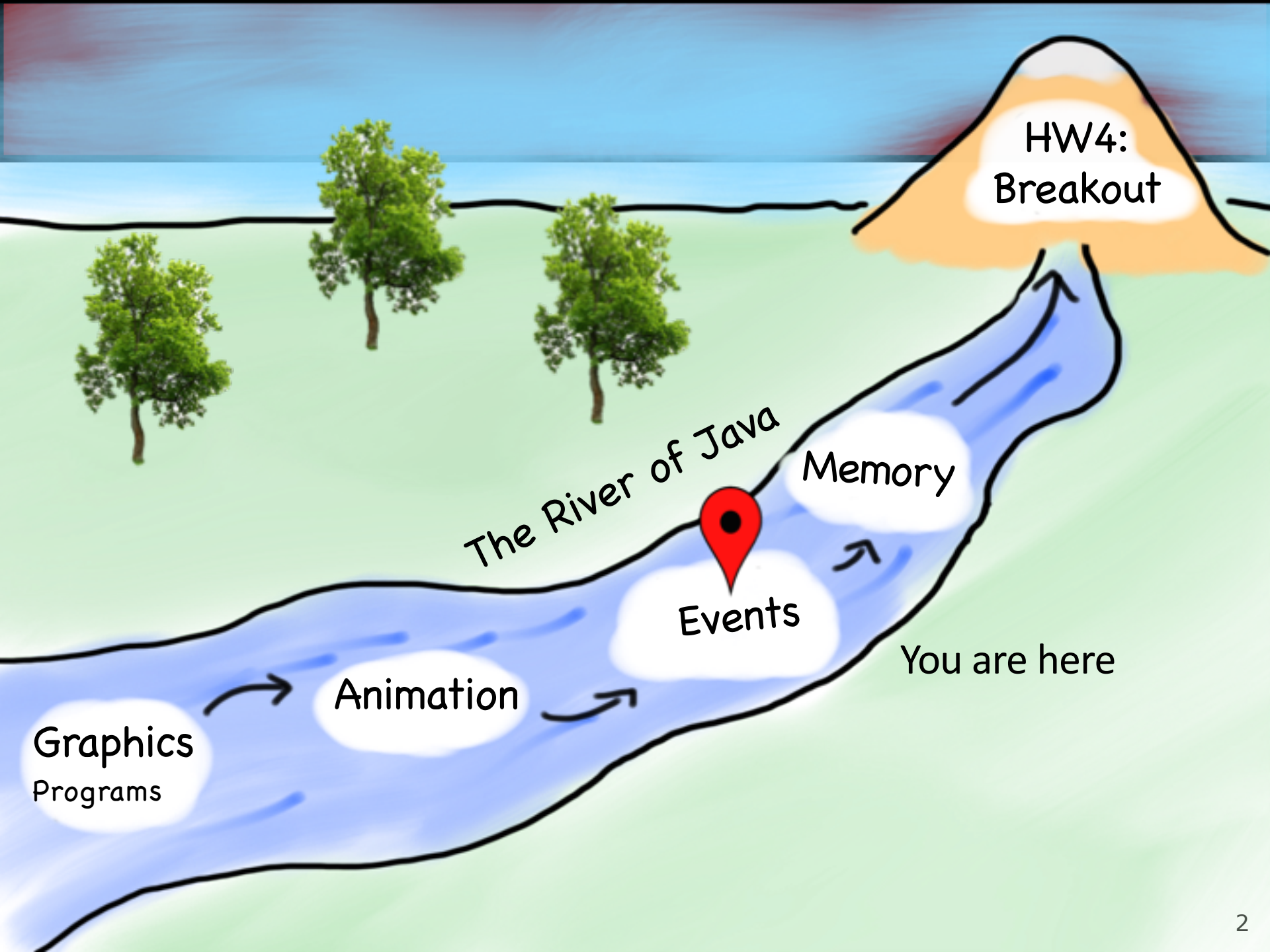


CS 106A, Lecture 14

Events and Instance Variables

Reading:

Art & Science of Java, Ch. 10.1-10.4



Plan for Today

- Announcements
- Review: animation and null
- Event-driven programming
- Instance variables
- Whack-A-Mole
- Extra: Googly Eyes

Announcements

- BlueBook download available on the website
 - try it out with the second practice exam!
- Assignment 3 due tomorrow
- Assignment 4 out tomorrow
 - not due until the following Monday, but could be good exam practice for graphics + events

Plan for Today

- Announcements
- **Review: animation and null**
- Event-driven programming
- Instance variables
- Whack-A-Mole
- Extra: Googly Eyes

Animation

- A Graphics program can be made to animate with a loop such as:

```
public void run() {  
    ...  
    while (test) {  
        update the position of shapes;  
        pause(milliseconds);  
    }  
  
}
```

- The best number of ms to pause depends on the program.
 - most video games \sim 50 frames/sec = 25ms pause

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);
```

Objects have the value **null** before being initialized.

```
Scanner myScanner; // initially null
```

Null

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

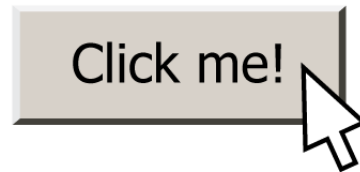
```
// may be a GObject, or null if nothing at (x, y)
GObject maybeAnObject = getElementAt(x, y);
if (maybeAnObject != null) {
    int x = maybeAnObject.getX(); // OK
} else {
    int x = maybeAnObject.getX(); // CRASH!
}
```


Plan for Today

- Announcements
- Review: animation and null
- **Event-driven programming**
- Instance variables
- Whack-A-Mole
- Extra: Googly Eyes

Events

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



- **event-driven programming:** A coding paradigm (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 {

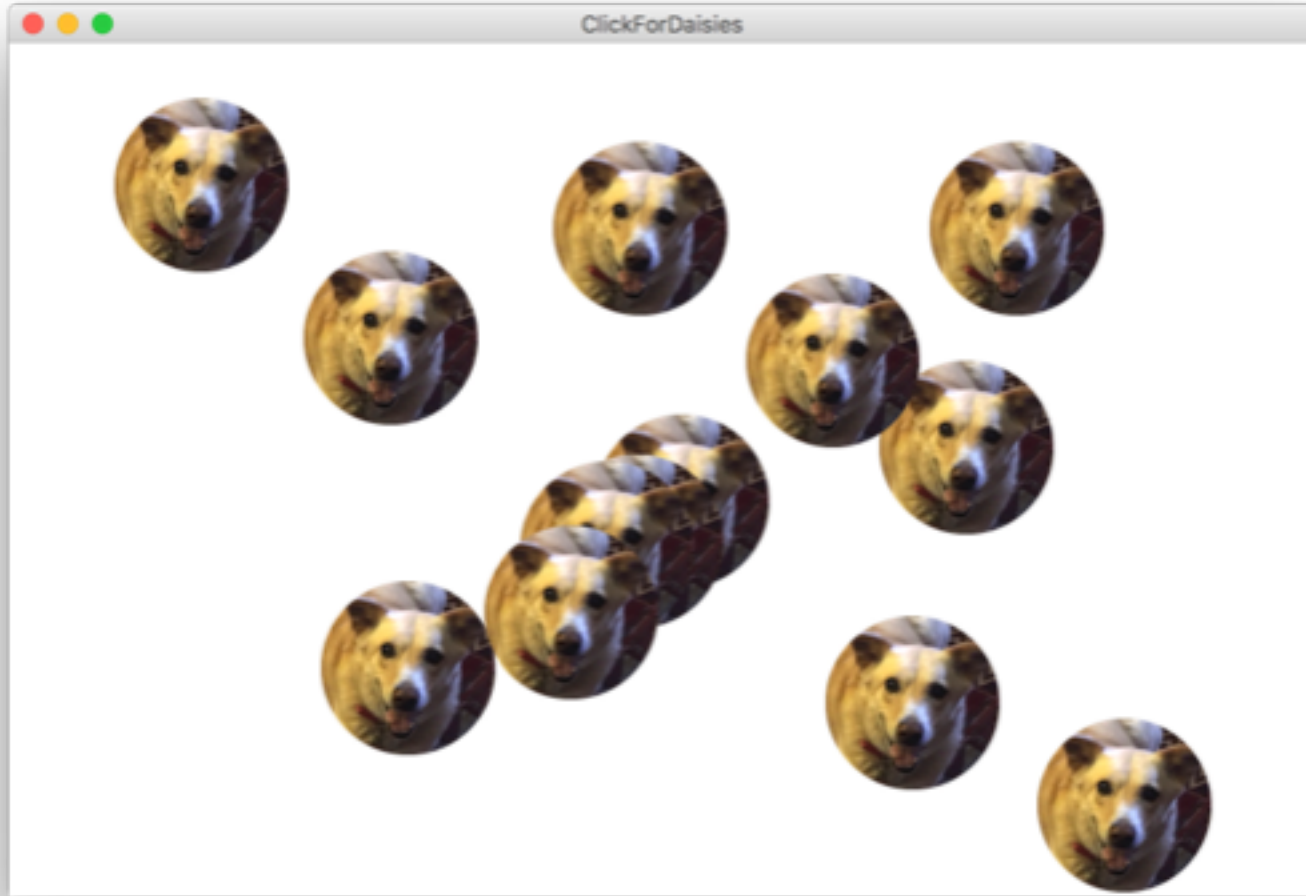
    // Add a Daisy image at 50, 50 on mouse click
    public void mouseClicked(MouseEvent event) {
        GImage daisy = new GImage("res/daisy.png", 50, 50);
        add(daisy);
    }
}
```

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

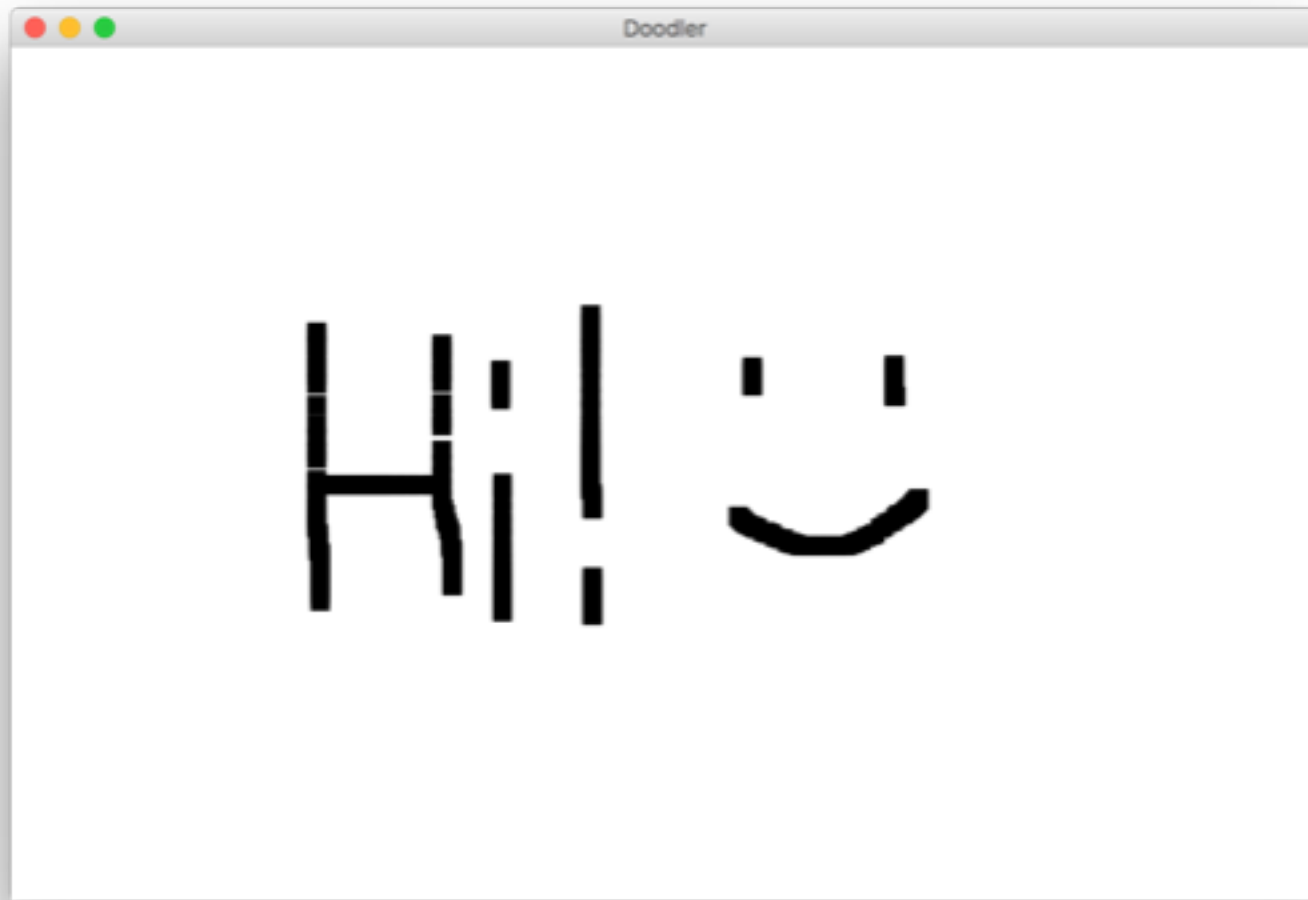
- There are many different types of mouse events.

- 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    |

# Example: 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

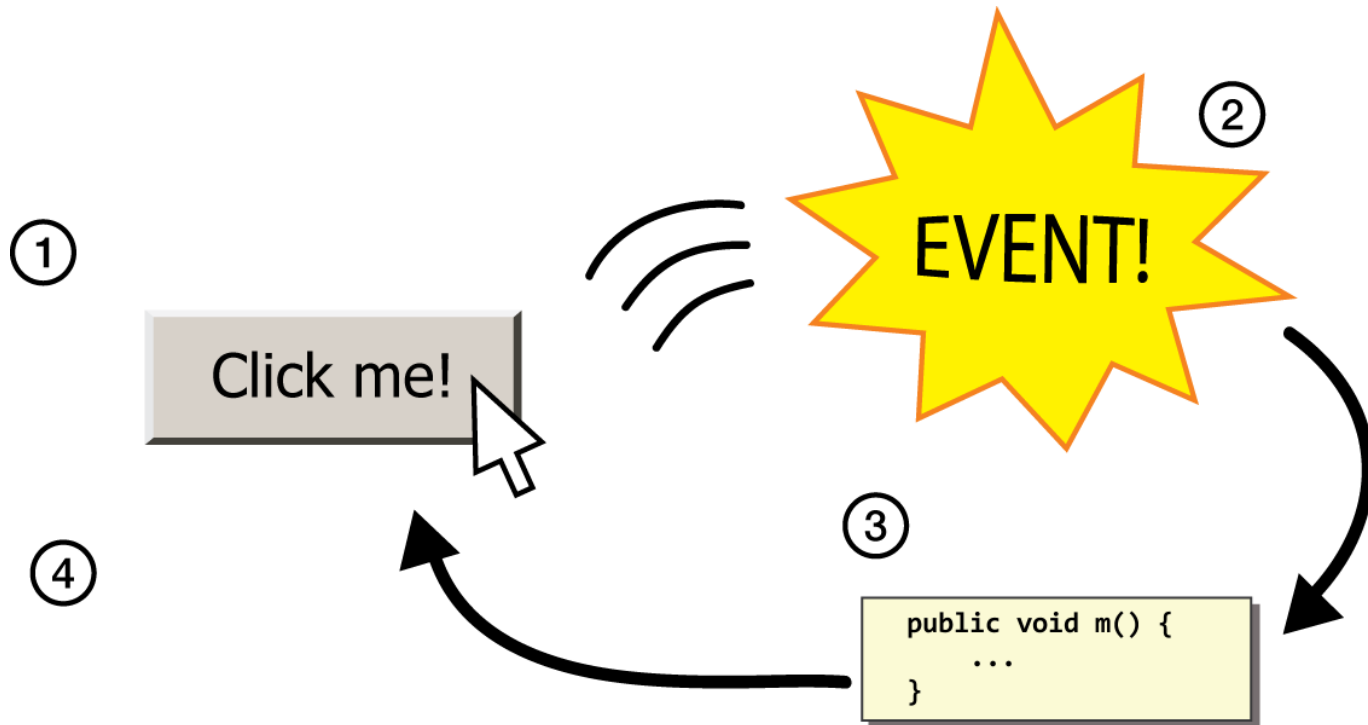
```
public void mouseDragged(MouseEvent event) {
 double mouseX = event.getX();
 double mouseY = event.getY();
 double rectX = mouseX - SIZE / 2.0;
 double rectY = mouseY - SIZE / 2.0;
 GRect rect = new GRect(rectX, rectY, SIZE, SIZE);
 rect.setFilled(true);
 add(rect);
}
```

# 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);
}
```

# Review: animation and null: 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

- Announcements
- Review: animation and null
- Event-driven programming
- **Instance variables**
- Whack-A-Mole
- Extra: Googly Eyes

# Instance Variables

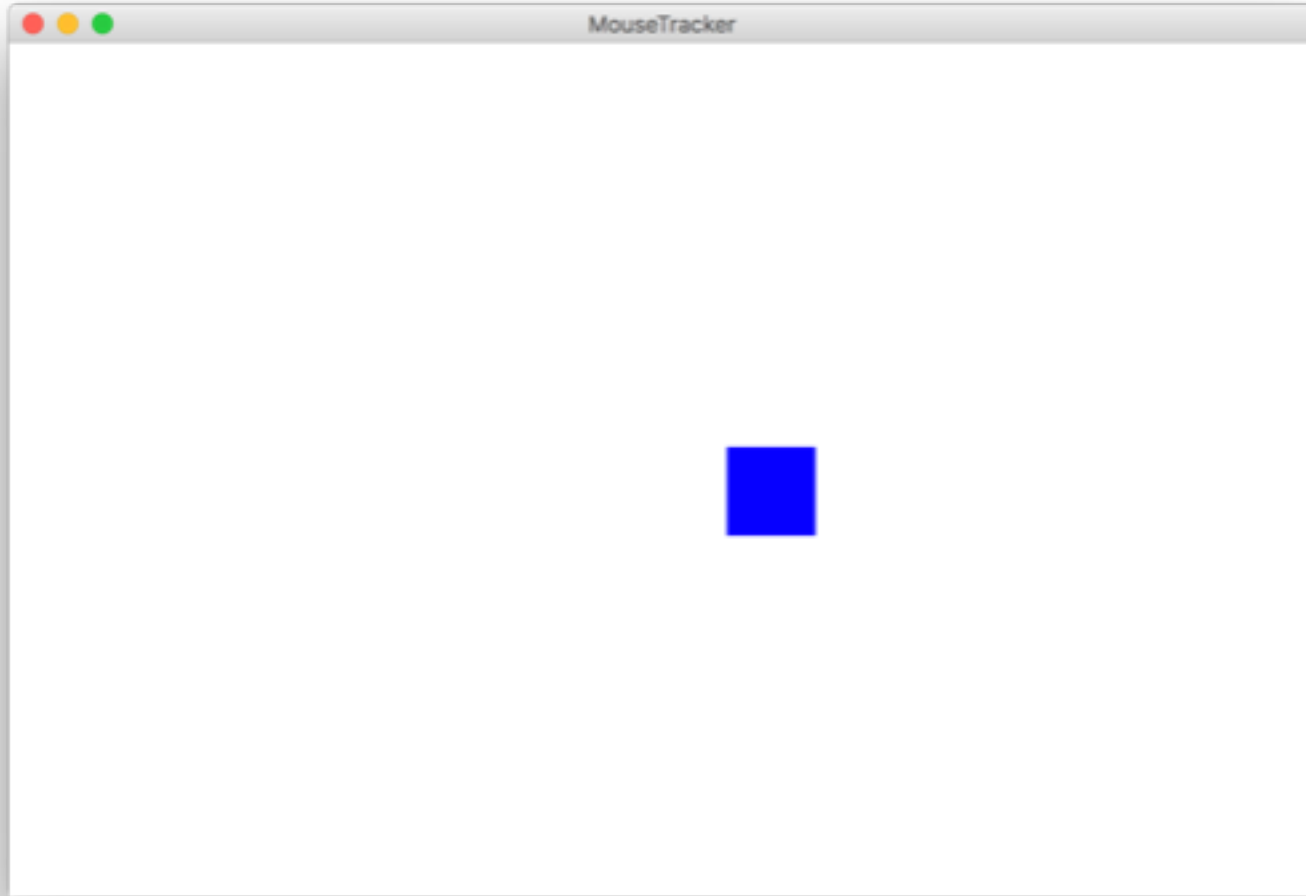
```
private type name; // declared outside of any method
```

- **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*

**DO NOT USE INSTANCE VARIABLES ON SNOWMAN!**



# Example: MouseTracker



# Plan for Today

- Announcements
- Review: animation and null
- Event-driven programming
- Instance variables
- **Whack-A-Mole**
- Extra: Googly Eyes

# Putting it all together



# Whack-A-Mole

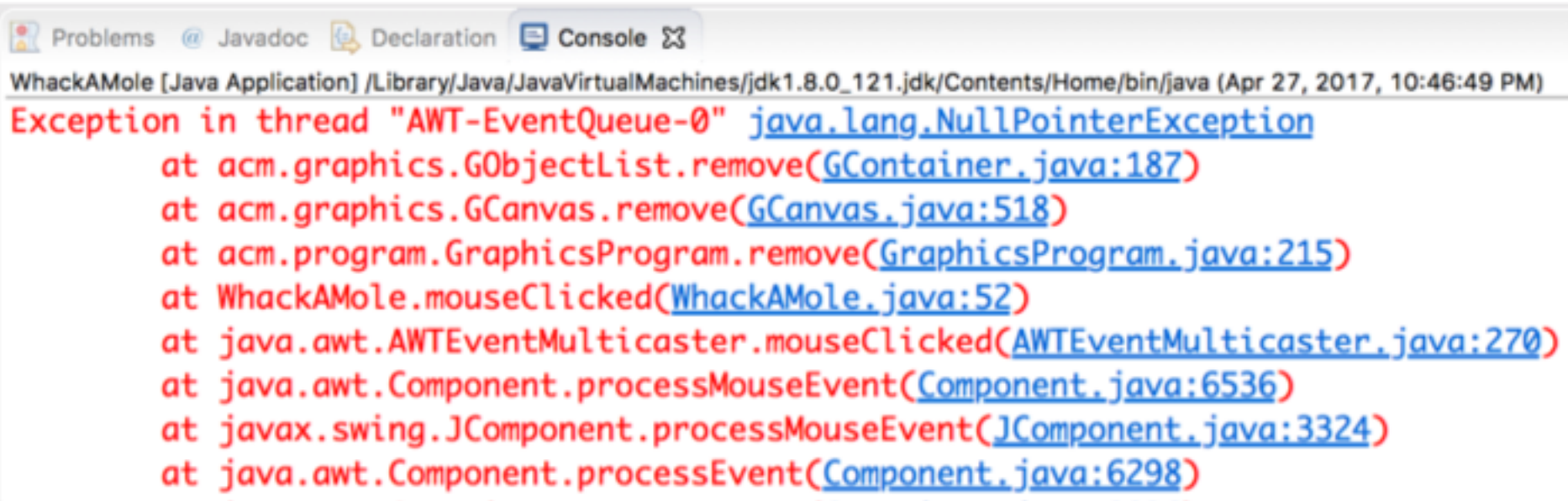
Let's use instance variables and events to make Whack-A-Mole!

- A mole should appear every second at a random location, and stop once the user has gotten at least 10 points.
- If the user clicks a mole, remove it and increase their score by 1
- There should be a GLabel in the left corner showing their score



# Exception

- If the user clicks an area with no mole, the program crashes.
  - A program crash in Java is called an **exception**.
  - When you get an exception, Eclipse shows red error text.
  - The error text shows the line number where the error occurred.
  - Why did this error happen?
  - How can we avoid this?



The screenshot shows the Eclipse IDE's Console window. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output shows a red error message: 'Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException'. Below this, a stack trace is displayed in red text, listing the sequence of method calls that led to the exception, with file names and line numbers in blue. The stack trace starts from the user's code and goes down to the Java AWT/Swing framework.

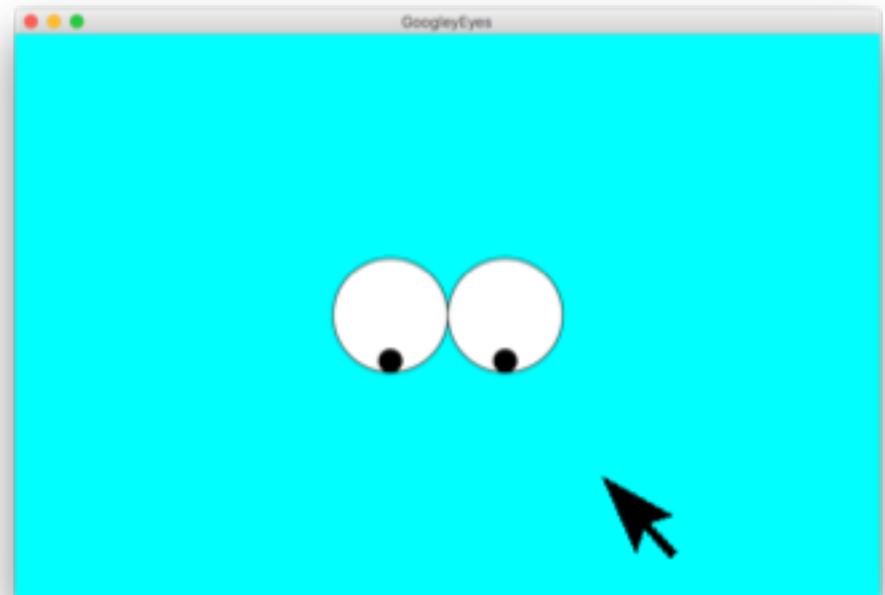
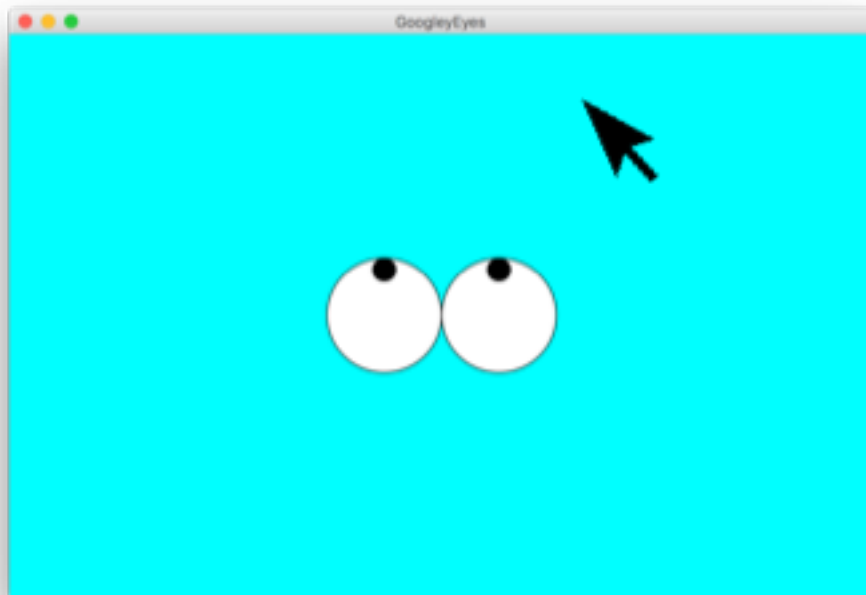
```
WhackAMole [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/bin/java (Apr 27, 2017, 10:46:49 PM)
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
 at acm.graphics.GObjectList.remove(GContainer.java:187)
 at acm.graphics.GCanvas.remove(GCanvas.java:518)
 at acm.program.GraphicsProgram.remove(GraphicsProgram.java:215)
 at WhackAMole.mouseClicked(WhackAMole.java:52)
 at java.awt.AWTEventMulticaster.mouseClicked(AWTEventMulticaster.java:270)
 at java.awt.Component.processMouseEvent(Component.java:6536)
 at javax.swing.JComponent.processMouseEvent(JComponent.java:3324)
 at java.awt.Component.processEvent(Component.java:6298)
```

# Plan for Today

- Announcements
- Review: animation and null
- Event-driven programming
- Instance variables
- Whack-A-Mole
- **Extra: Googly Eyes**

# Extra Practice: GooglyEyes

Let's write a program called **GooglyEyes** that draws eyes whose pupils follow the user's mouse vertically as it moves around.



# Plan for Today

- Announcements
- Review: animation and null
- Event-driven programming
- Instance variables
- Whack-A-Mole
- Extra: Googly Eyes

**Next Time: Memory**