

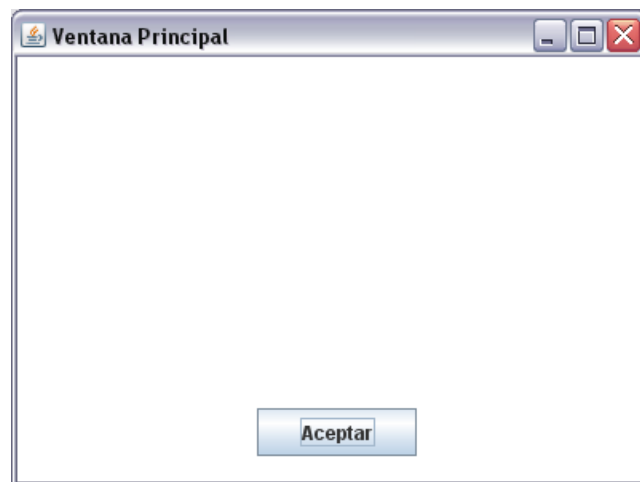
# Práctica 1

## Creación de una Aplicación con Interfaz Gráfica de Usuario

### 1. Introducción

Las aplicaciones con ventanas son el tipo de aplicación que más se utiliza habitualmente cuando se trabaja con el ordenador. Se trata de aplicaciones que utilizan como interfaz de usuario las tecnologías de las ventanas típicas de los sistemas operativos Mac OS, en la que nació, Windows y XWindows, el servidor gráfico para Linux y Unix.

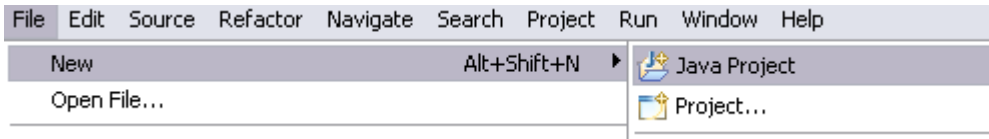
En esta primera práctica se trata de crear una aplicación que muestre una ventana que contenga un botón “Aceptar”:



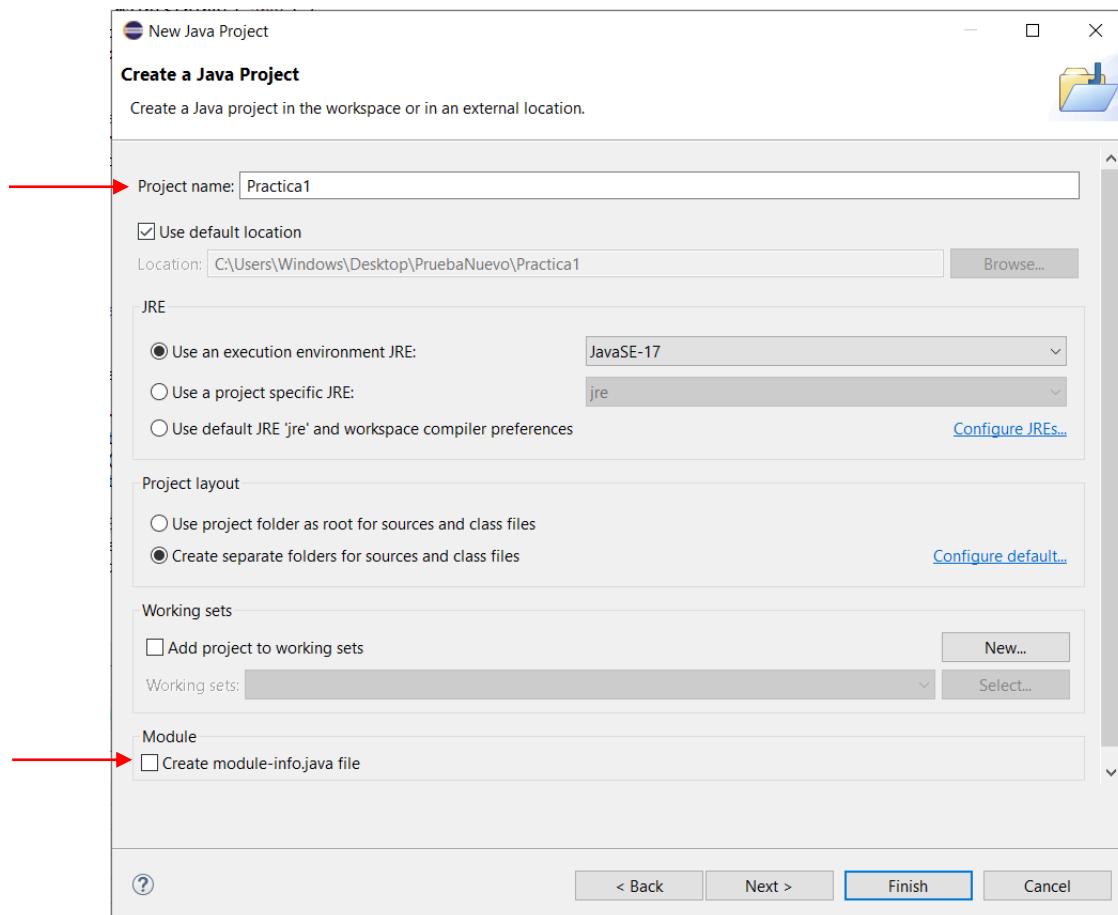
Aunque en las sucesivas prácticas comprobaremos que ésta es una tarea que resulta bastante sencilla si se utiliza alguna de las herramientas de diseño de Eclipse (como WindowBuilder), comenzaremos por incorporar nosotros todo el código necesario para conseguir la interfaz planteada con el fin de familiarizarnos con sentencias que, de otro modo, se generarían en Eclipse de manera automática.

### 2. Creación de un proyecto en Eclipse

Tras arrancar Eclipse, indica el nombre del espacio de trabajo (*workspace*) en el que Eclipse almacenará los proyectos creados. Si no tienes una carpeta específica para ello, crea una. En nuestro caso, utilizaremos una única carpeta para todos los proyectos de la asignatura (CPM). Una vez indicado el espacio de trabajo, Elige *File | New | Java Project* para iniciar el asistente para proyectos.



- **Introduce el nombre del proyecto.** Escribe *Practica1* en el campo *Project name*. Desmarca la casilla *Create module-info.java file* y pulsa *Finish*.

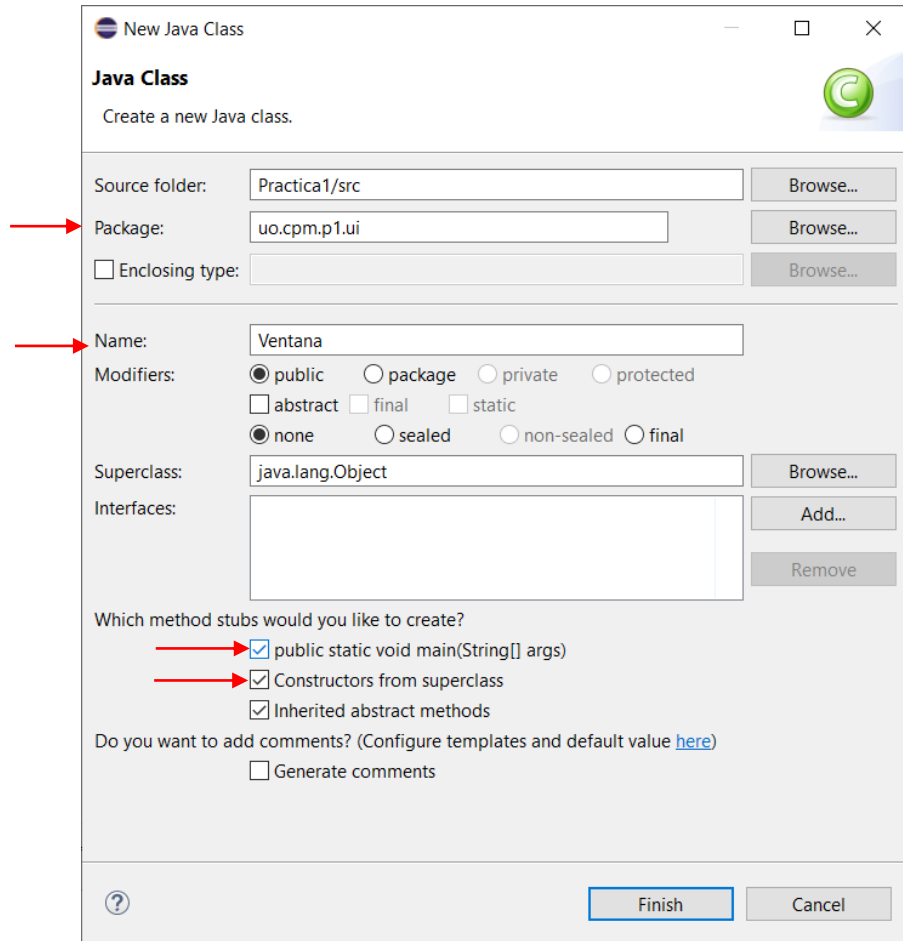


## 3. Añadir la clase al proyecto

En esta primera práctica, el proyecto va a estar compuesto de una única clase java que vamos a denominar “Ventana” en la que definiremos la interfaz. Esta nueva clase la crearemos como una *clase visual* a partir de la siguiente práctica, pero como el objetivo de ésta práctica es crear la interfaz visual “a mano”, vamos a crear una **nueva clase** java. Elige *File | New | Class*.

Introduce a continuación un nombre de paquete (*package*), que en nuestro caso será *uo.cpm.p1.ui* (se indica que es un paquete de la universidad de Oviedo, asignatura cpm, proyecto practica1 e interfaz de usuario).

Indicamos además el nombre de la nueva clase, que incorpore el método *main* y constructores de la superclase. Pulsa *Finish* para finalizar con la creación de la nueva clase.



**New Java Class**

Java Class

Create a new Java class.

Source folder: Practica1/src Browse...

Package: uo.cpm.p1.ui Browse...

☐ Enclosing type: Browse...

Name: Ventana

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static  
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☒ public static void main(String[] args)

☒ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

? Finish Cancel

El código fuente de la nueva clase incorporada es el siguiente:

```
Ventana.java x
1 package uo.cpm.p1.ui;
2
3 public class Ventana {
4
5     public Ventana() {
6         // TODO Auto-generated constructor stub
7     }
8
9     public static void main(String[] args) {
10        // TODO Auto-generated method stub
11    }
12 }
13
14 }
15 }
```

Para mostrar al usuario una ventana principal, la clase *Ventana* tiene que derivar de la clase base *JFrame*, perteneciente a la librería *Swing*, así que añadimos esta porción de código, realizando el *import* correspondiente:

```
import javax.swing.JFrame;  
public class Ventana extends JFrame
```

Al ser una clase que puede ser serializada, Eclipse nos indica que es necesario añadir un atributo que recoja el número de versión de la misma:

```
public class Ventana extends JFrame{  
  
    private static final long serialVersionUID = 1L;
```

En el **método main** crearemos un objeto ventana y estableceremos el atributo visible (inicialmente con valor false) a true.

```
public static void main(String[] args) {  
    Ventana ventana = new Ventana();  
    ventana.setVisible(true);  
}
```

Al ejecutar la aplicación, la ventana se mostrará en la pantalla:



En el **constructor de la ventana** iremos incorporando todas las sentencias relativas a la inicialización de la misma. El atributo principal es el tamaño de la ventana, así que estableceremos un valor para el mismo mediante el método `setBounds(int x,int y,int width,int height)`.

```
public Ventana() {  
    setBounds(100, 100, 450, 300);
```

Tras ejecutar de nuevo, se obtiene el siguiente resultado:



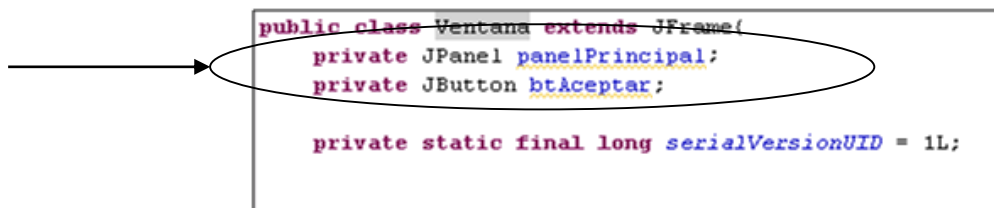
Podemos incorporar un título a la ventana mediante la siguiente sentencia añadida también en el constructor:

```
setTitle ("Ventana principal");
```

## 4. Añadir Contenedor y Botón

Hasta aquí ya hemos conseguido tener una ventana en la que incorporaremos un botón. Para ello, previamente se ha de incluir un contenedor ya que todos los controles (o componentes) que añadamos a una ventana han de estar dentro de un componente específico denominado contenedor.

En primer lugar, definimos dentro de la clase *Ventana* un contenedor (*JPanel*) y el único componente ( *JButton*) que vamos a incorporar por ahora: un botón. Para ello incorporamos dos nuevos atributos privados en la clase:



```
public class Ventana extends JFrame {  
    private JPanel panelPrincipal;  
    private JButton btAceptar;  
  
    private static final long serialVersionUID = 1L;  
}
```

Ambos componentes están definidos en la librería swing, así que podemos sustituir el import que ya habíamos incorporado por:

```
import javax.swing.*;
```

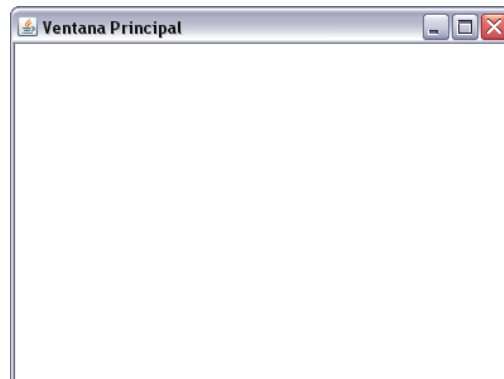
Creamos el panel principal y se lo asociamos a la ventana mediante las siguientes sentencias en el constructor de la ventana:

```
panelPrincipal = new JPanel();  
setContentPane (panelPrincipal);
```

Y establecemos, por ejemplo, el color blanco de fondo, siendo necesario importar el paquete *java.awt.Color* para tener acceso a la clase *Color*:

```
panelPrincipal.setBackground(Color.WHITE);
```

Ejecutamos:



Añadimos ahora el botón a la ventana. Para ello:

- Crear el botón y modificar los atributos:
  - Creamos el botón en el constructor de la ventana
  - Dimensionamos el botón: *setBounds(x,y,width,height)*
  - Le añadimos un texto: *setText("Aceptar")*
- “Colocar” el botón en el panel:
  - Modificamos el *layout* del panel poniéndolo a *null*<sup>1</sup>
  - Añadimos el botón al contenedor (método *add*):

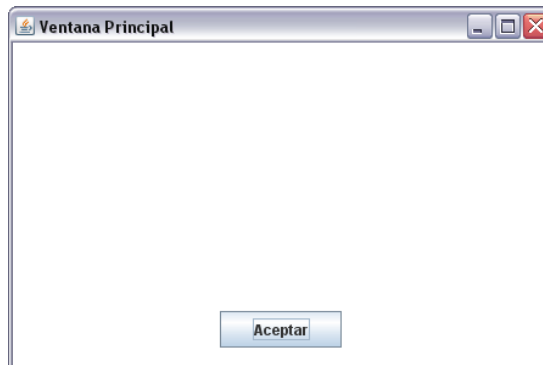
```
public Ventana() {  
    setBounds(100, 100, 450, 300);  
    setTitle ("Ventana Principal");  
  
    panelPrincipal = new JPanel();  
    setContentPane(panelPrincipal);  
    panelPrincipal.setBackground(Color.WHITE);  
    panelPrincipal.setLayout(null);  
  
    btAceptar = new JButton();  
    btAceptar.setBounds(170, 220, 100, 30);  
    btAceptar.setText("Aceptar");  
  
    panelPrincipal.add(btAceptar);  
}
```

Por último, ejecutamos de nuevo:

---

1

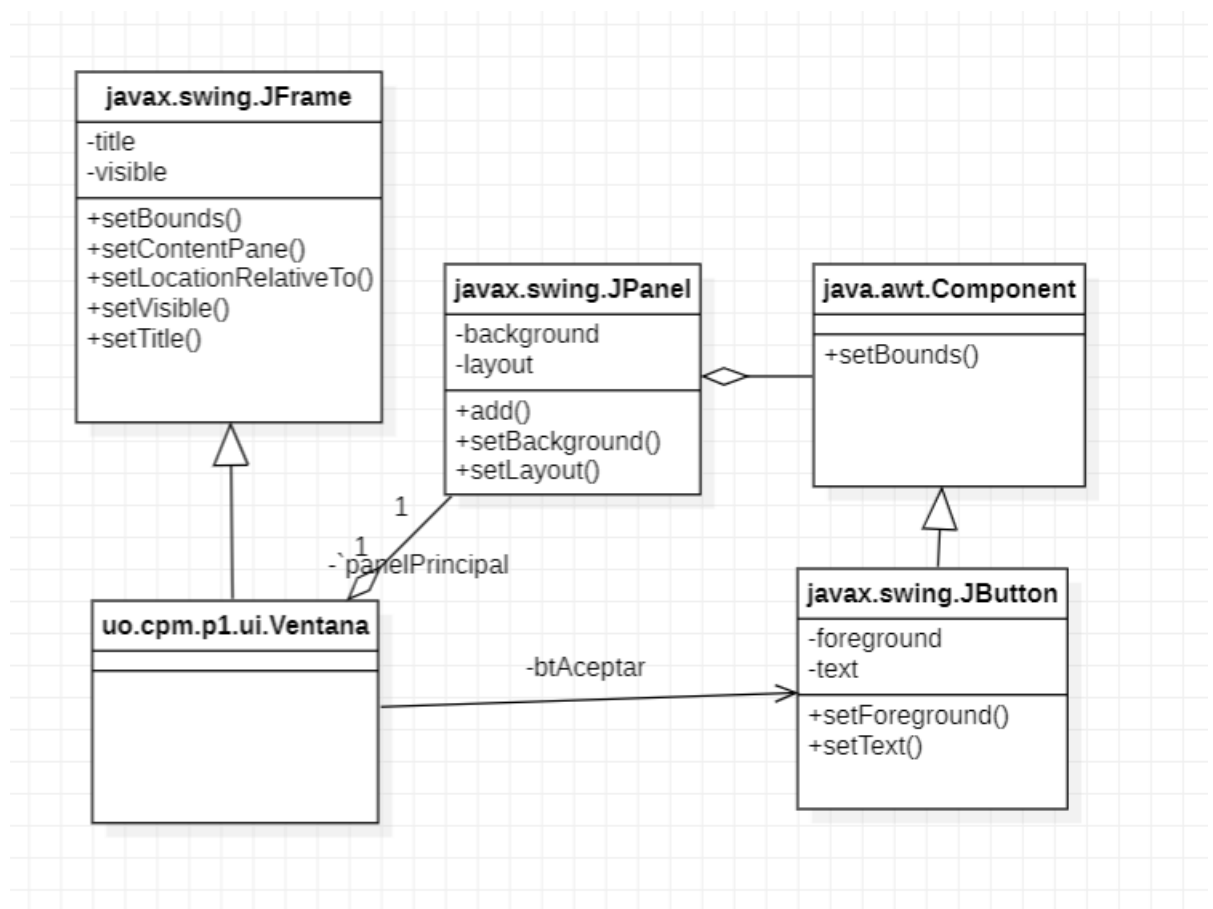
La colocación de los componentes en posición absoluta sobre un contenedor, o *null*, es la opción más adecuada para la creación de prototipos de interfaces ya que se pueden situar los componentes donde se desee. Sin embargo, la interfaz de usuario no cambia de tamaño correctamente cuando se modifica el tamaño de la ventana de la aplicación; por ello se recomienda no dejar nunca los contenedores en modo *null* para la distribución.



## 5. Centrar la ventana en la pantalla

Observamos que la ventana creada no está centrada en la pantalla. El siguiente código añadido en el constructor (siempre después de la invocación al método `setBounds()`) muestra la ventana en la parte central de la pantalla:

```
setLocationRelativeTo (null) ;
```



## Tareas práctica 1

1. **IMPORTANTE:** Estudiar los fundamentos teóricos asociados a esta práctica porque **formarán parte de los contenidos a evaluar tanto en el examen teórico como en el práctico de la asignatura.**
2. Cambiar a azul el texto del botón modificando la propiedad *foreground* del mismo.
3. Añadir un nuevo botón "Cancelar".
4. Añadir una etiqueta (clase *JLabel*) y un cuadro de texto (*TextField*) para introducir el nombre del usuario.

