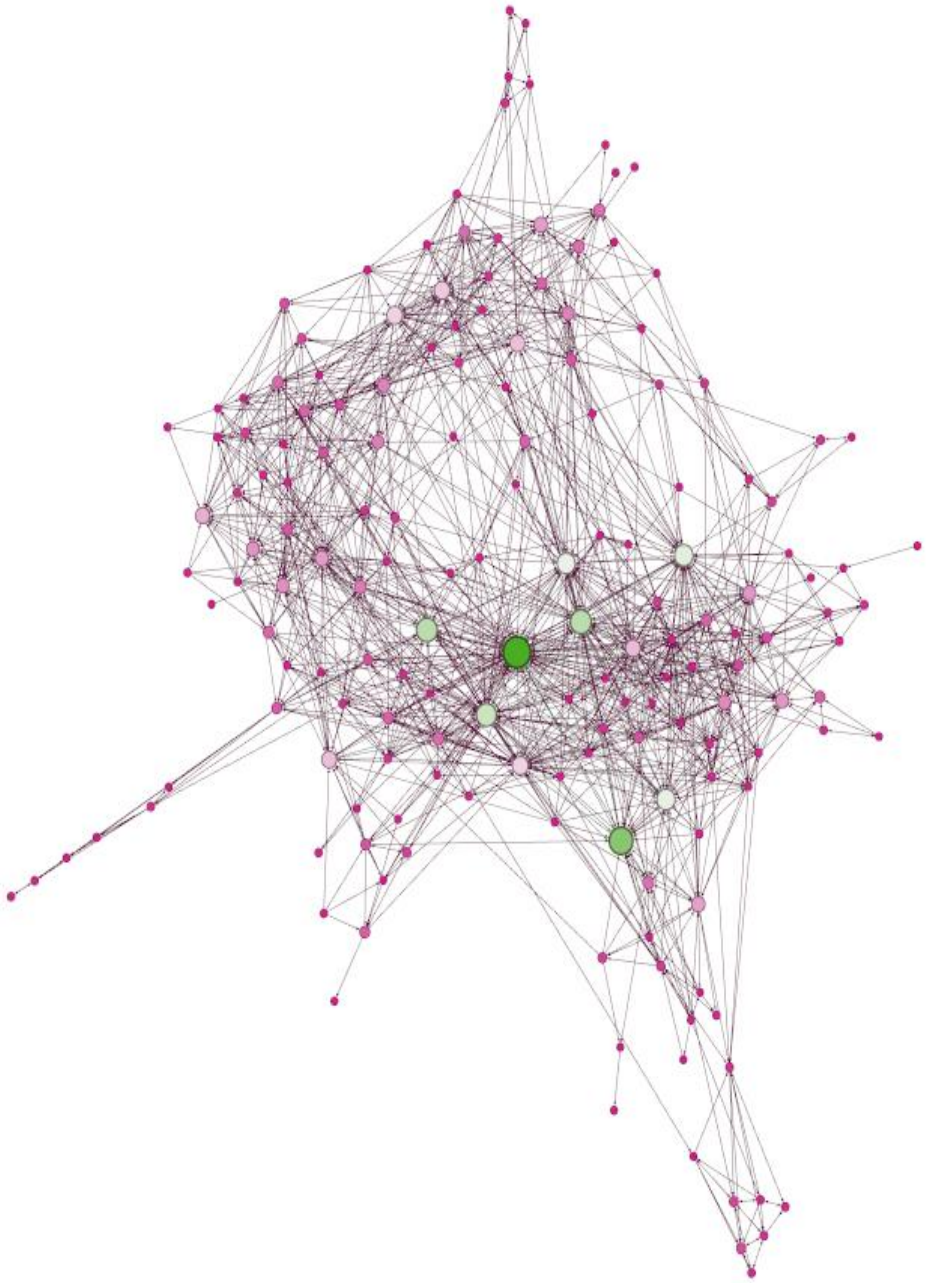


Programación Orientada a Objetos GUI

Fabián Andrés Giraldo

Libro Base: Daniel_Liang, Introduction_to_Java_Programming

Graphics

Introducción

Se puede dibujar formas personalizadas sobre una GUI.

La Clase **Graphics**

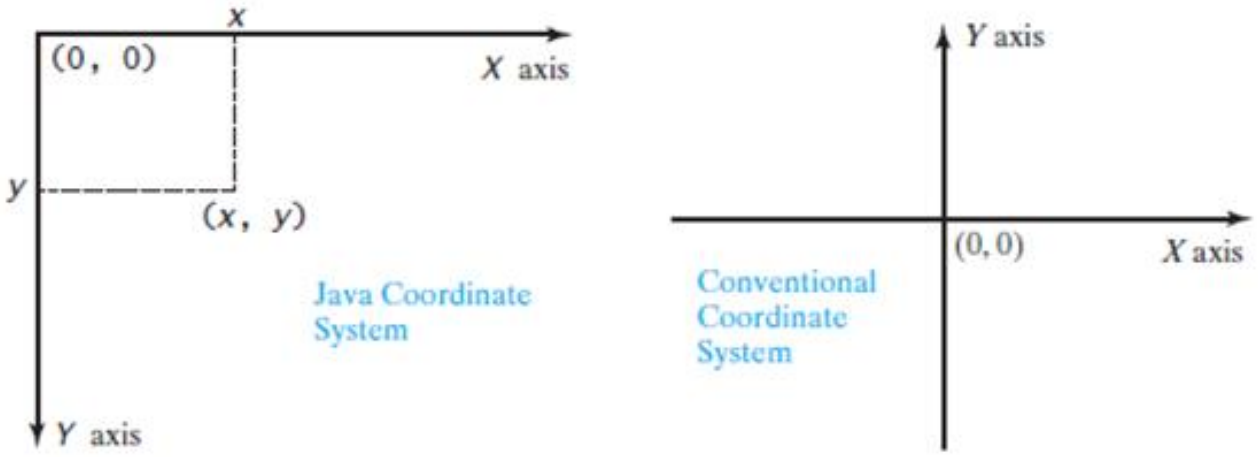
- *La clase* **Graphics** *contiene métodos para dibujar varias formas.*
- La clase **Graphics** provee los métodos para dibujar texto, líneas, rectángulos, óvalos, arcos, polígonos, y poli-lineas.

Graphics

Para dibujar sobre un componente, se necesita definir una clase que extienda de **JPanel** y sobrescribir el método **paintComponent** y especificar que quiere pintar. La signatura del método **paintComponent** es:

protected void paintComponent(Graphics g)

Sistema de Coordenas



Graphics

```
1  import javax.swing.*;
2  import java.awt.Graphics;
3
4  public class TestPaintComponent extends JFrame {
5    public TestPaintComponent() {
6      add(new NewPanel());
7    }
8
9    public static void main(String[] args) {
10     TestPaintComponent frame = new TestPaintComponent();
11     frame.setTitle("TestPaintComponent");
12     frame.setSize(200, 100);
13     frame.setLocationRelativeTo(null); // Center the frame
14     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15     frame.setVisible(true);
16   }
17 }
18
19 class NewPanel extends JPanel {
20   @Override
21   protected void paintComponent(Graphics g) {
22     super.paintComponent(g);
23     g.drawLine(0, 0, 50, 50);
24     g.drawString("Banner", 0, 40);
25   }
26 }
```

Graphics

java.awt.Graphics	
+setColor(color: Color): void	Sets a new color for subsequent drawings.
+setFont(font: Font): void	Sets a new font for subsequent drawings.
+drawString(s: String, x: int, y: int): void	Draws a string starting at point (x, y).
+drawLine(x1: int, y1: int, x2: int, y2: int): void	Draws a line from (x1, y1) to (x2, y2).
+drawRect(x: int, y: int, w: int, h: int): void	Draws a rectangle with specified upper-left corner point at (x,y) and width w and height h.
+fillRect(x: int, y: int, w: int, h: int): void	Draws a filled rectangle with specified upper-left corner point at (x, y) and width w and height h.
+drawRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void	Draws a round-cornered rectangle with specified arc width aw and arc height ah.
+fillRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void	Draws a filled round-cornered rectangle with specified arc width aw and arc height ah.
+draw3DRect(x: int, y: int, w: int, h: int, raised: boolean): void	Draws a 3-D rectangle raised above the surface or sunk into the surface.
+fill3DRect(x: int, y: int, w: int, h: int, raised: boolean): void	Draws a filled 3-D rectangle raised above the surface or sunk into the surface.
+drawOval(x: int, y: int, w: int, h: int): void	Draws an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+fillOval(x: int, y: int, w: int, h: int): void	Draws a filled oval bounded by the rectangle specified by the parameters x, y, w, and h.
+drawArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void	Draws an arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+fillArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void	Draws a filled arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+drawPolygon(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a closed polygon defined by arrays of x- and y-coordinates. Each pair of (x[i], y[i])-coordinates is a point.
+fillPolygon(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a filled polygon defined by arrays of x- and y-coordinates. Each pair of (x[i], y[i])-coordinates is a point.
+drawPolygon(g: Polygon): void	Draws a closed polygon defined by a Polygon object.
+fillPolygon(g: Polygon): void	Draws a filled polygon defined by a Polygon object.
+drawPolyline(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a polyline defined by arrays of x- and y-coordinates. Each pair of (x[i], y[i])-coordinates is a point.

Graphics

- Pintar polígonos

java.awt.Polygon	
+xpoints: int[]	x-coordinates of all points in the polygon.
+ypoints: int[]	y-coordinates of all points in the polygon.
+npoints: int	The number of points in the polygon.
+Polygon()	Creates an empty polygon.
+Polygon(xpoints: int[], ypoints: int[], npoints: int)	Creates a polygon with the specified points.
+addPoint(x: int, y: int): void	Appends a point to the polygon.
+contains(x: int, y: int): boolean	Returns true if the specified point (x, y) is contained in the polygon.

```
Polygon polygon = new Polygon();
polygon.addPoint(40, 20);
polygon.addPoint(70, 40);
polygon.addPoint(60, 80);
polygon.addPoint(45, 45);
polygon.addPoint(20, 60);
```

```
int x[] = {40, 70, 60, 45, 20};
int y[] = {20, 40, 80, 45, 60};
g.drawPolygon(x, y, x.length);
```

Graphics

Una imagen también se puede poner en un panel.

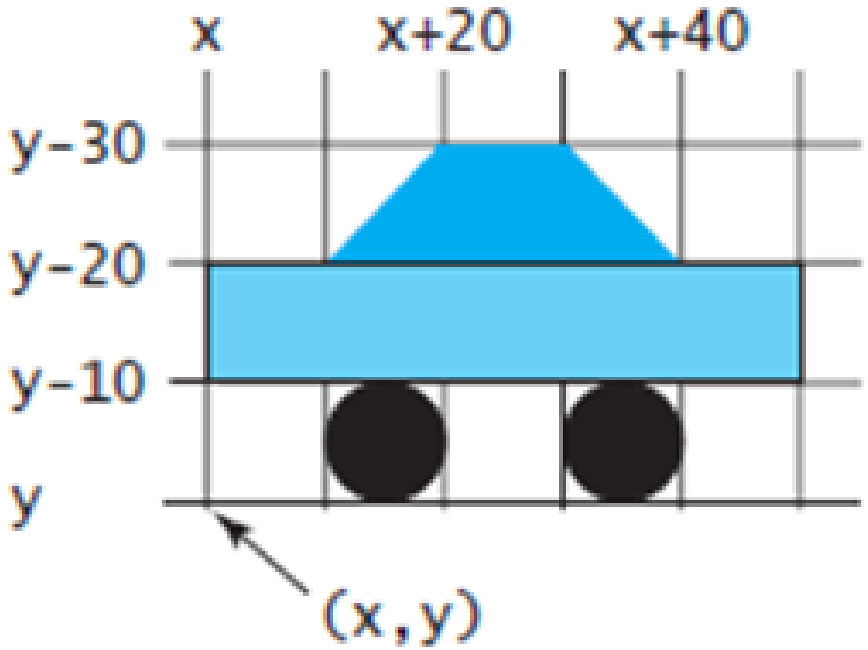
```
class ImagePanel extends JPanel {
  private ImageIcon imageIcon = new ImageIcon("image/us.gif");
  private Image image = imageIcon.getImage();

  @Override /** Draw image on the panel */
  protected void paintComponent(Graphics g) {
    super.paintComponent(g);

    if (image != null)
      g.drawImage(image, 0, 0, getWidth(), getHeight(), this);
  }
}
```

Ejercicio

- Usando los métodos de la clase Graphics se solicita dibujar el siguiente carro



Eventos

- Gestionando Eventos

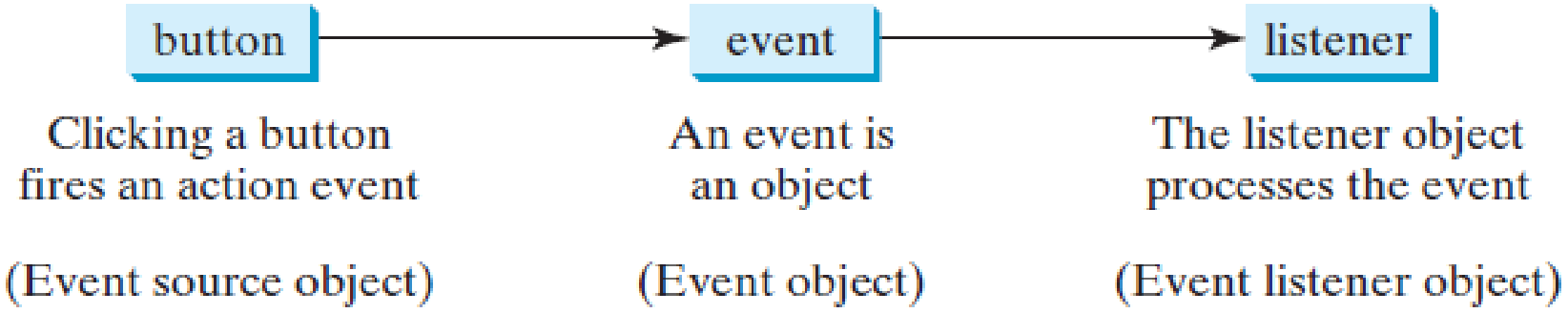
Con el fin de crear interfaces interactivas, se deben manejar los eventos en Java. Cuando el usuario da click en un componente, mueve el mouse sobre este, el sistema de interfaces de Java crea un tipo de objeto llamado evento que representa esta acción.

Eventos

- Para lograr que un componente particular responda a un evento (tal como el click a un botón) se usa un objeto llamado Listener.
- Un objeto Listener es un objeto que es notificado cuando un evento ocurre y ejecuta código que responda al evento.
- En Java, los Listeners son escritos implementando interfaces particulares. La interface para gestionar eventos es ActionListener. La interface ActionListener interface está en el paquete java.awt.event.

Eventos

Para responder al Click, se necesita escribir el código para procesar la acción. Por ejemplo, un botón es un objeto Fuente de evento — donde las acciones son originadas. Se requiere crear un objeto que tenga la capacidad de manejar el evento del botón. Este objeto es llamado el es evento escuchador (Listener).



Eventos

No todos los objetos pueden escuchar un evento. Para ser escuchador de un evento se quiere dos elementos

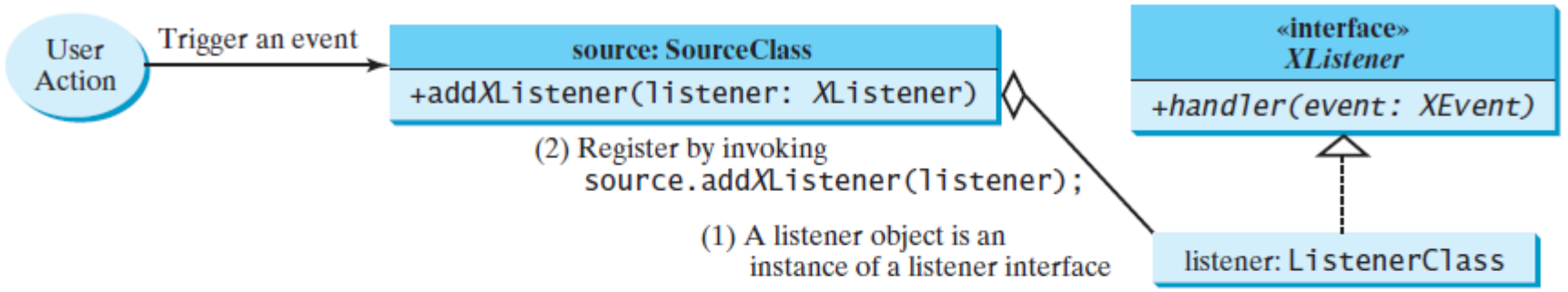
1. El objeto debe ser una instancia de la interface **ActionListener**. Esta interface define el comportamiento común de todos los escuchadores de eventos.
   – La interface **ActionListener** contiene el método **actionPerformed** para procesar el evento.

2. El objeto **ActionListener** debe ser registrado con el objeto Fuente del evento usando el método **source.addActionListener(listener)**.

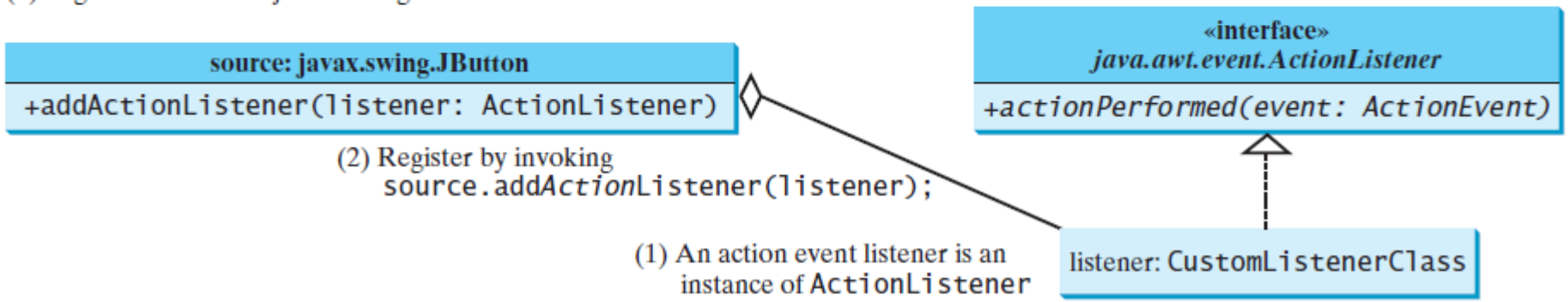
Eventos

User Action	*Source Object*	*Event Type Fired*	*Listener Interface*	*Listener Interface Methods*
Click a button	`JButton`	`ActionEvent`	`ActionListener`	`actionPerformed(ActionEvent e)`
Press Enter in a text field	`JTextField`	`ActionEvent`	`ActionListener`	`actionPerformed(ActionEvent e)`
Select a new item	`JComboBox`	`ActionEvent` `ItemEvent`	`ActionListener` `ItemListener`	`actionPerformed(ActionEvent e)` `itemStateChanged(ItemEvent e)`
Check or uncheck	`JRadioButton`	`ActionEvent` `ItemEvent`	`ActionListener` `ItemListener`	`actionPerformed(ActionEvent e)` `itemStateChanged(ItemEvent e)`
Check or uncheck	`JCheckBox`	`ActionEvent` `ItemEvent`	`ActionListener` `ItemListener`	`actionPerformed(ActionEvent e)` `itemStateChanged(ItemEvent e)`
Select a new item	`JComboBox`	`ActionEvent` `ItemEvent`	`ActionListener` `ItemListener`	`actionPerformed(ActionEvent e)` `itemStateChanged(ItemEvent e)`
Mouse pressed	`Component`	`MouseEvent`	`MouseListener`	`mousePressed(MouseEvent e)`
Mouse released				`mouseReleased(MouseEvent e)`
Mouse clicked				`mouseClicked(MouseEvent e)`
Mouse entered				`mouseEntered(MouseEvent e)`
Mouse exited				`mouseExited(MouseEvent e)`
Mouse moved			`MouseMotionListener`	`mouseMoved(MouseEvent e)`
Mouse dragged				`mouseDragged(MouseEvent e)`
Key pressed	`Component`	`KeyEvent`	`KeyListener`	`keyPressed(KeyEvent e)`
Key released				`keyReleased(KeyEvent e)`
Key typed				`keyTyped(KeyEvent e)`

Eventos

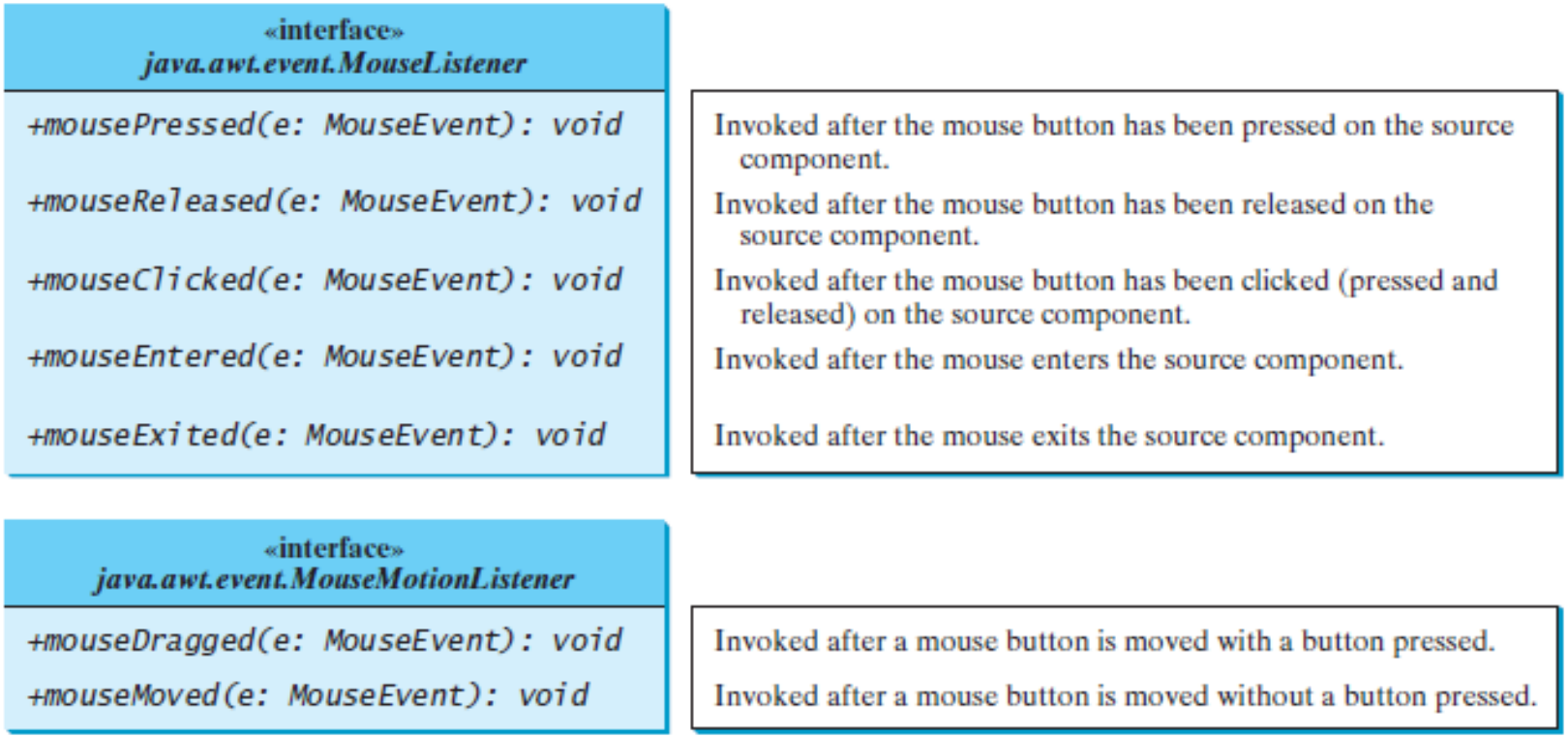


(a) A generic source object with a generic listener



Gestión Eventos

MouseListener permite gestionar los eventos del Mouse



Animaciones

- Animaciones usando la clase **Timer. La clase javax.swing.Timer** es un componente Fuente que lanza un **ActionEvent** a una tasa de tiempo predefinido.
- Un objeto **Timer** sirve como un objeto **ActionEvent**. El escuchado debe ser una instancia de **ActionListener** y debe ser registrado en el objeto **Timer**.
- Se puede crear un objeto **Timer** usando como parámetros del constructor la tasa de tiempo y el objeto listener, donde la tasa de tiempo **especifica el número de milisegundo entre dos eventos.**
- Se pude usar listener adicionales usando **addActionListener**
- Para iniciar un Time invocar el método **start()**; y para detener un Timer usar el método **stop().**

Animaciones

Invocar el método **repaint causa que el método paintComponent se ha llamado**. El método **repaint** el invocado para refrescar el área de la vista. Típicamente, se llama si tiene nuevas cosas que desplegar.

Animaciones

javax.swing.Timer	
+Timer(delay: int, listener: ActionListener)	Creates a Timer object with a specified delay in milliseconds and an ActionListener.
+addActionListener(listener: ActionListener): void	Adds an ActionListener to the timer.
+start(): void	Starts this timer.
+stop(): void	Stops this timer.
+setDelay(delay: int): void	Sets a new delay value for this timer.

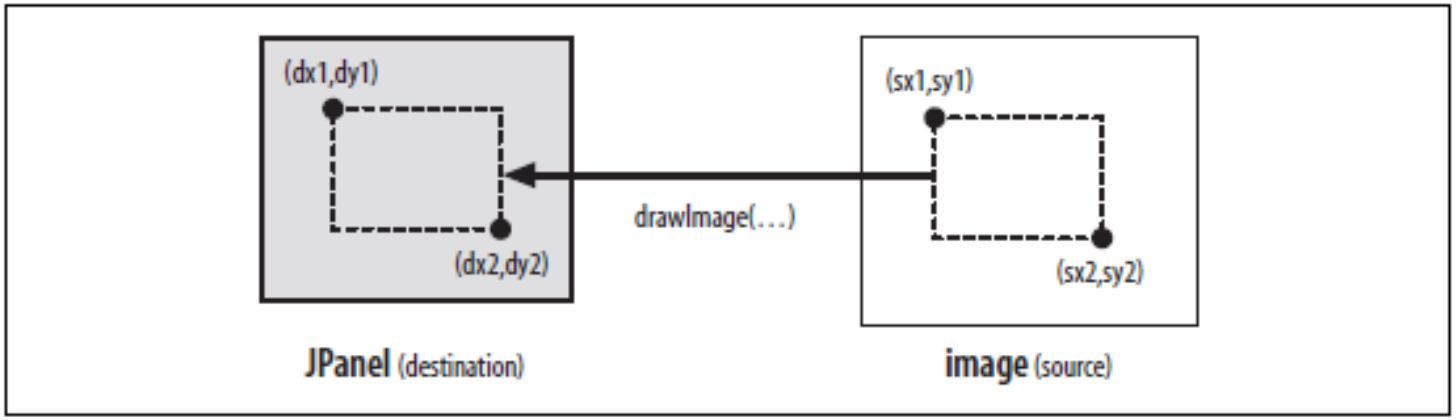
Reproducción de Sonidos

```
url = new URL("file:ball.wav");
AudioClip ac = Applet.newAudioClip(url);
ac.play();
```

Revisar paquete: Animation

Animaciones

- drawImage

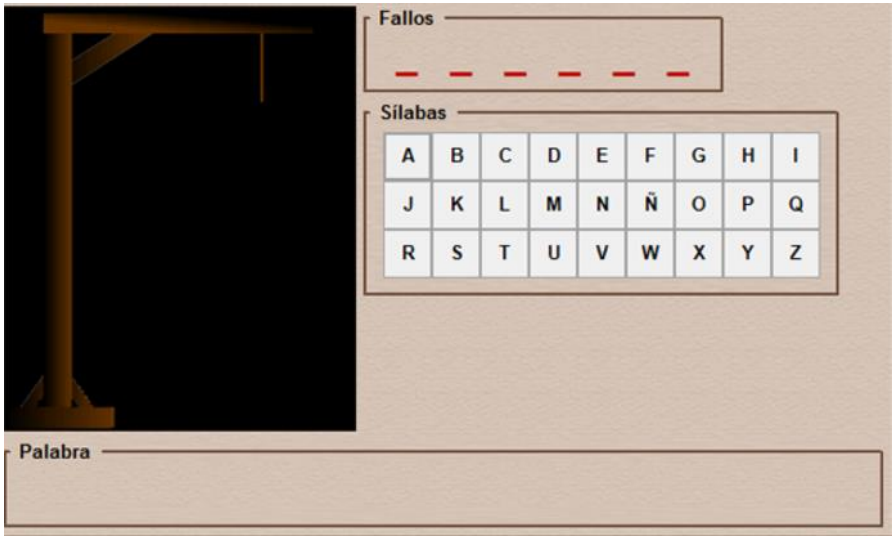


EJERCICIO 1

- Realizar un juego de memoria. Este consiste en pares de cartas que son barajadas y colocadas en un tablero. El jugador debe de levantar solo dos cartas y “memorizar” su posición para encontrar las cartas pares en un solo intento. Entre menos intentos, mayor memoria y ganas el juego.

EJERCICIO 2

- El juego del ahorcado es un clásico en el desarrollo de videojuegos. En este juego tendremos un tablero con las letras. Al igual que el juego de la vida real, seleccionaremos una letra de una palabra oculta. Si la letra coincide se ponen las letras adivinadas. En caso contrario se dibuja una parte del ahorcado hasta que es completamente dibujado.



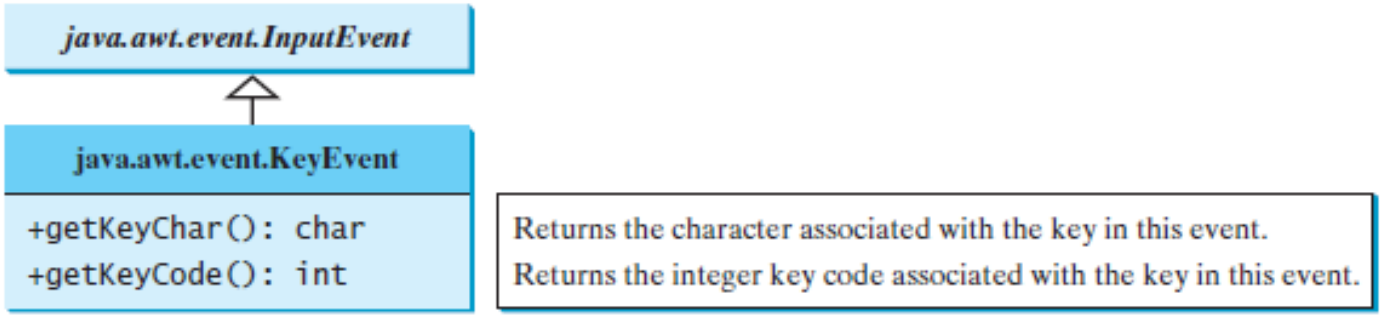
EJERCICIO 3

- Poner a volar el Ave. El ave debe despegar, volar y aterrizar.

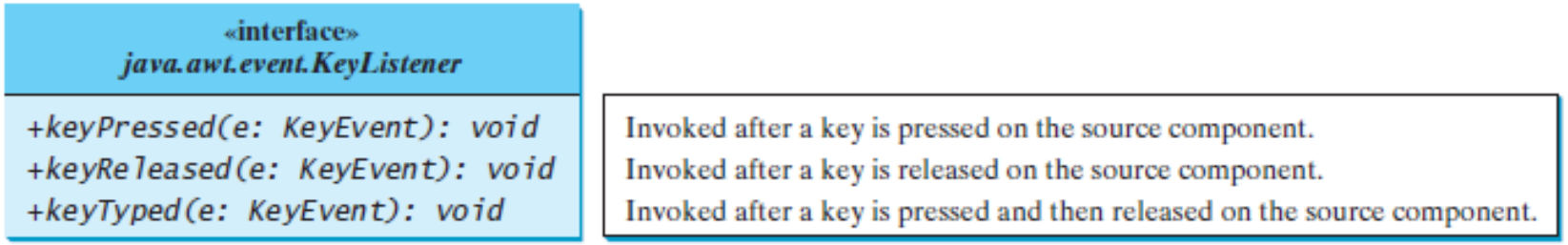
PRÓXIMA CLASE.

Gestión Eventos

- El evento KeyEvent encapsula la información de los eventos por presionar una tecla.



The `KeyEvent` class encapsulates information about key events.



Gestión Eventos

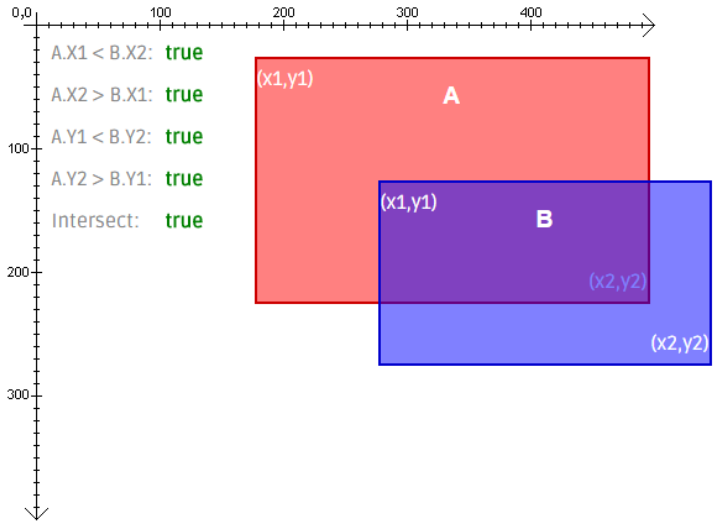
- Solo un componente **focused** puede recibir un evento **KeyEvent**. Para hacer un componente **focusable**, se debe definir su propiedad **focusable** a **true.**

- Cuando un botón se le da click, el panel pierde le focused. Invocando **canvas.requestFocusInWindow(),** resetea el focus sobre el **canvas. Así que el canvas** puede escuchar eventos del teclado.

Revisar el ejemplo: **ControCircle**, **ControlCircleWithMouseAndKey**

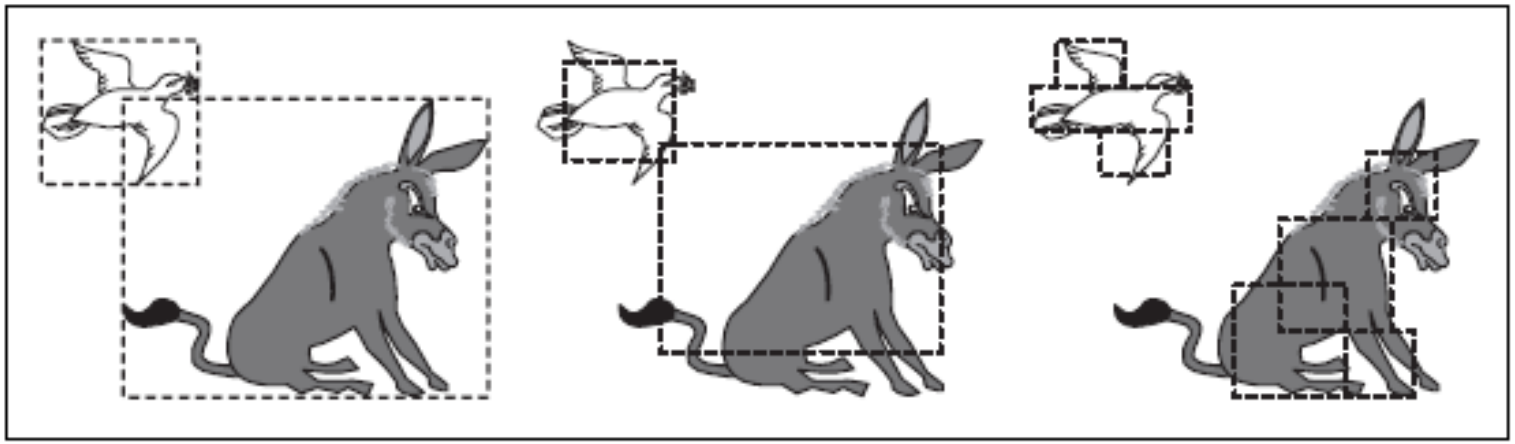
Gestión de Colisiones.

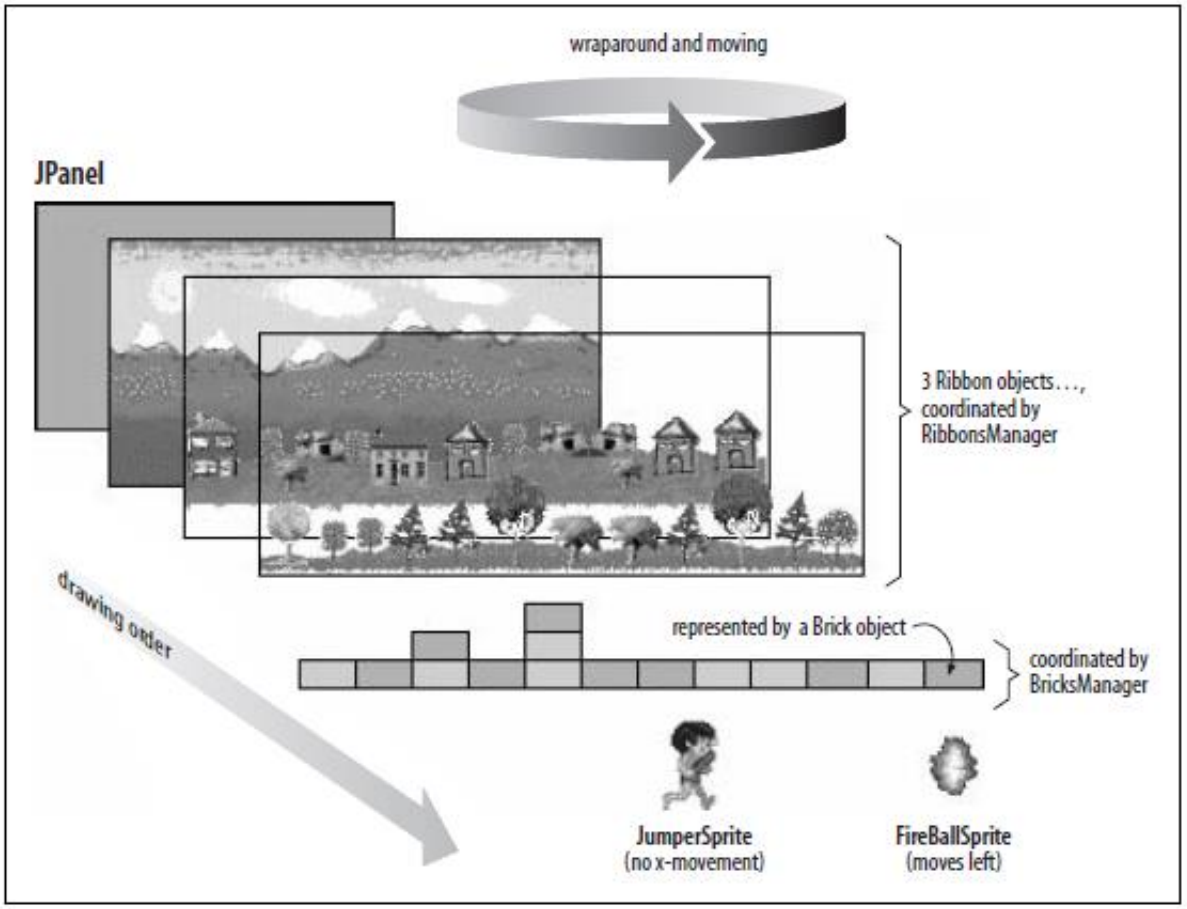
- Se presenta colisiones. Una forma sencilla de detectar colisiones se presenta cuando se intersectan dos rectángulos. En Java existe el método **intersects** en la clase Rectangle.



Revisar Game 3 Colisiones.

Gestión de Colisiones.




wraparound and moving
JPanel
3 Ribbon objects…,
coordinated by
RibbonsManager
drawing order
represented by a Brick object
coordinated by
BricksManager
JumperSprite
(no x-movement)
FireBallSprite
(moves left)

Material Adicional

- **Videos Youtube**

Detección de colisiones

https://www.youtube.com/watch?v=Otl24e_nuyc

Gravedad y Saltos.

https://www.youtube.com/watch?v=UzQSAVc4RYc

https://www.youtube.com/watch?v=qNK5B82IXvc