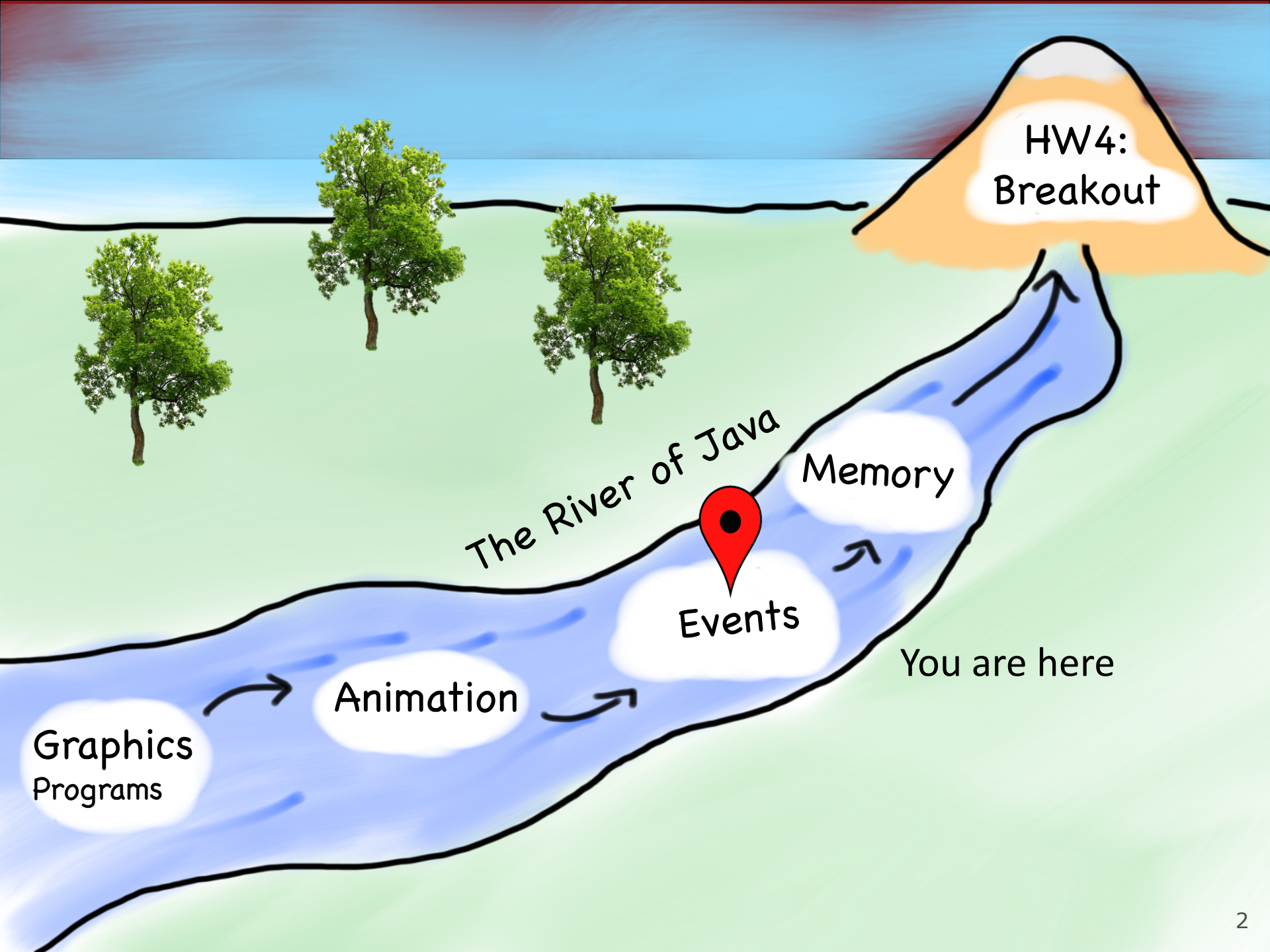
CS 106A, Lecture 14
Events and Instance Variables

Reading:

Art & Science of Java, Ch. 10.1-10.4


HW4:
Breakout
The River of Java
Memory
Events
You are here
Animation
Graphics
Programs

Learning Goals

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

Plan for Today

- Announcements
- Review: Animation
- Null
- Event-driven programming (with Daisy!)
- Instance Variables
- Whack-A-Mole

Plan for Today

- Announcements
- Review: Animation
- Null
- Event-driven programming (with Daisy!)
- Instance Variables
- Whack-A-Mole

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 ~= 50 frames/sec = 25ms pause

Plan for Today

- Announcements
- Review: Animation
- Null
- Event-driven programming (with Daisy!)
- Instance Variables
- Whack-A-Mole

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

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
}
```

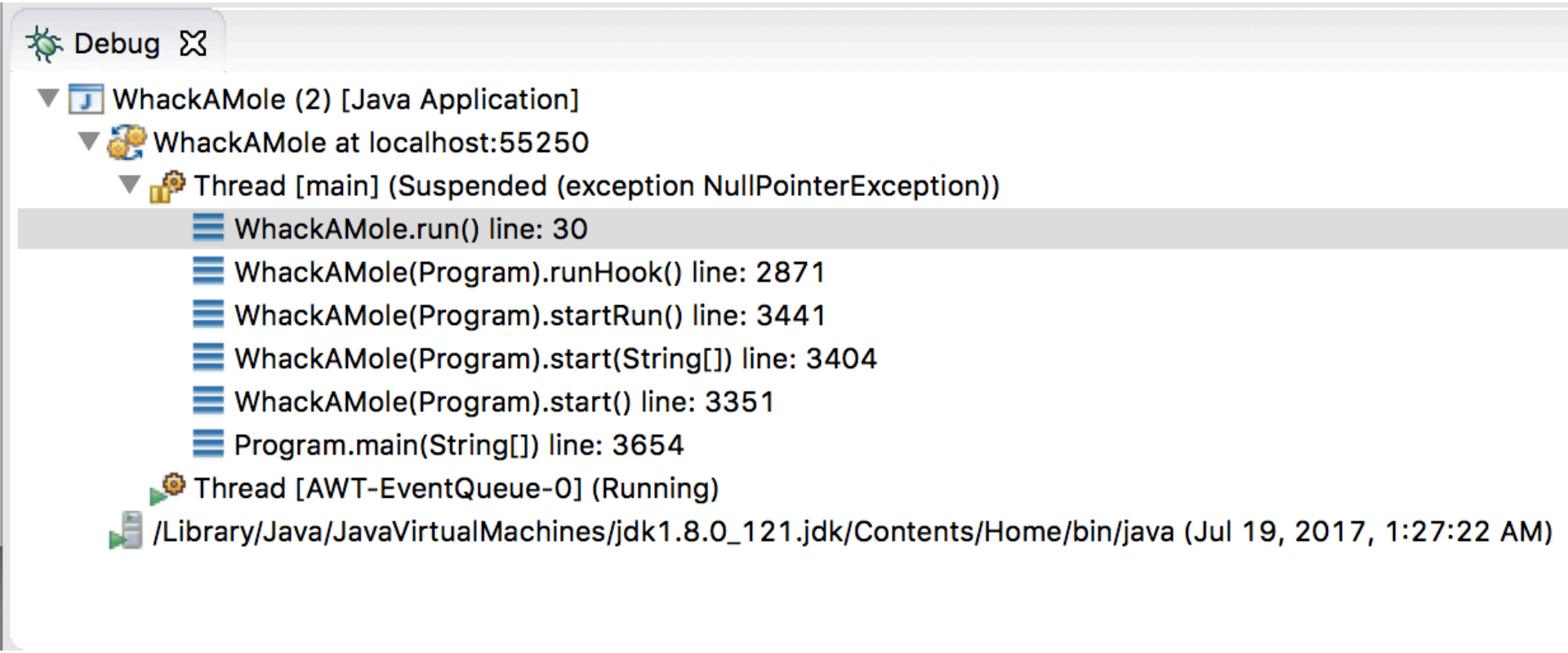
Null

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)

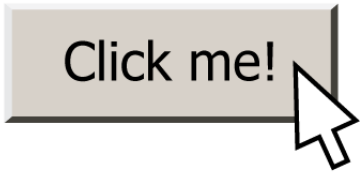


Plan for Today

- Announcements
- Review: Animation
- Null
- Event-driven programming (with Daisy!)
- Instance Variables
- Whack-A-Mole

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

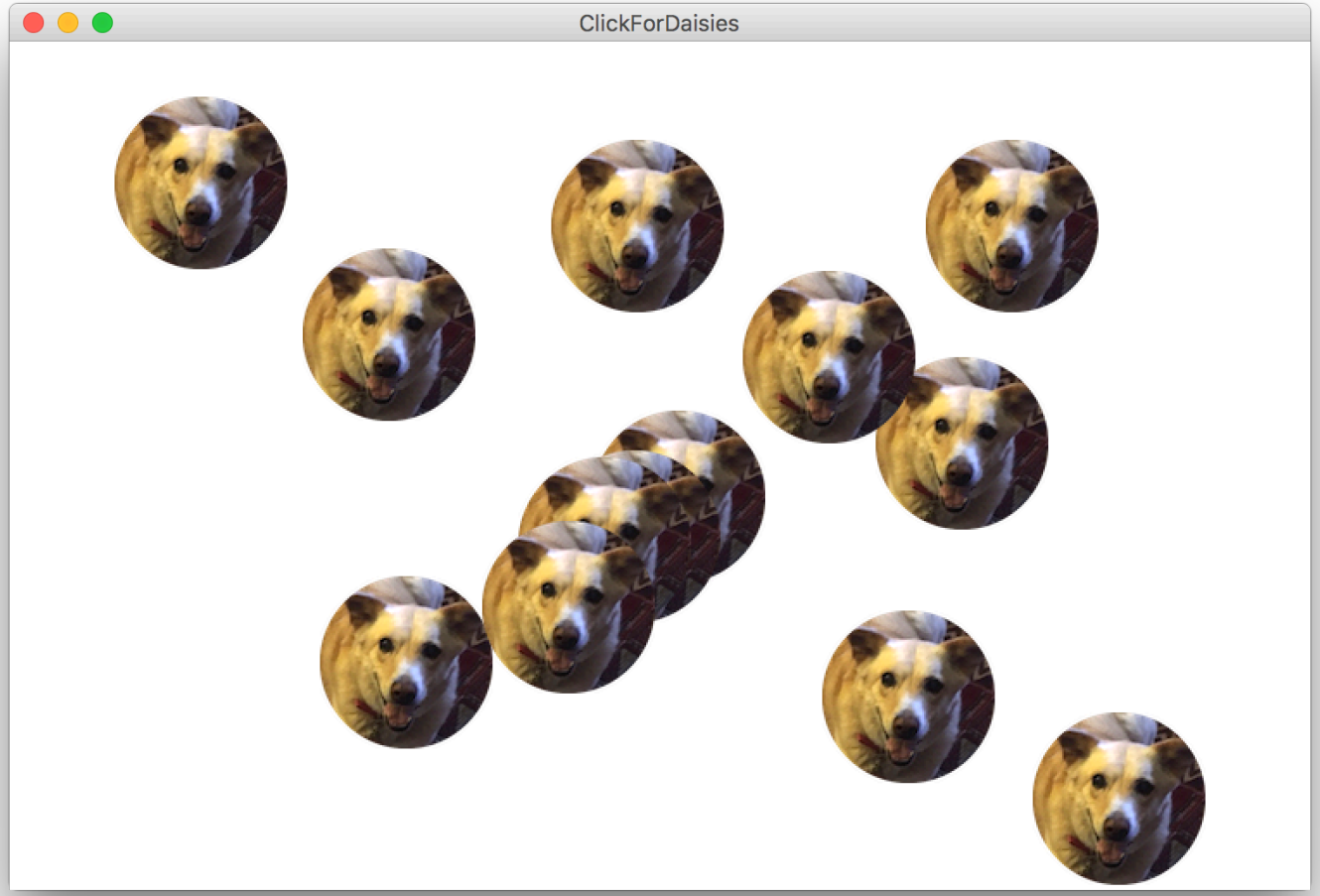
```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;      // NEW

public class ClickForDaisy extends GraphicsProgram {

    // 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
`e.getX()`	the *x*-coordinate of mouse cursor in the window
`e.getY()`	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

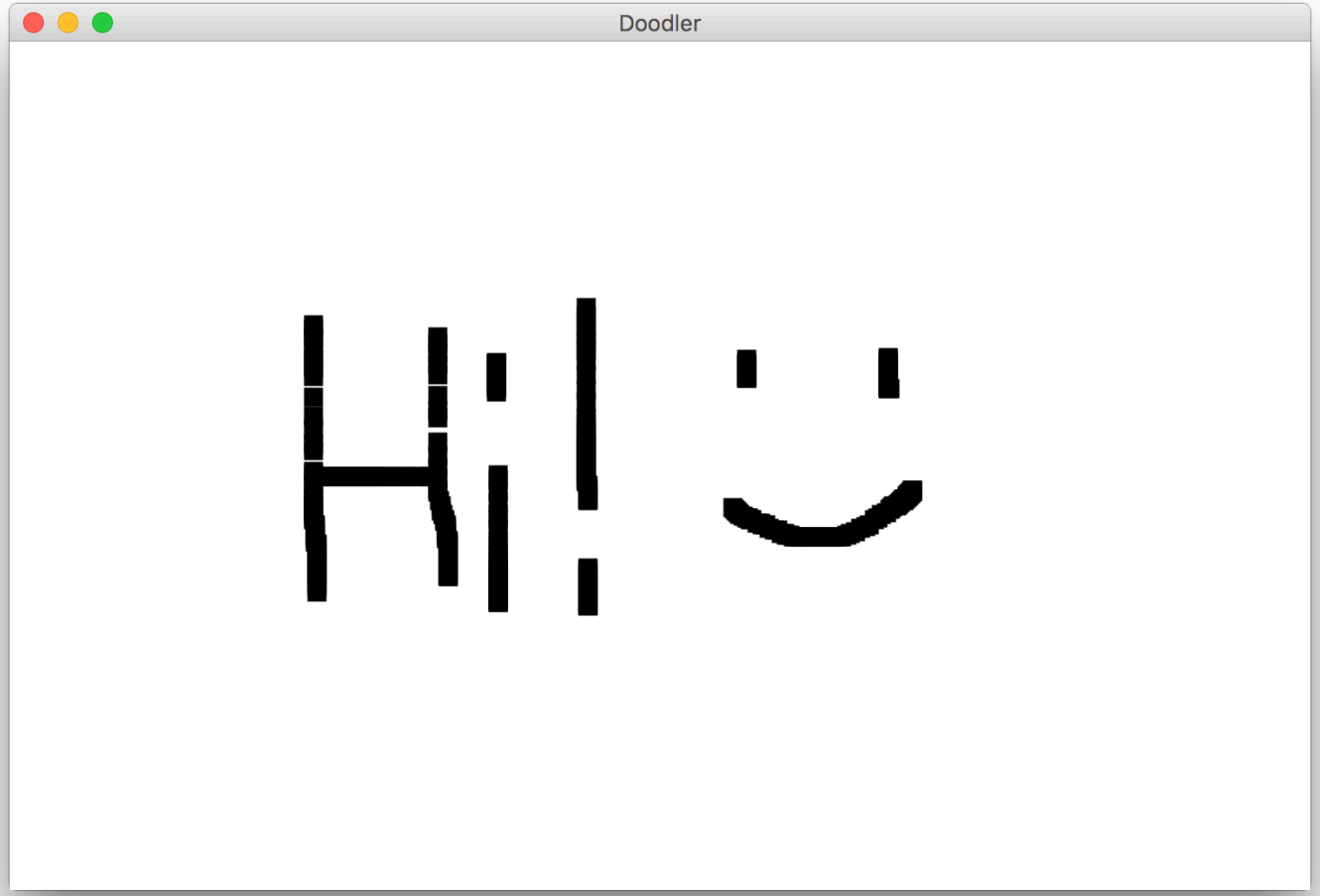
```
public class ClickForDaisies extends GraphicsProgram {

   // 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

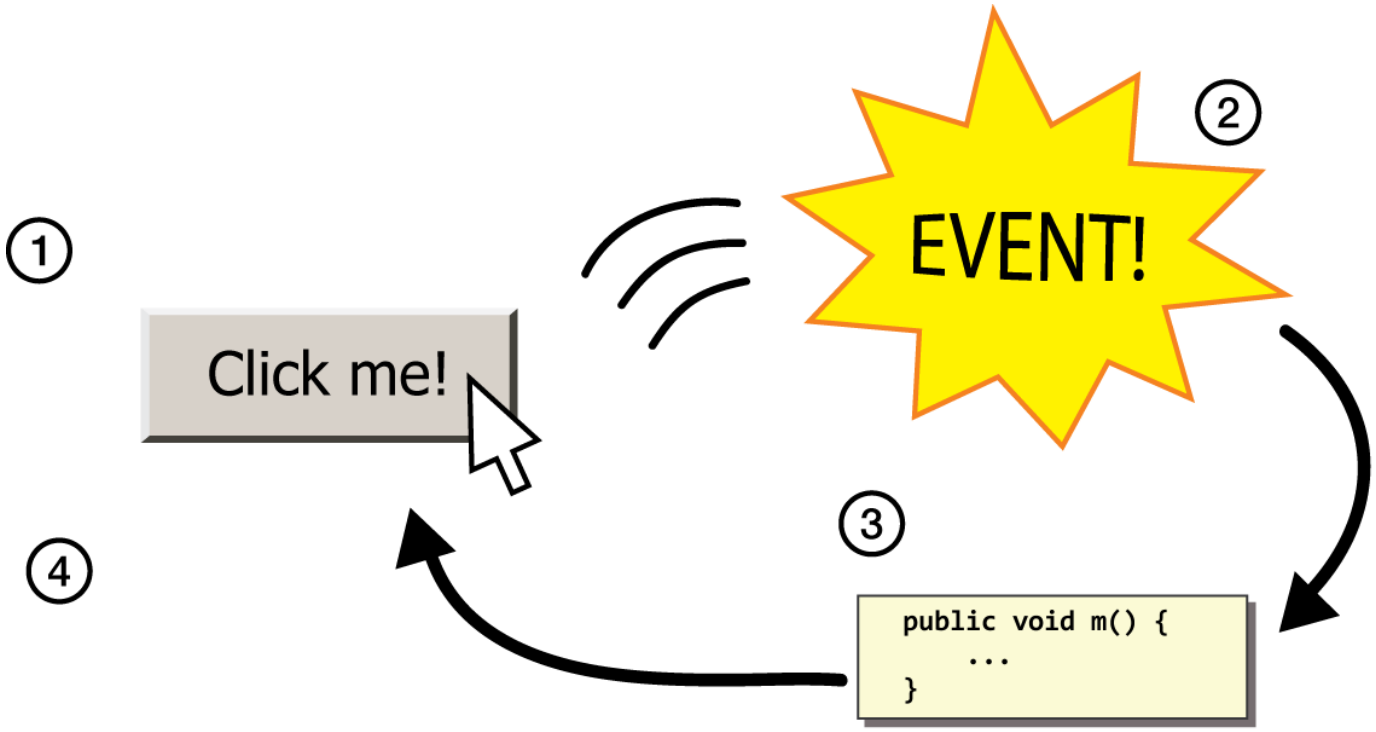
```
public void mouseDragged(MouseEvent event) {
    double mouseX = event.getX();
    double mouseY = event.getY();
    double rectX = mouseX - SIZE / 2.0;
    double rectY = mouseY - SIZE / 2.0;
    GRect rect = new GRect(rectX, rectY, SIZE, SIZE);
    rect.setFilled(true);
    add(rect);
}
```

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);
}
```

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

- Announcements
- Review: Animation
- Null
- Event-driven programming (with Daisy!)
- Instance Variables
- Whack-A-Mole

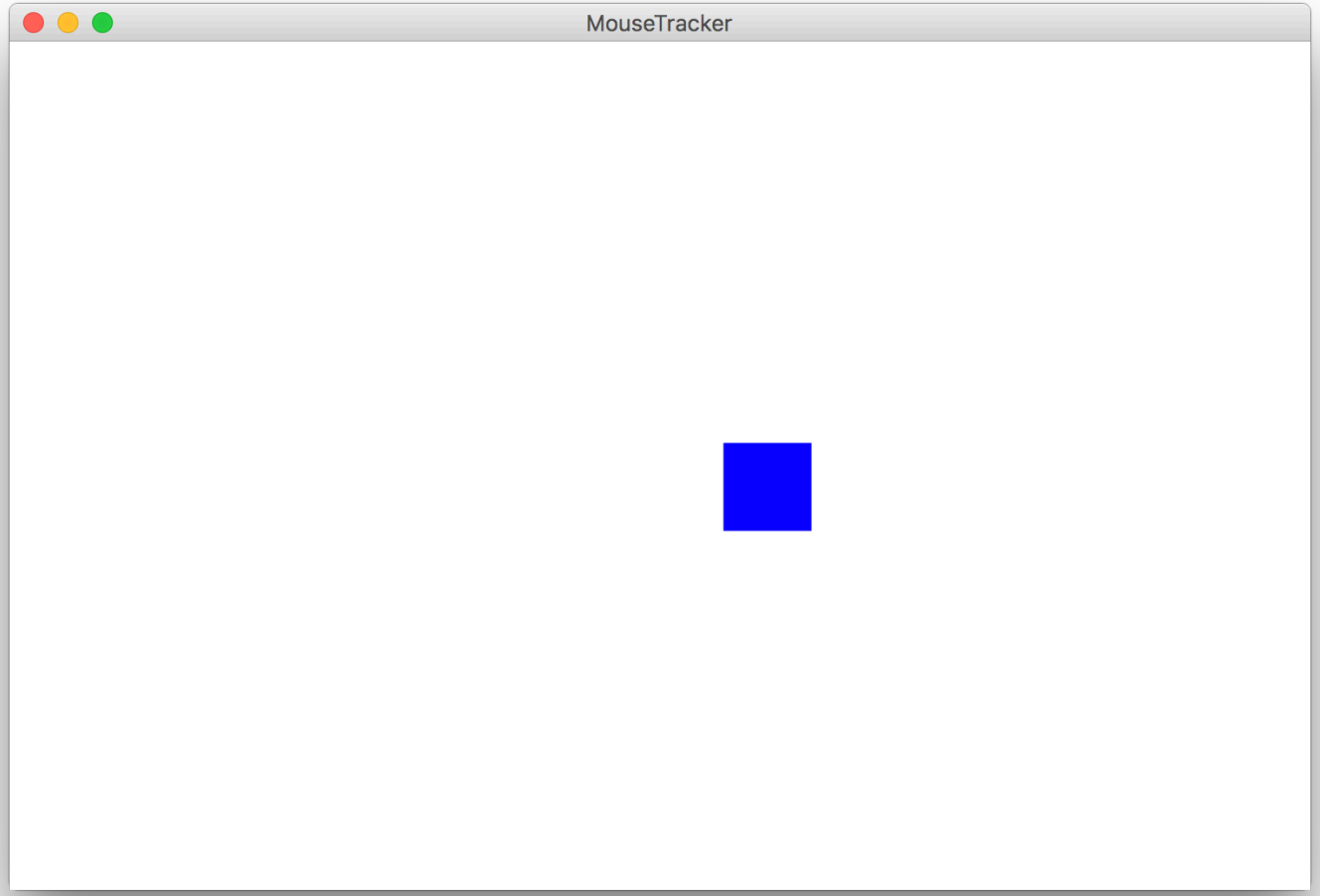
Instance Variables

```
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*

DO NOT USE INSTANCE VARIABLES ON HANGMAN!

Example: MouseTracker



Plan for Today

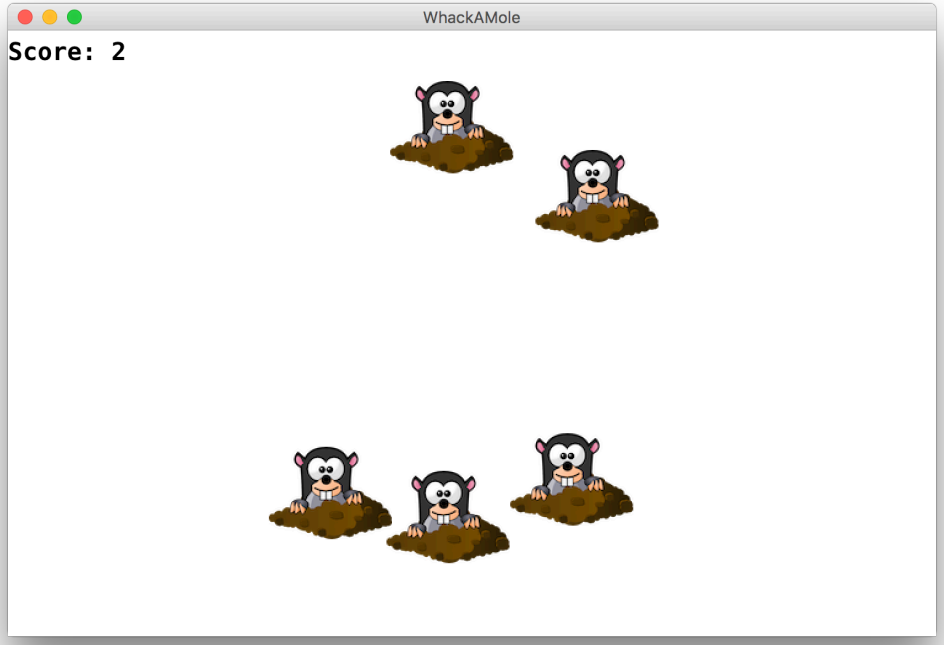
- Announcements
- Review: Animation
- Null
- Event-driven programming (with Daisy!)
- Instance Variables
- Whack-A-Mole

Putting it all together

Whack-A-Mole

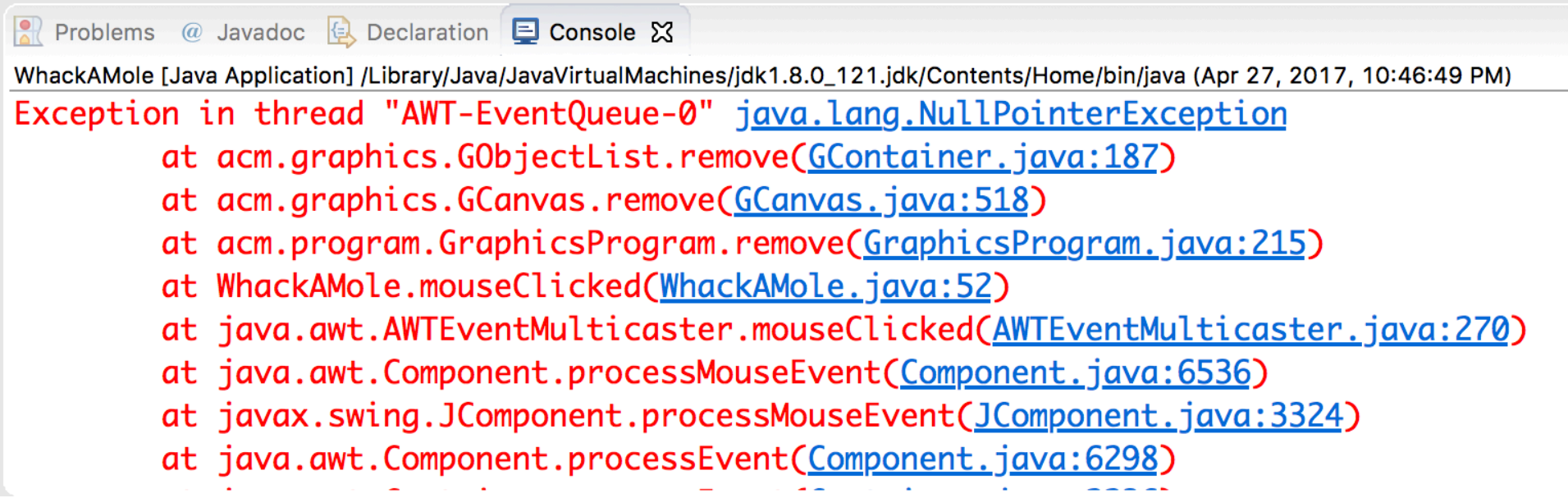
Let's use instance variables and events to make Whack-A-Mole!

- 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?



Recap

- Announcements
- Review: Animation
- Null
- Event-driven programming (with Daisy!)
- Instance Variables
- Whack-A-Mole

Next Time: More Events + Memory