Events

Chris Piech
CS106A, Stanford University
Catch Me If You Can
We’ve Gotten Ahead of Ourselves

Source: The Hobbit
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
Novelty

New Commands
• `rg.nextInt(max);`
• `addMouseListener();`
• `getElementAt(x, y);`
• `remove(obj);`

New Ideas
• The Listener Model
• Instance Variables
• `null`
Random Numbers

```java
// this variable can make random numbers
RandomGenerator rg = new RandomGenerator();

// make a random number between 0 and 5
int randNum = rg.nextInt(6);
```
Random Numbers Theory

\[ X_{n+1} = (aX_n + c) \% m \]

// actual code from the java library
int nextseed = (oldseed * multiplier + addend) % max;

Fun fact: the first random number comes from the CPU clock

* Just for fun. Not on exams or psets…
Random Numbers

```
// this variable can make random numbers
RandomGenerator rg = new RandomGenerator();

// now each run will be the same...
rg.setSeed(0);

// make a random number between 0 and 5
int randNum = rg.nextInt(6);
```
Catch Me If You Can
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
}
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
Hole Puncher
Now With Dancing Children
Normal Program

Run Method

Piech, CS106A, Stanford University
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);
    }
}
```
public void run() {
    while (true) {
        update();
        pause(DELAY);
    }
}
Run Method

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

Run Method
New Listener Characters

Mouse Listener

Mouse Moved Method

Piech, CS106A, Stanford University
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

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

public void run() {
    square = makeSquare();
    addSquareToCenter();
    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
/* Instance variable for the square to be tracked */
private GRect square = null;

public void run() {
    square = makeSquare();
    addSquareToCenter();
    addMouseListeners();
}

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() {
    GOval example = null;
    if(example == null) {
        println("initially example is null");
    }
    example = new GOval(5, 5);
    if(example != null) {
        println("now its not null.");
    }
}
```

Initially example is null
now its not null.
GObjects returned by `getElementAt` might be null!

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

```java
// 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)
Debris Sweeper

Piech, CS106A, Stanford University
Novelty

New Commands
• `rg.nextInt(max);`
• `addMouseListener();`
• `getElementAt(x, y);`
• `remove(obj);`

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

1. The \textbf{run} method should call \texttt{addMouseListeners}
2. Write definitions of any listener methods needed

<table>
<thead>
<tr>
<th>Function</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 \texttt{MouseEvent} object, which provides more data about event, such as the location of mouse.

Using portions of slides by Eric Roberts
Responding to Keyboard Events

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyPressed(e)</td>
<td>Called when the user presses a key</td>
</tr>
<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?