Events
Chris Piech
CS106A, Stanford University
We’ve Gotten Ahead of Ourselves
Start at the Beginning

Source: The Hobbit
Learning Goals

1. Write a program that can respond to mouse events
2. Use an instance variable in your program
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
}
public void run() {
    // 1. add mouse listeners
    addMouseListener();
}

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

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

YOU ARE HERE
Hole Puncher
Now With Dancing Children
Normal Program

Run Method
public void run() {
    while (true) {
        update();
        pause(DELAY);
    }
}
Normal Program

Run Method

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

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

Run Method

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

Run Method

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

Run Method

Piech, CS106A, Stanford University
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
addMouseListener();
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
Mouse Tracker
Mouse Tracker
1. Variables exist until their inner-most control block ends.
2. If a variable is defined outside all methods, its inner-most control block is the entire program!
3. We call these variables **instance variables**

```java
public class MouseTrackerSoln extends GraphicsProgram {
    /* Instance variable for the square to be tracked */
    GRect square = null;

    public void run() {
        addSquare();
        addMouseListener();
    }
}
```

* Instance variables have special meanings in programs with multiple files. For now you need to know that all methods can see them and that their initialization line is executed before run.
Often you need instance variables to pass information between the run method and the mouse event methods!

```java
public class MouseTrackerSoln extends GraphicsProgram {

    /* Instance variable for the square to be tracked */
    GRect square = null;

    public void run() {
        square = makeSquare();
        addMouseListener();
    }

    public void mouseMoved(MouseEvent e) {
        int x = e.getX() - SQUARE_SIZE/2;
        int y = e.getY() - SQUARE_SIZE/2;
        square.setLocation(x, y);
    }
}
```
Objects have a special value called **null** which means this variable is not associated with a value yet.

```java
public void run() {
    G0val example = null;
    if(example == null) {
        println("initially example is null");
    }
    example = new G0val(5, 5);
    if(example != null) {
        println("now its not null.");
    }
}
```

Initially example is null
now its not null.
GOblents returned by getElelementAt might be null!

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

```c
// 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!
}
```
Calling methods on an object that is **null** will crash your program! (throws a NullPointerException)
Debris Sweeper
New Concepts

New Commands
• `addMouseListener()``
• `getElementAt(x, y)``
• `remove(obj)`

New Ideas
• The Listener Model
• Instance Variables
• **null**
Responding to Mouse Events

1. The run method should call `addMouseListener`
2. Write definitions of any listener methods needed

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mouseClicked(e)</td>
<td>Called when the user clicks the mouse</td>
</tr>
<tr>
<td>mousePressed(e)</td>
<td>Called when the mouse button is pressed</td>
</tr>
<tr>
<td>mouseReleased(e)</td>
<td>Called when the mouse button is released</td>
</tr>
<tr>
<td>mouseMoved(e)</td>
<td>Called when the user moves the mouse</td>
</tr>
<tr>
<td>mouseDragged(e)</td>
<td>Called when the mouse is dragged with the button down</td>
</tr>
</tbody>
</table>

The parameter `e` is `MouseEvent` object, which provides more data about event, such as the location of mouse.
Responding to Keyboard Events

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

<table>
<thead>
<tr>
<th>keyPressed(e)</th>
<th>Called when the user presses a key</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyReleased(e)</td>
<td>Called when the key comes back up</td>
</tr>
<tr>
<td>keyTyped(e)</td>
<td>Called when the user types (presses and releases) a key</td>
</tr>
</tbody>
</table>

The parameter `e` is a **KeyEvent** object, which indicates which key is involved.
And Here We Are...
Catch Me If You Can?