

## Agenda

- Tipos de objetos y clases de la GUI
- Guía detallada para crear una interfaz gráfica de usuario
- Guía detallada para la gestión de eventos
- Problema 1 del boletín 5
- Problema 2 del boletín 5

## Contenedores y componentes

- Existen dos tipos de clases de interfaz gráfica de usuario
  - **Componentes**: JButton, JLabel, JTextField, etc.
  - **Contenedores**: que incluyen los componentes, p.ej. JFrame, JPanel, Container, etc.
- **Frame**
  - Un *frame* es un *contenedor de nivel superior* que ofrece un lugar para que el resto de componentes se “*pinten*” a sí mismos.
  - *Frame* = Ventana
  - Un JFrame contiene una pequeña barra de menú y un ContentPane

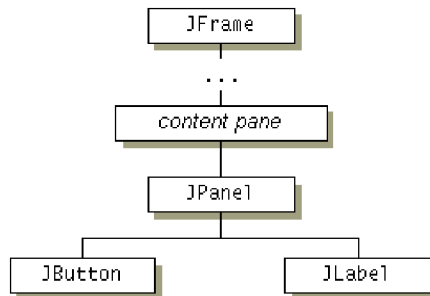
```
setTitle("Potencias de dos");
Container conPane = getContentPane();
```
  - ContentPane facilita el “lienzo” en el que se pueden agregar los demás componentes

```
conPane.add(XXXX);
```

# Contenedor

- **Panel**

- Forma más simple de un contenedor
- Su única finalidad es simplificar la ubicación de los botones, etiquetas y otros componentes

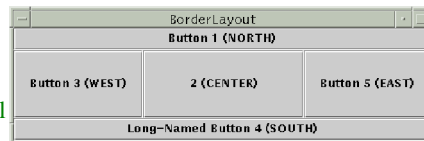


# Apariencia del contenedor

- Cada contenedor debe tener un administrador de apariencia que gestione el aspecto de los componentes dentro del contenedor

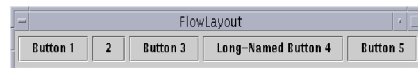
- BorderLayout

- Dispone los componentes en 5 áreas: norte, sur, este, oeste y centro. El resto de espacio adicional se coloca en el área central



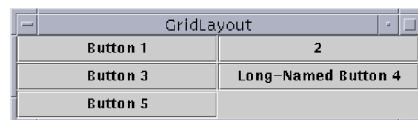
- FlowLayout

- Dispone los componentes de izquierda a derecha



- GridLayout

- Iguala el tamaño de un grupo de componentes y los muestra en el número solicitado de filas y columnas



## Componentes

- `JButton`: botón
- `JTextField`: campo de entrada de una línea
- `JLabel`: muestra texto
- `JComboBox`: lista desplegable en la que el usuario puede elegir un valor

## 5 pasos para construir una interfaz gráfica de usuario

1. Crear un contenedor de nivel superior
2. Definir la apariencia del contenedor
3. Crear los componentes
- 3a. *Crear contenedor(es) intermedios para organizar mejor los componentes*
4. Agregar componentes al contenedor
5. Agregar funciones de gestión de eventos

Boletín de problemas 5 , problema 1:

Cada vez que el usuario pulse el botón Double, el valor del campo de texto se duplica



## BP5-1, pasos 1, 2: crear un contenedor de nivel superior y definir su apariencia

- **Frame:**

- Al tratarse de una aplicación GUI independiente, la forma más sencilla es ampliar la clase JFrame

- Constructor

- Obtiene el contentPane
- Define su apariencia
- Define el título del *frame*
- .....
- Empaqueta el *frame* a un tamaño aceptable

```
public class Pow2Frame extends JFrame
{
    Public Pow2Frame()
    {
        Container conPane = getContentPane();
        conPane.setLayout(new BorderLayout());
        setTitle("Potencias de dos");
        .....
        pack();
    }
}
```

## BP5-1, paso 3: crear componentes

- **Tres componentes**



- Label

```
JLabel valueLabel = new JLabel("Valor:");
```

- TextField

- Valor inicial: 1
- Tamaño máx.: 6

```
JTextField valueField = new JTextField("1", 6);
```

- Button

```
doubleButton = new JButton("Doble");
```

## BP5-1, pasos 3a, 4: crear un contenedor intermedio y agregar componentes

- ¿Por qué?

- Para organizar mejor los componentes

- ¿Cómo?

- Se crea un panel
- Se define su apariencia en FlowLayout
- Se agregan todos los componentes al panel en el orden adecuado
- Se agrega el panel al final del *ContentPane* del *frame*



```
Jpanel buttonHolder = new JPanel();  
  
How?  
buttonHolder.add(valueLabel);  
buttonHolder.add(valueField);  
buttonHolder.add(doubleButton);  
  
conPane.add(buttonHolder, BorderLayout.SOUTH);
```

## BP5-1, paso 5: gestionar el evento

- La gestión de eventos es de vital importancia en los programas GUI
  - Puede asustar a los programadores que empiezan
  - Sin embargo, es MUCHO más fácil de lo que creen
- Resumen del proceso de gestión de eventos:
  1. ¿Qué evento se quiere gestionar? (pulsar un botón, una tecla, etc.)
  2. Buscar el tipo de evento (p.ej. `ActionEvent`)
  3. Implementar la interfaz `Listener` de dicho evento (p.ej. `ActionListener`)
  4. Agregar el `Listener` al objeto (p.ej. botón)
  5. Definir la funcionalidad del `Listener` (p.ej. qué haría el programa si el usuario pulsa el botón)

## BP5-1, paso 5: gestionar el evento

1. ¿Qué evento desea gestionar y en qué objeto? (pulsar un botón, una tecla...) `"Cada vez que el usuario pulse el botón Double, ..."`
2. Buscar el tipo de evento (p.ej. `ActionEvent`), el Listener y la interfaz
3. Implementar la interfaz Listener de dicho evento (p.ej. `ActionListener`)
4. Agregar el Listener al objeto (p.ej. botón)
5. Definir la interfaz del Listener (p.ej. qué haría el programa si el usuario pulsa el botón)

```
Evento:      ActionEvent
Listener:    ActionListener
Interfaz:    actionPerformed(...)
```

```
public class Pow2Frame extends JFrame
implements ActionListener

doubleButton.addActionListener(this);

public void actionPerformed(ActionEvent e)
{
    String valueString = valueField.getText();
    int value = Integer.parseInt(valueString);
    value *= 2;
    valueString = Integer.toString(value);
    valueField.setText(valueString);
}
```

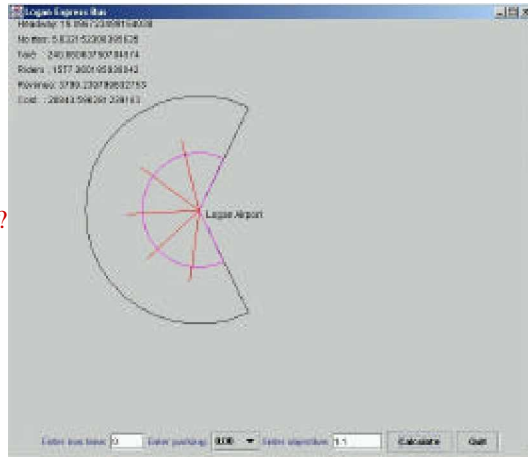
## Cómo gestionar el evento si hay varios objetos que lo escuchan

- **Problema:**
  - Si hay dos botones, ambos generarán un `ActionEvent`
  - ¿Cómo puede Java saber qué objeto desencadena el evento?
- **Solución:**
  - `e.getSource()`

```
Public void actionPerformed (ActionEvent e)
{
    if (e.getSource() == b1) //¿Qué objeto origina el evento?
        label.setText ("b1 se imprime");
    else if (e.getSource() == b2)
        label.setText ("b2 se imprime");
}
```

## BP5, problema 2: GUI

- ¿Cuál es el contenedor de nivel superior?
- ¿Cuál es la apariencia?
- ¿Cuáles son los componentes?
  - El centro del *contentPane* es un panel personalizado llamado *BusDrawing* (se suministra)
- ¿Necesitamos un panel? Si es así, ¿cuál es su apariencia?



## BP5, problema 2: eventos

1. ¿Qué evento desea gestionar y en qué objeto? (pulsar un botón, una tecla, etc.)
2. Buscar el tipo de evento (p.ej. *ActionEvent*), el *Listener* y la interfaz
3. Implementar la interfaz *Listener* de dicho evento (p.ej. *ActionListener*)
4. Agregar el *Listener* al objeto (p.ej. botón)
5. Definir la interfaz del *Listener* (p.ej. qué haría el programa si el usuario pulsa el botón)

## BP5, problema 2: sugerencias

- **BusSystem:**
  - Clase que optimiza un conjunto de rutas de autobús basadas en:
    - ✓ Horario del autobús
    - ✓ Aparcamiento
    - ✓ Peso objetivo
  - El usuario necesita definir los parámetros y llamar a la función `optimize()`
- **BusDrawing**
  - Panel personalizado que traza las rutas de autobús según el objeto `BusSystem` concreto

```
BusDrawing drawArea = new BusDrawing(bSystem);
conPane.add(drawArea, BorderLayout.CENTER);
```

- Cómo utilizar `BusDrawing`

```
drawArea.busLines();
drawArea.repaint();
```

## BP5, problema 2: sugerencias

- **Crear un JComboBox**
  - El constructor del `JComboBox` toma un *array* `String` que contiene las selecciones

```
// Cuadro combinado para seleccionar precio de aparcamiento
String[] parkingOptions = {"0.00", "2.00", "4.00", "6.00",
"8.00", "10.00"};
JComboBox parkingBox= new JComboBox(parkingOptions);
```

## BP5, problema 2: más sugerencias

- **BusFrame**
  - ¿Qué eventos deben gestionarse?
    - **¿Necesita gestionar eventos de los campos de texto y los cuadros combinados? ¿Por qué? ¿Por qué no?**
      - Pista: no necesita gestionar estos eventos
  - Cuando haya decidido qué event(os) deben gestionarse:
    - **¿Cómo se leen los datos de cada componente?**
      - Pista: utilizando `getSelectedItem()`, etc., como si respondiese a un evento de dicho origen
    - **¿Qué clas(es) implementará `ActionListener`?**
      - Pista: el panel del contenedor es una buena opción
    - **¿Cuál es la lógica de los métodos `ActionPerformed`?**
      - Pista: debe leer los cuadros combinados, definir los parámetros de `BusSystem`, invocar el cálculo de optimización y, después, los métodos de dibujo
  - ¿Cómo se conectan los orígenes de eventos y los *listeners*?
    - **¿Es posible ignorar algunos eventos sin que suponga un riesgo? Pista: Sí.**