

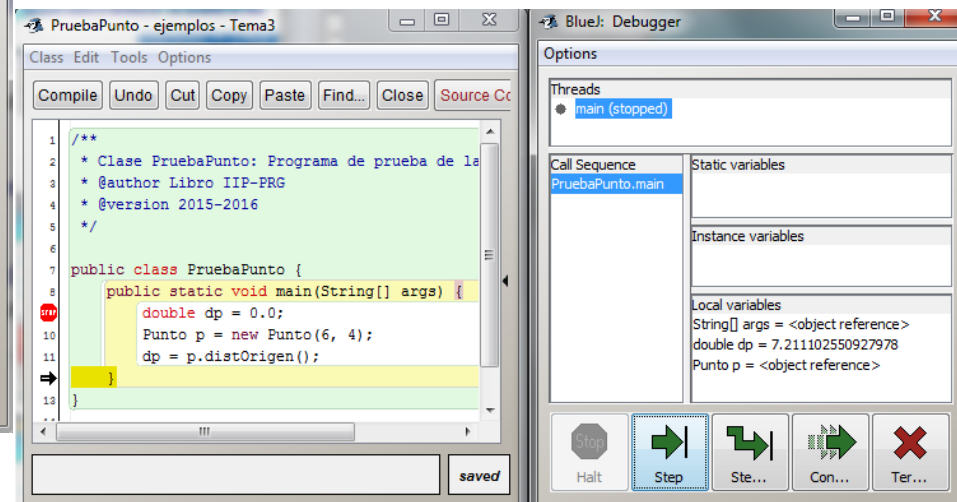
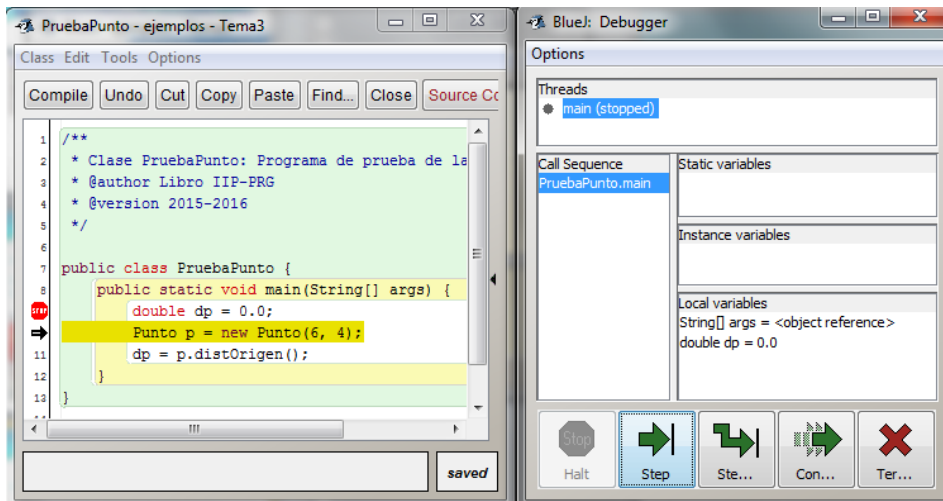
Tema 3. Variables: definición, tipos y uso en Java

Punto 2 - Parte 1: Uso de una variable Java

- Principios básicos: estado de una variable y su modificación. Traza de ejecución
- Asignación
 - Inicialización de variables, según su tipo y ámbito. Literales. Expresiones. Compatibilidad de tipos. Valores por defecto
 - Copia e intercambio
 - Objetos desreferenciados y Garbage Collector
- Otras operaciones sobre variables, según su tipo
 - Igualdad
 - Comparación
- Detalles, ejemplos y ejercicios con operadores de tipo primitivo: desbordamiento, compatibilidad (automática y forzosa), división (entera y real), precedencia (asociatividad por la izquierda y uso paréntesis) , operadores aritméticos no simples, operaciones con `char`, operadores relacionales y lógicos cortocircuitados

Uso de una variable: principios básicos

- Las variables **pueden cambiar de valor durante la ejecución de un programa**
¿Cómo?
- El **estado de una variable** es el valor que tiene en un determinado momento
 - La **ejecución de un programa** se puede ver como una sucesión de cambios de estado que transforman un cierto estado inicial (datos) en un determinado estado final (resultados)
 - El **estado de un programa** es el contenido de sus variables en un momento de la ejecución
 - La **Traza de la ejecución de un programa** es el seguimiento de la evolución de los valores de las variables durante su ejecución



Uso de una variable en Java: Asignación

- La **asignación** (operador **=**) se usa para dar valores a las variables o para reemplazar los valores que ya tienen por otros nuevos

```
identificador = expresión;
```

donde identificador y expresión deben ser de tipos iguales o **compatibles**

- Una expresión es una sucesión (sintácticamente correcta) de valores, variables, operadores y llamadas a métodos que se evalúa a un único valor, siendo el tipo de la expresión el tipo de este valor
- Al **ejecutar una asignación** se evalúa **primero** la expresión y, **después**, se almacena el valor resultante en la variable

```
int cantidadInicial;  
↓  
cantidadInicial = 50 * 3;
```

Uso de una variable en Java: Asignación

¿Cuándo se usa? Inicialización

Para usar una variable NO basta con declararla sino que es **IMPRESINDIBLE INICIALIZARLA**, i.e. **asignarle un valor inicial (operador =)**, típicamente en la línea en que se declara



BlueJ: ejemplos - Tema3

Escribe las siguientes **instrucciones** de declaración-**inicialización** de variables en el **CodePad** de BlueJ... **¿En qué se parecen? ¿En qué no? ¿Por qué?**

The screenshot shows the BlueJ IDE interface. On the left, there are buttons for 'Nueva Clase...', a dashed arrow, a solid arrow, and 'Compilar'. Below these is a 'Compilando... Listo' status bar. The main workspace is divided into two panes. The top pane shows a class diagram with a class named 'Circulo'. The bottom pane shows the source code of the 'Circulo' class. The code is as follows:

```
double radioC1 = 50.0;
int centroXC1 = 100;
int centroYC1 = 100;
Circulo c1 = new Circulo(radioC1, "amarillo", centroXC1, centroYC1);
c1
```

Blue annotations highlight the initialization process. A dashed blue arrow points from the text 'Al inicializar una variable SIEMPRE se le asigna un valor' to the equals sign in the first line of code. Another dashed blue arrow points from the same text to the equals sign in the fourth line of code. The variable 'radioC1' is circled in blue, and the variable 'c1' is also circled in blue. Below the code, the IDE shows the object reference for 'c1' as '<object reference> (Circulo) @5e21151e'.

Uso de una variable en Java: Asignación

Inicialización *según su tipo* (I): valores para Referencias



BlueJ: ejemplos - Tema3

Escribe las siguientes **instrucciones** de declaración-**inicialización** de variables en el **CodePad** de BlueJ... ¿En qué se parecen? ¿En qué no? ¿Por qué?

The screenshot shows the BlueJ IDE window titled "BlueJ: ejemplos - Tema3". On the left, there are buttons for "Nueva Clase...", a dashed arrow, a solid arrow, and "Compilar". Below these is a progress bar. The main area is divided into two panes. The top pane shows a class diagram with a class named "Circulo" (represented by an orange rectangle) and a document icon. To the right of the diagram, text reads: "Al **inicializar** una variable **SIEMPRE** se le asigna un valor ... PERO este valor debe ser el **“adecuado”** para su tipo". The bottom pane shows a code editor with the following Java code:

```
double radioC1 = 50.0;  
int centroXC1 = 100;  
int centroYC1 = 100;  
Circulo c1 = new Circulo(radioC1, "amarillo", centroXC1, centroYC1);
```

The word "new" in the last line is circled in blue. At the bottom of the window, a status bar says "Compilando... Listo".

Para **inicializar** una variable de **tipo Referencia** hay que asignarle el valor **new NombreClase(...)**

Uso de una variable en Java: Asignación

Inicialización según su tipo (II): valores para variables de tipo Primitivo

```
// Declaración e inicialización
// Inicialización a un valor del tipo o compatible, por asignación
char c = 'A'; // el valor asignado a c es un literal de tipo char
double d1 = 2; // el valor asignado a d1 es un literal int,
               // de tipo COMPATIBLE con double
double d2 = 3.0 + d1; // el valor asignado a d2 es el resultado de
                     // evaluar la expresión 3.0 + d1, i.e. 5.0
```



Bluej: ejemplos - Tema3

Copia las instrucciones anteriores –no los comentarios– en el **CodePad** de Bluej y luego “**traduce**” a Java los siguientes enunciados:

- **Mostrar** el **valor** de las variables c, d1 y d2 (en el *CodePad*)
- En la misma línea, **declarar** de tipo `int` e **inicializar** la variable d1Truncada a **2**
Mostrar su **valor** ¿En qué se diferencia del mostrado para la variable d1? ¿Por qué?
- **Evaluar** (la expresión) **3.5 + d1**, para mostrar su valor y tipo
- En la misma línea, **declarar** de tipo `int` e **inicializar** la variable d2Truncada al valor **3.5 + d1**. **Mostrar** su **valor** ¿En qué se diferencia del mostrado para la variable d1? ¿Por qué?

Uso de una variable en Java: Asignación Literales

Cara a **evaluar expresiones donde aparecen literales** es importante tener en cuenta que...

- Por defecto, los literales enteros son de tipo **int**; para **forzar** que un entero se interprete como un **long** se añade tras el valor la letra **L** o la **l**
- Por defecto, los literales reales son de tipo **double**; para **forzar** que un entero se interprete como un **float** se añade tras el valor la letra **F** o la **f**
- Un literal de tipo **char** se escribe entre **comillas simples** -se representa internamente como un valor entero positivo, pero sin la representación en complemento a dos
- De tipo **boolean** sólo hay dos literales: **true** y **false**
- Un literal de tipo **String** se escribe entre **dobles comillas**

Uso de una variable en Java

Inicialización *según su rol (ámbito)*

Según el **bloque** Java **donde se declara**, una variable tiene un **rol** u otro (atributo, variable local o parámetro) y **su inicialización se realiza en un lugar u otro, obligatoriamente o no**

- **Variables locales y parámetros:** inicialización obligatoria, en los métodos donde se usan
- **Atributos:** inicialización NO obligatoria, en los métodos constructores

Uso de una variable en Java

Inicialización según su rol (ámbito): variables *locales* (I)

Inicialización obligatoria, en el método donde se usan



BlueJ: ejemplos - Tema3

En la clase **PruebaCirculo** del proyecto...

- **Comprueba** que no hay errores de compilación
- **Elimina** la inicialización de la variable local **radio** de su método **main** y **compila... ¿Algún error de compilación? ¿Por qué?**

The screenshot shows the BlueJ IDE with the file "PruebaCirculo - ejemplos - Tema3" open. The code editor displays the following Java code:

```
7 public class PruebaCirculo {
8     public static void main(String[] args) {
9         // crear un circulo de radio
10        // y con centro en (100, 100)
11        double radio; //= 30.0;
12        int centroX = 50;
13        int centroY = 50;
14        String color = new String("amarillo");
15        Circulo c1 = new Circulo(radio, color, centroX, centroY);
16    }
```

A compilation error message is displayed in a dialog box and at the bottom of the IDE:

variable radio might not have been initialized

You are using a local variable that is not guaranteed to be initialised before it is used here. If in doubt, initialise it at its declaration.

The error message is highlighted in the code editor on line 15, where the variable `radio` is used in the `Circulo` constructor call. The variable `radio` is also highlighted in the code editor on line 11.

The IDE interface includes buttons for "Compilar", "Deshacer", "Cortar", "Copiar", "Pegar", "Encontrar...", "Cerrar", and "Implementación". The status bar at the bottom shows "variable radio might not have been initialized" and a "guardado" button.

Uso de una variable en Java

Inicialización según su rol (ámbito): variables *locales* (II)

Inicialización obligatoria, en el método donde se usan



BlueJ: ejemplos - Tema3

En la clase **PruebaCirculo** del proyecto ...

- **Comprueba** que no hay errores de compilación
- **Elimina** la inicialización de la variable local **color** de su método **main** y **compila**... ¿Error de compilación? ¿En qué se diferencia del anterior?

The screenshot shows the BlueJ IDE interface. The main window is titled "PruebaCirculo - ejemplos - Tema3". It contains a code editor with the following Java code:

```
7 public class PruebaCirculo {
8     public static void main(String[] args) {
9         // crear un círculo de radio
10        // y con centro en (100, 100)
11        double radio = 30.0;
12        int centroX = 50;
13        int centroY = 50;
14        String color; // = new String("rojo");
15        Circulo c1 = new Circulo(radio, color, centroX, centroY);
16    }
```

A "Message" dialog box is open, displaying the error: "variable color might not have been initialized". The message text reads: "You are using a local variable that is not guaranteed to be initialised before it is used here. If in doubt, initialise it at its declaration." The "OK" button is visible.

At the bottom of the IDE window, a status bar shows the error message "variable color might not have been initialized" and a "guardado" (saved) button.

Uso de una variable en Java

Inicialización según su rol (ámbito): variables *locales* (III)

Inicialización obligatoria, en el método donde se usan



BlueJ: ejemplos - Tema3

En la clase **PruebaCirculo** del proyecto ...

- **Comprueba** que no hay errores de compilación
- **Cambia** a **null** la inicialización de la variable local **c2** de su método **main** y **compila**... ¿Error de compilación?
- Si compila sin error, **ejecuta**... ¿Error de ejecución? ¿Por qué?

```
18      c1.setRadio(2 * c1.getRadio());
19
20      // crear un circulo estandar, i.e. de radio 50.0,
21      // color negro y centro en (100, 100)
22      Circulo c2 = null; //new Circulo();
23
24      // mostrar por pantalla c2, el circulo estandar
25      System.out.println(c2.toString());
26  }
27  }
28  }
```

java.lang.NullPointerException:
null

BlueJ Options

BlueJ: BlueJ: Ventana de Terminal - ejemplos -...

```
java.lang.NullPointerException
    at PruebaCirculo.main(PruebaCirculo.java:25)
```

Uso de una variable en Java

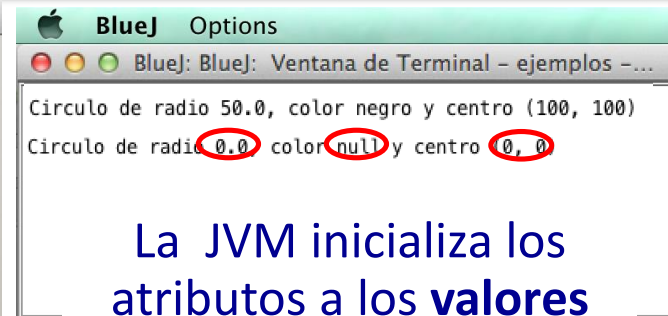
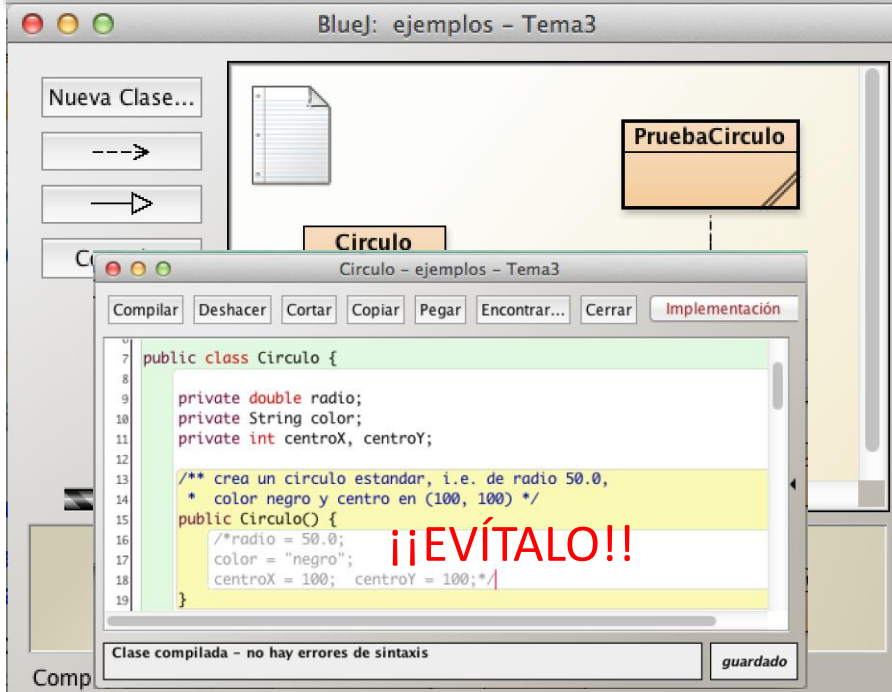
Inicialización según su rol (ámbito): *atributos (I)*

Inicialización **NO** obligatoria, en un método constructor



BlueJ: ejemplos - Tema3

- **Ejecuta** PruebaCirculo... ¿Cuál es el resultado? ¿Por qué?
- **Comprueba** la inicialización de los atributos de la clase **Circulo** en cada uno de sus métodos constructores. **Fíjate en su constructor por defecto**
- **Comenta** el código del constructor por defecto de la clase **Circulo** y **compila**
- **Ejecuta** PruebaCirculo... ¿Cuál es el resultado ahora? ¿Por qué?



La JVM inicializa los atributos a los **valores por defecto** de sus tipos

¡¡SIEMPRE!!

Uso de una variable en Java

Inicialización según su rol (ámbito): *atributos* (II)

Inicialización **NO** obligatoria, en un método constructor
¡OLVÍDALO!



BlueJ: ejemplos - Tema3

- **Comenta TODO** el constructor por defecto de la clase **Círculo**
- **Compila** la clase **Círculo**
- **Compila** la clase **PruebaCírculo** ... ¿Puedes?

```
7 public class Circulo {
8
9     private double radio;
10    private String color;
11    private int centroX, centroY;
12
13    /** crea un círculo estandar, i.e. de radio 50.0,
14     * color negro y centro en (100, 100) */
15    /*public Circulo() {
16        radio = 50.0;
17        color = "negro";
18        centroX = 100; centroY = 100;
19    }*/
```

cambiado

¿Y si la clase solo tiene el constructor por defecto?

¡OLVÍDALO!

```
7 public class Circulo {
8
9     private double radio = 50;
10    private String color = "negro";
11    private int centroX = 100, centroY = 100;
12}
```

Clase compilada - no hay errores de sintaxis guardado

¿Y si inicializo los atributos en la misma línea que los declaro?

¡EVÍTALO!

Uso de una variable en Java

Inicialización según su rol (ámbito): *atributos* (III)

En resumen...

- Las variables **ATRIBUTO** deben ser inicializadas en (el cuerpo de) los **métodos constructores** (tema 5)

¡¡Evita inicializarlas explícitamente al declararlas!!

- La **JVM** inicializa cada variable **ATRIBUTO** al valor **por defecto de su tipo** Java, automáticamente, al ejecutar el método constructor vía **new**
 - Tipos Primitivos:
 - **int** y el resto de tipos Enteros: literal **0**
 - **double** y el otro tipo Real: literal **0.0**
 - **char**: literal **'0'**
 - **boolean**: literal **false**
 - Tipos Referencia: literal **null**, que indica que **NO EXISTE OBJETO** referenciado
- En cualquier caso, la **secuencia de ejecución de la inicialización** de una variable **ATRIBUTO** es: **Por defecto** → **Explícitamente** → **En los constructores**

Inicialización de una variable según su tipo y ámbito:

Resumen

Inicialización de una variable: instrucción de **asignación** de un valor inicial a la variable ...

- **Obligatoria** para las variables **locales** —y **parámetros**—
- El **tipo** del valor asignado debe de ser **COMPATIBLE** con el tipo de la variable
- **¡Recuerda!** La JVM inicializa los atributos a los valores por defecto del tipo
- Sintaxis/Semántica dependen del tipo de la variable



Tipo Primitivo:

- `tipoDeclaracion identificador = valorTipoDeclaracion;`
- Típicamente, `valorTipoDeclaraciones` un **literal** (p.e. un valor por defecto)



Tipo Referencia:

- `TipoDeclaracion identificador = new TipoDeclaracion(...);`
- `identificador` es la **dirección del Heap** donde empieza el objeto creado
- **¡Peligro!** `TipoDeclaracion identificador = null;`