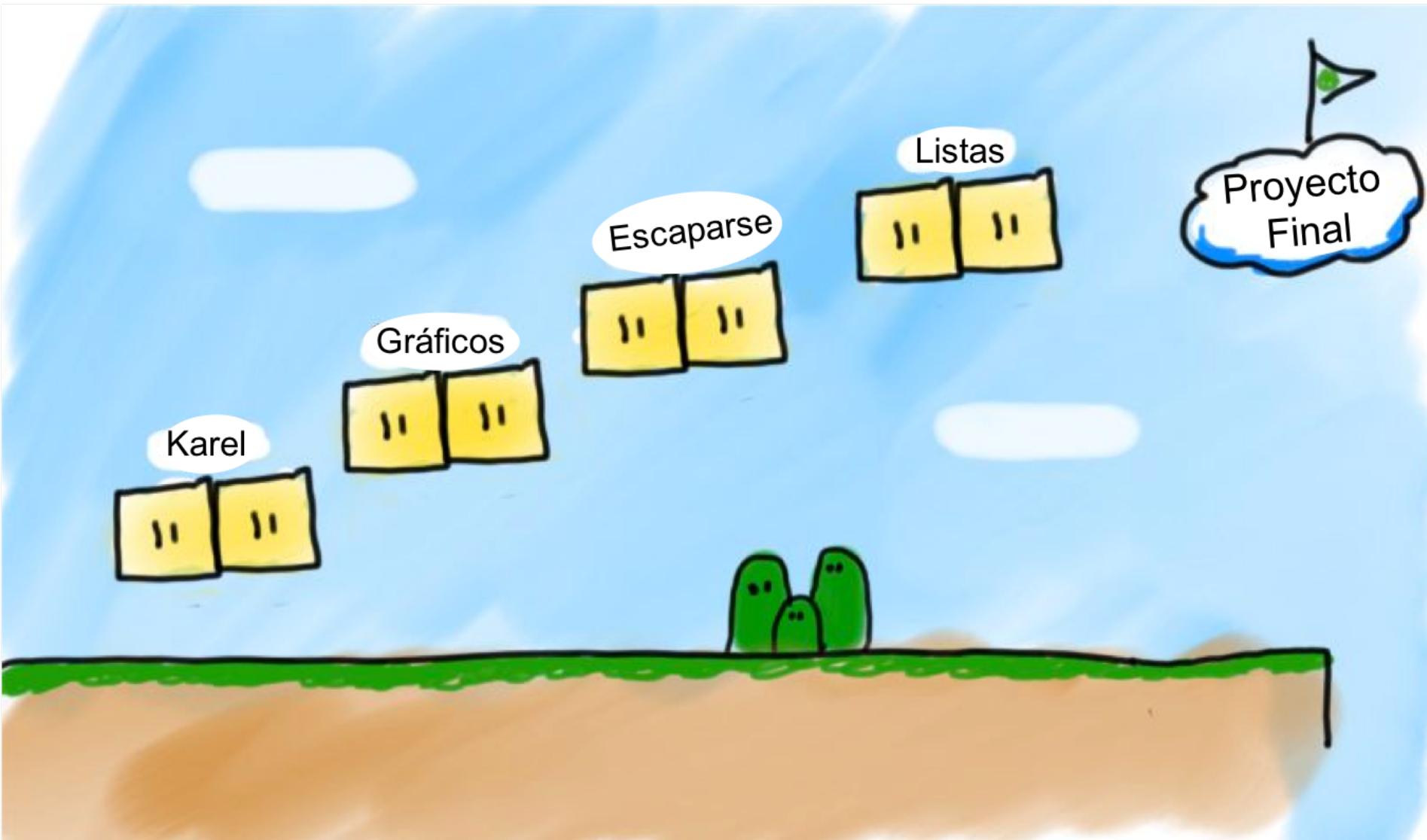




A large, energetic crowd of people is shown from behind, their hands raised in the air, suggesting they are at a concert or event. The scene is bathed in warm, golden stage lights. A blue rectangular box highlights the word "Eventos" below the seal.

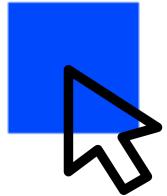
# Eventos

# ¿Dónde estamos?

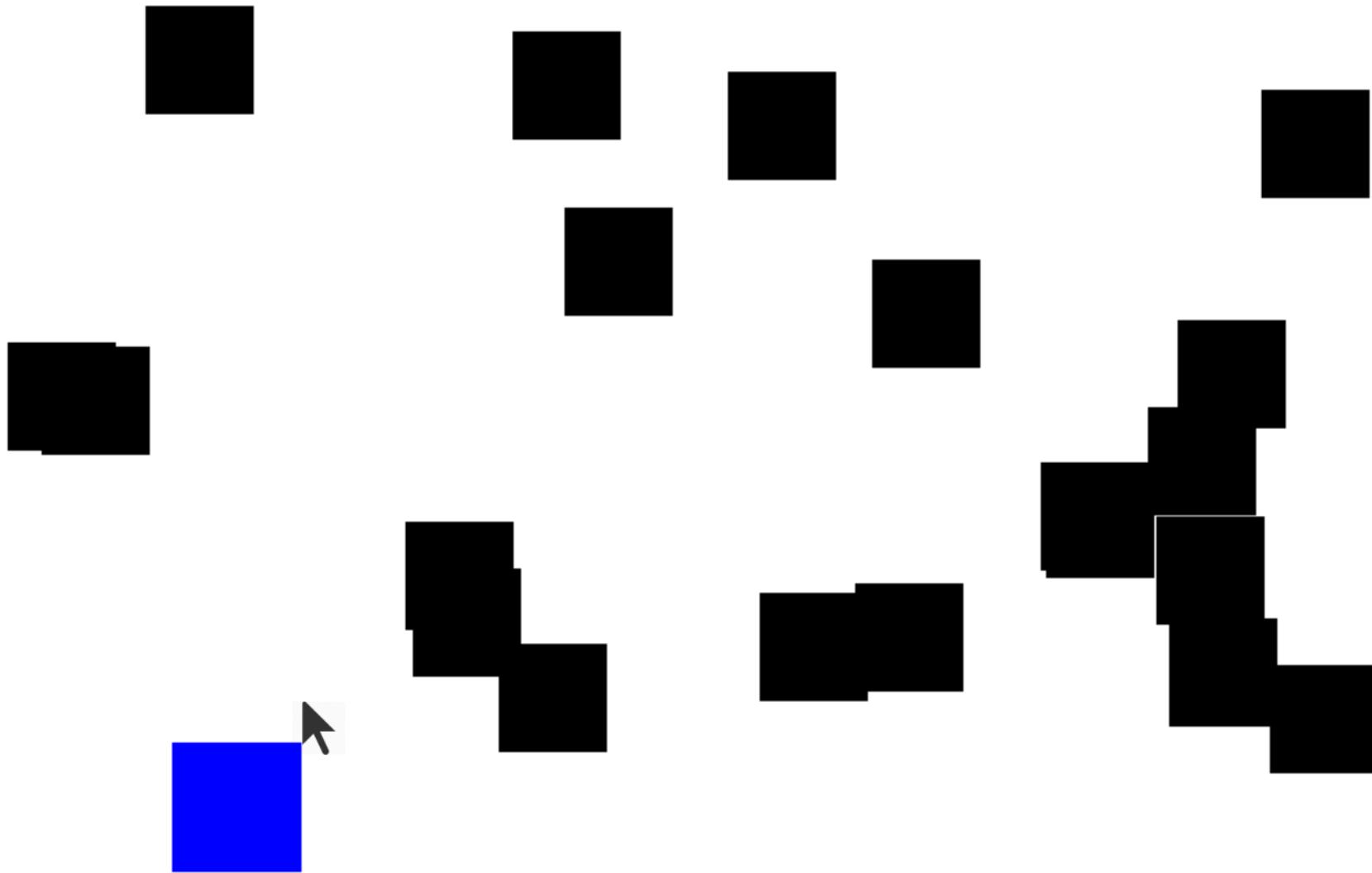


# Estampando

Estampando



# ¡Atrápame si puedes!



# Hemos ido más adelante de nosotros mismos



Source: The Hobbit

# Vamos a empezar por el inicio



Source: The Hobbit

# Objetivos de aprendizaje

1. Escribir un programa que puede responder a los eventos del mouse (ratón)
2. Utilizar una variable atributo en tu programa



# Modelo Listener

- Cuando los usuarios interactúan con el computador, generan eventos (e.g., mover/hacer clic en el mouse).
- Puede responder a eventos si tienen métodos listeners correspondientes.

**agregarMouseListeners()**

- Listeners toman el control del programa cuando sucede un evento.

# Respuestas a eventos del mouse

1. El método **run** debe llamar **agregarMouseListeners**
2. Escribir definiciones de cualquier método listener que sea necesario.

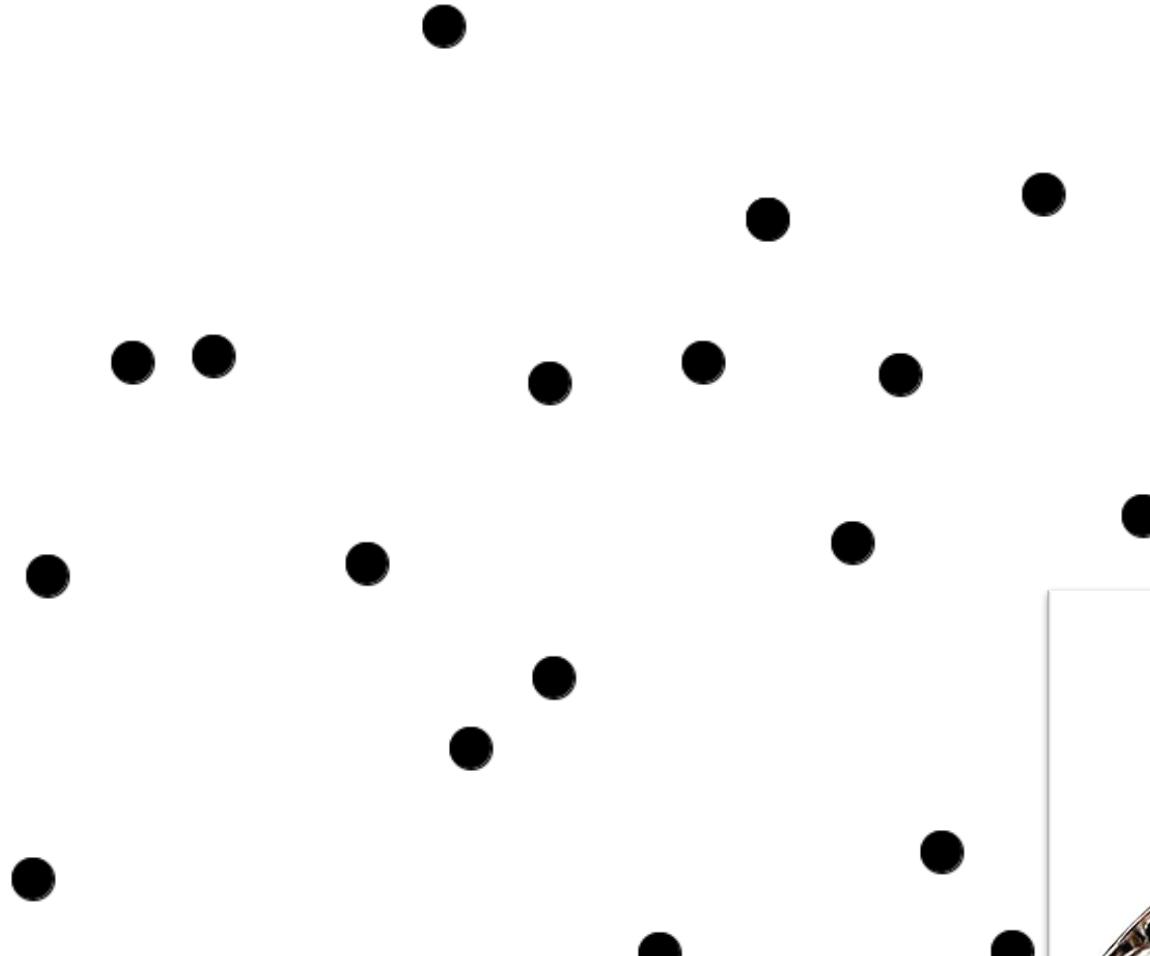
<b>mouseClickeado(<i>e</i>)</b>	Llamado cuando el usuario le da clic al mouse
<b>mousePulsado(<i>e</i>)</b>	Llamado cuando el botón del mouse está presionado
<b>mouseSoltado(<i>e</i>)</b>	Llamado cuando el botón del mouse es soltado
<b>mouseMovido(<i>e</i>)</b>	Llamado cuando el usuario mueve el mouse
<b>mouseArrastrado(<i>e</i>)</b>	Llamado cuando el mouse se mueve con el botón presionado.

El parámetro *e* es un objeto **MouseEvent**, que provee más información sobre un evento, como ubicación del mouse.

# Perforador



Perforador



Ahora bailando con niños

# Programa Normal

Método run



# Nuevos personajes listener

MouseListener



Método mouseClickeado



# El programa empieza a correr

Método run

Método mouseClickeado



# Adicionar MouseListener

Método run



Método mouseClicked



MouseListener



agregarMouseListeners();

# El programa corre normalmente

Método run



Método mouseClickeado



MouseListener



# Mouse Clickeado!

Método run



Método mouseClickeado



MouseListener



# Llama al método mouseClickeado

Método run



Método mouseClickeado



MouseListener



# Cuando termina, el método run continúa

Método run



Método mouseClicked



MouseListener



# Sigue haciendo lo suyo...

Método run



Método mouseClickeado



MouseListener



# El Mouse se movió

Método run



Método mouseClickeado



MouseListener



# Llama al método mouseClickeado

Método run



Método mouseClickeado



MouseListener



# Cuando termina, el método run continúa

Método run



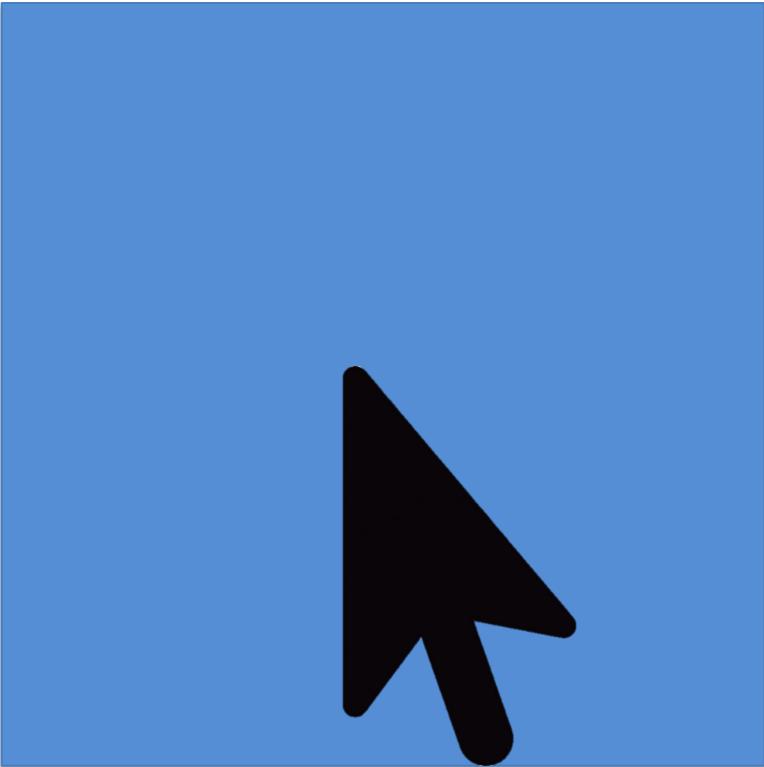
Método mouseClicked



MouseListener



# Seguimiento al Mouse



# ¿Dónde vive la casilla?

```
public class SeguidorMouse extends EsGraphics {  
  
    public void run() {  
        // SRect casilla = darCasilla(); ?  
        agregarMouseListeners();  
    }  
  
    public void mouseMovido(MouseEvento e) {  
        // SRect casilla = darCasilla(); ?  
        // mover la casilla  
    }  
}
```

# Atributos

1. Las variables salen únicamente cuando el bloque de código más interno se termina.
2. Si una variable se define por fuera de todos los métodos, el código de bloque más interno es todo el programa!
3. Estas variables se llaman **atributos**

```
public class SeguidorMouse extends EsGraphics {  
  
    /* Atributo que deja que los listeners usen la casilla */  
    SRect casilla = null;  
  
    public void run() {  
        casilla = darCasilla();  
        agregarMouseListeners();  
    }  
  
    public void mouseMovido(MouseEvent e) {  
        // mover la casilla  
    }  
}
```

\* Los atributos tiene un significado especial en programas con multiples archivos.  
Por ahora, necesitas saber que son visibles para todos los métodos y que su línea de inicialización es ejecutada antes de run.

# Atributos + Eventos

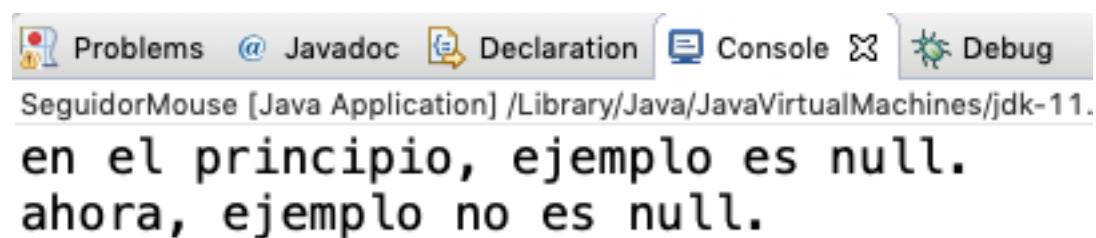
Muchas veces necesitas atributos para pasar información entre el método run y los métodos de eventos del mouse!

```
public class SeguidorMouse extends EsGraphics {  
  
    /* Atributo que deja que los listeners usen la casilla */  
    SRect casilla = null;  
  
    public void run() {  
        casilla = darCasilla();  
        agregarMouseListeners();  
    }  
  
    public void mouseMovido(MouseEvento e) {  
        int x = e.darX() - TAMANO / 2;  
        int y = e.darY() - TAMANO / 2;  
        casilla.cambiarUbicacion(x, y);  
    }  
}
```

# Null

Objetos tienen un valor especial que se llama **null**, lo que quiere decir que la variable aún no tiene ningún valor.

```
public void run() {  
    S0valo ejemplo = null;  
    if (ejemplo == null) {  
        imprimir("en el principio, ejemplo es null.");  
    }  
    ejemplo = new S0valo(5, 5);  
    if (ejemplo != null) {  
        imprimir("ahora, ejemplo no es null.");  
    }  
}
```



The screenshot shows an IDE interface with several tabs at the top: Problems, Javadoc, Declaration, Console, and Debug. The Console tab is active, displaying the following text:  
SeguidorMouse [Java Application] /Library/Java/JavaVirtualMachines/jdk-11.  
en el principio, ejemplo es null.  
ahora, ejemplo no es null.

# **darElementoEn(x, y);**

- **darElementoEn(x, y)** devuelve cualquier SObjeto en las coordenadas (x, y). Devuelve **null** si no hay ningún objeto en esas coordenadas.

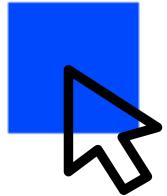
```
public void mouseClickeado(MouseEvent e) {  
    int x = e.darX();  
    int y = e.darY();  
    SObjeto objeto = darObjetoEn(x, y);  
    if (objeto != null) {  
        quitar(objeto);  
    }  
}
```

# Y aquí estamos...



# Estampando

Estampando



# Nuevos Conceptos

## Nuevos Comandos

- agregarMouseListeners();
- darElementoEn(x, y);

## Nuevas Ideas

- El modelo listener
- Atributos
- **null**

# Respuestas a eventos del mouse

1. El método **run** debe llamar **agregarMouseListeners**
2. Escribir definiciones de cualquier método listener que sea necesario.

<b>mouseClickeado(<i>e</i>)</b>	Llamado cuando el usuario le da clic al mouse
<b>mousePulsado(<i>e</i>)</b>	Llamado cuando el botón del mouse está presionado
<b>mouseSoltado(<i>e</i>)</b>	Llamado cuando el botón del mouse es soltado
<b>mouseMovido(<i>e</i>)</b>	Llamado cuando el usuario mueve el mouse
<b>mouseArrastrado(<i>e</i>)</b>	Llamado cuando el mouse se mueve con el botón presionado.

El parámetro *e* es un objeto **MouseEvent**, que provee más información sobre un evento, como ubicación del mouse.

# Respuestas a eventos del teclado

1. The **run** method should call **addKeyListeners**
2. Write definitions of any listener methods needed

<b>keyPressed(<i>e</i>)</b>	Called when the user presses a key
<b>keyReleased(<i>e</i>)</b>	Called when the key comes back up
<b>keyTyped(<i>e</i>)</b>	Called when the user types (presses and releases) a key

The parameter *e* is a **KeyEvent** object, which indicates which key is involved.

# Haciendo Pistas



# ¡Atrápame si puedes!

