CS 106A, Lecture 14
Events and Instance Variables

Reading:
Art & Science of Java, Ch. 10.1-10.4
HW4: Breakout

You are here

The River of Java

Memory

Events

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

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

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

Events

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

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>e.getX()</code></td>
<td>the x-coordinate of mouse cursor in the window</td>
</tr>
<tr>
<td><code>e.getY()</code></td>
<td>the y-coordinate of mouse cursor in the window</td>
</tr>
</tbody>
</table>
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);
    }
}
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:
    ```java
    public void eventMethodName(MouseEvent event) {
    ...}
    ```

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mouseMoved</td>
<td>mouse cursor moves</td>
</tr>
<tr>
<td>mouseDragged</td>
<td>mouse cursor moves while button is held down</td>
</tr>
<tr>
<td>mousePressed</td>
<td>mouse button is pressed down</td>
</tr>
<tr>
<td>mouseReleased</td>
<td>mouse button is lifted up</td>
</tr>
<tr>
<td>mouseClicked</td>
<td>mouse button is pressed and then released</td>
</tr>
<tr>
<td>mouseEntered</td>
<td>mouse cursor enters your program's window</td>
</tr>
<tr>
<td>mouseExited</td>
<td>mouse cursor leaves your program's window</td>
</tr>
</tbody>
</table>
Example: 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);
}
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);
}
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);
}
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);
}
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.
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
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

Score: 2

[Image of a computer screen showing a game with four mole characters and a score of 2]
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?

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)
Recap

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

Next Time: More Events + Memory