

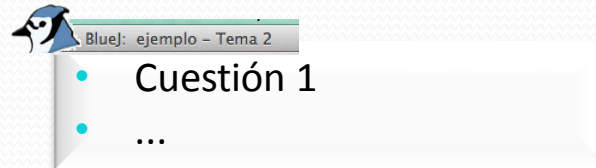
# Tema 2.- Objetos, clases y programas

Duración: 2 sesiones

Índice:

1. **Introducción:** estrategia POO de resolución de un problema
2. **Definición de una clase Java:** tipo y estructura básica
3. **Creación y manipulación de un objeto Java:** operadores *new* y *.* (punto)
4. Documentación de clases Java
5. Errores de compilación y ejecución de clases Java
6. Organización de clases Java en librerías (*packages*). La librería estándar *java.lang* y sus clases *Object*, *String* y *System*.

**NOTA:** en este tema usaremos el **proyecto BlueJ ejemplo – Tema 2**. En muchas de sus transparencias aparecen cuestiones sobre él en un cuadro de texto con el siguiente formato:



**Para que lo tengas operativo en un ordenador es necesario...**

1. **Instalar BlueJ en tu ordenador personal** siguiendo la guía de instalación (Mac, Windows o Linux) que hay en la carpeta de PoliformaT **Recursos / Software para desarrollo en Java**
2. **Descargar el zip del proyecto** de mi carpeta del Tema 2 de la PoliformaT de IIP **Recursos / Profesores / Galiano Ronda, Isabel. Grupo 1FLIP / Material Propio**

**RECUERDA:** descarga el zip en **tu carpeta W:\IIP\Tema 2**

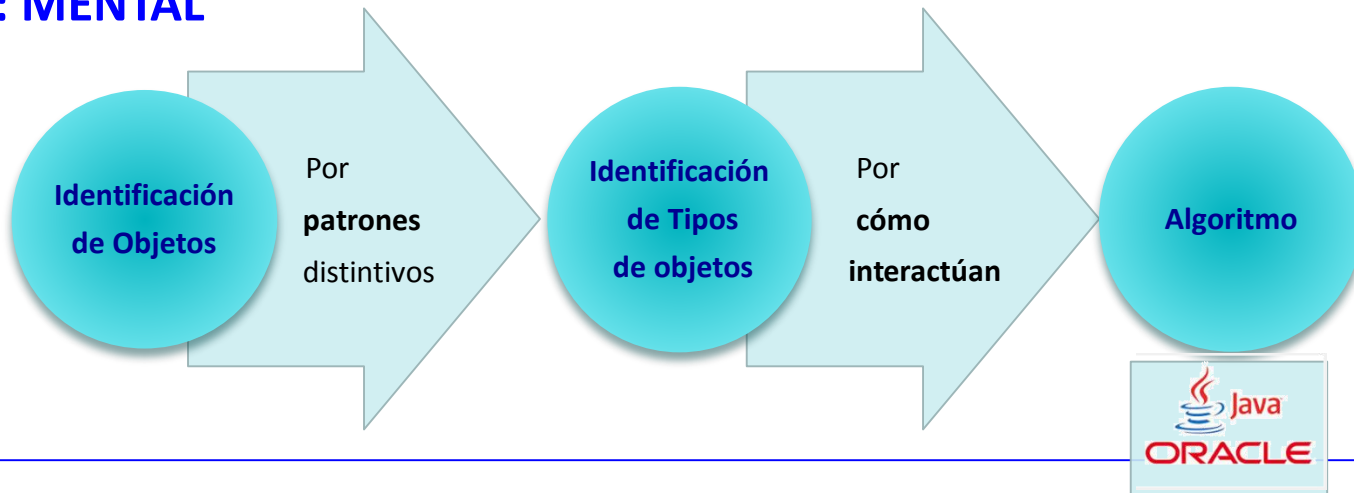
3. **Descomprimir el zip y abrir el proyecto BlueJ**

# Introducción

## ¿Cómo resolver un problema siguiendo una estrategia POO?

- ¿Qué es programar en Java? -

### Fase 1: MENTAL



### Fase 2: MENTAL

- **Escribir en Java el Algoritmo** que describe la resolución del problema, una instrucción por paso: **Programa Java**
- **Escribir en Java la definición de cada Tipo de objetos** que se crean y manipulan en el Algoritmo: **Clases Java de los Datos**
- **Ejecutar el Programa Java**

### Entorno de Desarrollo Integrado (IDE) Java

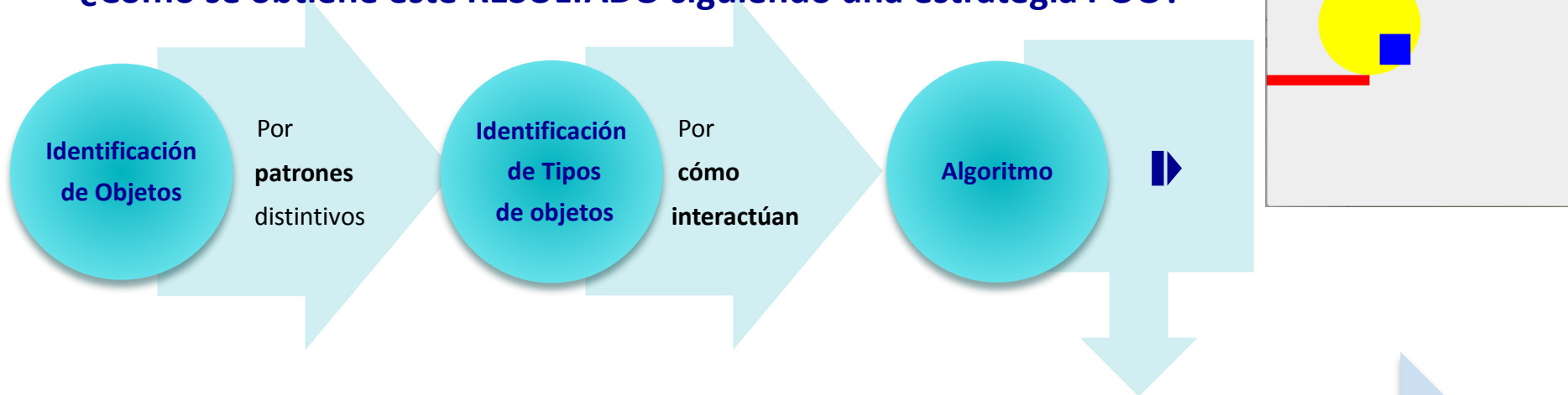
- Editar (**.java**)
- Compilar (**.class**)
- Ejecutar (**bytecode**)

# Introducción

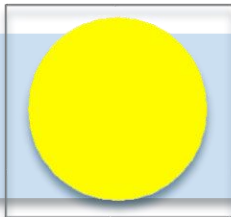
## Actividad Ejemplo - Tema 2: la fase MENTAL

**PROBLEMA:** disponer de un espacio de dibujo a modo de pizarra, que pueda tener un tamaño variable, un título y sobre la cuál se puedan dibujar círculos y rectángulos de diferentes tamaño, colores y en diferentes posiciones. En esa pizarra podría dibujar, **POR EJEMPLO** ...

¿Cómo se obtiene este **RESULTADO** siguiendo una estrategia POO?



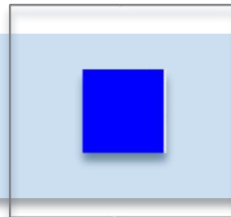
**Paso 1:**  
**Crear** Pizarra *miPizarra*  
- con título **ESPACIO DIBUJO**  
- con base y altura  $\approx 8\text{ cm}$



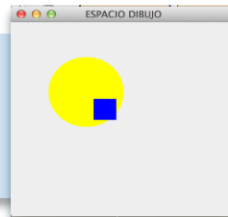
**Paso 2:**  
**Crear** Círculo *c1*  
- con radio  $\approx 1.3\text{ cm}$   
- con color **amarillo**  
- con centro en  $\approx (3, 3)$



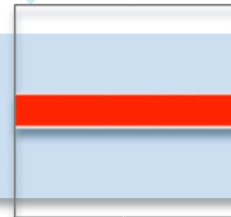
**Paso 3:**  
**Añadir a** *miPizarra* el Círculo *c1*



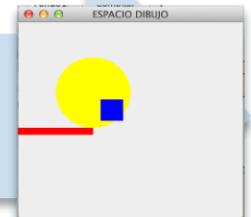
**Paso 4:**  
**Crear** Rectáng. *r1*  
- con base y altura  $\approx 0.8\text{ cm}$   
- con color **azulón**  
- con centro en  $\approx (3.3, 3.3)$



**Paso 5:**  
**Añadir a** *miPizarra* el Rectángulo *r1*



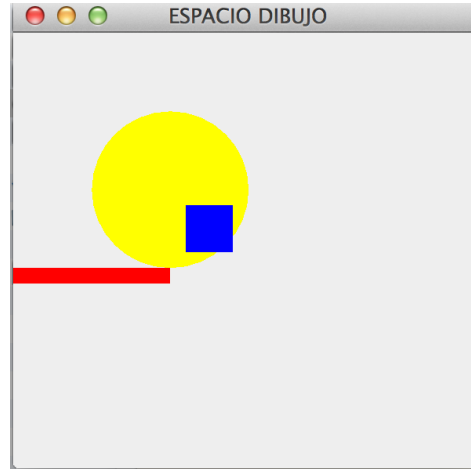
**Paso 6:**  
**Crear** Rectáng. *r2*  
- con base  $\approx 3\text{ cm}$   
- con altura  $\approx 0.3\text{ cm}$   
- con color **rojo**  
- con centro en  $\approx (1.3, 4.1)$



**Paso 7:**  
**Añadir a** *miPizarra* el Rectángulo *r2*

# Introducción

## Actividad Ejemplo - Tema 2: detalles de la fase MENTAL



### Objetos identificados

Pizarra **miPizarra**, de título **ESPACIO DIBUJO** y tamaño  $\approx 8 \times 8$  cm

Círculo **c1** de radio  $\approx 1.3$  cm, color **amarillo** y centro en  $\approx (3, 3)$

Rectángulo **r1** de base=altura  $\approx 0.8$  cm, color **amarillo** y centro en  $(3.3, 3.3)$

Rectángulo **r2** de base  $\approx 3$  cm, altura  $\approx 0.3$  cm, de color **rojo** y centro en  $\approx (1.3, 4.1)$

### Patrones distintivos

**Atributos:** *título, base, altura*  
**Acciones:** *Crear, Añadir una figura dada, ...*

**Atributos:** *radio, color, centro*  
**Acciones:** *Crear, Área, Obtener o Cambiar radio, ...*

**Atributos:** *base, altura, color, centro*  
**Acciones:** *Crear, Área, Obtener o cambiar base, ...*

¿ Algoritmo ?

USA A

¡¡TODOS!!

### Tipos identificados

Pizarra — Pizarra **USA**

Círculo ← Círculos

Pizarra **USA**  
Rectángulos

Rectángulo ←

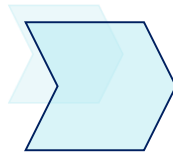
**Ejecutable**  
(Crear objetos y Manipularlos)

# Introducción

## Actividad Ejemplo - Tema 2: la fase MENTAL

- **Escribir en Java el Algoritmo** que describe la resolución del problema, una instrucción por paso:  
**Programa Java**
- **Escribir en Java la definición de cada Tipo de objetos** que se crean y manipulan en el Algoritmo:  
**Clases Java de los Datos**
- **Ejecutar el Programa Java**

**Algoritmo**



BlueJ

University of  
**Kent**

### **Entorno de Desarrollo Integrado (IDE) Java**

- Editar (**.java**)
- Compilar (**.class**)
- Ejecutar (**bytecode**)

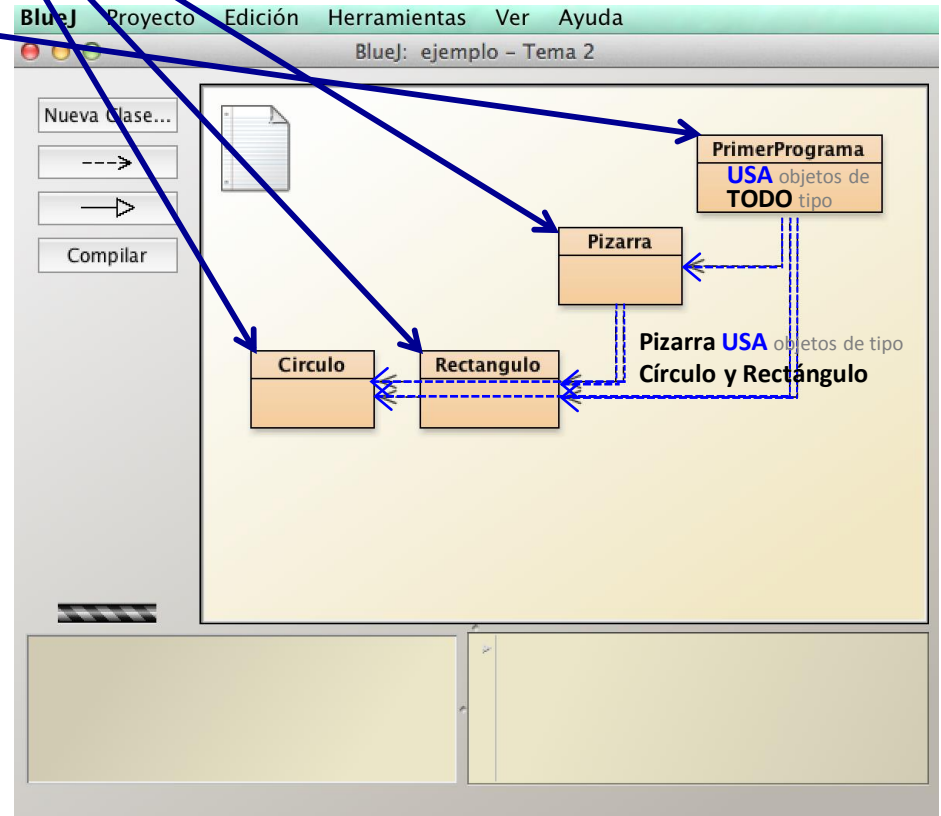
# Introducción

## Actividad Ejemplo - Tema 2: la fase MENTAL en BlueJ

Clases Java de Datos

Programa Java

Algoritmo



# Definición de una clase Java

## Estructura básica (I): cabecera y cuerpo

```
[modificadores] class NombreDeLaClase [ extends OtraClase ] {  
    cabecera (de declaración)
```

```
// Definición, o declaración, de Atributos
```

```
[modificadores] tipo nomVar1;
```

```
[modificadores] tipo nomVar2;
```

```
... ..
```

```
[modificadores] tipo nomVarN; ]
```

```
// Definición de Métodos
```

```
[modificadores] tipo nomMetodo1([listaParams]) { cuerpo }
```

```
[modificadores] tipo nomMetodo2([listaParams]) { cuerpo }
```

```
... ..
```

```
[modificadores] tipo nomMetodoM([listaParams]) { cuerpo } ]
```

```
}
```

cuerpo



BlueJ: ejemplo – Tema 2

A la vista de la estructura general de una clase Java, abre todas las clases de este proyecto e indica qué tienen en común todas ellas

# Definición de una clase Java

## Estructura básica (II): bloques

- Java es un lenguaje orientado a bloques, o “**unidades**” de código
- Delimitadores de bloque: llaves de inicio ( **{** ) y final de bloque ( **}** )
- **Bloques de instrucciones**
  1. Ejemplo: cuerpo de una clase
  2. Secuencia de cero o más instrucciones comprendidas entre las llaves **{ y }**
  3. Las instrucciones están separadas por **;** y serán ejecutadas, también, secuencialmente.
  4. Propósito: agrupar EN UNA SOLA las instrucciones que contiene; de esta forma, **un bloque se puede utilizar en los mismos sitios que una instrucción simple.**



# Definición de una clase Java

## Estructura básica de su cuerpo (I): atributos

- Los **atributos** (nomVar1, nomVar2, ..., nomVarN) **representan las componentes que TIENE UNA clase**, se declaran de un tipo de datos determinado y, habitualmente, se definen como **privados** (**modificador private**)
- El tipo de datos de un atributo define los valores que éste puede tomar y las operaciones que sobre él se pueden realizar

```
// Definición de Atributos  
[[modificadores] tipo nomVar1;  
[modificadores] tipo nomVar2;  
...  
[modificadores] tipo nomVarN; ]
```



BlueJ: ejemplo - Tema 2

- Abre las clases **PrimerPrograma** y **Circulo** del proyecto...¿Cuál tiene atributos y cuál no? ¿Por qué?
- Para cada atributo de la clase que sí los tenga, identifica los elementos de su declaración (**bloc**)

# Definición de una clase Java

## Estructura básica de su cuerpo (II): métodos

- Los **métodos** definen las **operaciones** que **TIENE UNA** clase, i.e. las que se pueden aplicar sobre los objetos de la clase. Constan de 2 partes:
  - Cabecera o perfil:** modificador de visibilidad, tipo de su resultado, nombre y, en su caso, lista de parámetros (si hay más de uno, separados entre sí por comas).  
*Es posible que un método **NO** devuelva un resultado (tipo void)*
  - Cuerpo:** instrucciones que permiten obtener su resultado, al ejecutarlo.  
*La instrucción **return** es obligatoria, salvo si el tipo de su resultado es void, y permite devolver el resultado calculado*

### // Definición de Métodos

```
[modificadores] tipo nomMetodo1([listaParams]) { cuerpo }  
[modificadores] tipo nomMetodo2([listaParams]) { cuerpo }  
...  
[modificadores] tipo nomMetodoM([listaParams]) { cuerpo } ]
```



Bluej: ejemplo - Tema 2

Abre la clase **Círculo** e identifica los elementos de las **9 primeras** cabeceras de sus métodos (**bloc**)

# Definición de una clase Java

## Estructura básica de su cuerpo (II): tipos de métodos

- **Constructores:** crean un objeto, inicializando sus atributos
- **Modificadores:** alteran el estado de un objeto, cambiando los valores de sus atributos
- **Consultores:** muestran el estado de un objeto, sin alterarlo, devolviendo el valor de uno o más de sus atributos



BlueJ: ejemplo - Tema 2

- Compara entre sí las cabeceras de los métodos de la clase **Círculo** del proyecto e identifica sus métodos constructores, consultores y modificadores (**bloc**)
- La clase **Rectángulo**, ¿tiene métodos equivalentes a los de la clase **Círculo**? ¿Por qué?
- La clase **PrimerPrograma**, ¿tiene métodos equivalentes a los de la clase **Círculo**? ¿Por qué?

# Definición de una clase Java

## Estructura básica de su cuerpo (IV): método `main`

El método `main`...

1. Marca el **punto de inicio de la ejecución de una aplicación** (Clase Programa)
2. La aplicación más sencilla en Java es una clase cuyo único método público es `main`



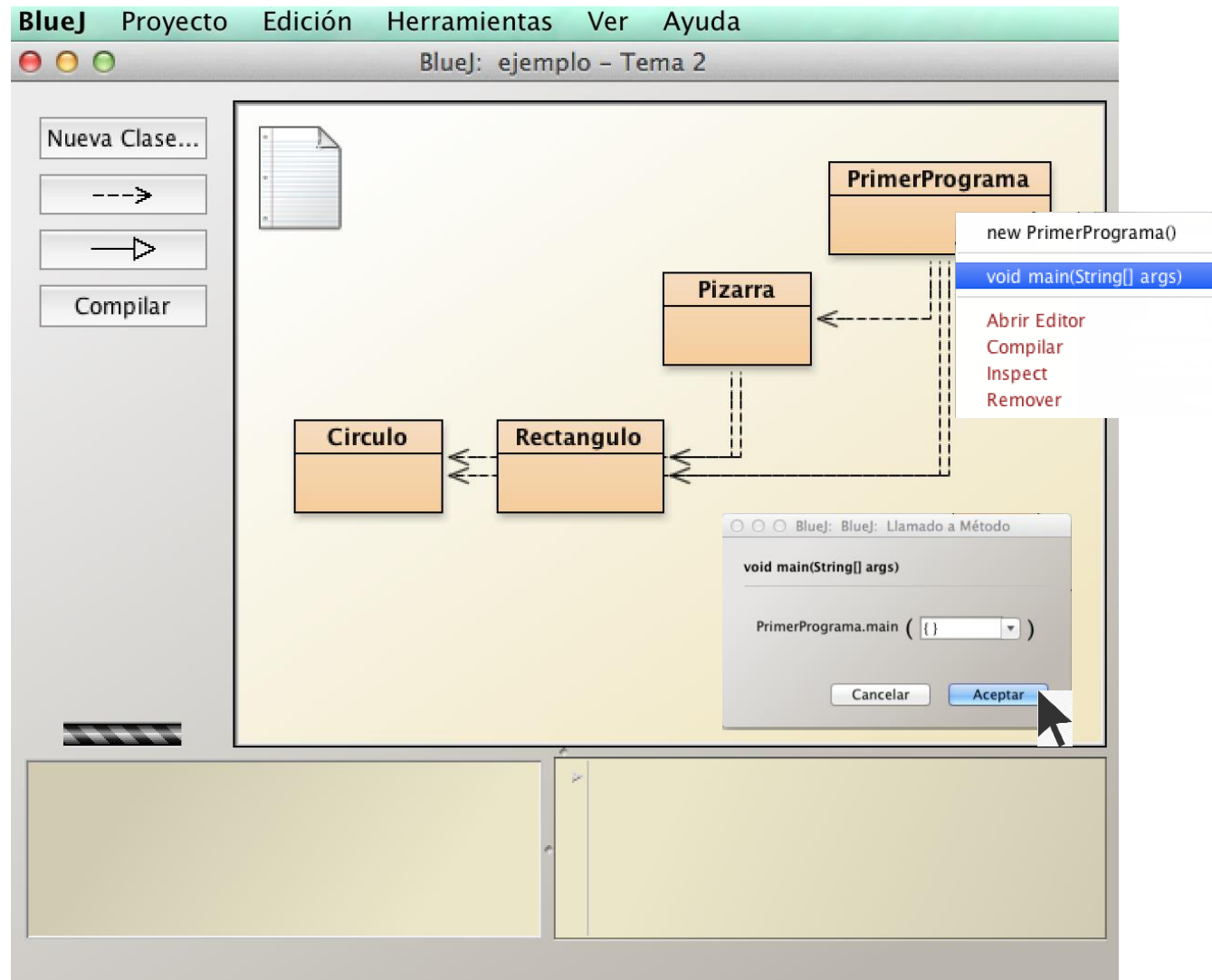
BlueJ: ejemplo - Tema 2

¿Cuál de las clases del proyecto tiene un método `main`? ¿Por qué?

Observa, en la siguiente transparencia, qué ocurre cuando se ejecuta este método y, luego, repite el proceso descrito en tu proyecto.

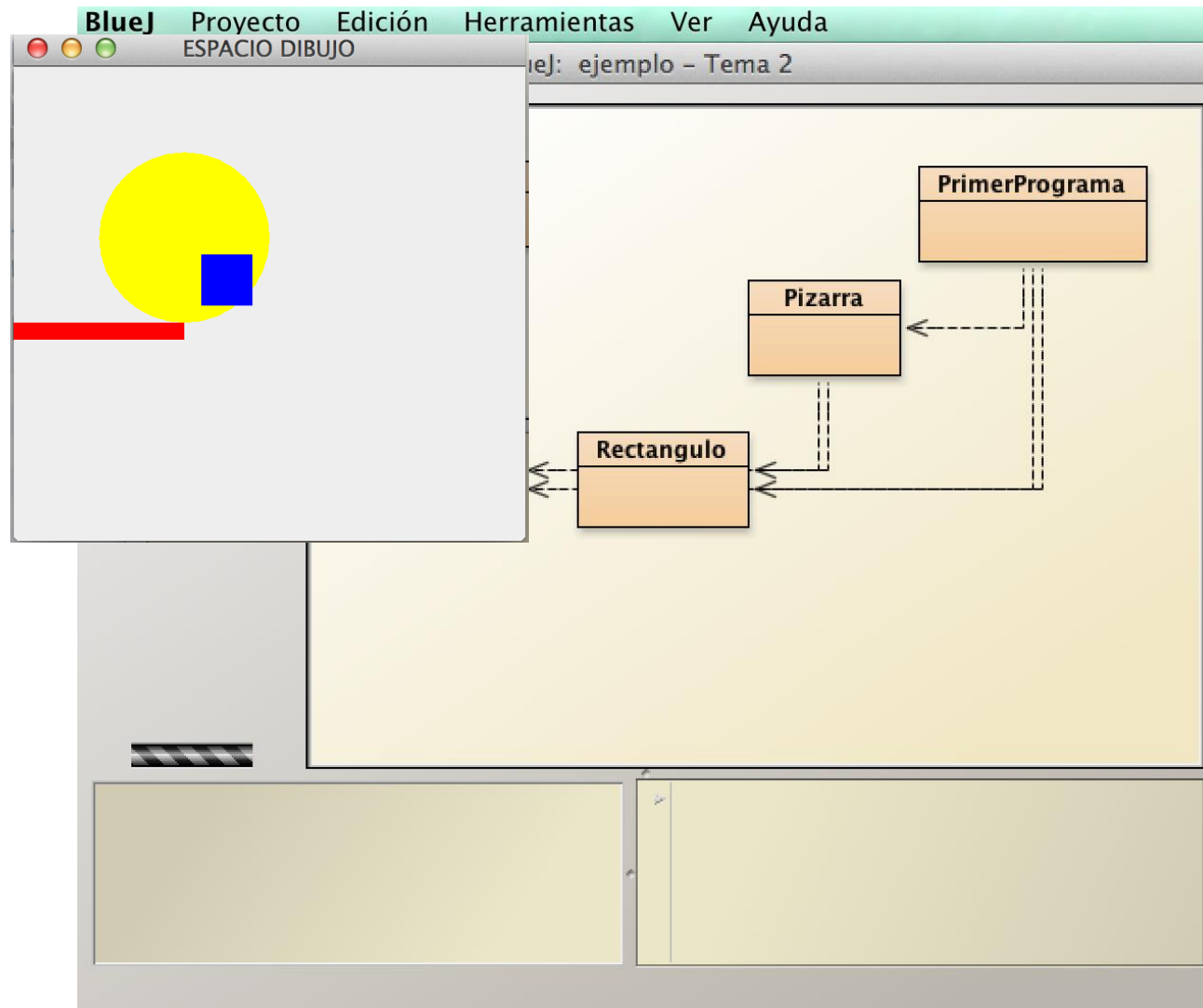
# Definición de una clase Java

## Estructura básica de su cuerpo (V): ejecución del main (I)



# Definición de una clase Java

## Estructura básica de su cuerpo (VI): ejecución del main (II)



# Definición de una clase Java

## Tipos de clases

- Según el **uso** que se va a hacer de ellas:
  1. Clase Tipo\_de\_Datos: definición de un Tipo\_de\_Objeto, i.e. de sus atributos y métodos. **NO tiene método main**
  2. Clase Programa: la única ejecutable, que “lanza” la aplicación. **SÍ tiene método main**
  3. Clase de\_Utilidades: son repositorios de métodos que pueden utilizarse desde otras clases.
- Según el **autor**: tus clases y las estandarizadas –bien por el propio lenguaje (estándar de Java), bien por otros autores.

# Uso de una clase: creación y manipulación de objetos

## operadores new y .

- El **operador new** se utiliza para crear un objeto de una clase
- El **operador punto** (.) se emplea para seleccionar el atributo deseado o el método específico que se desea aplicar (ejecutar) sobre el objeto.

```
public class PrimerPrograma {
```

```
    public static void main(String[] args) {
```

```
        // crear miPizarra para dibujar figuras, ...
```

```
        Pizarra miPizarra = new Pizarra("ESPACIO DIBUJO", 300, 300);
```

Creación e identificación de un objeto

```
        // crear un círculo c1 de radio 50, amarillo, con centro en (100,100)
```

```
        Círculo c1 = new Círculo(50, "amarillo", 100, 100);
```

Creación e identificación de un objeto

```
        // añadir c1 a miPizarra
```

```
        miPizarra.add(c1); invocación (de la ejecución) de un método distinto de main
```

```
        ...
```

```
    }
```

```
}
```



# Uso de una clase: creación y manipulación de objetos

## Ejercicio: La clase SegundoPrograma



Bluej: ejemplo - Tema 2

- Crea una clase **SegundoPrograma** en el proyecto; edita la clase para su único método sea el **main** que figura a continuación.
- Compila la clase **SegundoPrograma** y, si no tiene errores, ejecútala ¿Cuál es el resultado?
- Observa cómo y en que orden se han usado los operadores **new** y punto ( **.** ) para conseguir que el resultado del algoritmo sea el que es.

```
public static void main(String[] args) {  
    // crear un Circulo estandar c1 e incrementar su radio un 30%  
    Circulo c1 = new Circulo();  
    c1.crece();  
    // Crea un Circulo c2 con el radio, color y centro actuales de c1  
    double radioC2 = c1.getRadio();  
    String colorC2 = c1.getColor();  
    int centroXC2 = c1.getCentroX(), centroYC2 = c1.getCentroY();  
    Circulo c2 = new Circulo(radioC2, colorC2, centroXC2, centroYC2);  
    // Crear una Pizarra estandar pEstandar y dibujar c2 en ella  
    Pizarra pEstandar = new Pizarra();  
    pEstandar.add(c2);  
}
```

# Uso de una clase: creación y manipulación de objetos

## Ejercicio: La clase PruebaCirculo



BlueJ: ejemplo - Tema 2

- Crea una clase **PruebaCirculo** en el proyecto; edita la clase para su único método sea el **main** que figura a continuación. Antes de cada instrucción, escribe un comentario que indique qué hace
- Compila la clase **PruebaCirculo** y, si no tiene errores, ejecútala ¿Cuál es el resultado?
- Observa cómo y en que orden se han usado los operadores **new** y punto ( **.** ) para conseguir que el resultado del algoritmo sea el que es.

```
public static void main(String[] args) {  
    Circulo c1 = new Circulo();  
    System.out.println(c1.toString());  
    c1.decrece();  
    System.out.print("Decrece el radio a " + c1.getRadio());  
    System.out.println(" y su área es " + c1.area());  
}
```

# Resumen de la sesión

Para que te familiarices con los conceptos básicos sobre tipos, definición, elementos y uso de clases Java, hemos preparado un cuestionario en PoliformaT

**Examen** poli **[formaT]**

**Actividad Tema 2 - Clases Java: definición y uso**

# Documentación de clases Java:

## ¿Por qué es importante? (I)

```
public class PrimerPrograma {  
  
    public static void main(String[] args) {  
        // crear miPizarra para dibujar figuras, ...  
        Pizarra miPizarra = new Pizarra("ESPACIO DIBUJO", 300, 300);  
        // crear un circulo c1 de radio 50, amarillo, con centro en (100,100)  
        Circulo c1 = new Circulo(50, "amarillo", 100, 100);  
        // añadir c1 a miPizarra  
        miPizarra.add(c1);  
        ...  
    }  
}
```

Si no sé nada de la clase Pizarra, ¿cómo puedo usarla?

Si no recuerdo algún detalle de la clase Circulo, ¿qué hago?

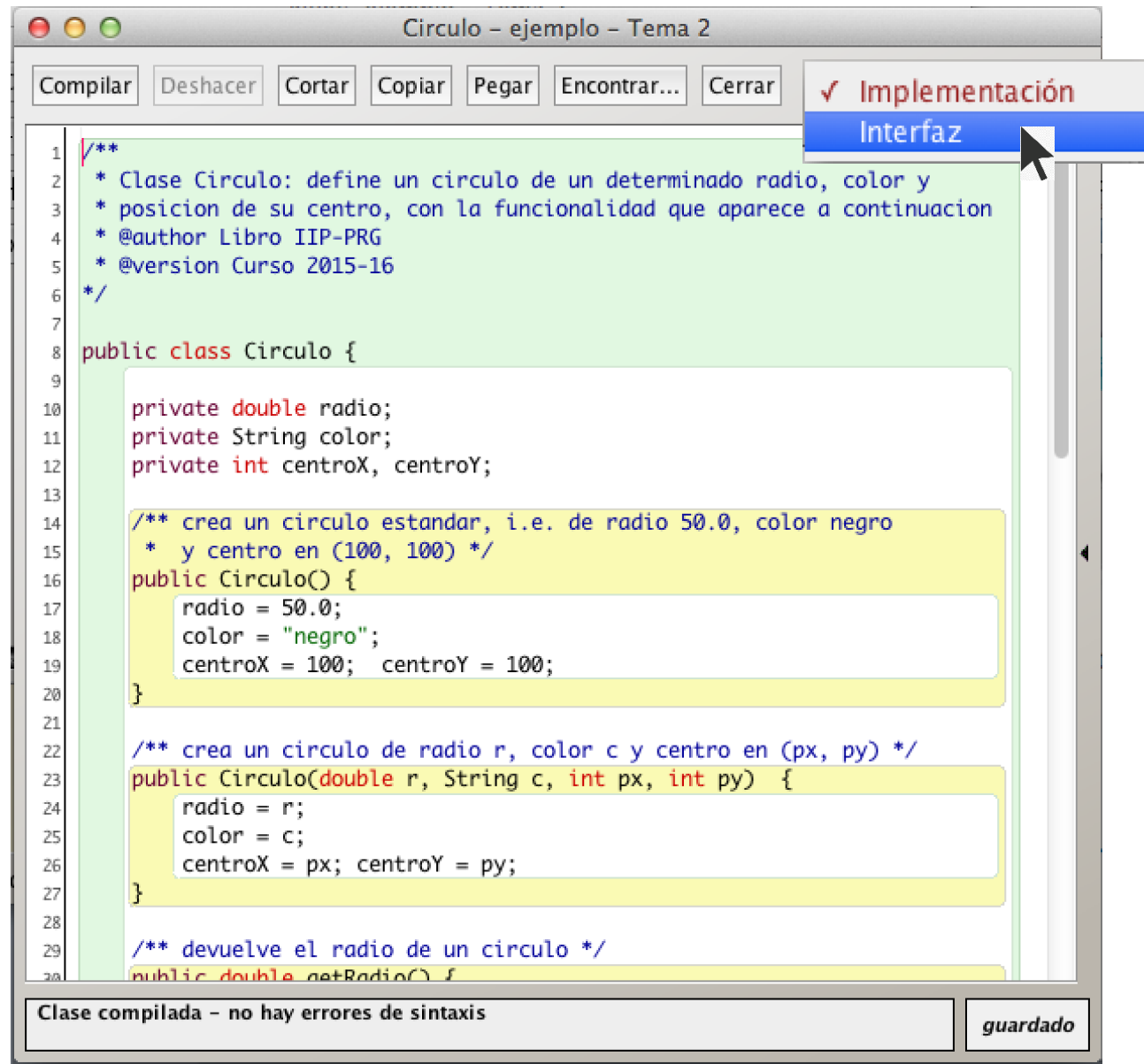


Bluej: ejemplo – Tema 2

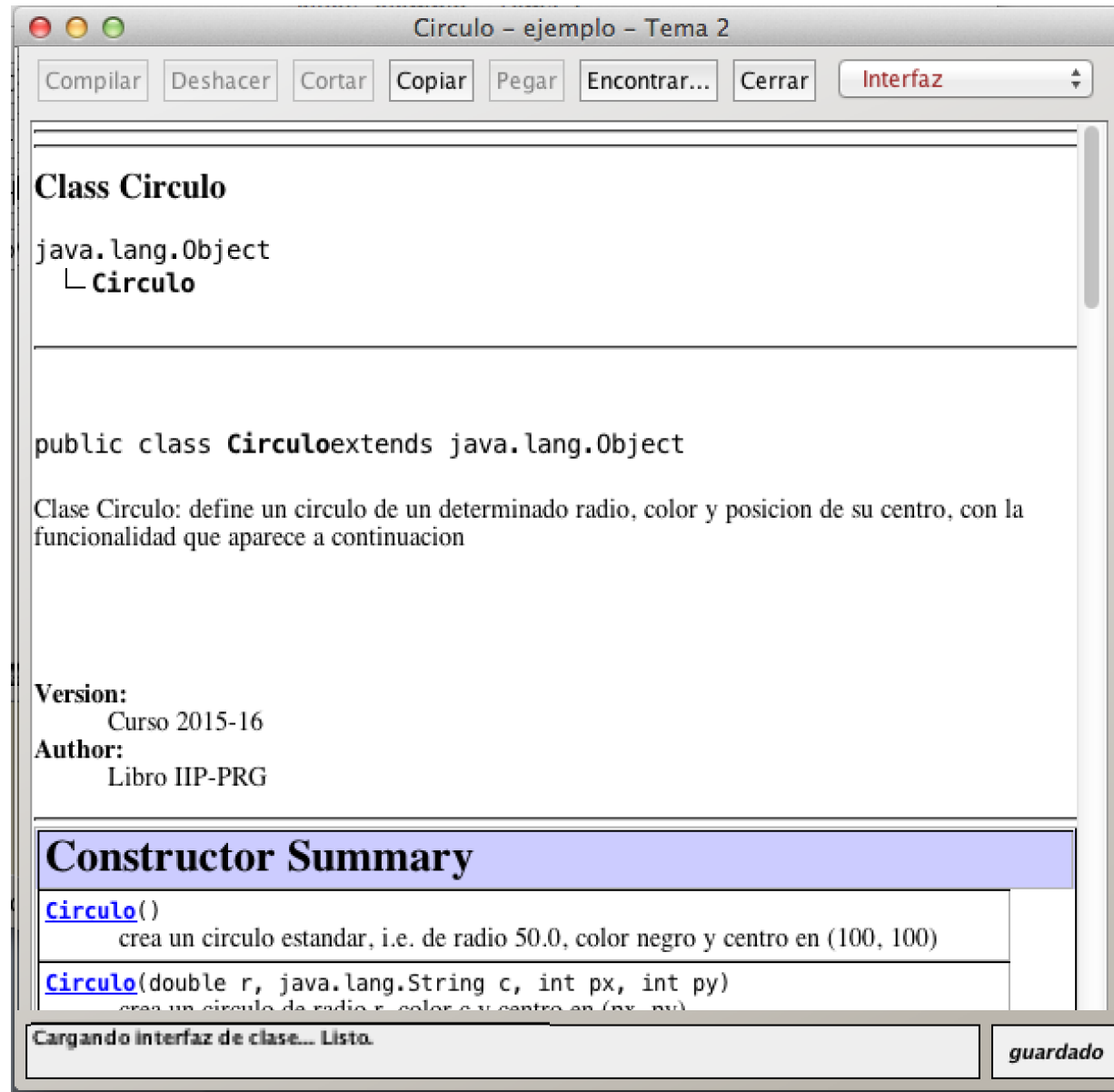
Abre, por ejemplo, la clase **Circulo** del proyecto y sigue las instrucciones que te indicamos en la siguiente transparencia.

¿Qué aparece en lugar del código Java?

# Documentación de clases Java: ¿por qué es importante? (II)



# Documentación de clases Java: ¿por qué es importante? (II)



The screenshot shows a Java IDE window titled "Circulo - ejemplo - Tema 2". The window has a menu bar with buttons: "Compilar", "Deshacer", "Cortar", "Copiar", "Pegar", "Encontrar...", "Cerrar", and a dropdown menu currently showing "Interfaz".

The main content area displays the source code for the **Class Circulo**, which extends `java.lang.Object`. The code is as follows:

```
public class Circulo extends java.lang.Object
```

Below the code, there is a descriptive paragraph: "Clase Circulo: define un circulo de un determinado radio, color y posicion de su centro, con la funcionalidad que aparece a continuacion".

Metadata information is provided below the description:

**Version:**  
Curso 2015-16

**Author:**  
Libro IIP-PRG

A section titled "Constructor Summary" is highlighted with a blue background. It lists two constructors:

- [Circulo\(\)](#)  
crea un circulo estandar, i.e. de radio 50.0, color negro y centro en (100, 100)
- [Circulo\(double r, java.lang.String c, int px, int py\)](#)  
crea un circulo de radio r, color c y centro en (px, py)

At the bottom of the window, there is a status bar with the text "Cargando interfaz de clase... Listo." and a button labeled "guardado".

# Documentación de clases Java: ¿por qué es importante? (III)



BlueJ: ejemplo - Tema 2

+ <http://docs.oracle.com/javase/8/docs/api/>

- Accede a la subcarpeta *doc* de este proyecto y abre su fichero *Circulo.html* para ver la documentación de la clase *Circulo*, generada por BlueJ al editar su interfaz.
- Genera la documentación de la clase *Pizarra* de este proyecto, que muestra (incompleta) la siguiente imagen

## Class Pizarra

```
java.lang.Object
├── java.awt.Component
│   ├── java.awt.Container
│   │   ├── java.awt.Window
│   │   │   ├── java.awt.Frame
│   │   │   │   ├── javax.swing.JFrame
│   │   │   │   └── Pizarra
```

```
public class Pizarra
extends JFrame
```

Clase Pizarra: define una Pizarra sobre la que se pueden dibujar elementos de tipo Circulo, Rectangulo y Cuadrado

### Version:

2011

### Author:

Libro-IIP-PRG

## Constructor Summary

### Pizarra()

Construye una Pizarra por defecto en la que es posible situar elementos gráficos.

### Pizarra(String titulo, int dimX, int dimY)

Construye una Pizarra con cierto título y tamaño en la que es posible situar elementos gráficos.

## Method Summary

void add(Object o)

Añade un objeto gráfico a la Pizarra y lo dibuja.

# Errores de Compilación y Ejecución en Java

## PrimerPrograma.java

```
/** Programa de prueba de las clases Circulo, Rectangulo y Pizarra
 * @author Libro IIP-PRG
 * @version 2015-2016
 */
public class PrimerPrograma {
    public static void main (String[] args) {
        // Iniciar el espacio para dibujar dándole nombre y dimensión
        Pizarra miPizarra = new Pizarra("ESPACIO DIBUJO",300,300);

        // Crear Circulo de radio 50, amarillo, con centro en (100,100)
        Circulo c1 = new Circulo(50,"amarillo",100,100);
        // Añadirlo a la Pizarra y dibujarlo
        miPizarra.add(c1);

        // Crear Rectangulo de 30 por 30, azul, con centro en (125,125)
        Rectangulo r1 = new Rectangulo(30,30,"azul",125,125);
        // Añadirlo a la Pizarra y dibujarlo
        miPizarra.add(r1);

        // Crear Rectangulo de 100 por 10, rojo, con centro en (50,155)
        Rectangulo r2 = new Rectangulo(100,10,"rojo",50,155);
        // Añadirlo a la Pizarra y dibujarlo
        miPizarra.add(r2);
    }
}
```



**javac** PrimerPrograma.java

PrimerPrograma.class

... **bytecodes** ...

**SI no hay errores de ejecución**

**java** PrimerPrograma

**SI no hay errores de compilación**



# Errores de Compilación y ejecución en Java

```
public class Circulo {  
    private double radio;  
    private String color;  
    private int centroX, centroY;  
  
    plubic Circulo() { radio=50; color="negro"; centroX=100; centroY=100; }  
  
    public double getRadio() { return radio; }  
    public void setRadio(double nuevoRadio) { radio = nuevoRadio; }  
  
    public void decrece() { radio = radio / 1.3; }  
  
    public double area() { return 3.14 * radio * radio; }  
  
    public String toStringo {  
        return "Circulo de radio " + radio + ", color " + color  
            + " y centro (" + centroX + ", " + centroY + ")";  
    }  
    ... // más métodos  
}
```



BlueJ: ejemplo - Tema 2

Abre la clase **Circulo** del proyecto y modifica su código en los métodos y forma que se te indica en esta transparencia (subrayado en rojo). Luego compila... ¿Qué pasa?

# Errores de Compilación y Ejecución en Java

```
public class PrimerPrograma {  
  
    public static void main(String[] args) {  
  
        Pizarra miPizarra;  
        Circulo c1 = new Circulo(50, "amarillo", 100, 100);  
        miPizarra.add(c1);  
        Rectangulo r1 = new Rectangulo(30, 30, "azul", 125, 125);  
        miPizarra.add(r1);  
        Rectangulo r2 = new Rectangulo(100, 10, "rojo", 50, 155);  
        miPizarra.add(r2);  
    }  
}
```



Bluej: ejemplo - Tema 2

Abre la clase **PrimerPrograma** del proyecto y modifica la primera instrucción de su método **main** tal como se muestra en esta transparencia (rojo y negrita). Luego compila... ¿Qué pasa?

# Errores de Compilación y Ejecución en Java

## (Excepciones)

```
public class PrimerPrograma {  
    public static void main(String[] args) {  
  
        Pizarra miPizarra = null;  
  
        Circulo c1 = new Circulo(50, "amarillo", 100, 100);  
        miPizarra.add(c1);  
        Rectangulo r1 = new Rectangulo(30, 30, "azul", 125, 125);  
  
        miPizarra.add(r1);  
  
        Rectangulo r2 = new Rectangulo(100, 10, "rojo", 50, 155);  
        miPizarra.add(r2);  
    }  
}
```




BlueJ: ejemplo - Tema 2

Abre la clase **PrimerPrograma** del proyecto y modifica su código en la forma que se te indica en esta transparencia (en rojo y negrita).  
Compila y ejecuta... ¿Qué pasa?

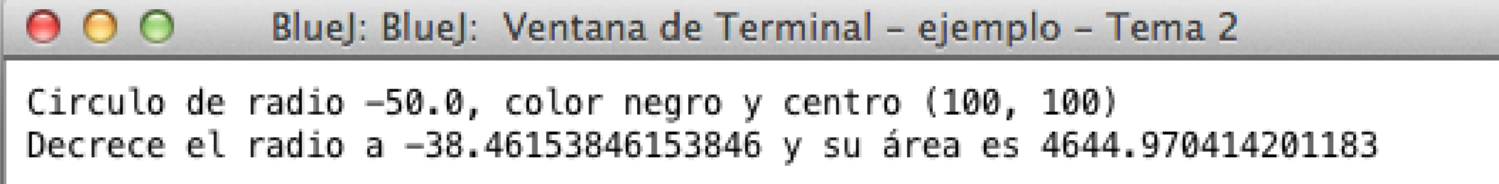
# Errores de Compilación y Ejecución en Java (Lógicos)

```
public class Circulo {  
    ...  
    public Circulo() {  
        radio = -50;  
        color="negro"; centroX = 100; centroY = 100;  
    }  
    ...  
    public void decrece() { radio = radio * -1.3; }  
    ...  
}
```



BlueJ: ejemplo - Tema 2

- Abre la clase **Circulo** del proyecto y modifica su código en los métodos y forma que se te indica en esta transparencia (en rojo y negrita). Compila.
- Compila y ejecuta ahora **PruebaCirculo** (código en transparencia 20).
- Analiza el resultado de la ejecución de **PruebaCirculo** que aparece en el terminal de BlueJ, justo la que se muestra a continuación:



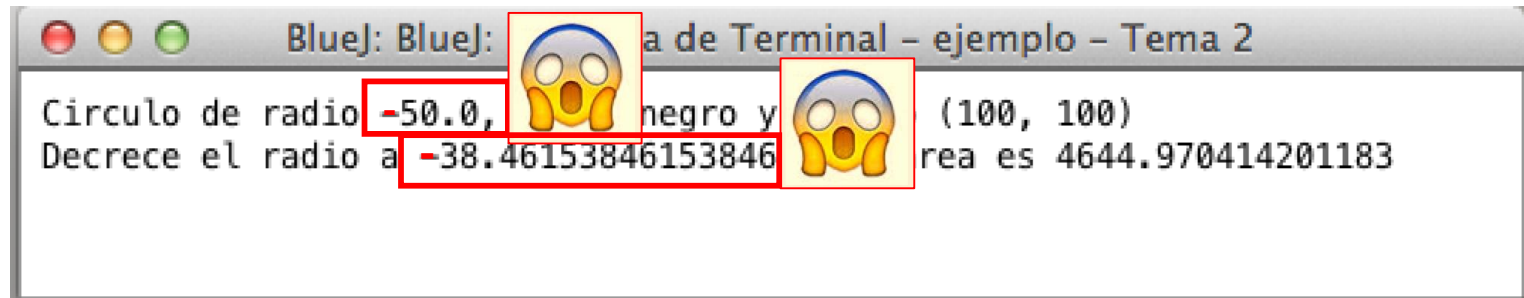
BlueJ: BlueJ: Ventana de Terminal - ejemplo - Tema 2

```
Circulo de radio -50.0, color negro y centro (100, 100)  
Decrece el radio a -38.46153846153846 y su área es 4644.970414201183
```

**¿Es correcto?** Si no lo es, ¿dónde están los errores y cuáles son? (en la siguiente transparencia tienes algunas **PISTAS**)

# Errores de Compilación y Ejecución en Java (Lógicos)

```
public class Circulo {  
    ...  
    public Circulo() {  
        radio = -50;  
        color="negro";  
        centroX = 100; centroY = 100;  
    }  
    ...  
    public void decrece() { radio = radio * -1.3; }  
    ...  
}
```



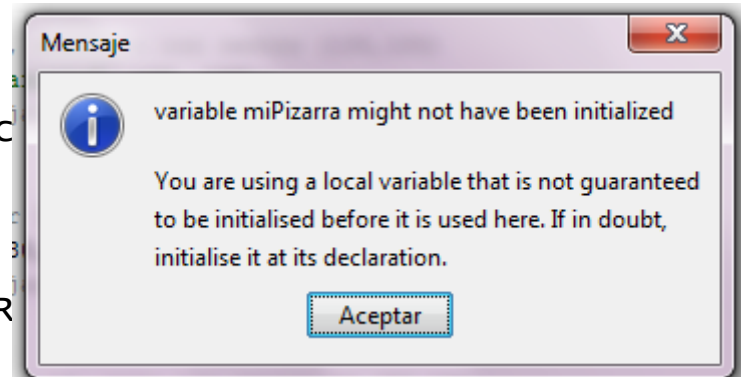
# Organización de clases Java en librerías (*packages*)

Un **package** (paquete o librería) de Java es un agregado de clases que pueden ser importadas y, tras ello, utilizadas en otras clases.

- En Java, las clases se estructuran siempre en paquetes. Cuando no se indica explícitamente, están en uno especial: **anonymous**.
- El paquete **java.lang** se importa por defecto. Forman parte de este paquete las clases **Object**, **String** y **System**.
- Facilita la organización y uso de las clases ya definidas, así como la definición y uso de nuevas vía directiva **import**

# Errores de Compilación y Ejecución en Java

```
public class PrimerPrograma {  
    public static void main(String[] args) {  
        Pizarra miPizarra;  
        Circulo c1 = new Circulo(100, 10, "rojo", 50, 155);  
        miPizarra.add(c1);  
        Rectangulo r1 = new Rectangulo(100, 10, "rojo", 50, 155);  
        miPizarra.add(r1);  
        Rectangulo r2 = new Rectangulo(100, 10, "rojo", 50, 155);  
        miPizarra.add(r2);  
    }  
}
```



BlueJ: ejemplo - Tema 2

Abre la clase **PrimerPrograma** del proyecto y modifica la primera instrucción de su método **main** tal como se muestra en esta transparencia (rojo y negrita). Luego compila... ¿Qué pasa?

# Errores de Compilación y Ejecución en Java (Excepciones)

```
public class PrimerPrograma {  
    public static void main(String[] args) {  
  
        Pizarra miPizarra = null;  
        Circulo c1 = new Circulo(50, "amarillo", 100, 100);  
        miPizarra.add(c1);  
        Rectangulo r1 = new Rectangulo(30, 30, "azul", 125, 125);  
        miPizarra.add(r1);  
        Rectangulo r2 = new Rectangulo(100, 10, "rojo", 50, 155);  
        miPizarra.add(r2);  
    }  
}
```

java.lang.NullPointerException:  
null



BlueJ: ejemplo - Tema 2

Abre la clase **PrimerPrograma** del proyecto y modifica su código en la forma que se te indica en esta transparencia (en rojo y negrita). Compila y ejecuta... ¿Qué pasa?