Los contenidos de este curso se organizan en dos partes:

Parte I. Fundamentos de programación orientada a objetos	
Tema 1	Capítulo 1. Objetos y clases
Tema 2	Capítulo 2. Definiciones de clases
Tema 3	Capítulo 3. Interacción de objetos
Tema 4	Capítulo 4. Agrupación de objetos
Tema 5	Capítulo 6. Comportamientos más sofisticados
Tema 6	Capítulo 7. Colecciones de tamaño fijo: matrices
Parte 2. Estructuras de las aplicaciones	
Tema 7	Capítulo 8. Diseño de clases
Tema 8	Capítulo 9. Objetos con un buen comportamiento
Tema 9	Capítulo 10. Mejora de la estructura mediante la herencia
Tema 10	Capítulo 11. Más sobre la herencia
Tema 11	Capítulo 12. Técnicas de abstracción adicionales
Tema 12	Capítulo 14. Tratamiento de errores

6 EDICION	**5 EDICION**
1 y APENDICE B	1 y APENDICE B
2 y APENDICES B,C,D	2 y APENDICES B,C,D
3 y APENDICES B,F	3 y APENDICES B,F
4 ; 7.1 a 7.4	4
6.	5
8. y APENDICE E	6. y APENDICE E
9	7
10,11	8, 9
12	10

Programación orientada a objetos

Capítulo 1

Objetos y clases

Tutor: Manuel Fernández Barcell
Centro Asociado de Cádiz
http://prof.mfbarcell.es

Tema 1: Objetos y clases. Semana 1

1- Los conceptos de objeto y clase 2- Creación de objetos 3- Invocación de objetos 4- Parámetros 5- Tipos de datos 6- Instanciación de objetos 7- Estado de un objeto 8- ¿Qué representa un objeto? 9- Interacción entre objetos 10- Codigo fuente 11- Valores de retorno 12- Objetos como parametros	1- Estudiar el capítulo 1 y el Apéndice B del libro base de la Unidad Didáctica I 2- Instalar el software BlueJ y realizar los ejercicios sugeridos en el libro base 3- Definir las clases necesarias en el problema de la práctica

Tema 1. Se presentan los conceptos más fundamentales de la orientación a objetos (objetos, clases y métodos). Ofrece una introducción sólida y práctica de estos conceptos sin entrar en los detalles de la sintaxis Java, aunque presenta un primer acercamiento al código.

Tema 1:

1- Asimilar los conceptos generales asociados a la programación en lenguaje Java. Los objetos y las clases.
2- Ser capaz de instanciar objetos de una clase e invocar métodos mediante el interfaz BlueJ.
3- Ser capaz de hacer pequeñas modificaciones en un código Java.

1.1 Objetos y clases

- La clase es la abstracción de una categoría de objeto
  - Una clase sirve para describir de un modo abstracto los objetos de un determinado tipo

Concepto

Los objetos Java modelan objetos que provienen del dominio de un problema.

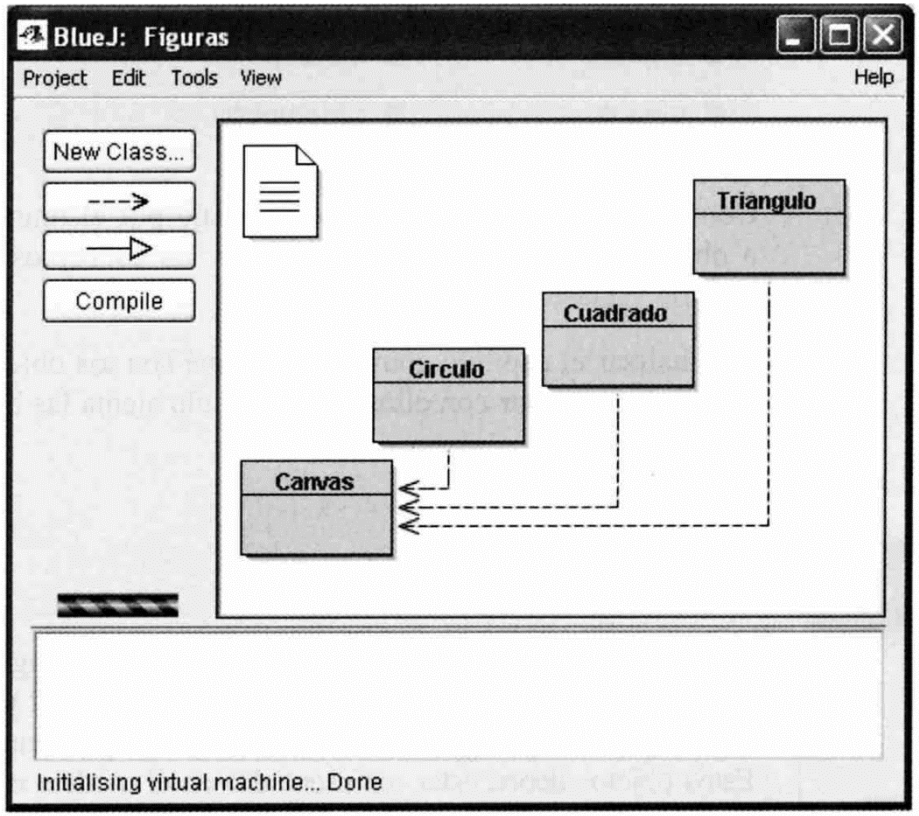
Concepto

Los objetos se crean a partir de clases. La clase describe la categoría del objeto. Los objetos representan casos individuales de una clase.

Generalmente, cuando nos referimos a un objeto en particular hablamos de una *instancia*. De aquí en adelante usaremos regularmente el término «instancia». Instancia es casi un sinónimo de objeto. Nos referimos a objetos como instancias cuando queremos enfatizar que son de una clase en particular (como por ejemplo, cuando decimos «este objeto es una instancia de la clase auto»).

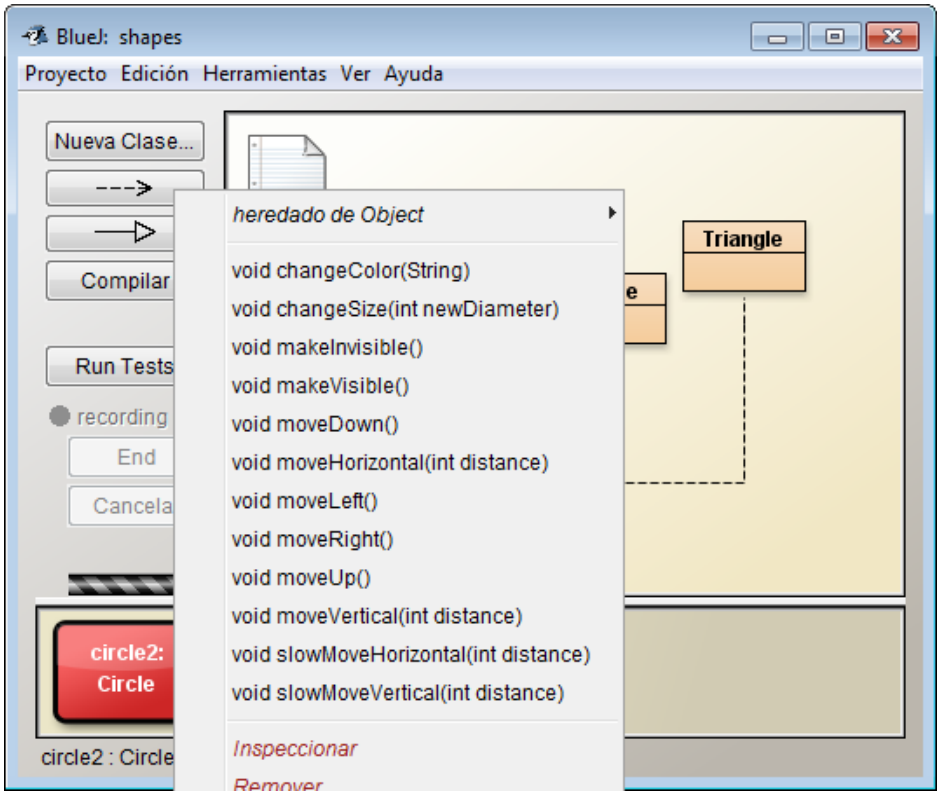
1.2 Crear Objeto

- Ejemplo: *shapes*



Convención Los nombres de las clases comienzan con una letra mayúscula (como `Circulo`) y los nombres de los objetos con letras minúsculas (circulo1). Esto ayuda a distinguir de qué elemento estamos hablando.

1.3 Invocar un método



Podemos comunicarnos con los objetos invocando sus métodos. Generalmente, los objetos hacen algo cuando invocamos un método.

Los elementos del menú contextual del círculo representan las operaciones que se pueden usar para manipular el círculo. En Java, estas operaciones se denominan *métodos*. Usando la terminología común, decimos que estos métodos son *llamados* o *invocados*. De aquí en adelante usaremos esta terminología que es más adecuada. Por ejemplo, podríamos pedirle que «invoque el método `moverDerecha` de `circulo1`».

1.4 Parámetros

Concepto

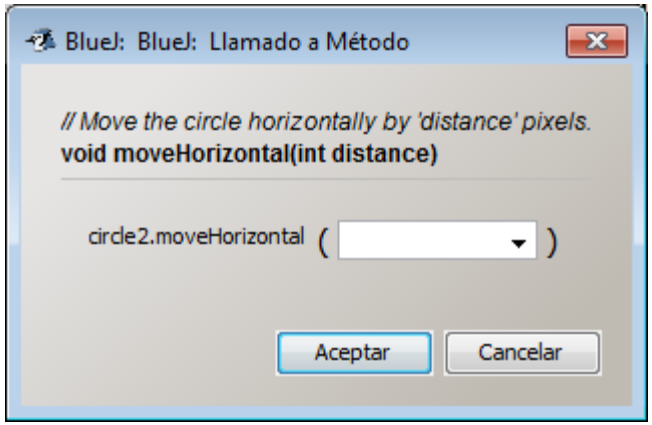
El encabezado de un método se denomina su *signatura* y proporciona la información necesaria para invocarlo.

Concepto

Los métodos pueden tener *parámetros* para proporcionar información adicional para realizar una tarea.

```
void moverHorizontal(int distancia)
```

signatura



signatura

```
void cambiarTamanio (int nuevoAlto, int nuevoAncho)
```

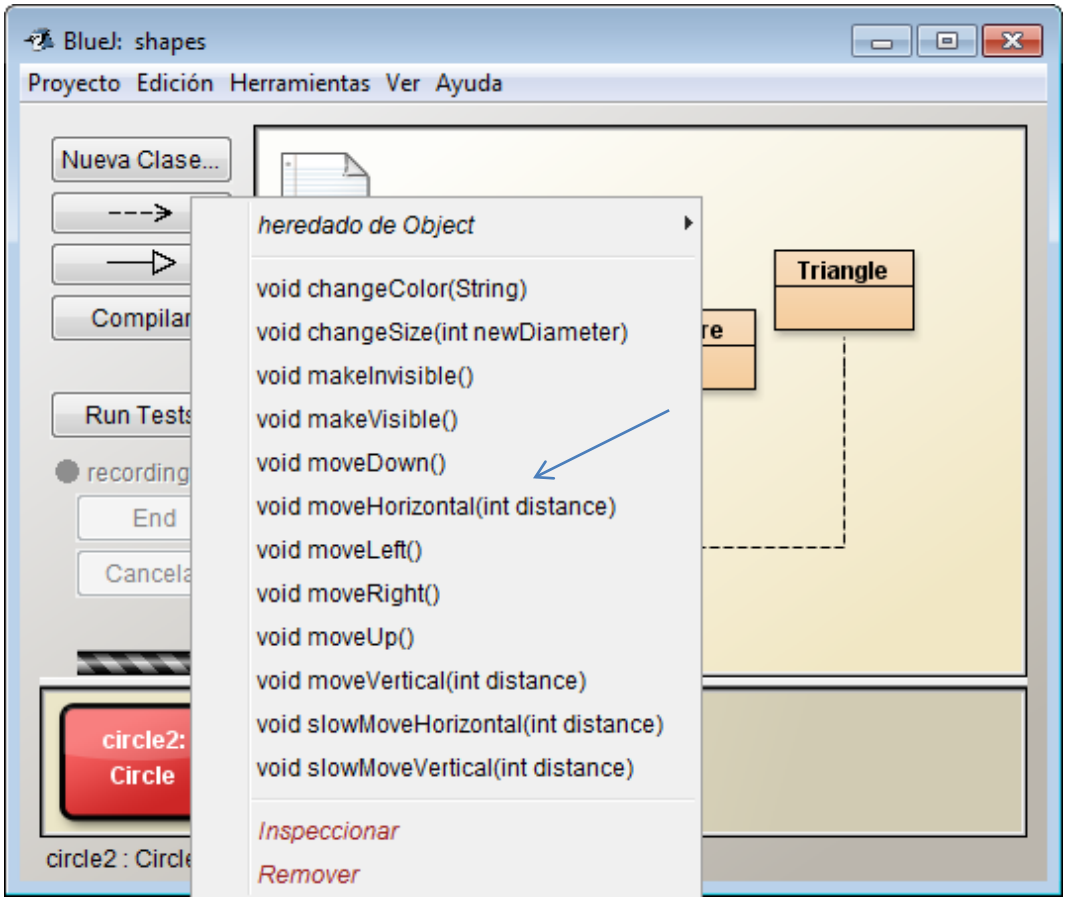
Este es un ejemplo de un método que tiene más de un parámetro. Este método tiene dos parámetros que están separados por una coma en la signatura. De hecho, los métodos pueden tener cualquier número de parámetros.

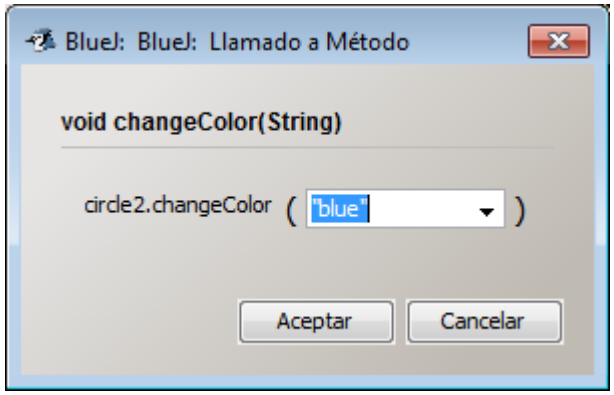
1.5 Tipos de datos

- String “entre comillas dobles”

Concepto

Los parámetros tienen tipos de dato. El tipo de dato define la clase de valores que un parámetro puede tomar.





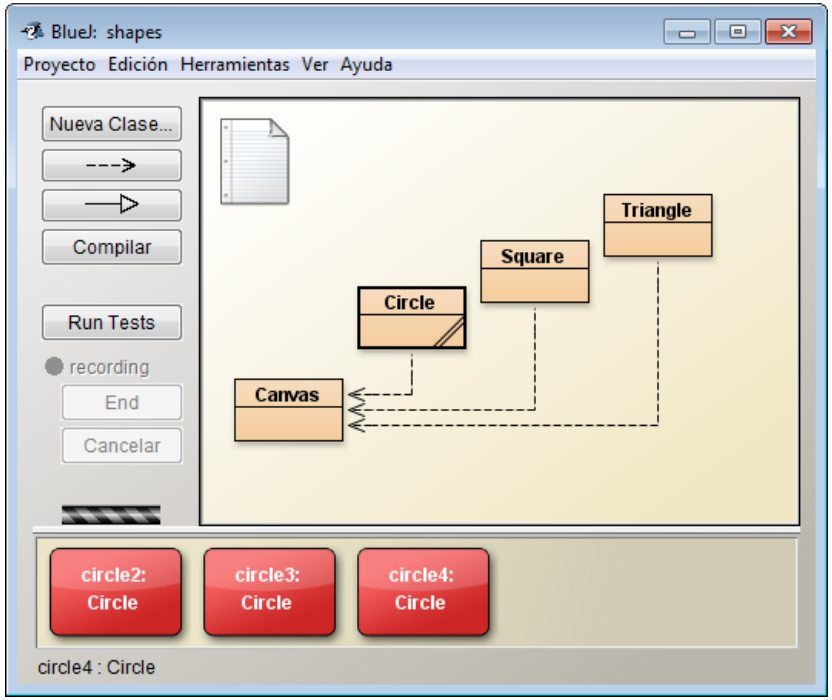
Cuidado Un error muy común entre los principiantes es olvidar las comillas dobles cuando escriben un valor de tipo `String`. Si escribe *verde* en lugar de «*verde*» aparecerá un mensaje de error diciendo «Error: cannot resolve symbol». («*no puede resolver el símbolo*»).

1.6 Instancias múltiples

- Cada instancia puede tener “valores” distintos en sus atributos

Concepto

Instancias múltiples.
Se pueden crear muchos objetos similares a partir de una sola clase.

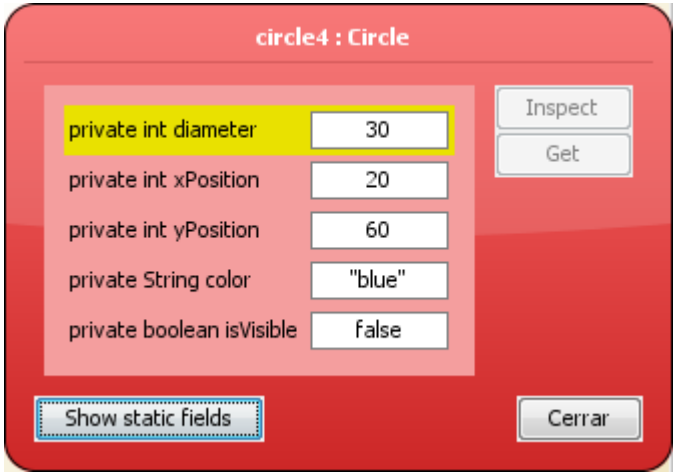


1.7 Estado

- Java se refiere a los atributos de los objetos como *campos* (fields)

Concepto

Los objetos tienen un estado. El estado está representado por los valores almacenados en sus campos.



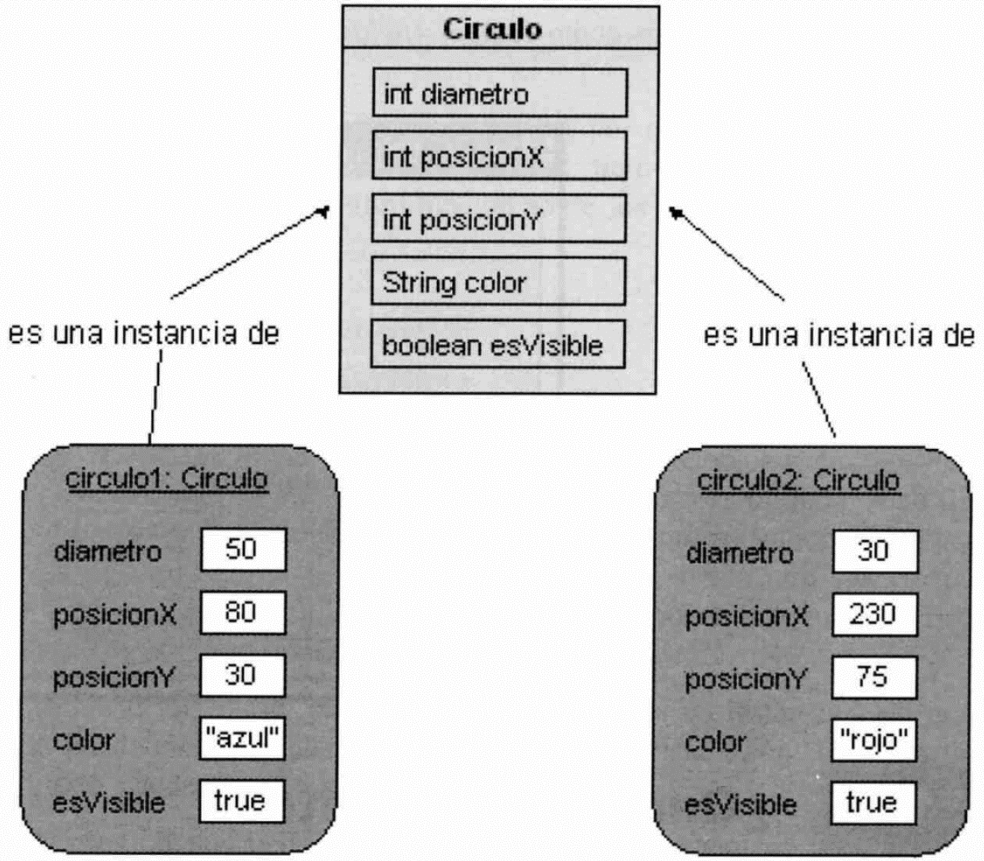
Se hace referencia al conjunto de valores de todos los atributos que definen un objeto (tales como las posiciones *x* e *y*, el color, el diámetro y el estado de visibilidad para un círculo) como el *estado* del objeto. Este es otro ejemplo de terminología común que usaremos de aquí en adelante.

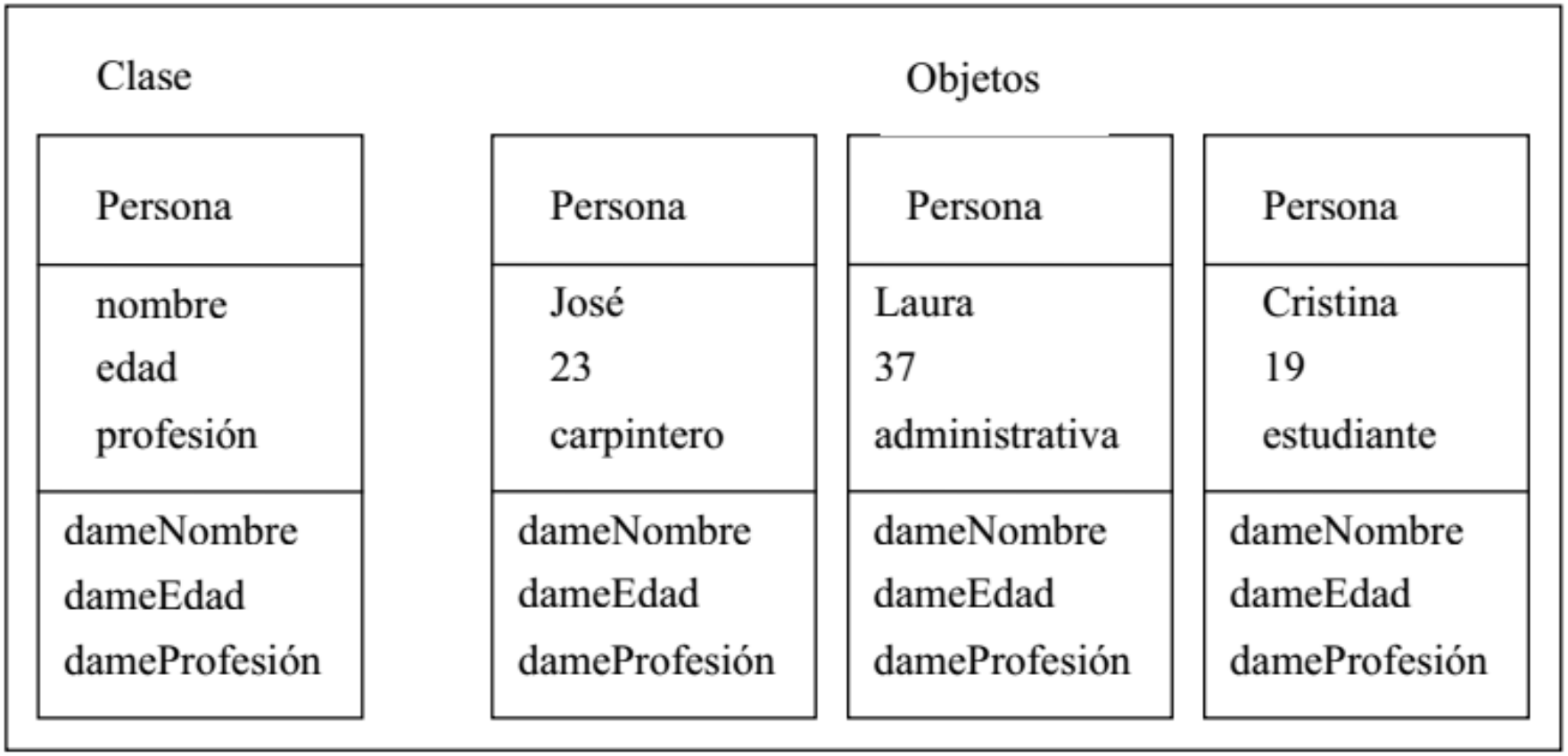
En BlueJ, el estado de un objeto se puede inspeccionar seleccionando la función *Inspect* del menú contextual del objeto. Cuando se inspecciona un objeto, se despliega una ventana similar a la que se muestra en la Figura 1.5 denominada *Inspector del Objeto (Object Inspector)*.

1.8 ¿Qué es un Objeto?

Al inspeccionar objetos diferentes observará que todos los objetos de la misma clase tienen los mismos campos; es decir que el número, el tipo de dato y los nombres de los campos de una misma clase son los mismos, mientras que el valor de un campo en particular de cada objeto puede ser distinto. Por el contrario, los objetos de clases diferentes pueden tener diferentes campos. Por ejemplo, un círculo tiene un campo «`diametro`», mientras que un triángulo tiene los campos «`ancho`» y «`alto`».

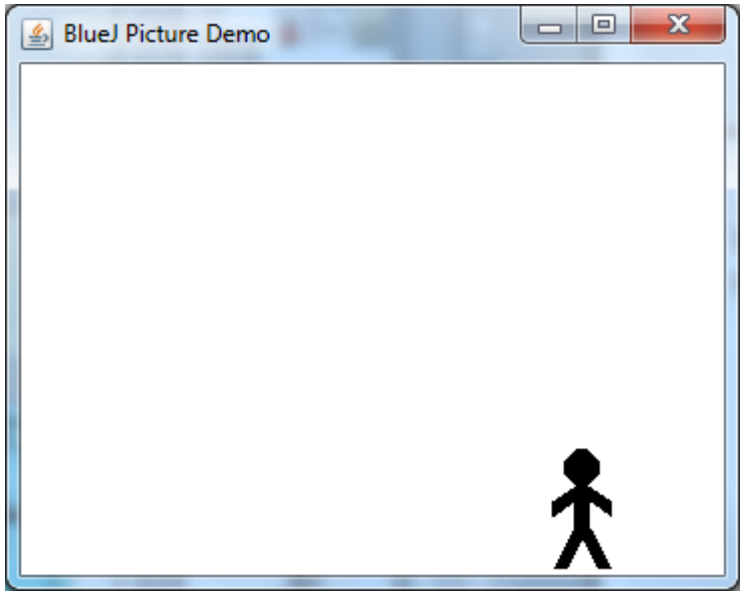
Figura 1.6
Una clase y sus objetos con campos y valores

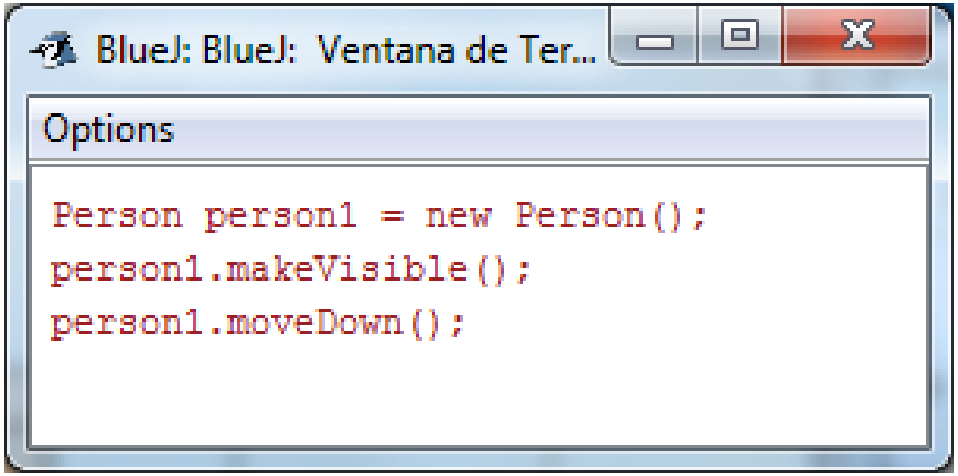



Clase
Objetos
Persona
nombre
edad
profesión
dameNombre
dameEdad
dameProfesión
Persona
José
23
carpintero
dameNombre
dameEdad
dameProfesión
Persona
Laura
37
administrativa
dameNombre
dameEdad
dameProfesión
Persona
Cristina
19
estudiante
dameNombre
dameEdad
dameProfesión

1.9 Código Java

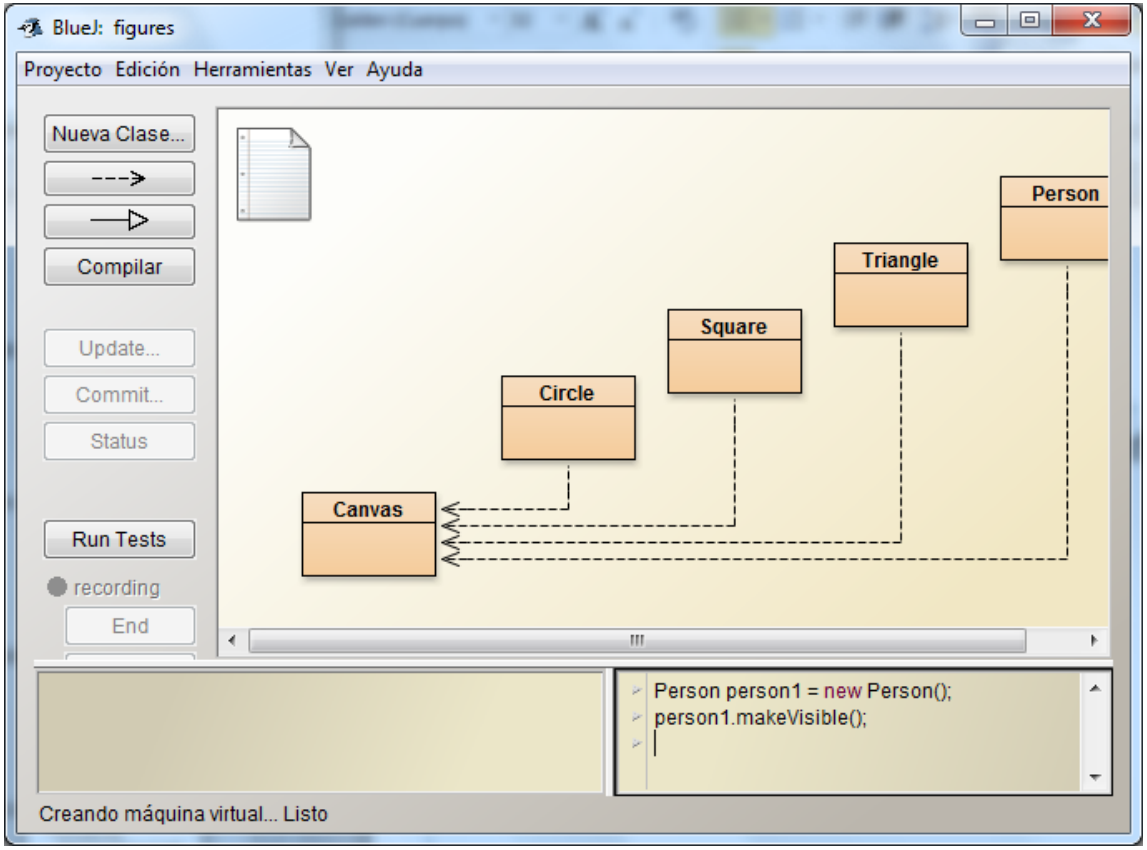
- VENTANA DE TERMINAL
  - USANDO EL EJEMPLO *figures*
    - Menú ver→ mostrar terminal
    - En opciones de la ventana del terminal: registrar llamadas a métodos





Bloque de código

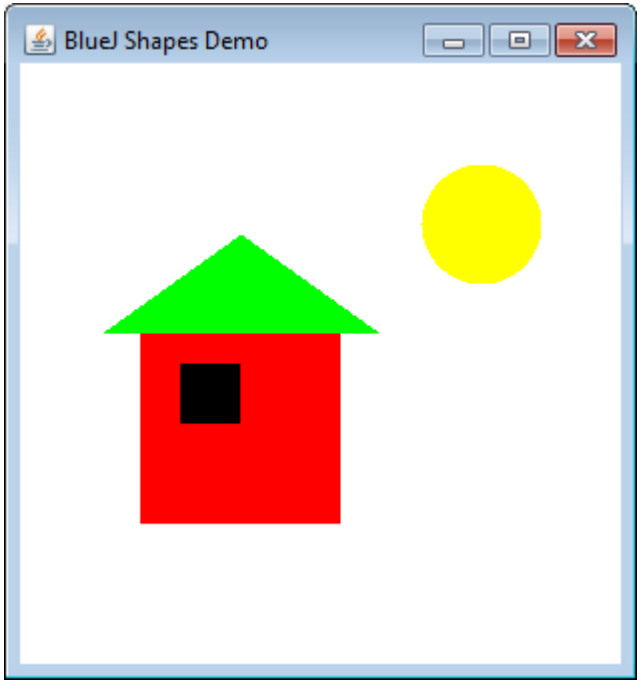
- Para escribir código directamente
- Menú ver → Show Code Pac

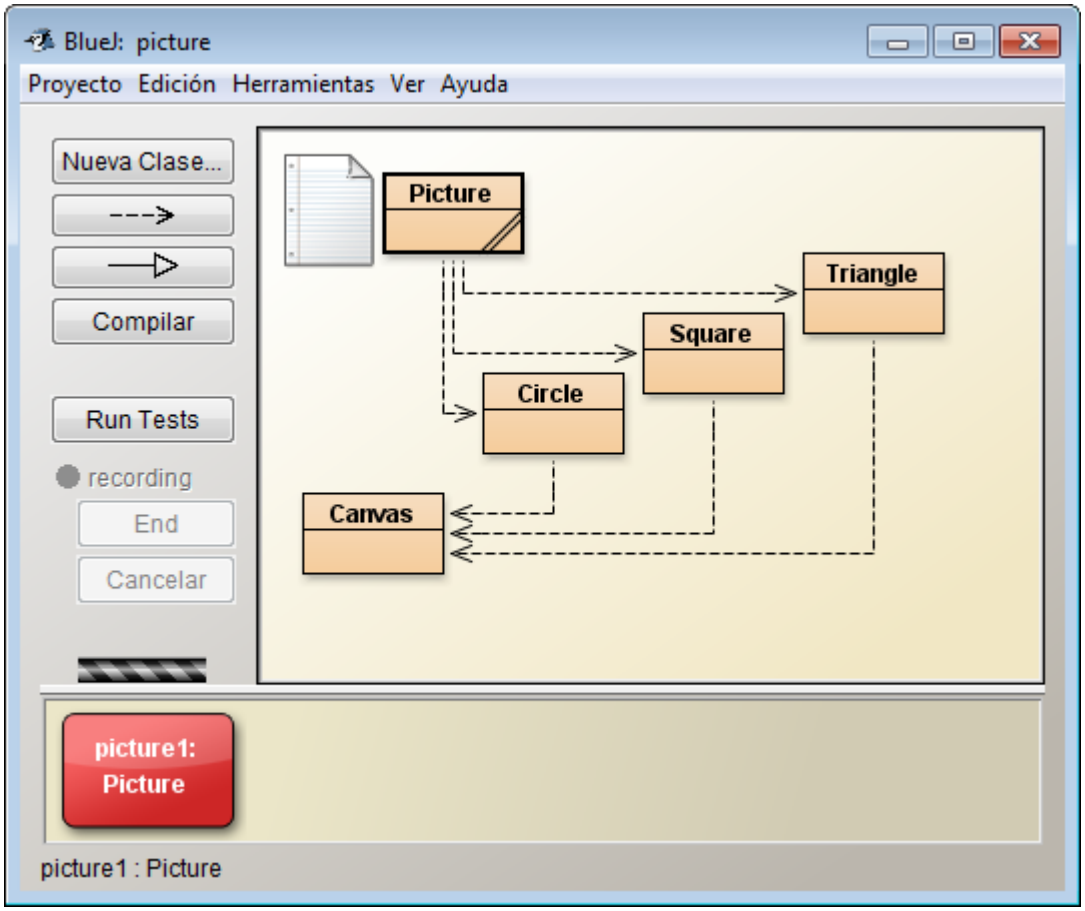


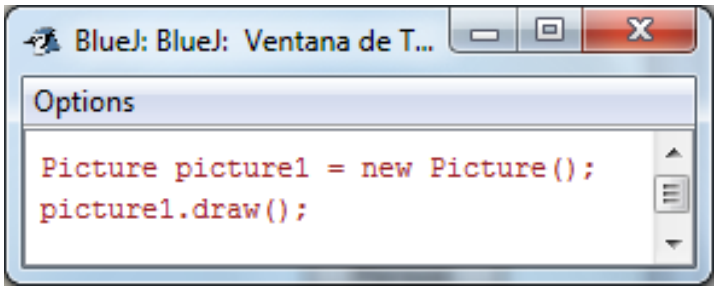
1.10 Interacción entre objetos

Concepto

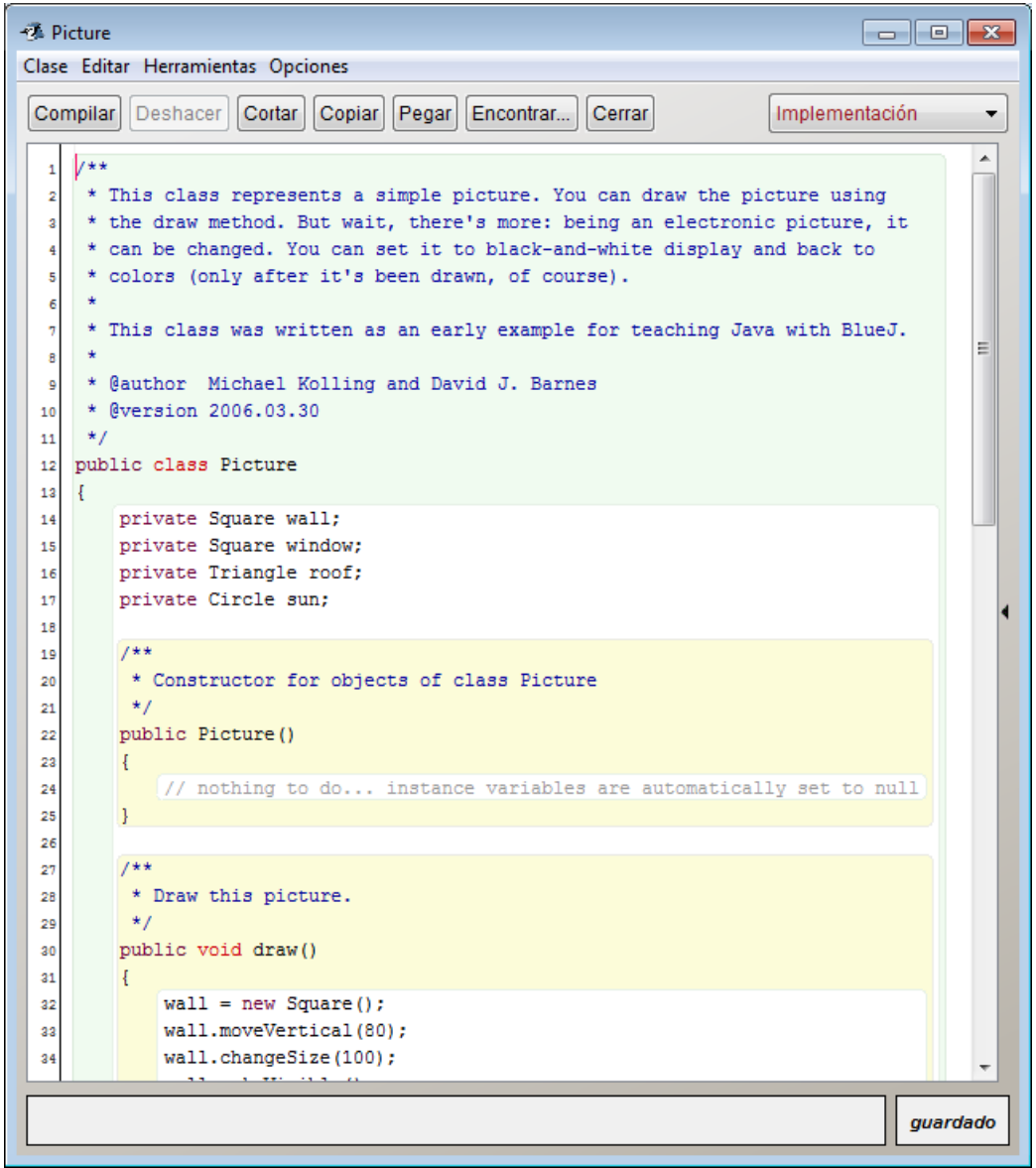
Llamada de métodos. Los objetos se pueden comunicar entre ellos invocando los métodos de los otros objetos.







1.11 Código Fuente



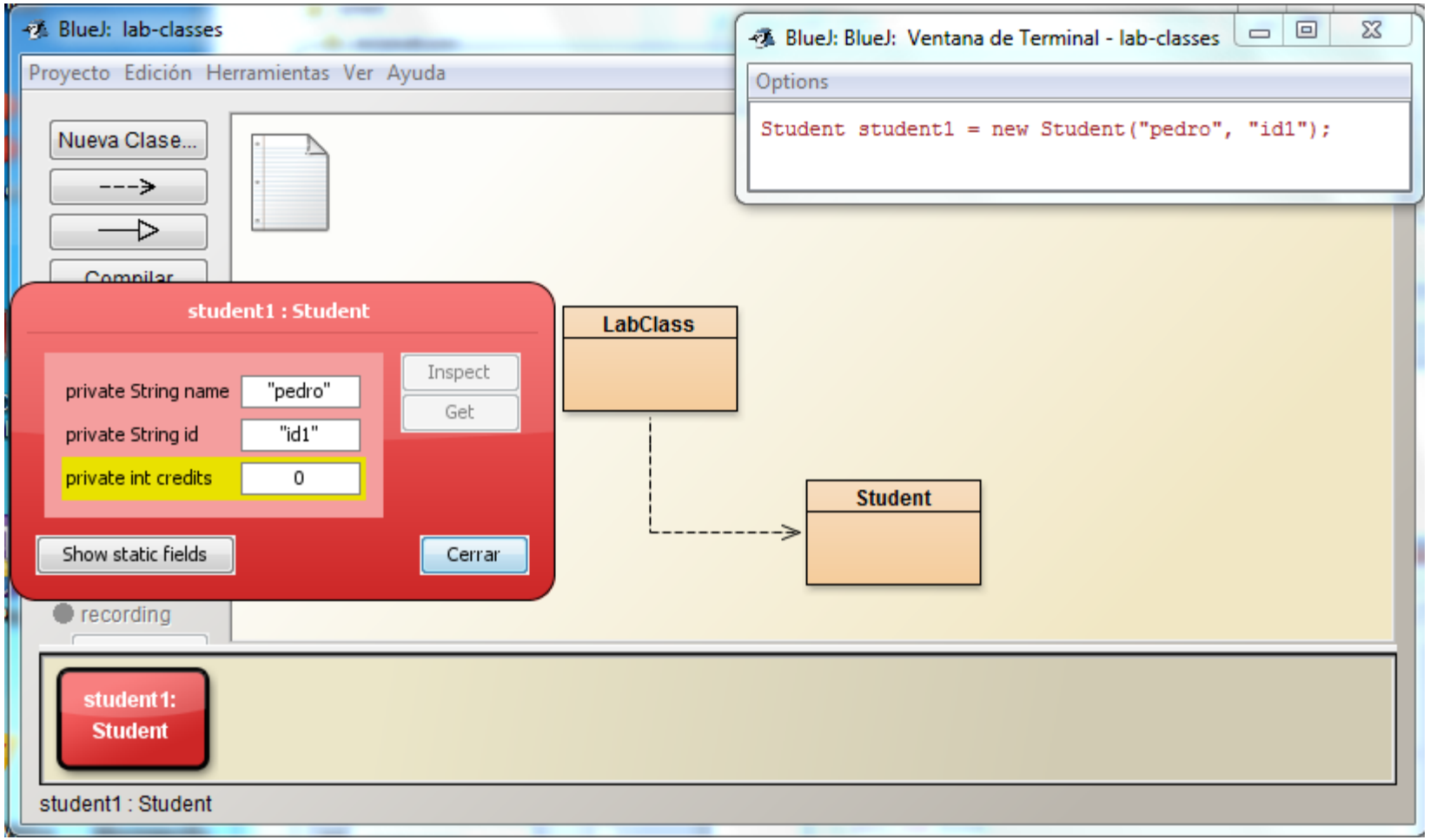

```
/**
 * This class represents a simple picture. You can draw the picture using
 * the draw method. But wait, there's more: being an electronic picture, it
 * can be changed. You can set it to black-and-white display and back to
 * colors (only after it's been drawn, of course).
 *
 * This class was written as an early example for teaching Java with BlueJ.
 *
 * @author  Michael Kolling and David J. Barnes
 * @version 2006.03.30
 */
public class Picture
{
    private Square wall;
    private Square window;
    private Triangle roof;
    private Circle sun;

    /**
     * Constructor for objects of class Picture
     */
    public Picture()
    {
        // nothing to do... instance variables are automatically set to null
    }

    /**
     * Draw this picture.
     */
    public void draw()
    {
        wall = new Square();
        wall.moveVertical(80);
        wall.changeSize(100);
```

Concepto

El **código fuente** de una clase determina la estructura y el comportamiento (los campos y los métodos) de cada uno de los objetos de dicha clase.

1.12 Otro Ejemplo (lab-classes)

- Crear un objeto de la clase *student*
- Los parámetros *string* hay que introducirlo entre comillas



1.13 Valores de retorno

Concepto

Resultados. Los métodos pueden devolver información de algún objeto mediante un **valor de retorno**.

La signatura de `obtenerNombre` (tal como muestra el menú contextual del objeto) está definida de la siguiente manera:

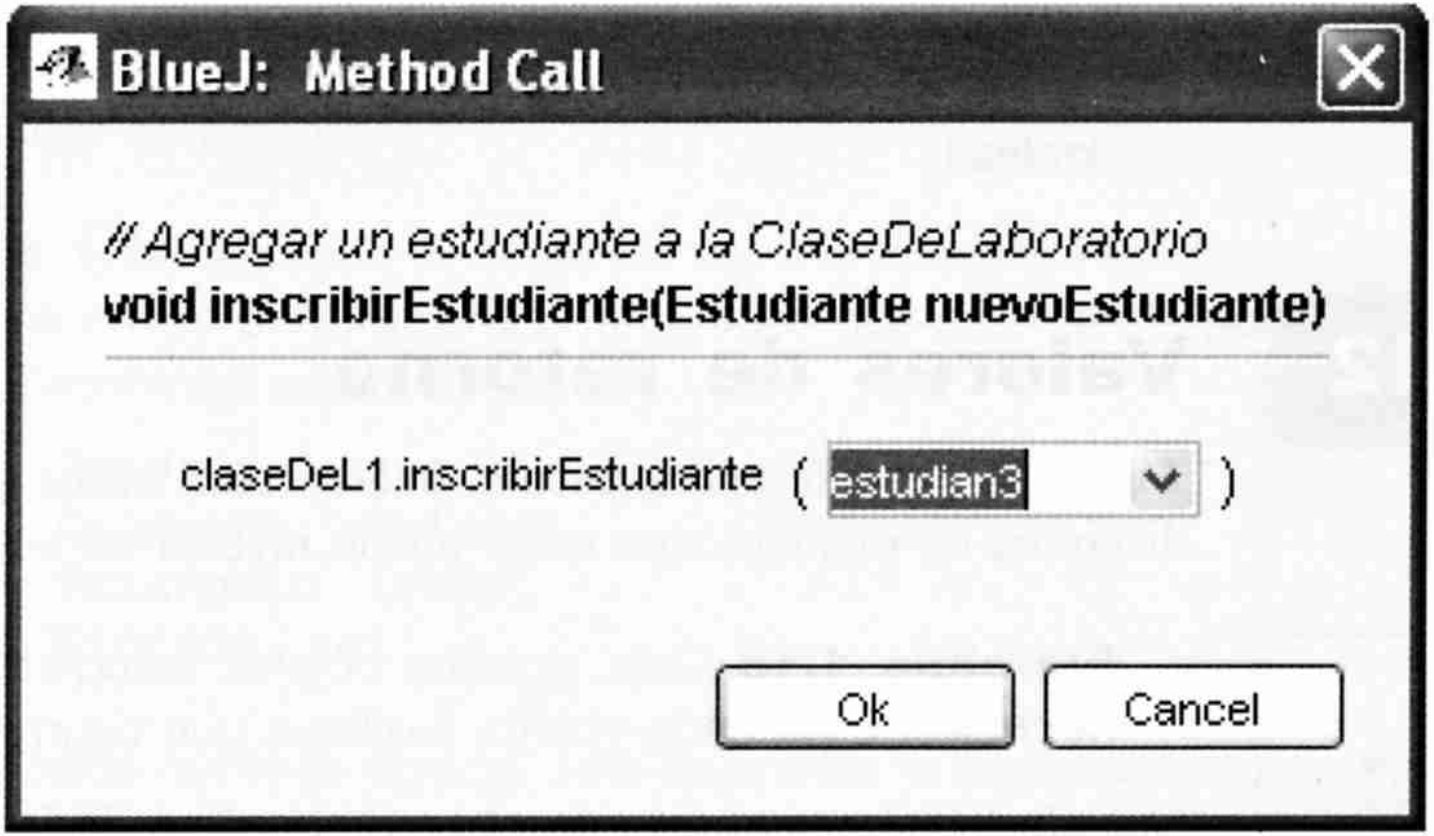
```
String obtenerNombre()
```

La palabra `String` que aparece antes del nombre del método especifica el tipo de retorno. En este caso, establece que este método devolverá un resultado de tipo `String` cuando sea invocado. La signatura de `cambiarNombre` es:

```
void cambiarNombre(String nuevoNombre)
```

La palabra `void` indica que este método no retorna ningún resultado.

1.14 Objetos como parámetros



Resumen de conceptos

- **objeto** Los objetos Java modelan los objetos del dominio de un problema.
- **clase** Los objetos se crean a partir de las clases. La clase describe la categoría del objeto; los objetos representan instancias individuales de la clase.
- **método** Podemos comunicarnos con los objetos invocando sus métodos. Generalmente, los objetos hacen algo cuando invocamos un método.
- **parámetro** Los métodos pueden tener parámetros para aportar información adicional para realizar una tarea.
- **signatura** El encabezado de un método se denomina su signatura. Proporciona la información necesaria para invocar dicho método.
- **tipo** Los parámetros tienen tipos. El tipo define la clase de valor que un parámetro puede tomar.
- **instancias múltiples** Se pueden crear muchos objetos similares a partir de una sola clase.
- **estado** Los objetos tienen un estado. El estado está representado por los valores almacenados en los campos.
- **llamar métodos** Los objetos se pueden comunicar invocando los métodos de cada uno de los otros objetos.
- **código fuente** El código de una clase determina la estructura y el comportamiento (los campos y los métodos) de cada uno de los objetos de dicha clase.
- **resultado** Los métodos pueden devolver información de un objeto mediante valores de retorno.

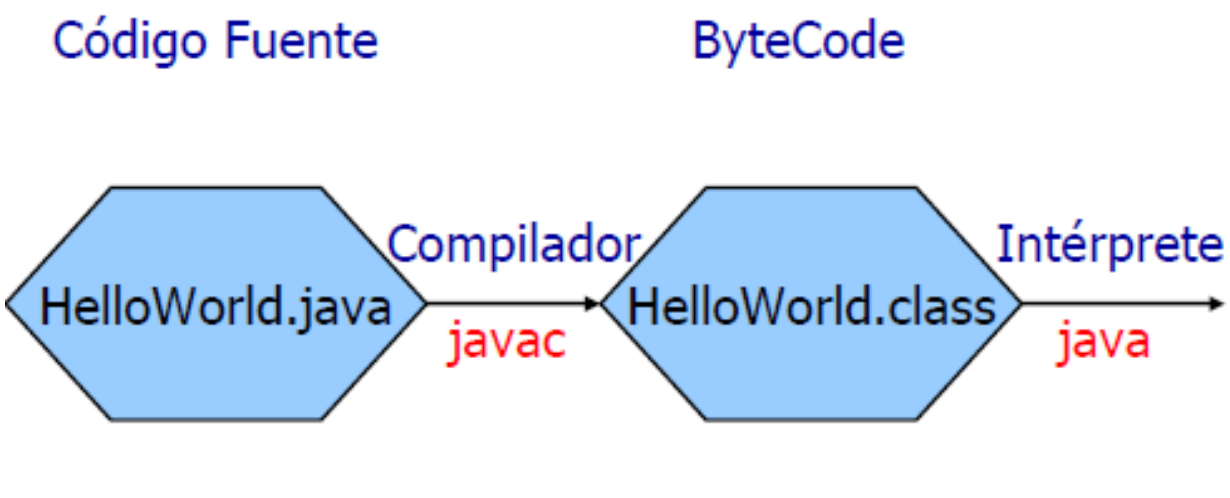
http://www.fdi.ucm.es/profesor/jpavon/poo/

Java

- Compilador: javac
- Interprete: java
- Plataforma de ejecución: JRE (Java Runtime Environment):
  - Incluye JVM
- Plataforma de desarrollo: Java SDK (Java Software Development Kit):
  - Incluye Compilador, etc.
  - Incluye JRE

```java
/**
 * Nombre:      HelloWorld.java
 * Descripción: Esta es mi primera clase escrita en Java
 * Autor:       Manuel Pereira
 */

public class HelloWorld {

    /**
     * @param args Argumentos recibidos por línea de parámetros
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

}
```

```java
/**
* Programa en Java que escribe texto en pantalla
*/
public class Hola {
  public static void main (String[] args){
    System.out.println ("Hola a todos");
    }
}
```


Código Fuente
ByteCode
Ejecución
HelloWorld.java
Compilador
javac
HelloWorld.class
Intérprete
java
Command Prompt
C:\Practicas>javac HelloWorld.java
C:\Practicas>java HelloWorld
Hello World!
C:\Practicas>