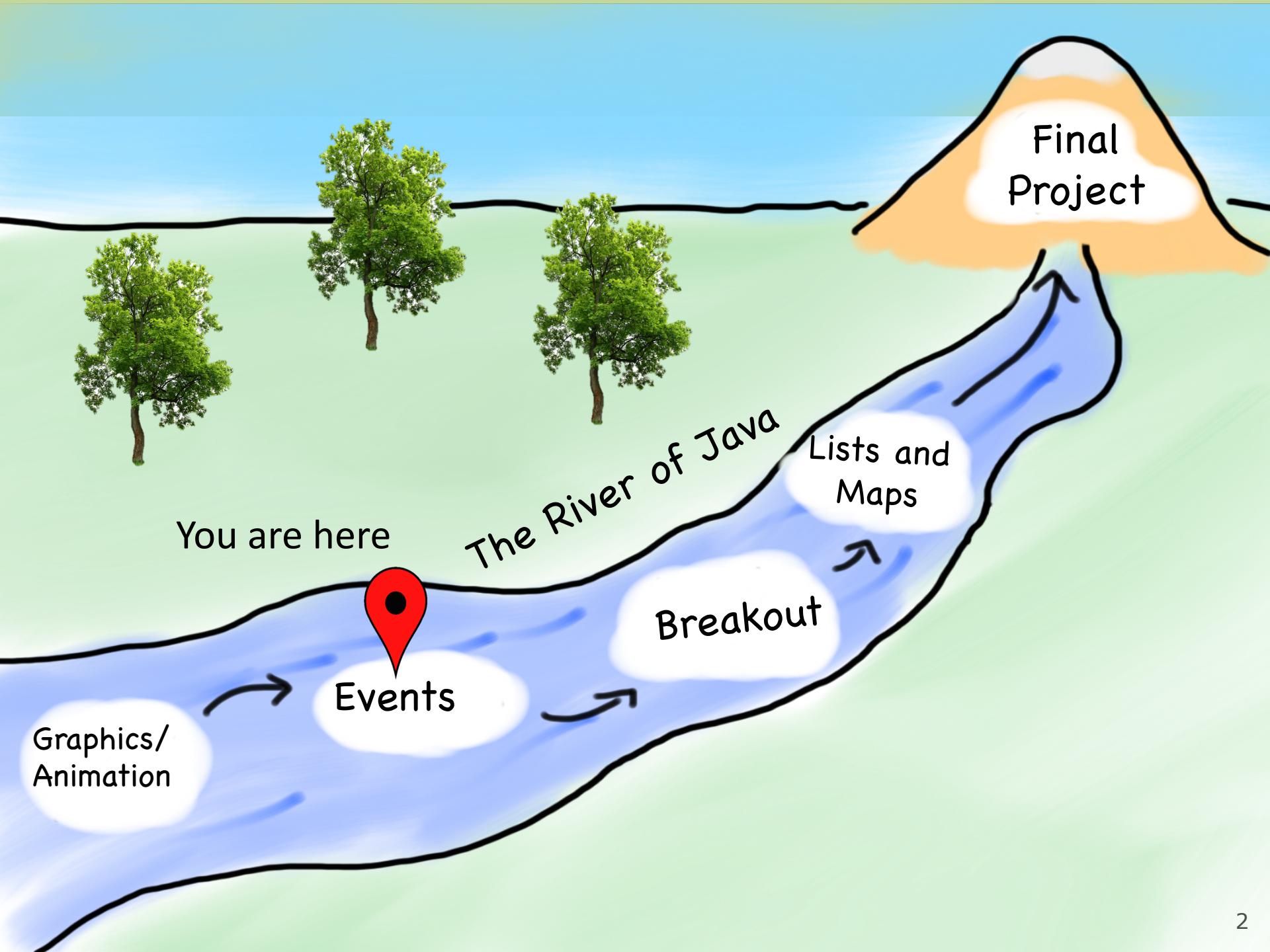
A photograph of a concert crowd from behind, showing many people with their hands raised. The scene is lit with warm, orange-yellow stage lights, creating a festive atmosphere. A blue rectangular frame highlights a central area where the word "Events" is placed.

Events



Learning Goals

- Learn to respond to mouse events in **GraphicsPrograms**
- Learn to use *instance variables* to store information outside of methods



Plan for Today

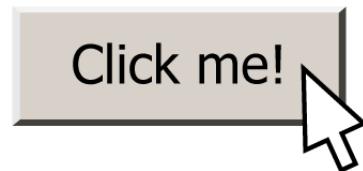
- Event-driven programming
- *Demo:* Click for Daisies
- *Demo:* Doodler
- Instance Variables
- **null** and **getElementAt**
- *Demo:* Whack-a-Mole

Plan for Today

- **Event-driven programming**
- *Demo:* Click for Daisies
- *Demo:* Doodler
- Instance Variables
- null and getElementAt
- *Demo:* Rubbish Sweeper

Events

- **event**: Some external stimulus that your program can respond to.



- **event-driven programming**: A coding style (common in graphical programs) where your code is executed in response to user events.

Events

- Program launches

Events

- Program launches
- Mouse motion
- Mouse clicking
- Keyboard keys pressed
- Device rotated
- Device moved
- GPS location changed
- and more...

Events

- Program launches
- Mouse motion
- Mouse clicking
- Keyboard keys pressed
- Device rotated
- Device moved
- GPS location changed
- and more...

Events

```
public void run() {  
    // Java runs this when program launches  
}
```

Events

```
public void run() {  
    // Java runs this when program launches  
}  
  
public void mouseClicked(MouseEvent event) {  
    // Java runs this when mouse is clicked  
}
```

Events

```
public void run() {  
    // Java runs this when program launches  
}
```

```
public void mouseClicked(MouseEvent event) {  
    // Java runs this when mouse is clicked  
}
```

```
public void mouseMoved(MouseEvent event) {  
    // Java runs this when mouse is moved  
}
```

Example: ClickForDaisy

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;      // NEW

public class ClickForDaisy extends GraphicsProgram {

    public void run() {
        addMouseListeners();
    }

    // Add a Daisy image at 50, 50 on mouse click
    public void mouseClicked(MouseEvent event) {
        GImage daisy = new GImage("res/daisy.png", 50, 50);
        add(daisy);
    }
}
```

addMouseListeners()

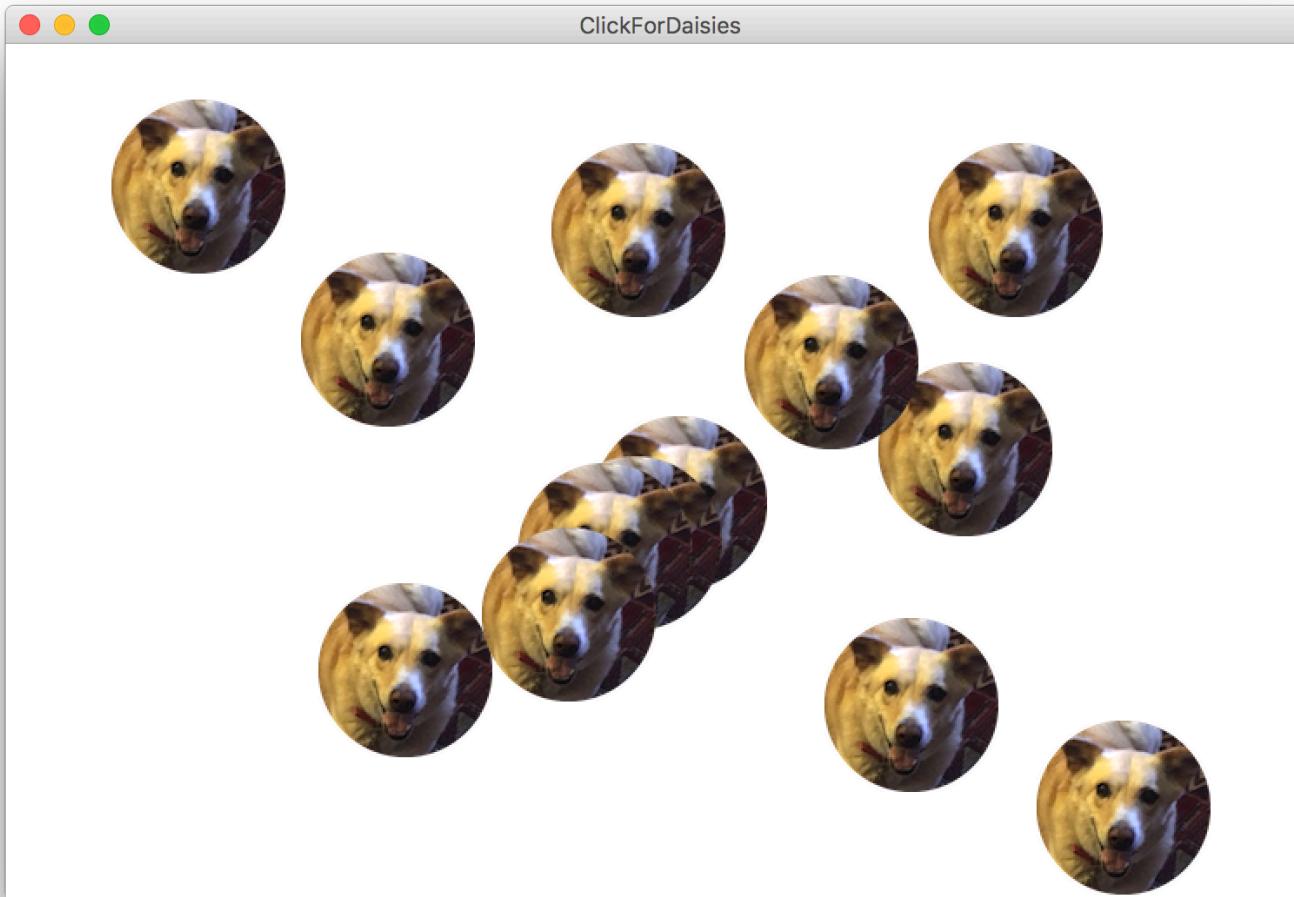
- You must call addMouseListeners() in your run method if you want to respond to mouse events in your program.
- After this is called, your program will start “listening” for mouse events.

MouseEvent Objects

- A MouseEvent contains information about the event that just occurred:

Method	Description
<code>e.getX()</code>	the x-coordinate of mouse cursor in the window
<code>e.getY()</code>	the y-coordinate of mouse cursor in the window

Example: ClickForDaisies



Example: ClickForDaisies

```
public class ClickForDaisies extends GraphicsProgram {  
  
    // Add a Daisy image where the user clicks  
    public void mouseClicked(MouseEvent event) {  
        // Get information about the event  
        double mouseX = event.getX();  
        double mouseY = event.getY();  
  
        // Add Daisy at the mouse location  
        GImage daisy = new GImage("res/daisy.png", mouseX, mouseY);  
        add(daisy);  
    }  
}
```

Example: ClickForDaisies

```
public class ClickForDaisies extends GraphicsProgram {  
  
    // Add a Daisy image where the user clicks  
    public void mouseClicked(MouseEvent event) {  
        // Get information about the event  
        double mouseX = event.getX();  
        double mouseY = event.getY();  
  
        // Add Daisy at the mouse location  
        GImage daisy = new GImage("res/daisy.png", mouseX, mouseY);  
        add(daisy);  
    }  
}
```

Types of Mouse Events

- There are many different types of mouse events.

- Each takes the form:

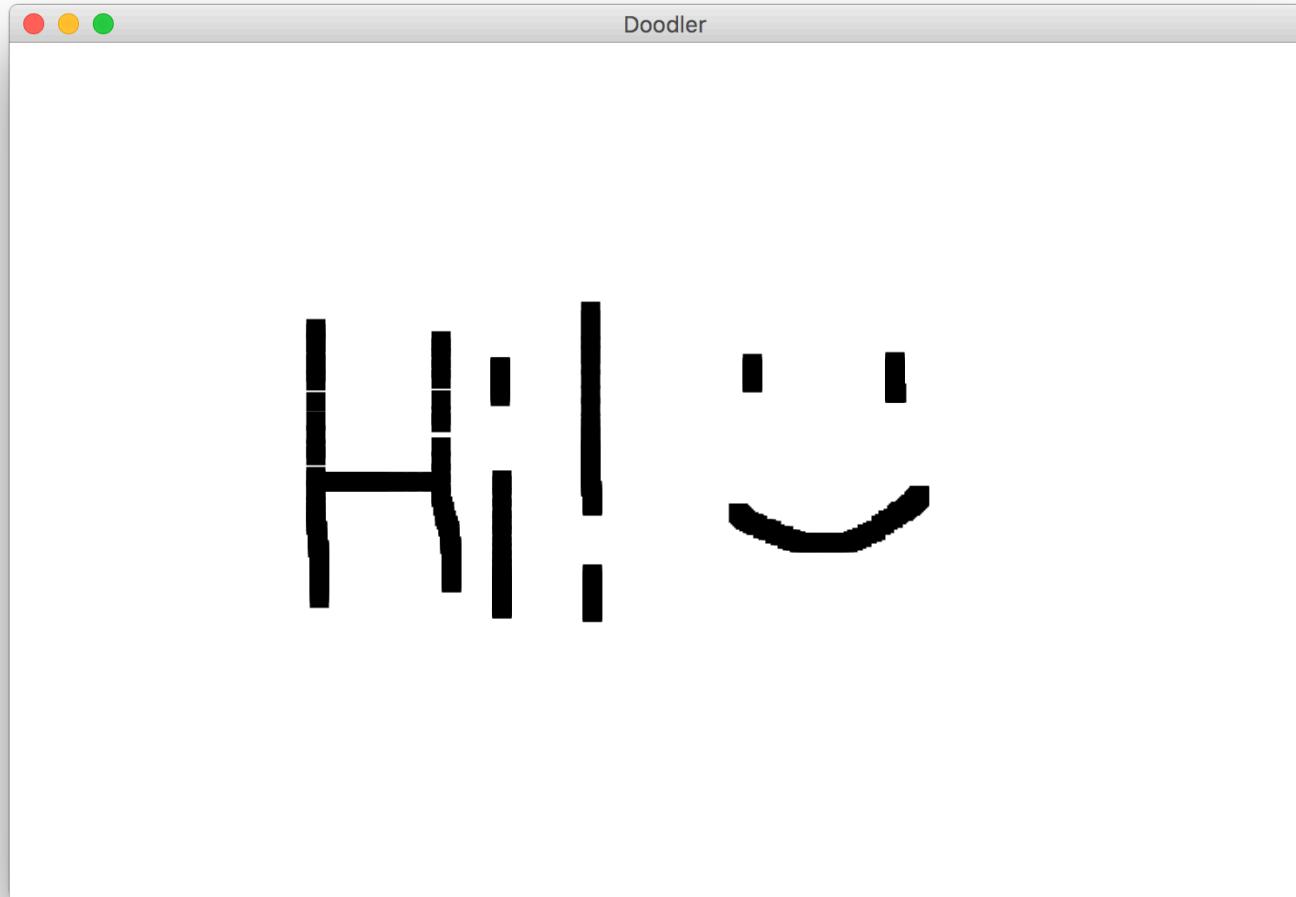
```
public void eventMethodName(MouseEvent event) { ... }
```

Method	Description
mouseMoved	mouse cursor moves
mouseDragged	mouse cursor moves while button is held down
mousePressed	mouse button is pressed down
mouseReleased	mouse button is lifted up
mouseClicked	mouse button is pressed and then released
mouseEntered	mouse cursor enters your program's window
mouseExited	mouse cursor leaves your program's window

Plan for Today

- Event-driven programming
- *Demo:* Click for Daisies
- ***Demo: Doodler***
- Instance Variables
- `null` and `getElementAt`
- *Demo:* Whack-a-Mole

Coding Together: Doodler



Doodler

```
private static final int SIZE = 10;  
...  
  
public void mouseDragged(MouseEvent event) {  
    double mouseX = event.getX();  
    double mouseY = event.getY();  
    double rectX = mouseX - SIZE / 2.0;  
    double rectY = mouseY - SIZE / 2.0;  
    GRect rect = new GRect(rectX, rectY, SIZE, SIZE);  
    rect.setFilled(true);  
    add(rect);  
}
```

Doodler

```
public void mouseDragged(MouseEvent event) {  
    double mouseX = event.getX();  
    double mouseY = event.getY();  
    double rectX = mouseX - SIZE / 2.0;  
    double rectY = mouseY - SIZE / 2.0;  
    GRect rect = new GRect(rectX, rectY, SIZE, SIZE);  
    rect.setFilled(true);  
    add(rect);  
}
```

Doodler

```
public void mouseDragged(MouseEvent event) {  
    double mouseX = event.getX();  
    double mouseY = event.getY();  
    double rectX = mouseX - SIZE / 2.0;  
    double rectY = mouseY - SIZE / 2.0;  
    GRect rect = new GRect(rectX, rectY, SIZE, SIZE);  
    rect.setFilled(true);  
    add(rect);  
}
```

Doodler

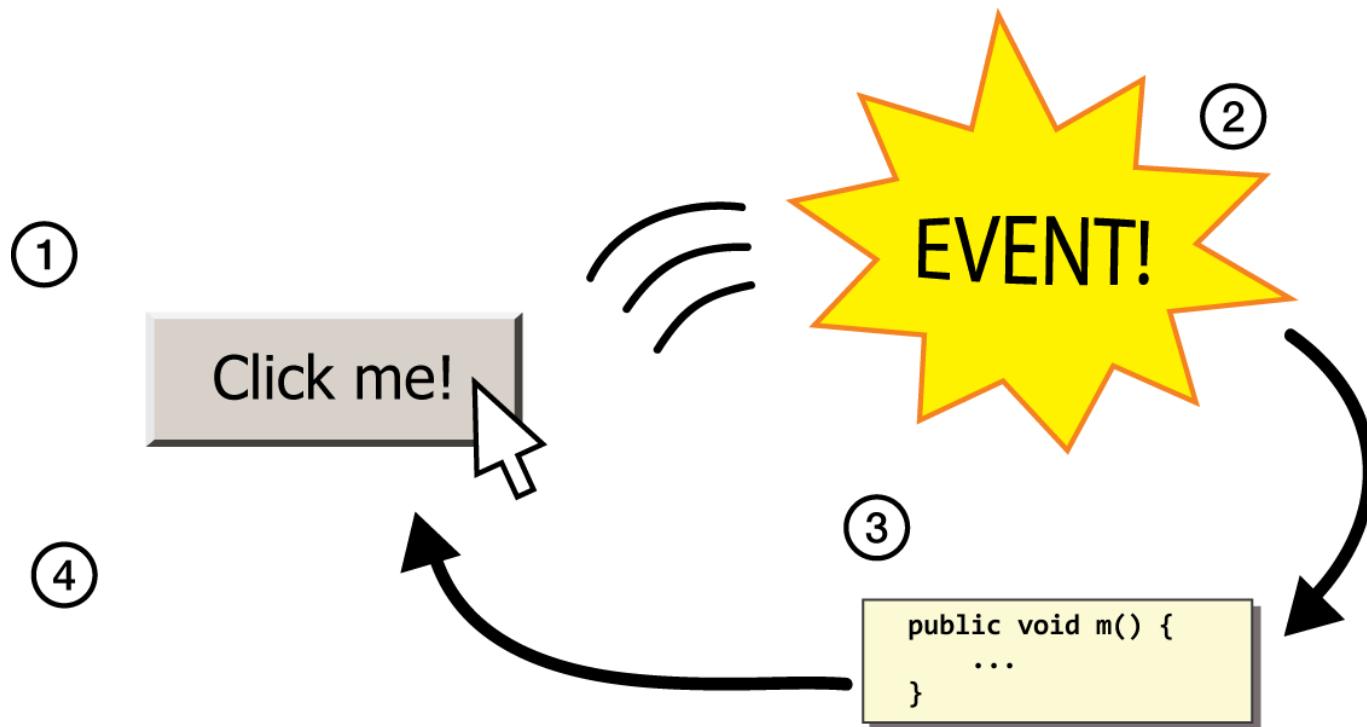
```
public void mouseDragged(MouseEvent event) {  
    double mouseX = event.getX();  
    double mouseY = event.getY();  
double rectX = mouseX - SIZE / 2.0;  
double rectY = mouseY - SIZE / 2.0;  
    GRect rect = new GRect(rectX, rectY, SIZE, SIZE);  
    rect.setFilled(true);  
    add(rect);  
}
```

Doodler

```
public void mouseDragged(MouseEvent event) {  
    double mouseX = event.getX();  
    double mouseY = event.getY();  
    double rectX = mouseX - SIZE / 2.0;  
    double rectY = mouseY - SIZE / 2.0;  
    GRect rect = new GRect(rectX, rectY, SIZE, SIZE);  
    rect.setFilled(true);  
    add(rect);  
}
```

Recap: Events

- 1) User performs some action, like moving / clicking the mouse.
- 2) This causes an event to occur.
- 3) Java executes a particular method to handle that event.
- 4) The method's code updates the screen appearance in some way.



Revisiting Doodler

```
public void mouseDragged(MouseEvent event) {  
    double mouseX = event.getX();  
    double mouseY = event.getY();  
    double rectX = mouseX - SIZE / 2.0;  
    double rectY = mouseY - SIZE / 2.0;  
    GRect rect = new GRect(rectX, rectY, SIZE, SIZE);  
    rect.setFilled(true);  
    add(rect);  
}
```

What if we wanted the *same* GRect to track the mouse, instead of making a new one each time?

Plan for Today

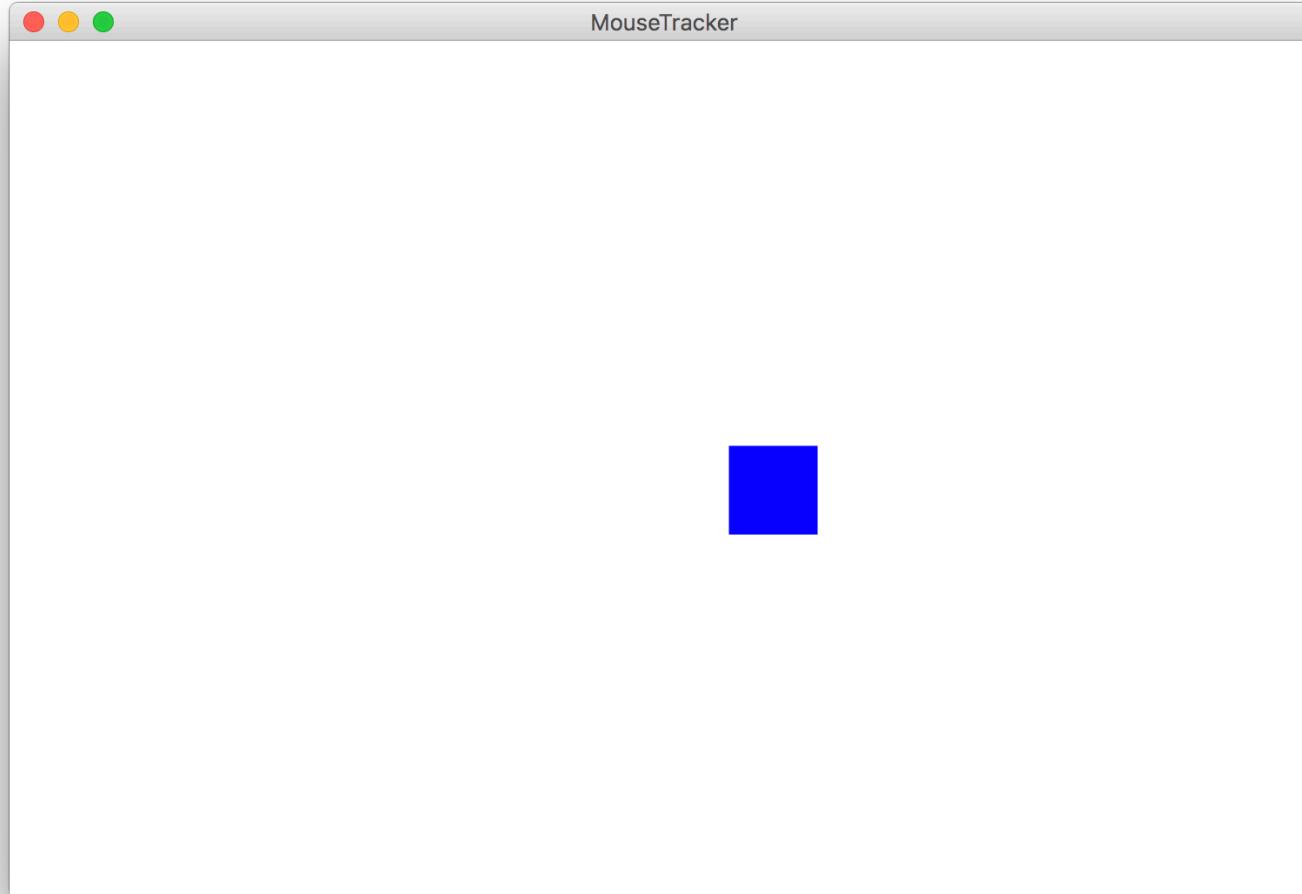
- Event-driven programming
- *Demo:* Click for Daisies
- *Demo:* Doodler
- **Instance Variables**
- null and getElementAt
- *Demo:* Whack-a-Mole

Instance Variables

```
private type name; // declared outside of any method
```

- **Instance variable:** A variable that lives outside of any method.
 - The *scope* of an instance variable is throughout an entire file (class).
 - Useful for data that must persist throughout the program, or that cannot be stored as local variables or parameters (event handlers).
 - *It is bad style to overuse instance variables*

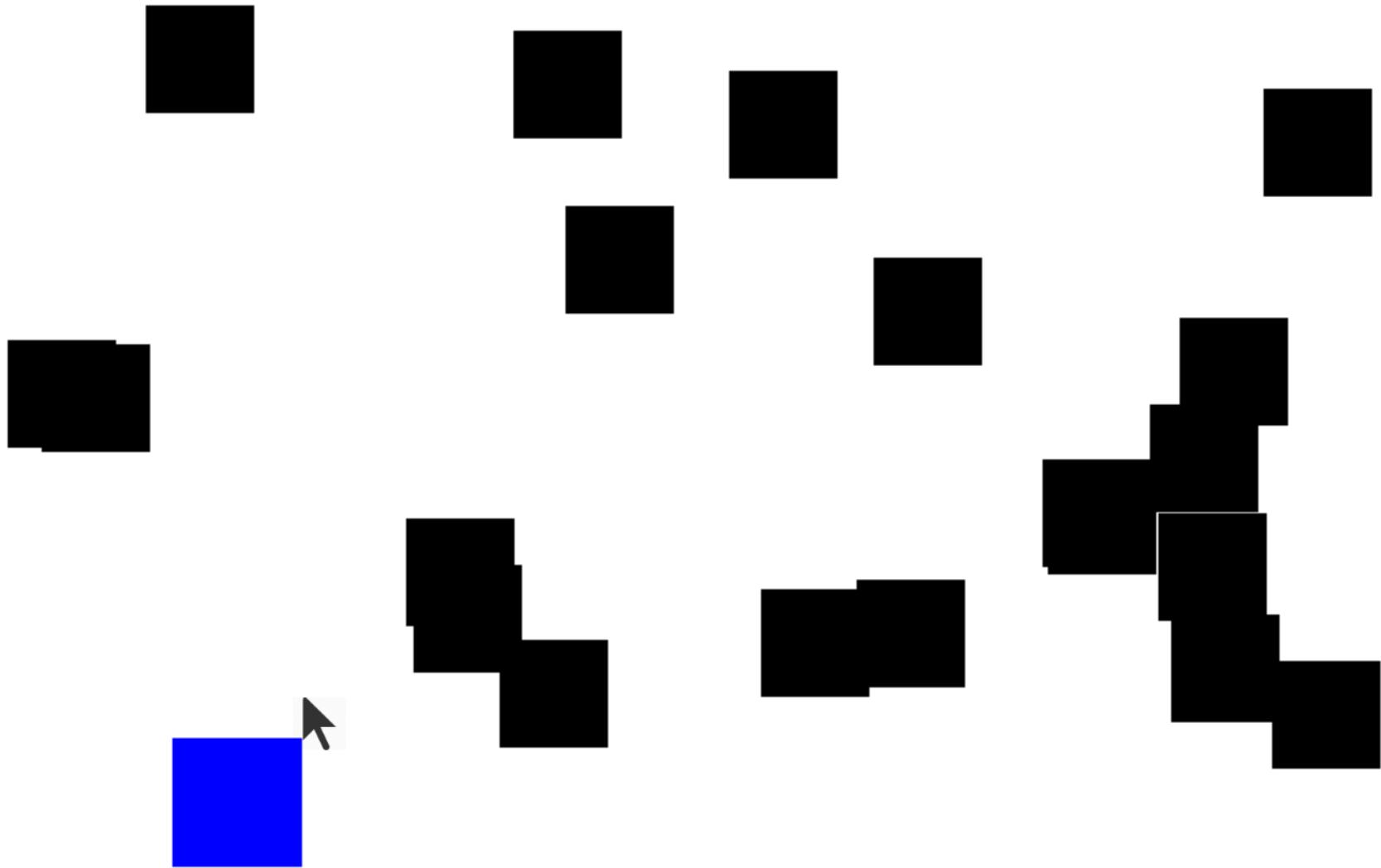
Example: MouseTracker



Plan for Today

- Event-driven programming
- *Demo:* Click for Daisies
- *Demo:* Doodler
- Instance Variables
- **null** and **getElementAt**
- *Demo:* Whack-a-Mole

getElementAt



getElementAt

getElementAt returns the object at this location on the canvas

```
GObject objectHere = getElementAt(x, y);
```

getElementAt

getElementAt returns the object at this location on the canvas

```
GObject objectHere = getElementAt(x, y);  
if (objectHere != null) {  
    // do something with objectHere  
} else {  
    // null – nothing at that location  
}
```

Null

Null is a special variable value that objects can have that means “nothing”. Primitives cannot be null.

If a method returns an object, it can return **null** to signify “nothing”. (just say **return null;**)

```
// may be a GObject, or null if nothing at (x, y)
GObject maybeAnObject = getElementAt(x, y);
```

Null

You can check if something is null using == and !=.

```
// may be a GObject, or null if nothing at (x, y)
GObject maybeAnObject = getElementAt(x, y);
if (maybeAnObject != null) {
    // do something with maybeAnObject
} else {
    // null - nothing at that location
}
```

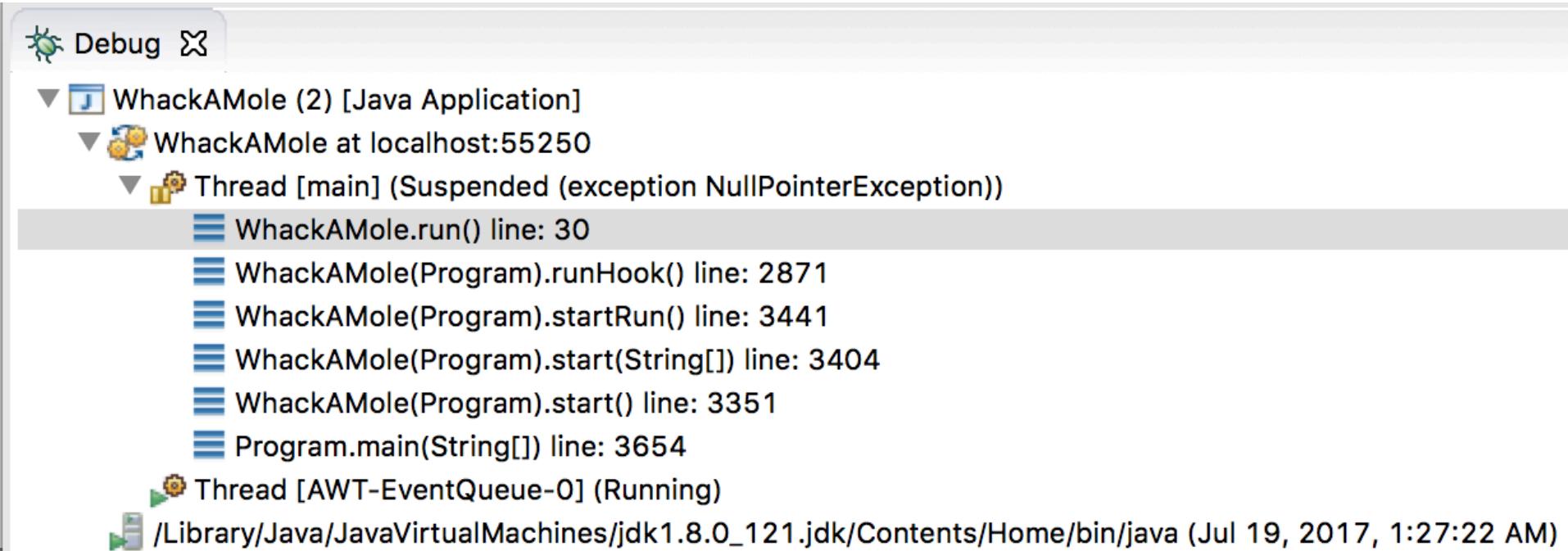
Null

Calling methods on an object that is **null** will crash your program!

```
// may be a GObject, or null if nothing at (x, y)
GObject maybeAnObject = getElementAt(x, y);
if (maybeAnObject != null) {
    int x = maybeAnObject.getX(); // OK
} else {
    int x = maybeAnObject.getX(); // CRASH!
}
```

Null

Calling methods on an object that is **null** will crash your program! (throws a NullPointerException)



Putting it all together



Whack-A-Mole

Let's make Whack-A-Mole!

- Moles should initially appear at random locations on the screen
- If the user clicks a mole, remove it



Whack-A-Mole

Let's add to our program by continuously adding more moles as the game plays.



Normal Program

Run Method



Normal Program

Run Method



```
public void run() {  
    while(true) {  
        update();  
        pause(DELAY);  
    }  
}
```

Normal Program

Run Method



```
public void run() {  
    while(true) {  
        update();  
        pause(DELAY);  
    }  
}
```

Normal Program

Run Method



```
public void run() {  
    while(true) {  
        update();  
        pause(DELAY);  
    }  
}
```

Normal Program

Run Method



```
public void run() {  
    while(true) {  
        update();  
        pause(DELAY);  
    }  
}
```

Normal Program

Run Method



```
public void run() {  
    while(true) {  
        update();  
        pause(DELAY);  
    }  
}
```

Normal Program

Run Method



New Listener Characters

Mouse Listener



Mouse Moved Method



Program with a Mouse Method

Run Method

Mouse Moved Method



Program Starts Running

Run Method

Mouse Moved Method



Add Mouse Listener

Run Method



Mouse Moved Method



Mouse Listener



addMouseListeners();

Program Runs as Usual

Run Method



Mouse Moved Method



Mouse Listener



Mouse Moved!

Run Method



Mouse Moved Method



Mouse Listener



Calls Mouse Moved Method

Run Method



Mouse Moved Method



Mouse Listener



When done, Run continues.

Run Method



Mouse Moved Method



Mouse Listener



Keeps Doing Its Thing...

Run Method



Mouse Moved Method



Mouse Listener



Mouse Moved!

Run Method



Mouse Moved Method



Mouse Listener



Calls Mouse Moved Method

Run Method



Mouse Moved Method



Mouse Listener



When done, Run continues.

Run Method



Mouse Moved Method



Mouse Listener



Recap

- Event-driven programming
- *Demo:* Click for Daisies
- *Demo:* Doodler
- Instance Variables
- **null** and **getElementAt**
- *Demo:* Whack-a-Mole