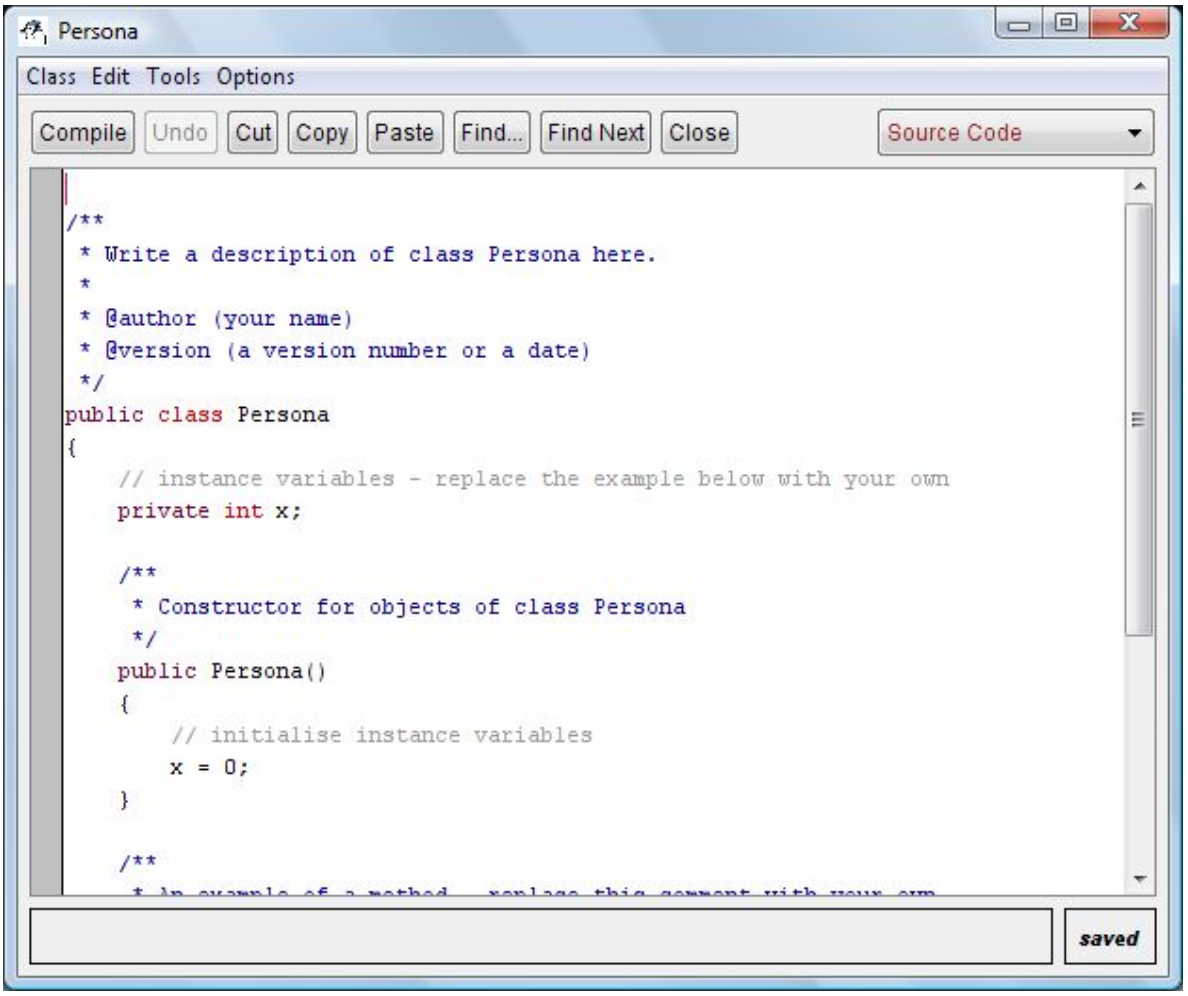




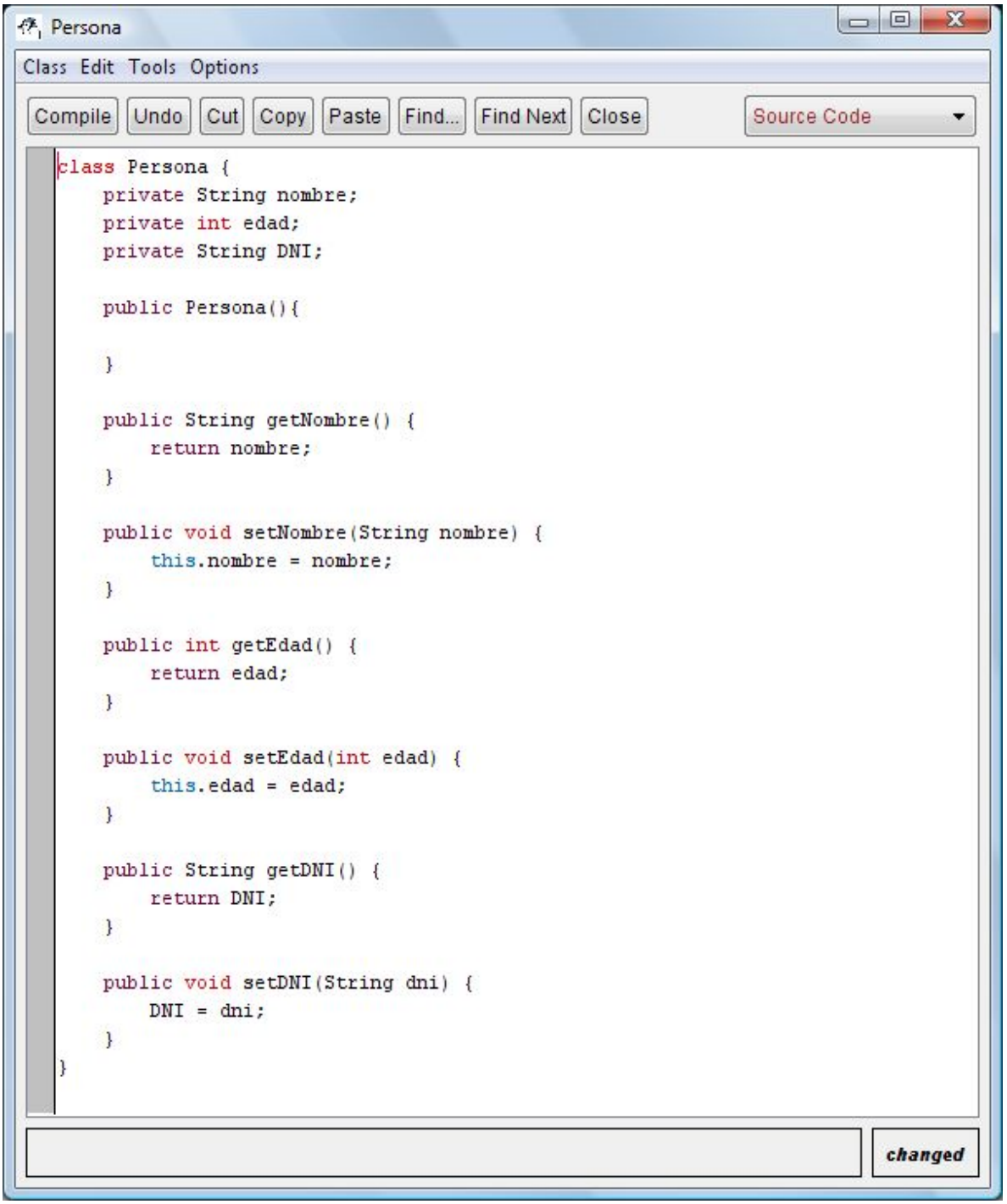
Luego de ingresar el nombre, debe aparecer en la pantalla principal la clase en un pseudo UML.

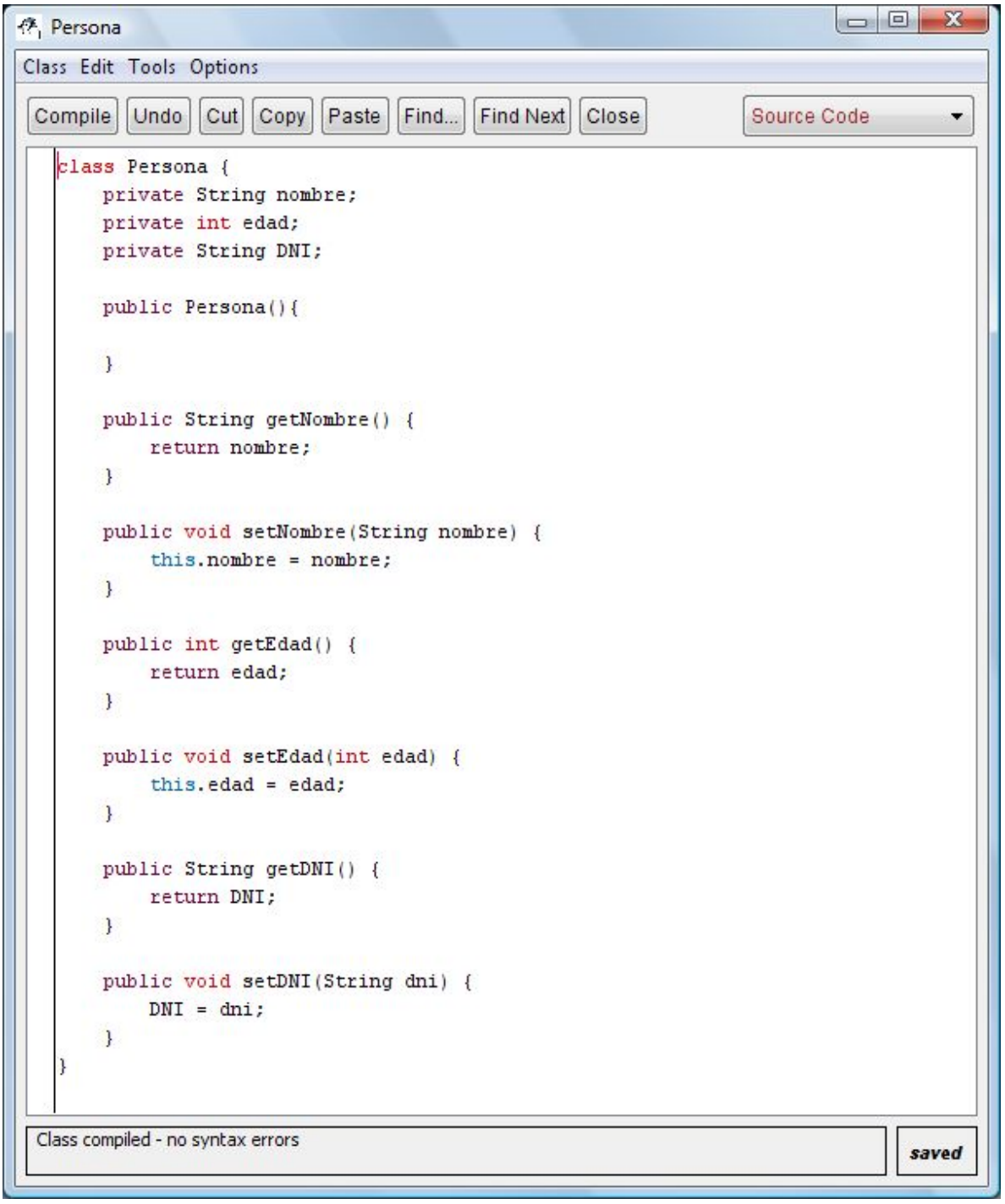


Haciendo doble click sobre la misma ingresaremos a la ventada de edicion/codificacion de la clase. Ahí pueden observar codigo JAVA de ejemplo.

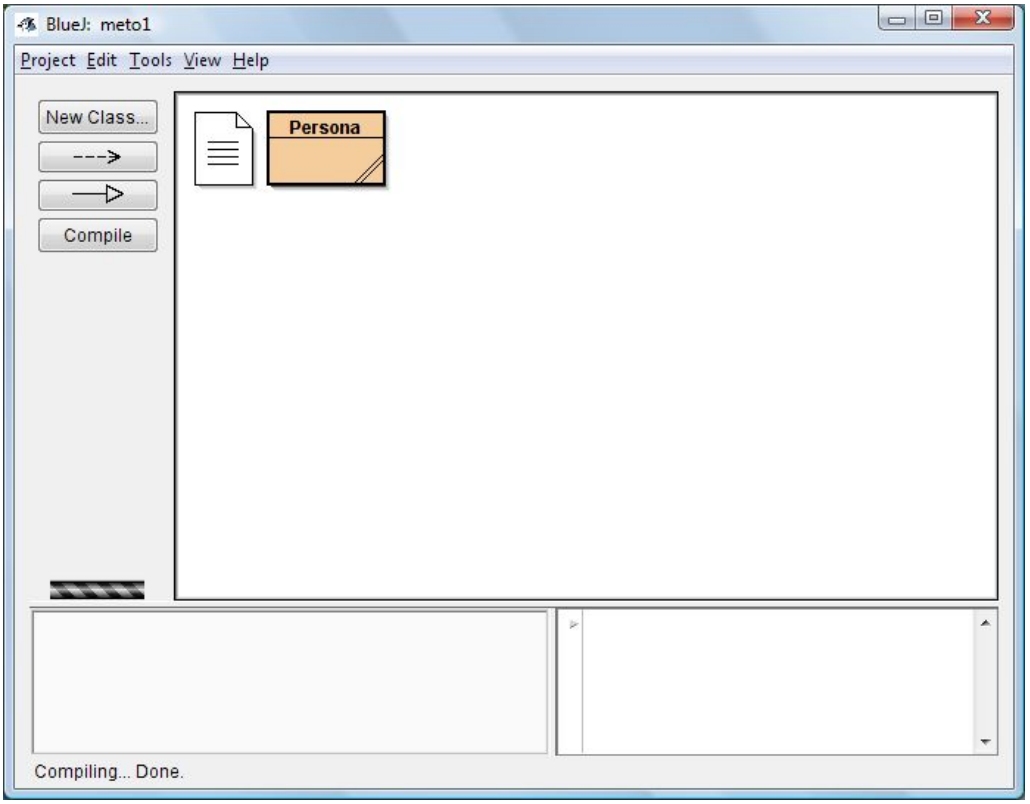


Borramos dicho código y copiamos y escribimos nuestro código de la clase persona.

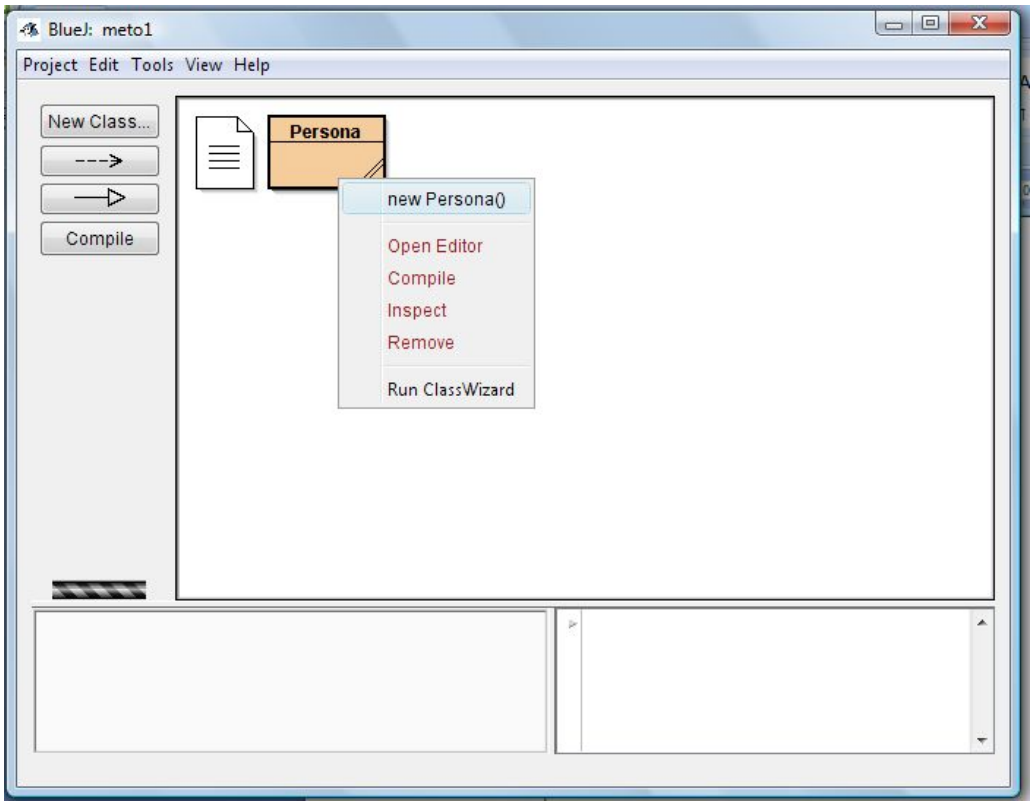

Persona
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Find Next Close
Source Code
class Persona {
private String nombre;
private int edad;
private String DNI;
public Persona(){
}
public String getNombre() {
return nombre;
}
public void setNombre(String nombre) {
this.nombre = nombre;
}
public int getEdad() {
return edad;
}
public void setEdad(int edad) {
this.edad = edad;
}
public String getDNI() {
return DNI;
}
public void setDNI(String dni) {
DNI = dni;
}
}
changed



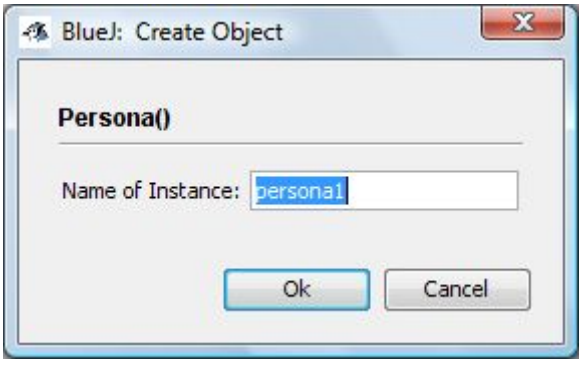
Guardamos (save) y compilamos (compile). Y al cerrar la ventana de edicion se puede ver en la ventana principal del IDE que el diagrama de clase en diferente. Ya no se ven las rayas en diagonal cubriendo todo la clase.



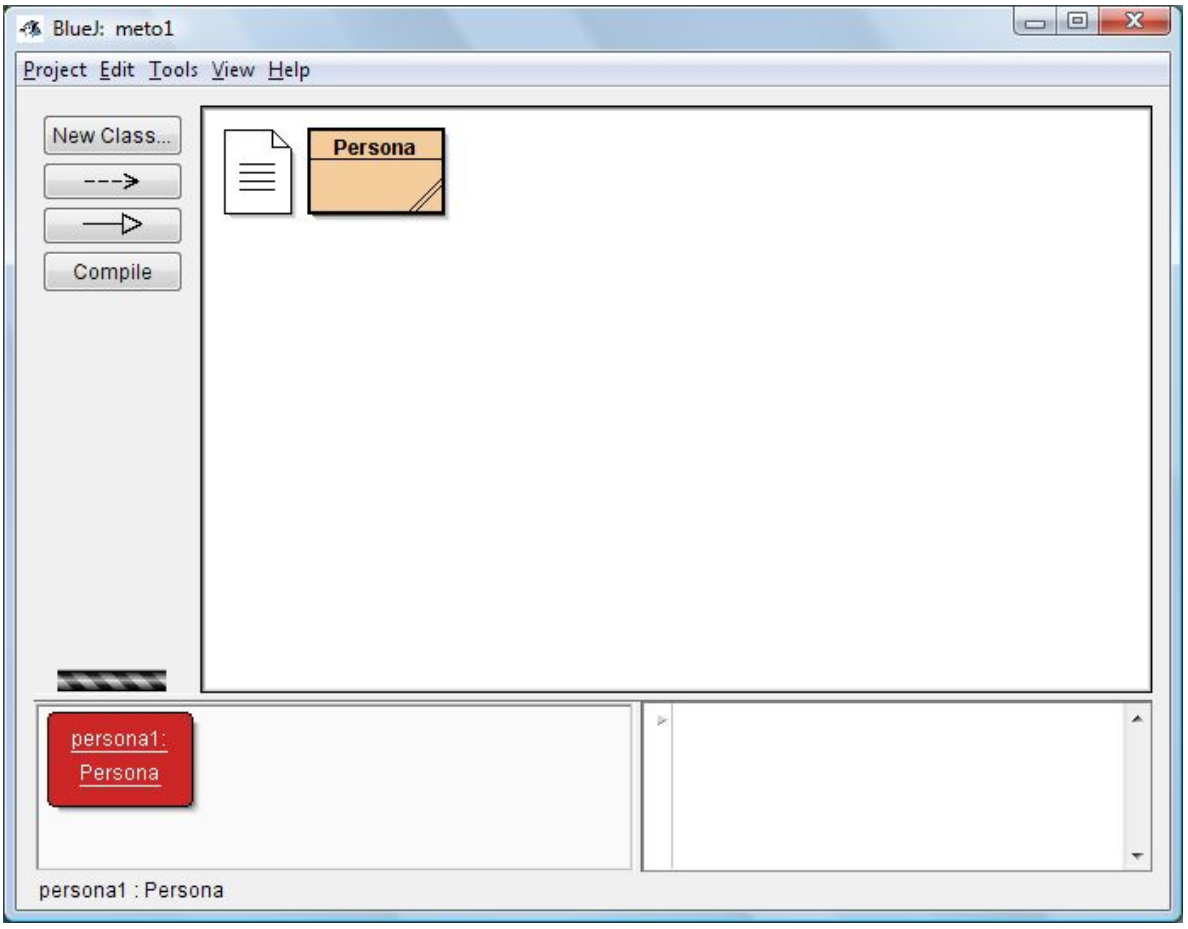
Luego instanciamos (new) un nuevo objeto de la clase persona. Haciendo click con el botón de la derecha.



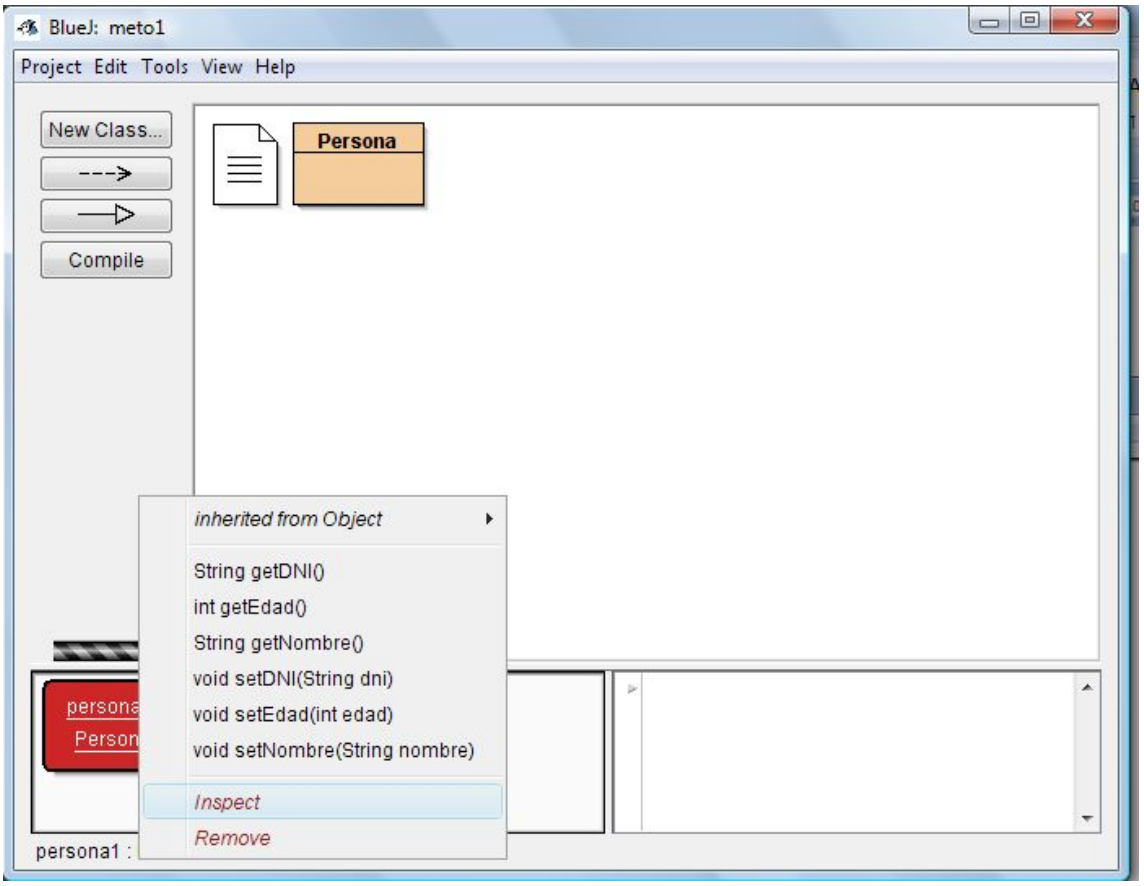
El IDE nos pedirá que le asignemos un nombre a ese Objeto



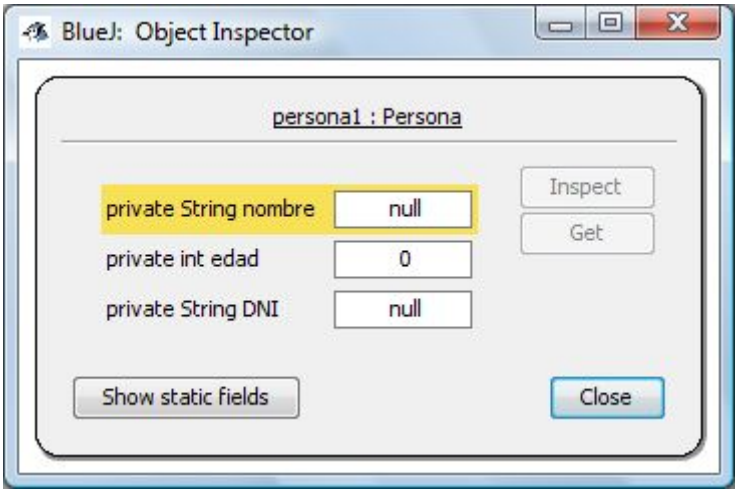
Y luego dicho objeto debe aparecer en el area de objetos instanciados



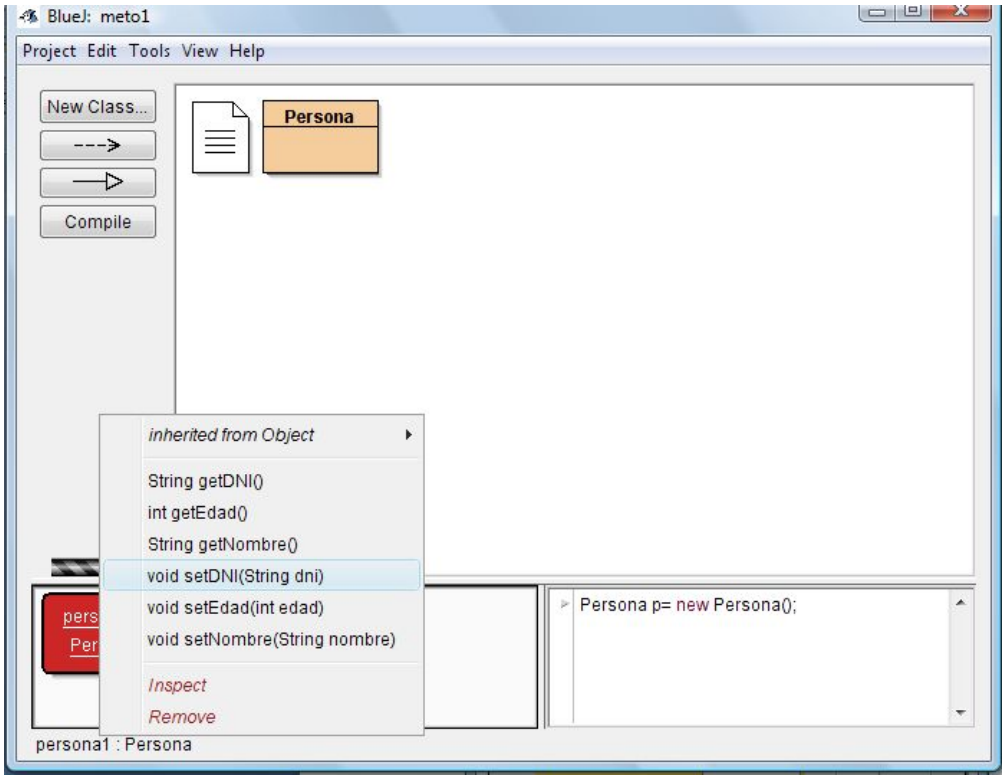
Luego podemos inspeccionar el objeto y ver los valores de sus atributos.

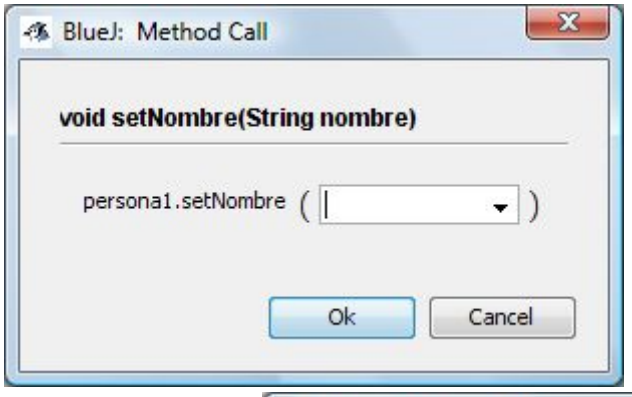


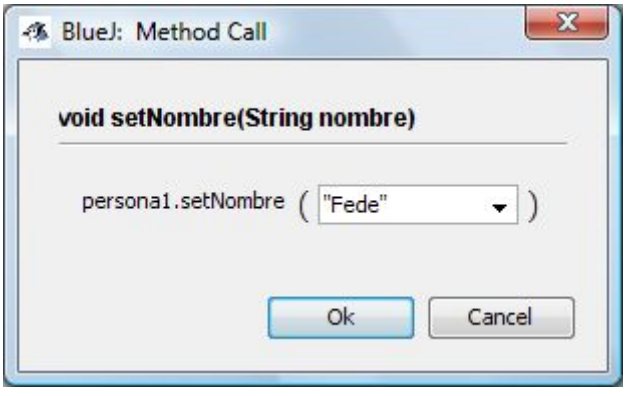
En este caso los atributos no tienen datos (null)



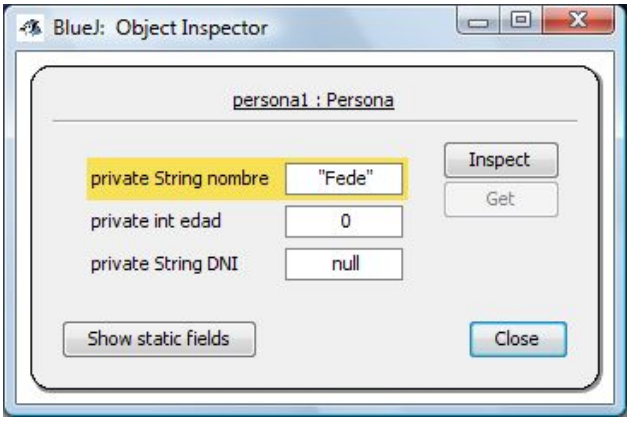
Vamos a setear valores a dicho objeto mediante los setters indicados


BlueJ: meto1
Project Edit Tools View Help
New Class...
Compile
Persona
inherited from Object
String getDNI()
int getEdad()
String getNombre()
void setDNI(String dni)
void setEdad(int edad)
void setNombre(String nombre)
Inspect
Remove
Persona p= new Persona();
persona1 : Persona

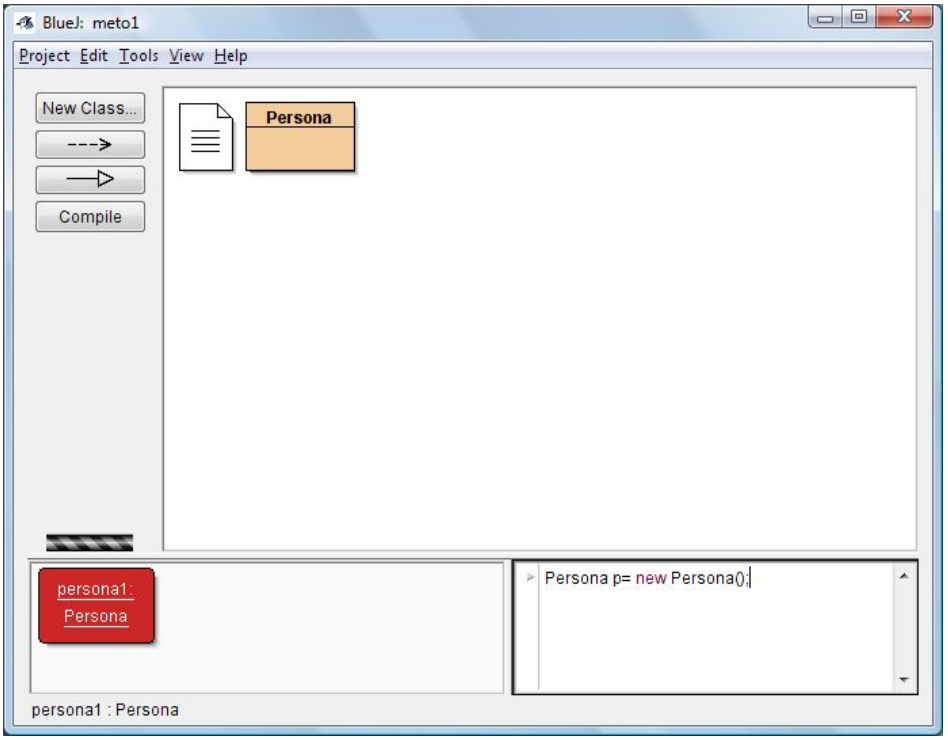

BlueJ: Method Call
void setNombre(String nombre)
persona1.setNombre ( )
Ok
Cancel

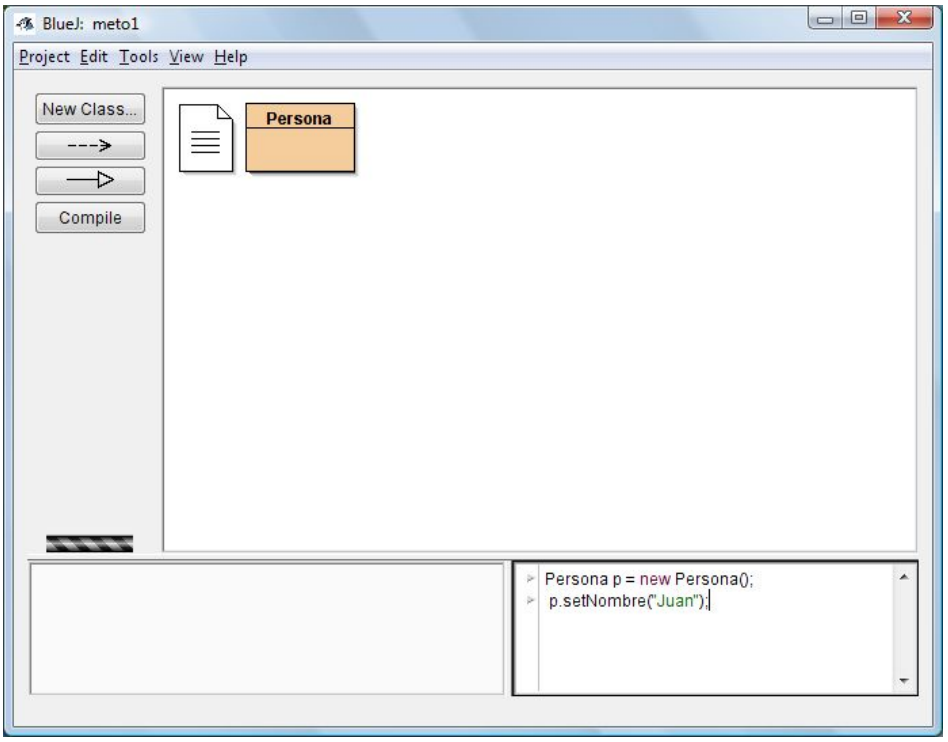

BlueJ: Method Call
void setNombre(String nombre)
persona1.setNombre ("Fede")
Ok
Cancel

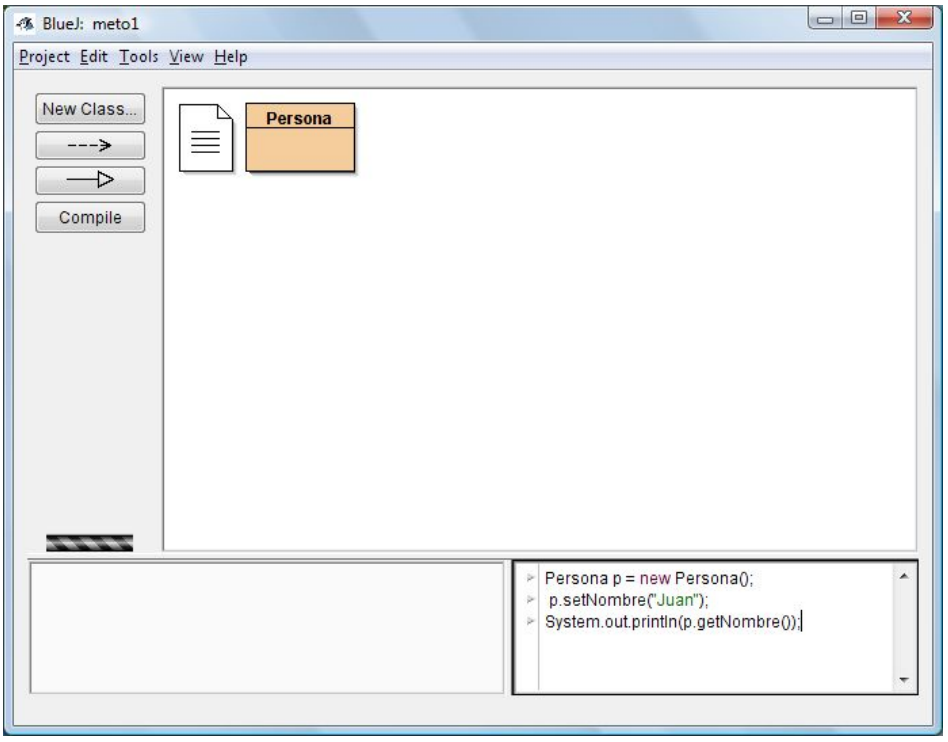
Si volvemos a inspeccionar el objeto veremos que los valores de los atributos cambiaron

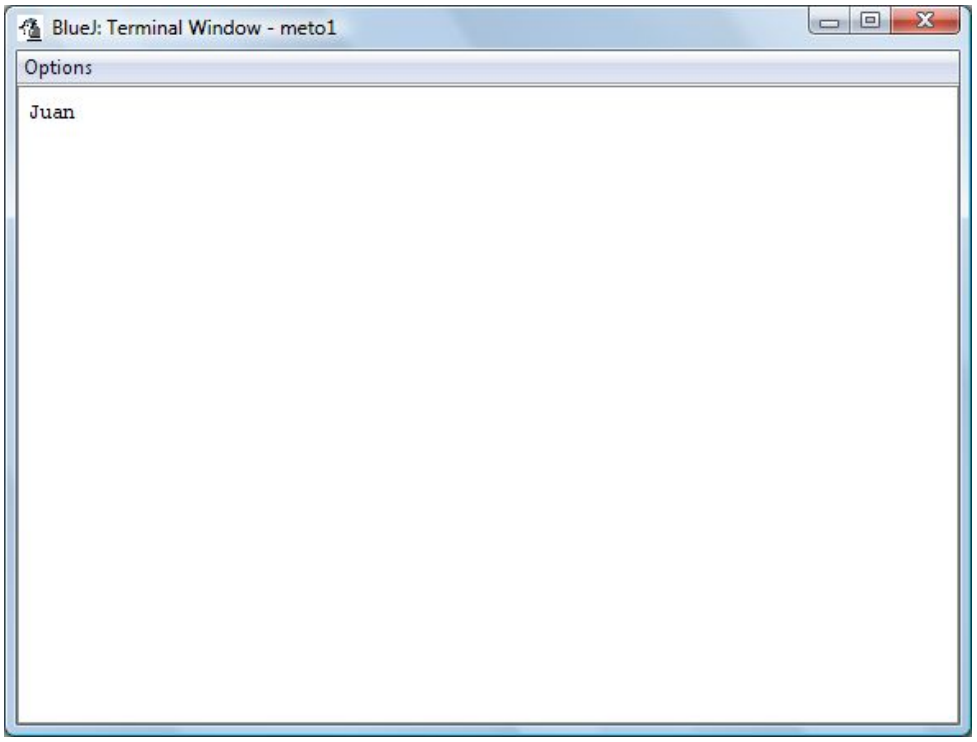


Usando el Workspace y la salida por consola




BlueJ: meto1
Project Edit Tools View Help
New Class...
Compile
Persona
Persona p = new Persona();
p.setNombre("Juan");


BlueJ: meto1
Project Edit Tools View Help
New Class...
Compile
Persona
Persona p = new Persona();
p.setNombre("Juan");
System.out.println(p.getNombre());


BlueJ: Terminal Window - meto1
Options
Juan