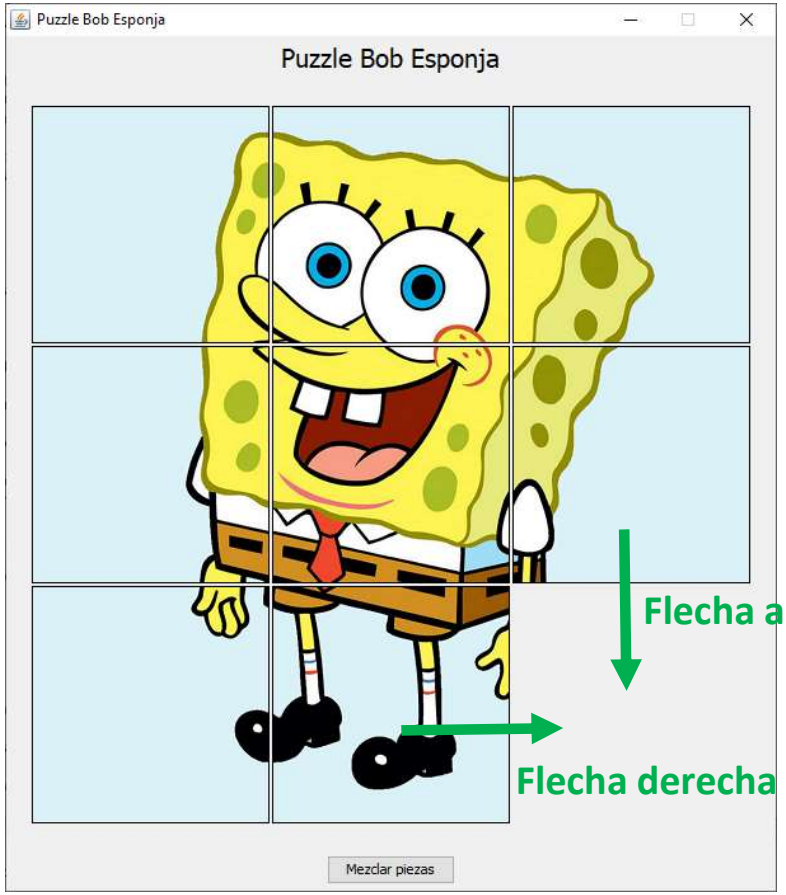
Práctica: Puzzle

- Descripción
- Componentes
- Desarrollo

Descripción



- Las piezas se mueven con las flechas del cursor
- Solo se permite mover una pieza al hueco existente.
- El botón “Mezclar piezas” desordena las piezas del puzzle.
- En cada movimiento se comprueba si el puzzle se ha finalizado.

Componentes y eventos

- JFrame
- JPanel
- GridLayout
- JLabel
- JButton
- ImageIcon
- JOptionPane
- KeyEvent / KeyListener
- MouseEvent / MouseListener

JFrame

Ventana principal de la aplicación. Tiene una barra de menu (opcional) y un panel de contenido.

```
// Constructor
JFrame frame = new JFrame("Título de la ventana");

// Salir al cerrar
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)

// Añadir un botón al panel de contenido
frame.getContentPane().add(new JButton("Botón 1"));

// Mostrar la ventana
frame.setVisible(true);
```

- Métodos
  - **setVisible**()
  - **setDefaultCloseOperation**(): Acción al cerrar
  - **get/setContentPane**(): Panel de contenido (Container)
  - **get/setJMenuBar**(): Barra de menú (JMenuBar)
  - **pack**(): Ajusta el tamaño
  - **get/setResizable**()
  - Otros: background, title, iconImage

- Listeners / Eventos:
  - **WindowListener**: windowActivated, windowClosed, windowClosing, windowOpened.
  - **WindowFocusListener**: windowGainedFocus, windowLostFocus.
  - **WindowStateListener**: windowStateChanged (maximizada, minimizada...)

JPanel

Contenedor multiuso. Dispone sus componentes hijo en base a un layout.

```
// Constructor por defecto
JPanel panel = new JPanel();

// Constructor con layout BoxLayout
JPanel panel = new JPanel(new BoxLayout());

// Establecer un borde con línea negra
panel.setBorder(BorderFactory.createLineBorder(Color.black))
;
```

- Métodos
  - **add**(<componente>, <posición>): Añade un componente al panel, opcionalmente en una posición del layout.
  - **remove**(<comp>) / **removeAll**(): Borra componentes
  - **setLayout**(layout): Establece el layout
  - **getComponent**(): componentes por índice o coordenadas
  - **setBorder**(): Bordes del panel. Usa los métodos de *BorderFactory*.
  - **setFocusable**() / **requestFocus**(): permite recibir el foco.
  - Otros: foreground, background, cursor

- Eventos / Listeners
  - No tiene eventos específicos

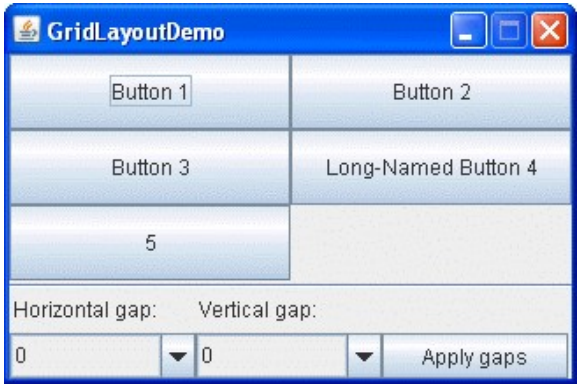
GridLayout

Dispone los componentes en una rejilla de N x M celdas. No permite celdas vacías, ni combinar celdas.

```
// Grid de 10 filas por 4 columnas
GridLayout layout = new GridLayout(10, 4);

// Idem, con un hueco entre celdas de 10 puntos horizontal y 20
vertical
GridLayout layout = new GridLayout(10, 4, 10, 20);

JPanel panel = new JPanel(layout);
panel.add(new JLabel("texto")); // Fila 1, columna 1
panel.add(new JButton("botón")); // Fila 1, columna 2
```



JLabel

Muestra texto o imágenes. El contenido no es seleccionable. Admite texto HTML.

```
// Etiqueta con texto
JLabel label = new JLabel("Texto");

// Etiqueta con texto e icono a la izquierda
ImageIcon icono = new ImageIcon("imagen.png");
JLabel label = new JLabel("Texto", icono,
SwingConstants.LEADING);

// Etiqueta con texto HTML
JLabel label = new JLabel();
label.setText("<html><b>Texto en negrita</b></html>");
```

- Métodos
  - **setOpaque**(): Opacidad (por defecto transparente)
  - **setFont**()
  - **setForeground**()
  - **setBackground**()
  - **setText**()
  - **setIcon**()
  - **setHorizontalAlignment**() / **setVerticalAlignment**(): Usa *SwingConstant* para los valores
  - **setHorizontalTextPosition**() / **setVerticalTextPosition**(): Usa *SwingConstant* para los valores
  - Otros: disabledIcon.

- Eventos / Listeners
  - No tiene eventos específicos

JButton

Botón básico, con texto y/o imagen

```
// Botón con texto
JButton boton = new JButton("Texto");

// Botón con texto e icono a la izquierda
ImageIcon icono = new ImageIcon("imagen.png");
JButton boton = new JButton ("Texto", icono,
SwingConstants.LEADING);


// Establecer el botón por defecto para la ventana
JFrame frame = new JFrame();
JButton boton = new JButton("Boton por defecto");
frame.getContentPane().add(boton);
frame.getRootPane().setDefaultButton(boton);

// Establecer acción para boton
boton.setActionCommand("imprimir");
boton.addActionListener(...)
```

- Métodos
  - **setEnabled**()
  - **setForeground**()
  - **setBackground**()
  - **setText**()
  - **setIcon**()
  - **setHorizontalAlignment**() / **setVerticalAlignment**(): Usa *SwingConstant* para los valores
  - **setHorizontalTextPosition**() / **setVerticalTextPosition**(): Usa *SwingConstant* para los valores
  - **setMnemonic**(): Tecla asociada. Usa constantes de *KeyEvent*.
  - **setActionCommand**(): Acción asociada al botón
  - Otros: disabledIcon, selectedIcon, margin

- Eventos / Listeners
  - ActionListener: actionPerformed
  - MouseListener: mouseClicked, mousePressed...

ImageIcon

Muestra imágenes a partir de un fichero o URL. Se usa sola o asociada a etiquetas, botones o tabs. Admite formatos GIF, JPEG y PNG.

```
// Cargar imagen desde resource en una etiqueta
URL url = App.class.getResource("imagenes/img1.png");
ImageIcon imagen = new ImageIcon(url);
JLabel lbPieza = new JLabel();
lbPieza.setIcon(imagen);
```

- Métodos
  - **setDescription**()
  - **setIconWidth**()
  - **setIconHeight**()

- Eventos / Listeners
  - No tiene eventos específicos

JOptionPane

Muestra diálogos “pop-up” configurables. Tiene predeterminados para información, confirmación y entrada de dato.

```
// Mensaje de alerta
JOptionPane.showMessageDialog(null, "alert", "alert",
   JOptionPane.ERROR_MESSAGE);

// Mensaje de confirmación
JOptionPane.showConfirmDialog(null,"Confirmación", "¿Seguro?",
   JOptionPane.YES_NO_OPTION);

// Mensaje para leer un dato
String inputValue = JOptionPane.showInputDialog("Introduzca un
   valor");
```

- Métodos
  - **showConfirmDialog**()
  - **showInputDialog**()
  - **showMessageDialog**()
  - **showOptionDialog**(): Combinación de los 3 anteriores
- Valores para “tipo de mensaje”: ERROR_MESSAGE, INFORMATION_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE
- Valores para “botones”: YES_NO_OPTION, YES_NO_CANCEL_OPTION, OK_CANCEL_OPTION
- Valores de retorno: YES_OPTION, NO_OPTION, CANCEL_OPTION, OK_OPTION, CLOSED_OPTION
- Eventos / Listeners
  - No tiene eventos específicos

KeyEvent / KeyListener

Evento lanzado por las acciones en el teclado.

```
private KeyListener leerPulsacion = new KeyAdapter() {
      @Override
      public void keyPressed(KeyEvent e) {
            switch (e.getKeyCode()) {
                  case KeyEvent.VK_UP:
                    System.out.println("Arriba");
                    break;
                  case KeyEvent.VK_DOWN:
                    System.out.println("Abajo");
                    break;

                  default:
                    break;
            }
      }
}
```

- Propiedades del evento:
  - **getKeyCode**(): Tecla asociada al evento. Usa las constantes definidas en *KeyEvent*
  - getKeyChar(): Carácter asociado al evento.

- Eventos:
  - **keyTyped**(): Independiente de la plataforma. Informa del carácter pulsado en getKeyChar()
  - **keyPressed**() / **keyReleased**() : Dependen de la plataforma. Informan de la tecla pulsada o liberada en getKeyCode()

MouseEvent / MouseListener

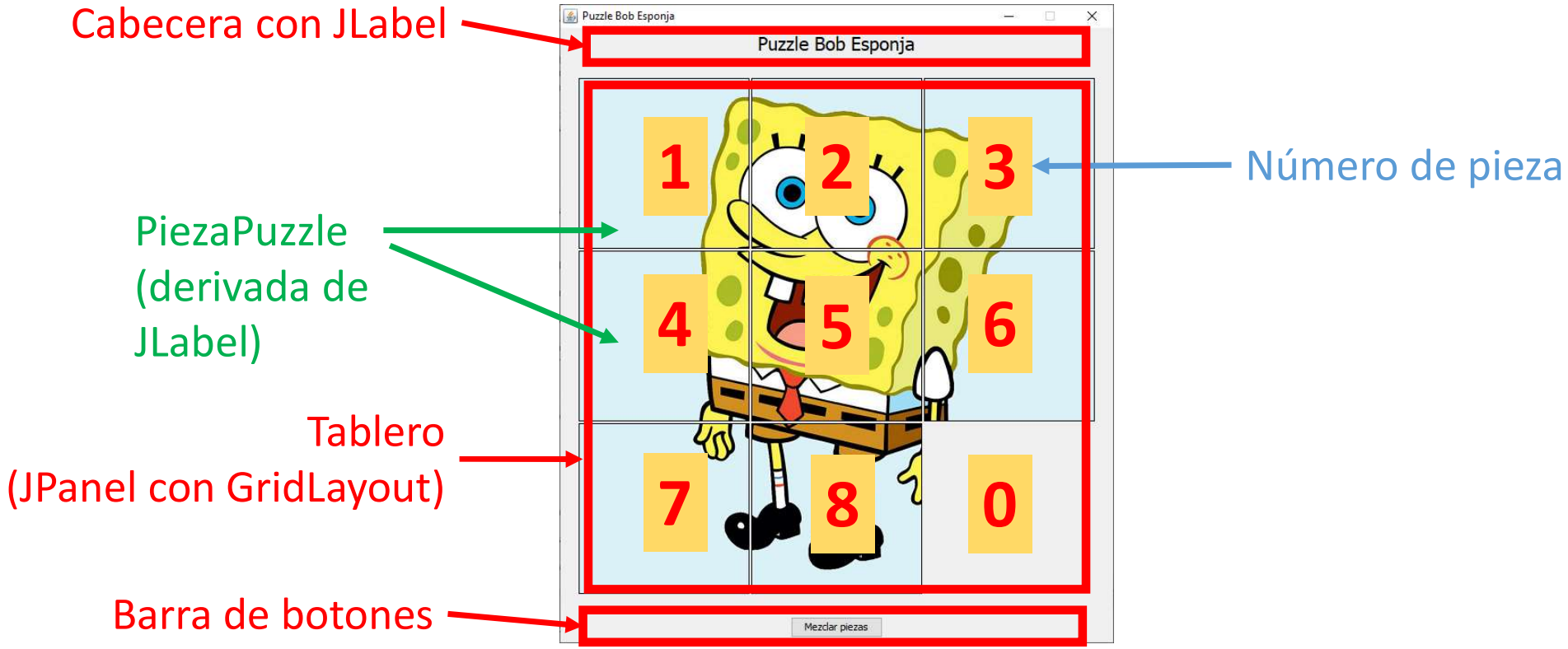
Evento lanzado por las acciones realizadas con el ratón.

```
JButton btMezclar = new JButton("Mezclar piezas");
  btMezclar.addMouseListener(new MouseAdapter() {
   @Override
   public void mouseClicked(MouseEvent e) {
      System.out.println("Botón pulsado: " + e.getButton())
   }
});
```

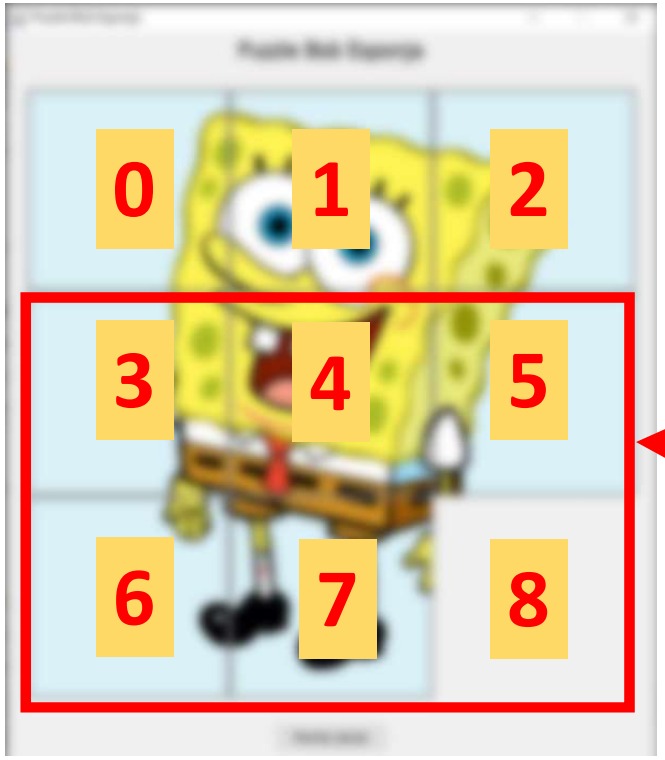
- Propiedades del evento:
  - **getX(), getY():** Posición X e Y del ratón, relativas al componente
  - **getXOnScreen(), getYOnScreen()**: Idem relativo a la ventana
  - **getButton()**: Botón pulsado (*BUTTON1...3, NOBUTTON*)
  - **isAltDown(), isControlDown(), isShiftDown()**: indican si dichas teclas están pulsadas.

- Eventos:
  - **mouseClicked()**: se ha hecho click en el componente.
  - **mouseEntered()**: se pasa sobre un componente (hover)
  - **mouseExited()**: se sale de un componente.
  - **mousePressed()**: se ha pulsado el botón sobre el componente
  - **mousedReleased()**: se ha soltado el botón sobre el componente.

Puzzle - Diseño



Movimiento de piezas hacia abajo

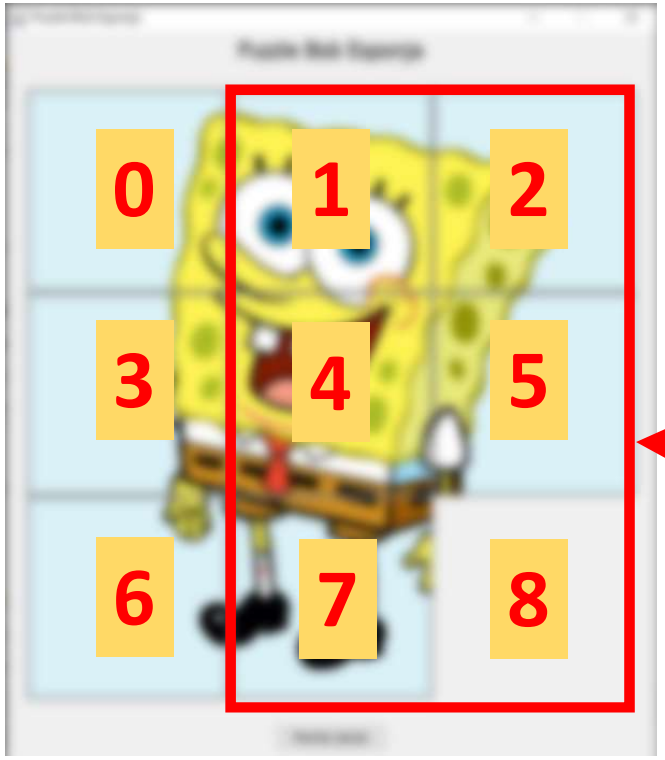


Posiciones permitidas del "hueco" para mover una pieza abajo:

índice > = 3

nueva posición del hueco = índice - 3

Movimiento de piezas a la derecha



Posiciones permitidas del "hueco" para mover una pieza abajo:

índice% 3 > 0

nueva posición del hueco = índice - 1

Puzzle - Diseño

- **PiezaPuzzle**
  - Extiende JLabel
  - Almacena el numero de la pieza
  - En el constructor carga la imagen de la pieza como icono del JLabel

Desarrollo

- Crear proyecto y clase principal
- Modelo:
  - Tablero: contenedor de casillas del puzzle
  - Piezas: Lista de piezas del puzzle.
    - El orden en la lista de cada pieza determina su posición en el tablero.

Desarrollo

Construir la ventana principal (constructor)

- Llamar al constructor de la clase padre (super())
- Establecer propiedades básicas
- Establecer layout del contenido
- Añadir componentes: Cabecera, Tablero, Piezas, Botones
  - Secuencia:
    - Crear componente
    - Establecer propiedades / Inicializar datos
    - Añadir al contenedor padre
- Ajustar tamaño (pack)

Desarrollo

Métodos auxiliares:

- Componer el tablero a partir del modelo (lista de piezas)
- Mover piezas arriba, abajo, derecha e izquierda
  - Captura de evento de pulsación de teclas
- Comprobar si el puzzle está finalizado
- Mezclar las piezas