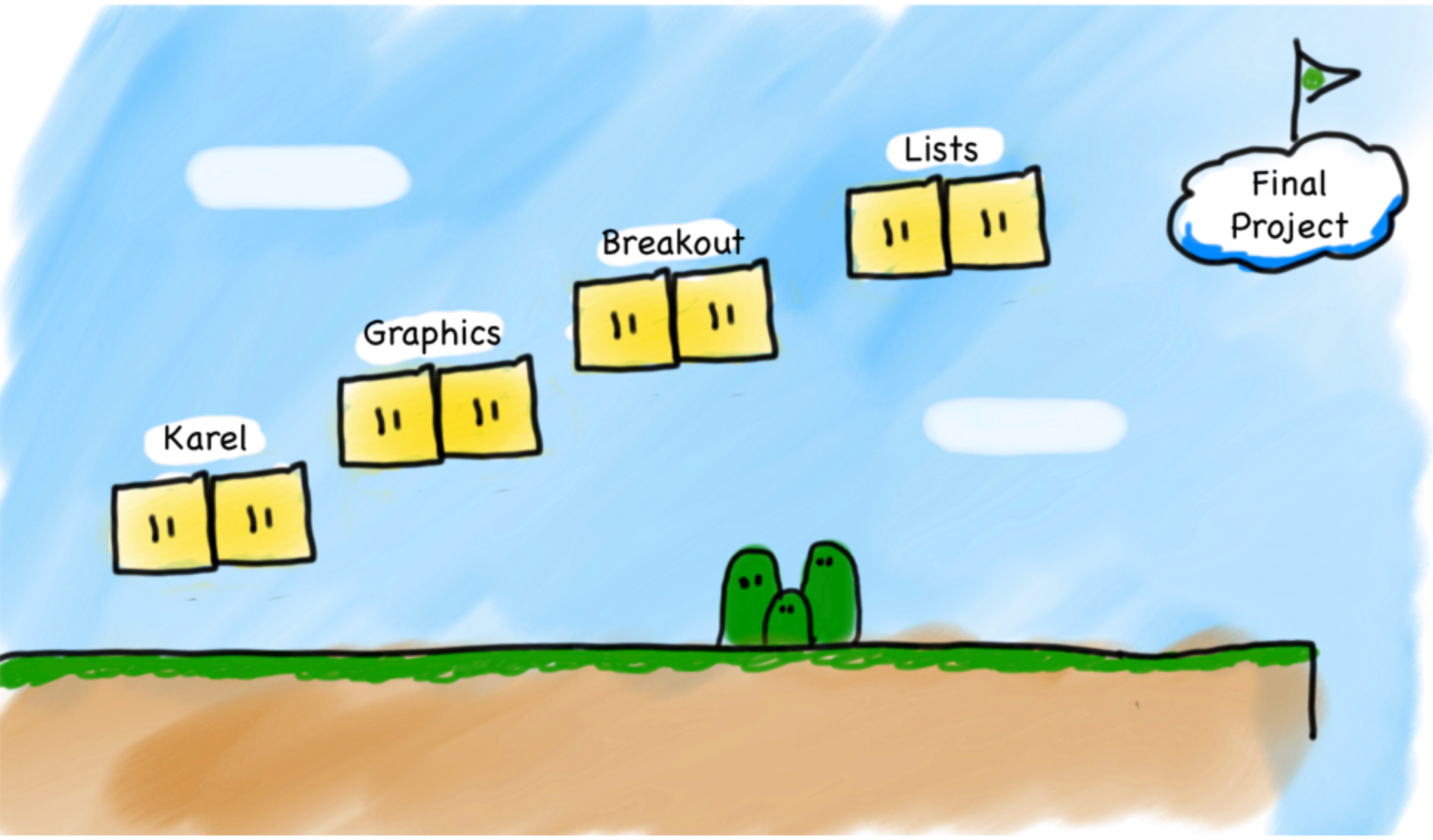
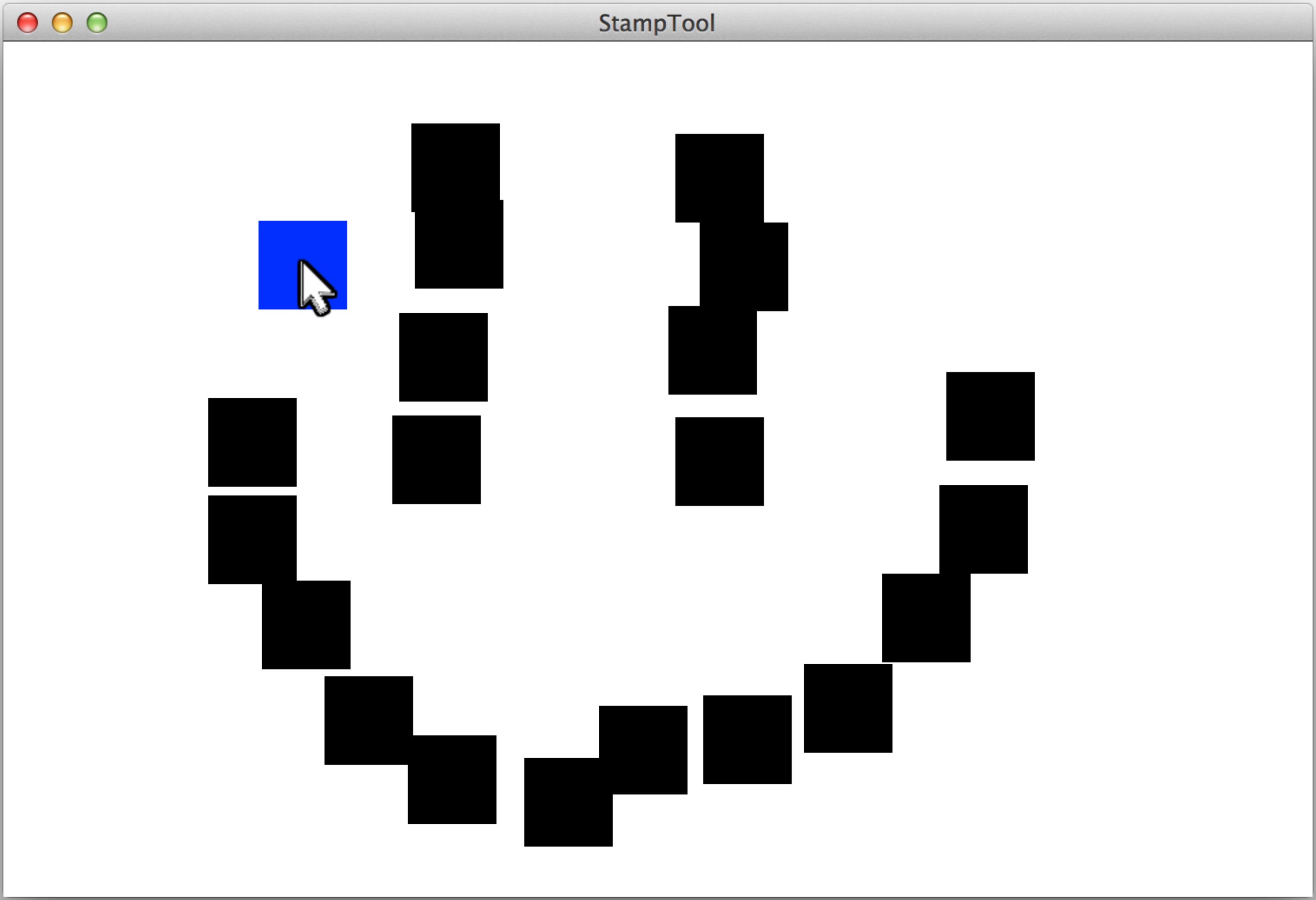
Events

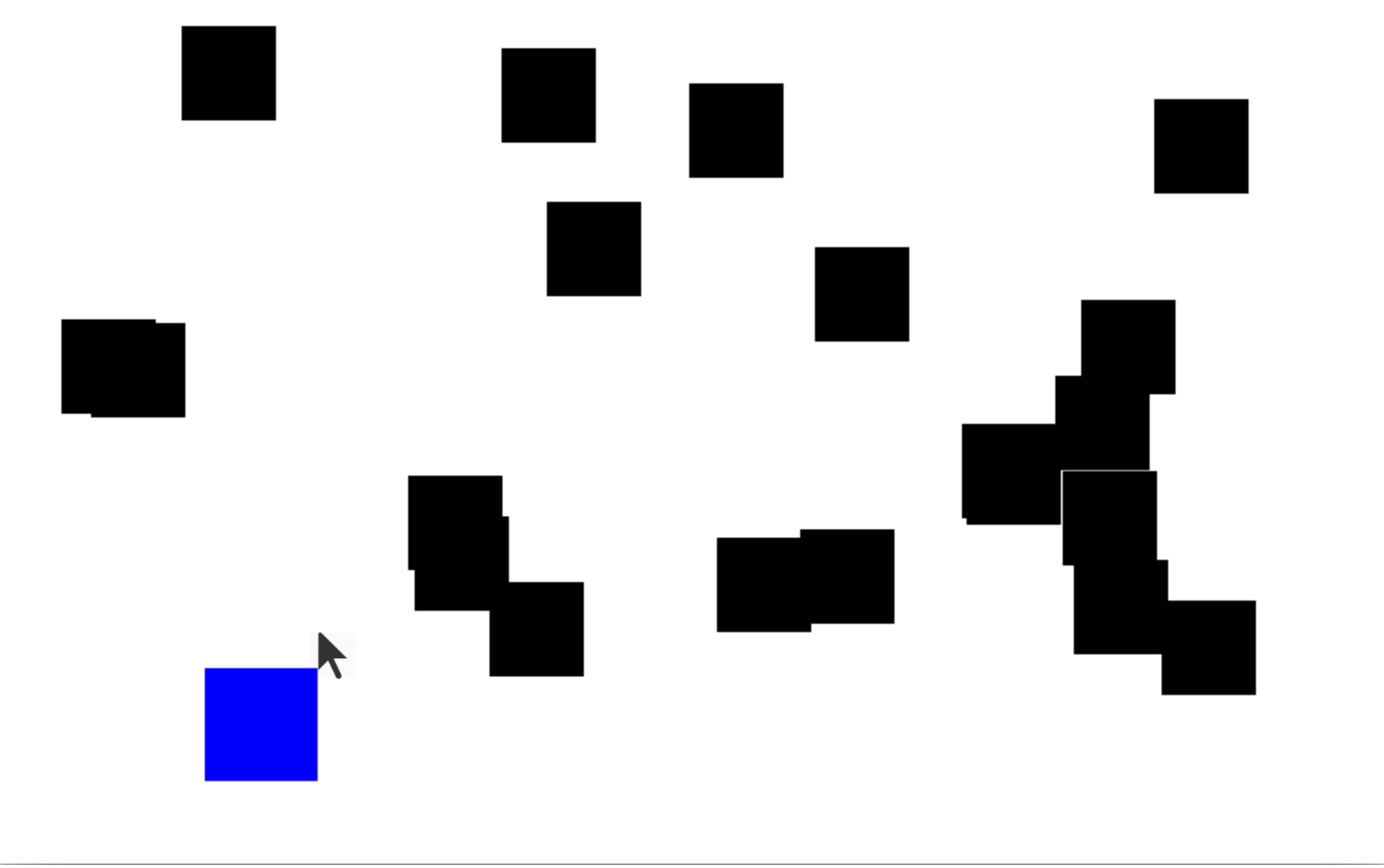
Where are we?



Stamp Tool



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

Listener Model

- When users interact with computer they generate events (e.g., moving/clicking the mouse)
- Can respond to events by having **<u>listener</u>** for events

```
addMouseListeners()
```

- Listeners get control of the program when an event happens.

Responding to Mouse Events

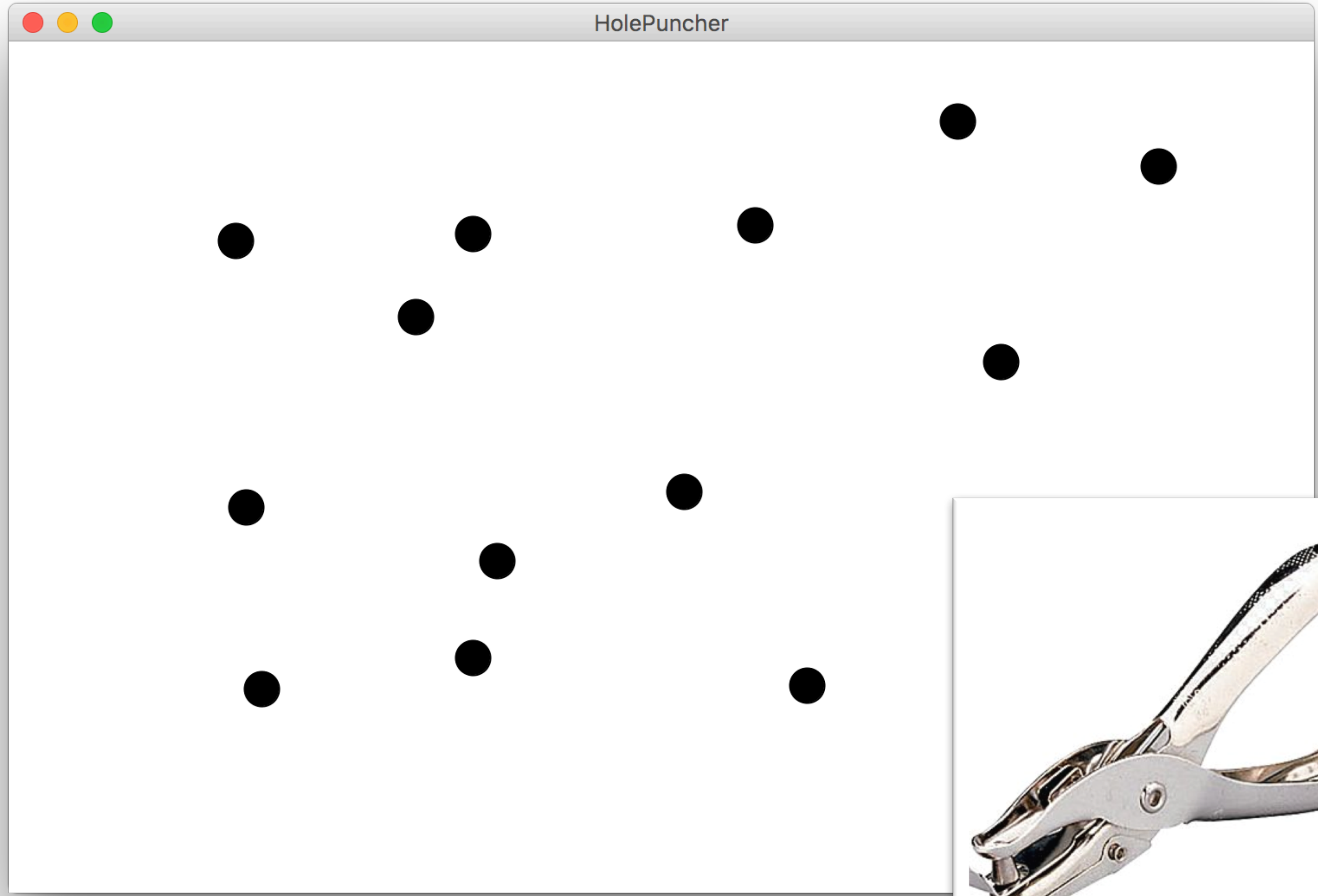
1. The **run** method should call **addMouseListeners**
2. Write definitions of any listener methods needed

mouseClicked(*e***)**	Called when the user clicks the mouse
mousePressed(*e***)**	Called when the mouse button is pressed
mouseReleased(*e***)**	Called when the mouse button is released
mouseMoved(*e***)**	Called when the user moves the mouse
mouseDragged(*e***)**	Called when the mouse is dragged with the button down

The parameter *e* is **MouseEvent** object, which provides more data about event, such as the location of mouse.

Using portions of slides by Eric Roberts

Hole Puncher



Now With Dancing Children

Normal Program

Run Method

New Listener Characters

Mouse Listener

Mouse Clicked Method

Program Starts Running

Run Method

Mouse Clicked Method

Add Mouse Listener

Run Method

Mouse Clicked Method

Mouse Listener

```
addMouseListeners();
```

Program Runs as Usual

Run Method

Mouse Clicked Method

Mouse Listener

Mouse Clicked!

Run Method

Mouse Clicked Method

Mouse Listener

Calls Mouse Clicked Method

Run Method

Mouse Clicked Method

Mouse Listener

When done, Run continues.

Run Method

Mouse Clicked Method

Mouse Listener

Keeps Doing Its Thing…

Run Method

Mouse Clicked Method

Mouse Listener

Mouse Moved!

Run Method

Mouse Clicked Method

Mouse Listener

Calls Mouse Clicked Method

Run Method

Mouse Clicked Method

Mouse Listener

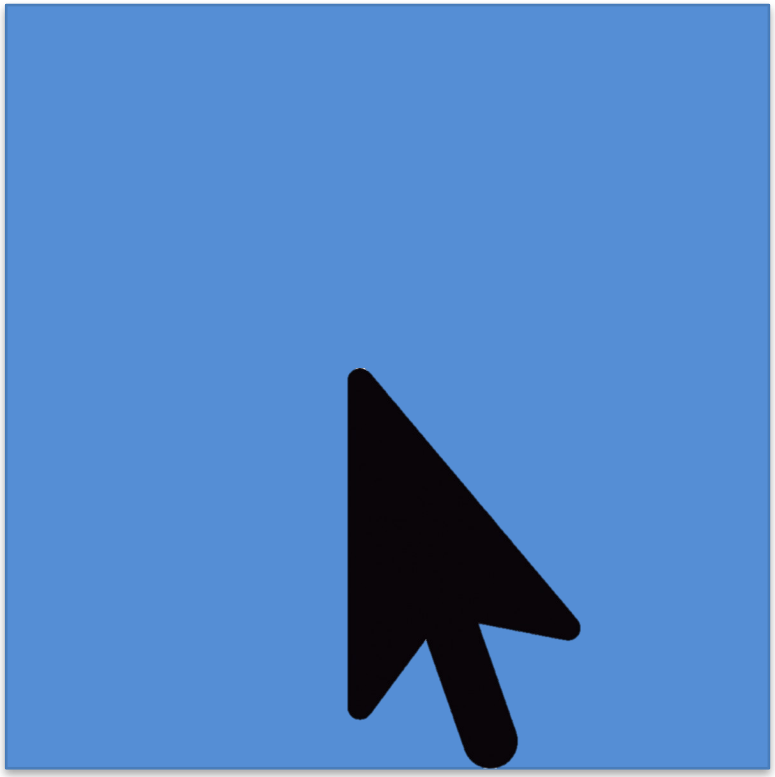
When done, Run continues.

Run Method

Mouse Clicked Method

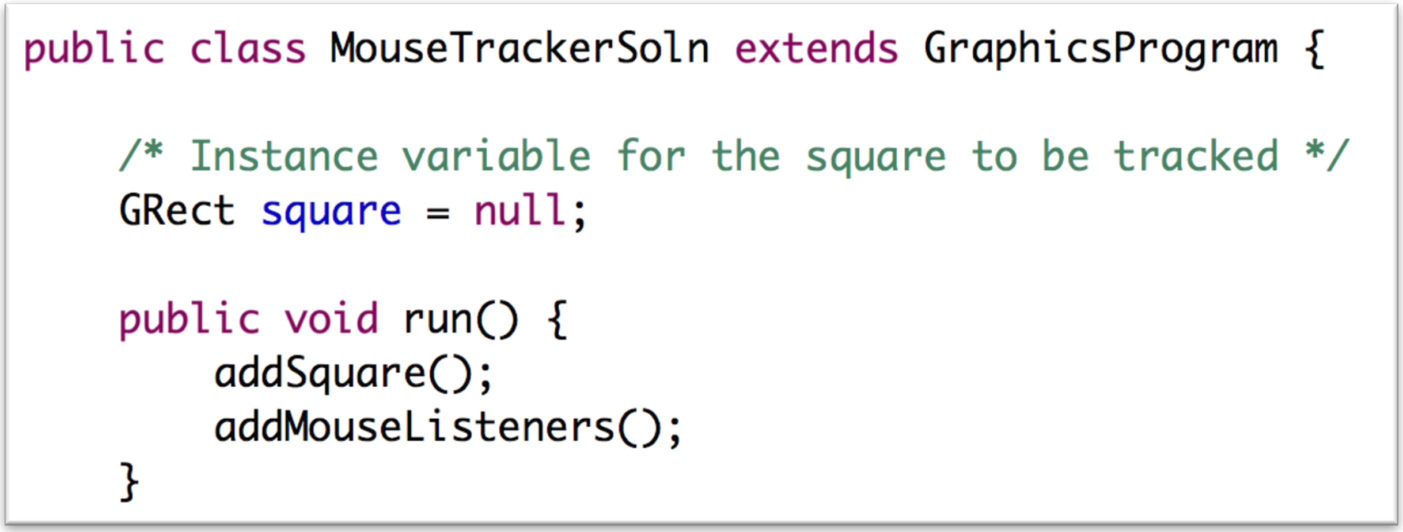
Mouse Listener

Mouse Tracker



Instance Variables

1. Variables exist until their inner-most code block ends.
2. If a variable is defined outside all methods, its inner-most code block is the entire program!
3. We call these variables **instance variables**

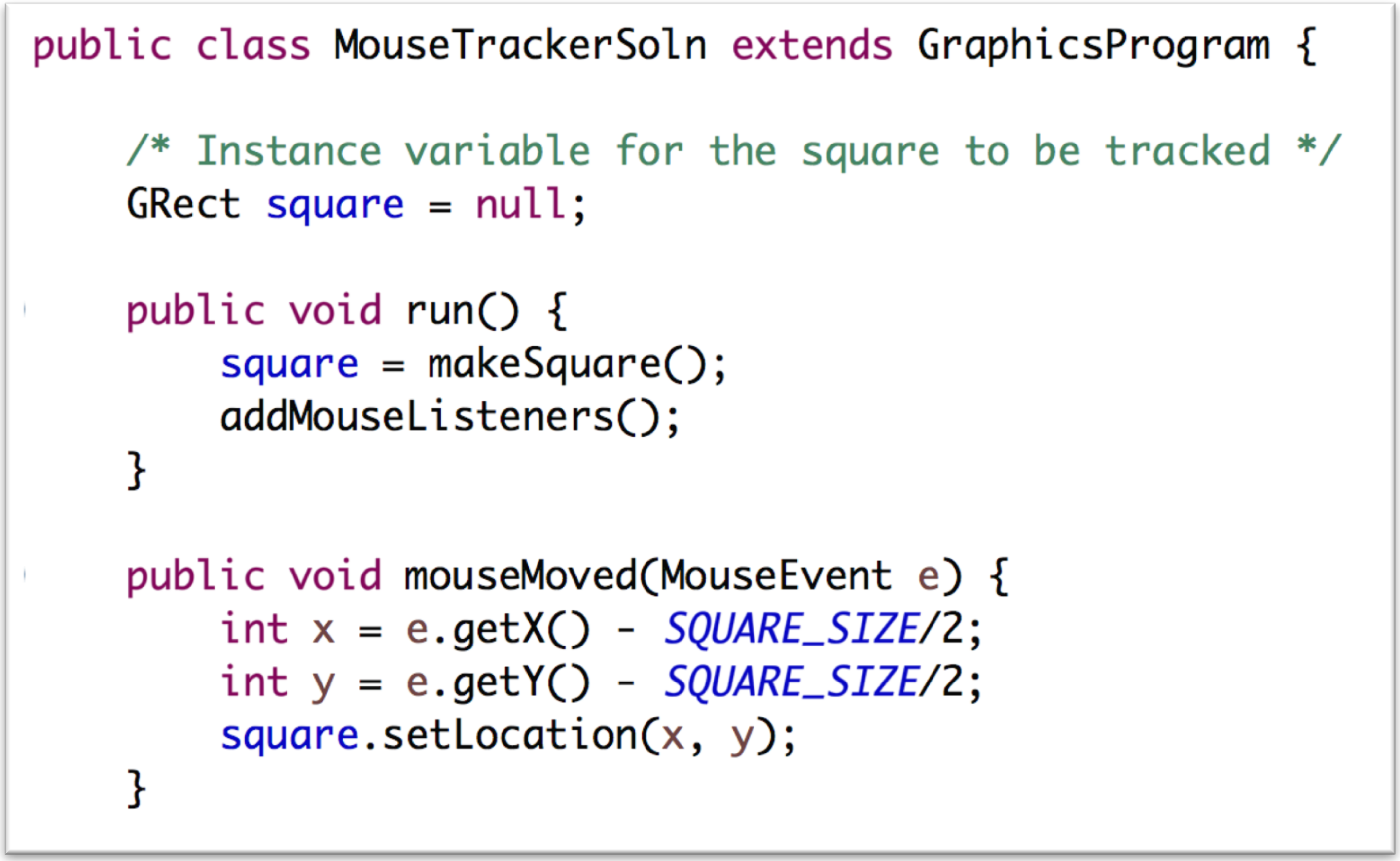


```
public class MouseTrackerSoln extends GraphicsProgram {

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

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

* 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.

Instance Variables + Events

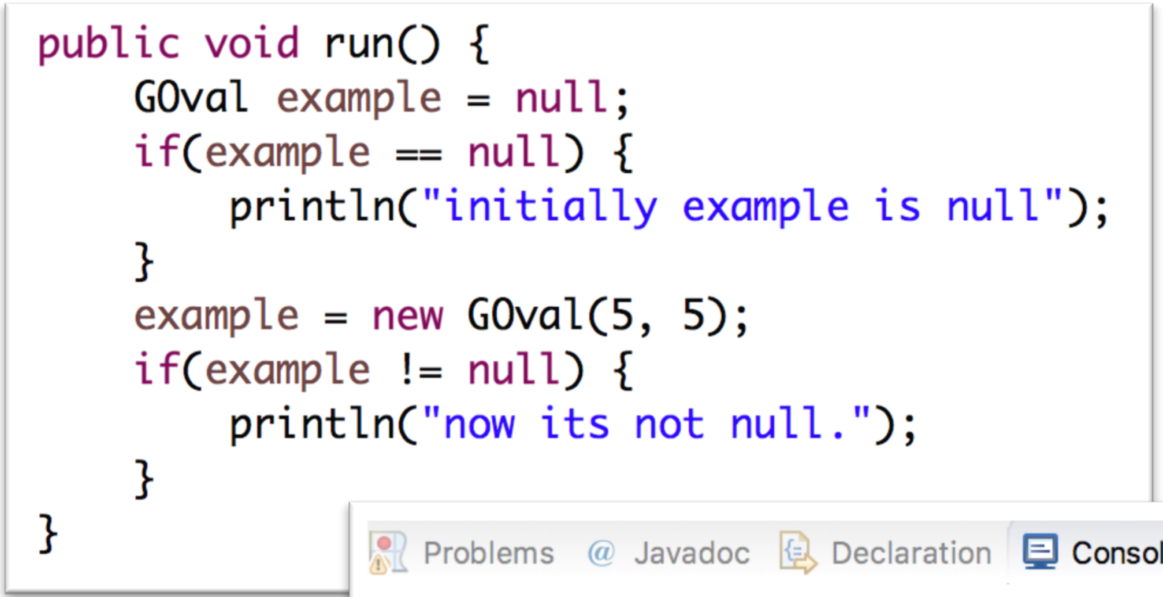
Often you need instance variables to pass information between the run method and the mouse event methods!



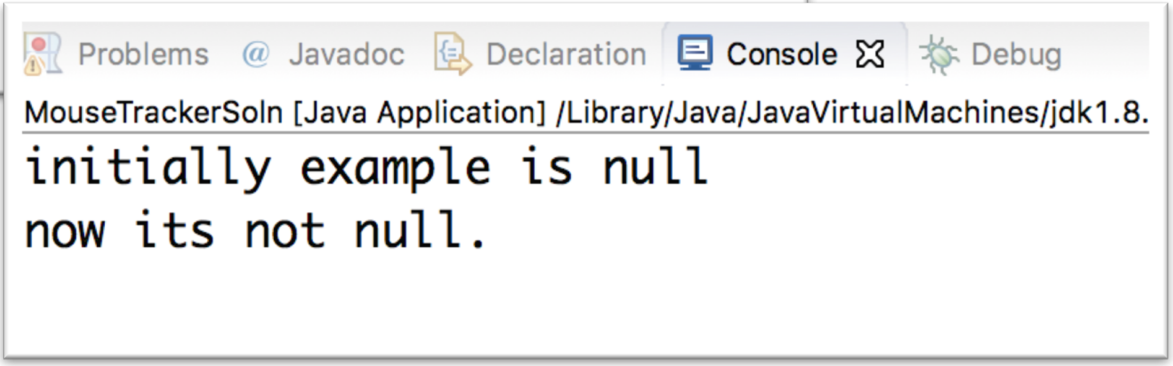
```java
public class MouseTrackerSoln extends GraphicsProgram {

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

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

    public void mouseMoved(MouseEvent e) {
        int x = e.getX() - SQUARE_SIZE/2;
        int y = e.getY() - SQUARE_SIZE/2;
        square.setLocation(x, y);
    }
```

Null

Objects have a special value called `null` which means this variable is not associated with a value yet.

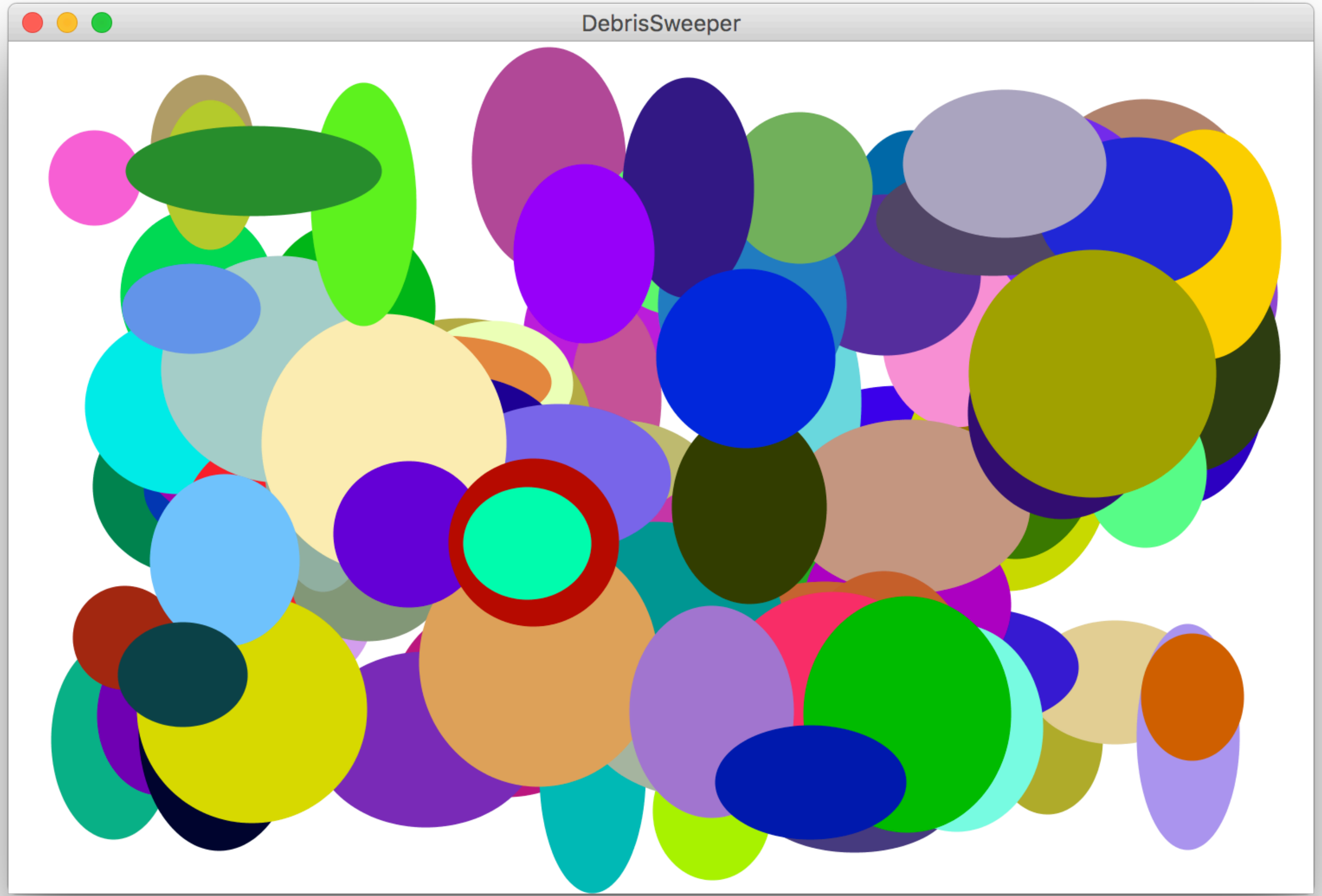


```java
public void run() {
    GOval example = null;
    if(example == null) {
        println("initially example is null");
    }
    example = new GOval(5, 5);
    if(example != null) {
        println("now its not null.");
    }
}
```



```
MouseTrackerSoln [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.
initially example is null
now its not null.
```

Debris Sweeper



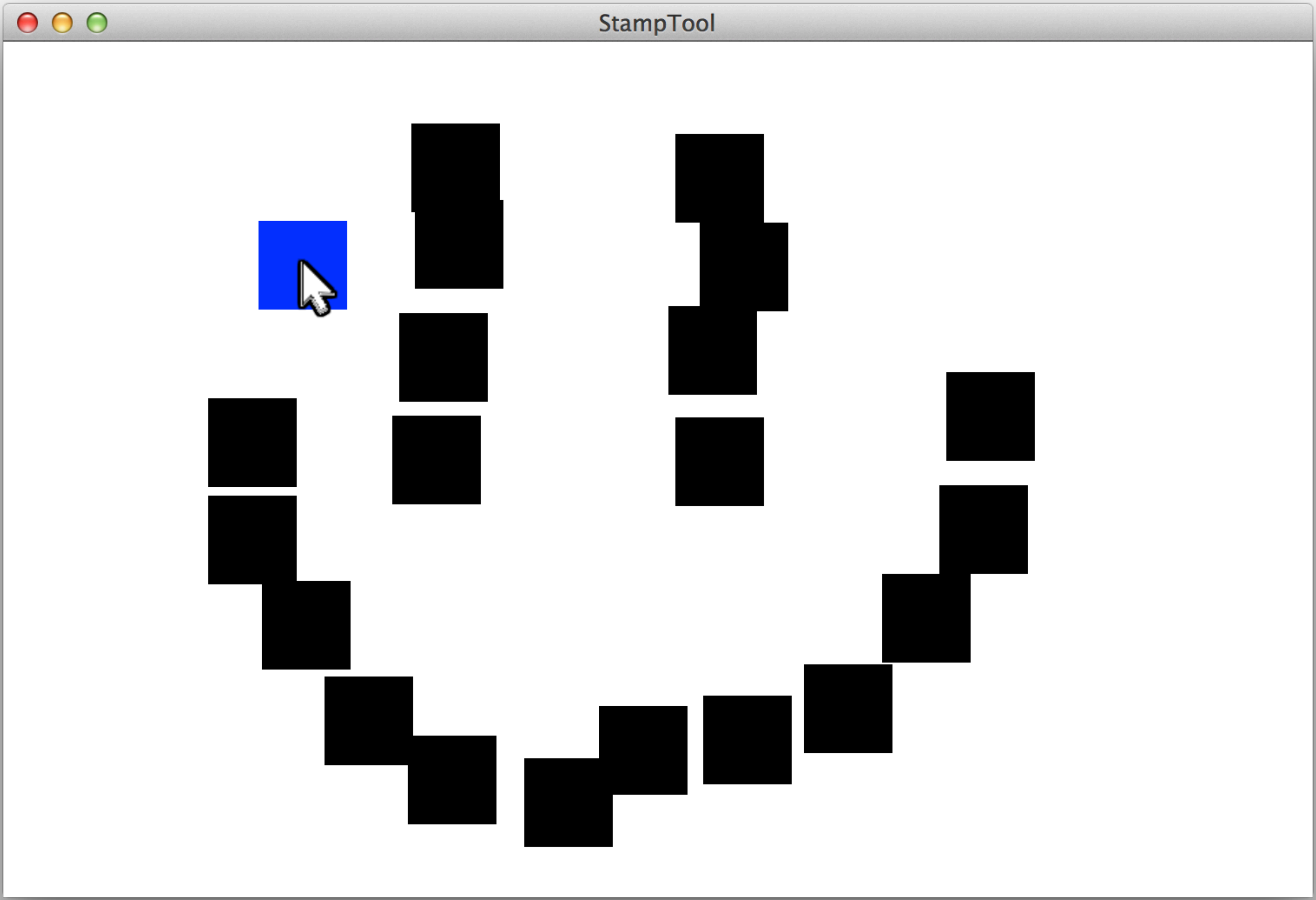
getElementAt(x, y);

- **`getElementAt(x, y)`** will return any GObject at the (x, y) coordinates. It will return **`null`** if there is no object at those coordinates.

```
public void mouseClicked(MouseEvent e) {
    int x = e.getX();
    int y = e.getY();
    GObject object = getElementAt(x, y);
    if (object != null) {
        remove(object);
    }
}
```

And Here We Are...

Stamp Tool



New Concepts

New Commands

- `addMouseListeners();`
- `getElementAt(`*`x`*`, `*`y`*`);`

New Ideas

- The Listener Model
- Instance Variables
- **`null`**

Responding to Mouse Events

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

mouseClicked(*e***)**	Called when the user clicks the mouse
mousePressed(*e***)**	Called when the mouse button is pressed
mouseReleased(*e***)**	Called when the mouse button is released
mouseMoved(*e***)**	Called when the user moves the mouse
mouseDragged(*e***)**	Called when the mouse is dragged with the button down

The parameter *e* is **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

keyPressed(*e***)**	Called when the user presses a key
keyReleased(*e***)**	Called when the key comes back up
keyTyped(*e***)**	Called when the user types (presses and releases) a key

The parameter *e* is a **KeyEvent** object, which indicates which key is involved.

Using portions of slides by Eric Roberts

Warm Up: Making Tracks

Catch Me If You Can?