
Primitives, Objects
and Traces

Primitives and Objects

- **Primitives:** int, double, boolean, char,…
- **Objects**: GRect, GOval, GLine, int[], … (anything with **new**, and that you call methods on)

Parameters

- **When passing parameters, make a copy of whatever is on the stack.**
- **Primitives:** the *actual value* is on the stack (pass by value)
- **Objects:** a *heap address* where the information lives is on the stack. (pass by reference)

Parameters: Primitives

```
public void run() {
     int x = 2;
     addTwo(x);
     println(x);  // x is still 2!
}

private void addTwo(int y) {
     y += 2;
}
```

Parameters: Objects

```
public void run() {
     GRect rect = new Grect(0,0,50,50);
     fillBlue(rect);
     add(rect);    // rect is blue!
}

private void fillBlue(GRect rect) {
     rect.setFilled(true);
     rect.setColor(Color.BLUE);
}
```

Program Traces

- **Approaching program traces**
  - Local variables are *separate* across methods
  - Parameters are just assigned names by the order in which they're passed
  - Write values above variable names as you go through the program (or draw stack frame boxes)
  - Pass-by-reference vs. pass-by-value

```
private void mystery(int[][] arr) {
    bloom(arr);
    frolic(arr[1][1]);
}

private void bloom(int[][] field) {
    for(int i = 0; i < field[0].length; i++) {
        field[0][i] += field[0][i - 1];
    }
}

private void frolic(int num) {
    int birds = num * 2;
    int bees = num % 2;
    num =  birds + bees;
}
```

0	**1**	**2**
3	**4**	**5**

Input to **mystery()**

What is **arr** after?

Take 1

```
private void mystery(int[][] arr) {
    bloom(arr);
    arr[1][1] = frolic(arr[1][1]);
}

private void bloom(int[][] field) {
    for(int i = 0; i < field[0].length; i++) {
        field[0][i] += field[0][i - 1];
    }
}

private int frolic(int num) {
    int birds = num * 2;
    int bees = num % 2;
    return birds + bees;
}
```

0	1	2
3	4	5

Input to **mystery()**

What is **arr** after?

Take 2


Event Drive
Programs

Events

- **Three** types of events: keyboard, mouse, interactors
- **Two** ways for Java to execute your code: from run() and from event handlers (mouseClicked, mouseMoved, actionPerformed, etc.)
- These programs are **asynchronous**; your code is not run in order anymore, since you don't know when the user will interact with your program!

Mouse Events

1. Sign up for notifications for key or mouse events
2. Implement the method corresponding to what event you care about (e.g. mousePressed, mouseMoved)
3. Java will call that method whenever the corresponding event occurs.

Interactors

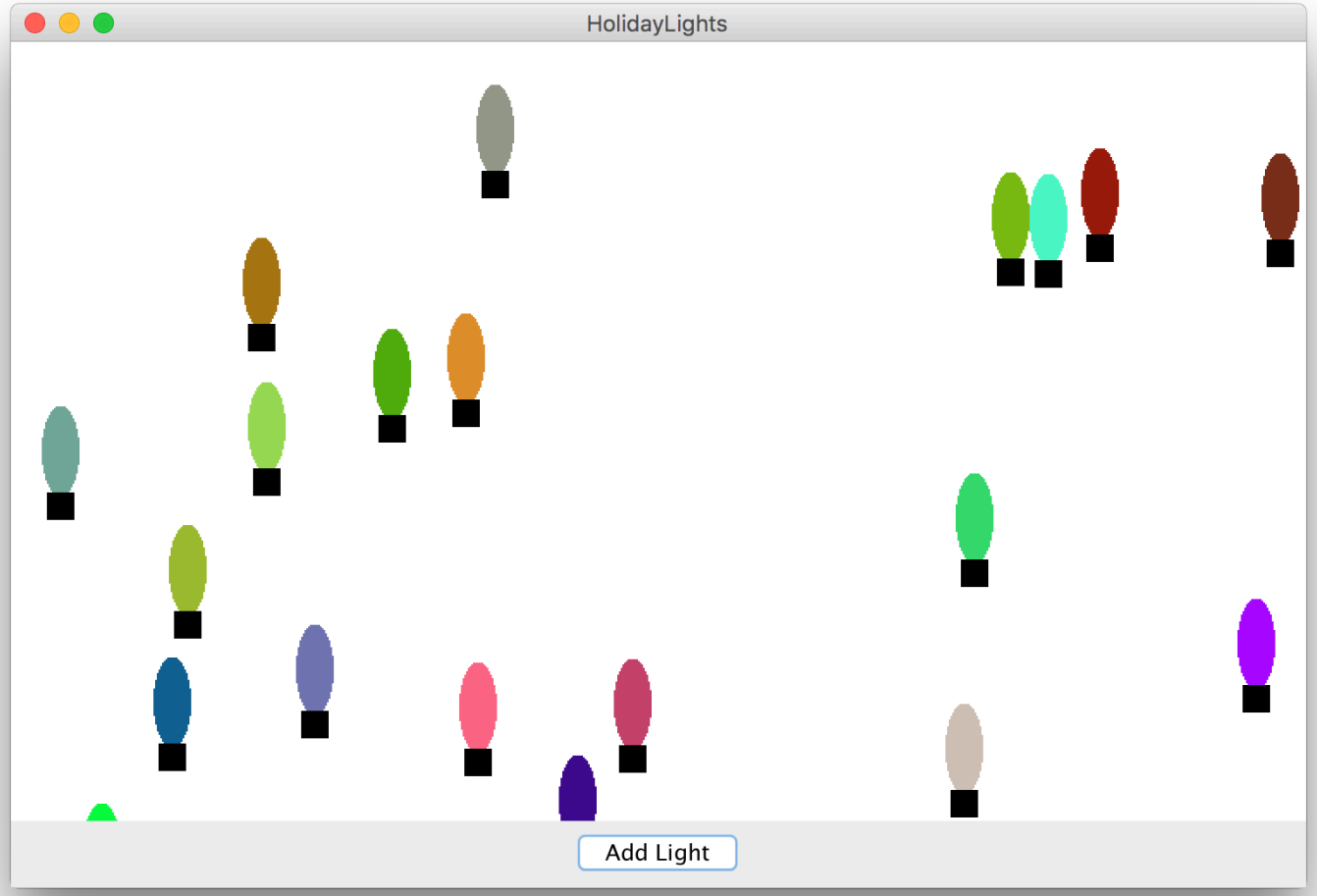
1. Add interactors in init()
2. addActionListeners() to listen for button presses
3. .addActionListener(this) on text fields for ENTER
4. Implement actionPerformed
5. Java will call actionPerformed whenever an action event occurs.

Interactors

```
public void actionPerformed(ActionEvent e) {
      if (e.getActionCommand().equals("My Button")) {
            ...
      }
}

// ... equivalent to ...
public void actionPerformed(ActionEvent e) {
      if (e.getSource() == myJButton) {
            ...
      }
}
```

Midsummer's Night Lights



Midsummer's Night Lights

- Add a new light by clicking "Add Light"
- Every second, every light on screen changes to a new random color

```
private static final int BULB_V_RADIUS = 25;
private static final int BULB_H_RADIUS = 10;
private static final int BULB_BASE_SIZE = 15;
private static final int PAUSE_TIME = 1000;
```

Instance Variables

```
// List of all bulbs on the screen
private ArrayList<GOval> bulbs = new ArrayList<GOval>();

// To change the bulb color and position bulbs
private RandomGenerator rgen = RandomGenerator.getInstance();
```

Init()

```
public void init() {
    add(new JButton("Add Light"), SOUTH);
    addActionListeners();
}
```

run()

```
public void run() {
    while (true) {
        for (int i = 0; i < bulbs.size(); i++) {
            bulbs.get(i).setColor(rgen.nextColor());
        }
        pause(PAUSE_TIME);
    }
}
```

```
public void actionPerformed(ActionEvent e) {
      if (e.getActionCommand().equals("Add Light")) {
            GOval bulb = new GOval(2*BULB_H_RADIUS, 2*BULB_V_RADIUS);
            bulb.setFilled(true);
            bulb.setColor(Color.RED);

            GRect base = new GRect(BULB_BASE_SIZE, BULB_BASE_SIZE);
            base.setFilled(true);
            base.setColor(Color.BLACK);

            int x = rgen.nextInt(0, getWidth());
            int y = rgen.nextInt(0, getHeight());
            add(bulb, x – BULB_H_RADIUS, y);
            add(base, x - BULB_BASE_SIZE / 2.0, y + 2*BULB_V_RADIUS);
            bulbs.add(bulb);
      }
}
```

```
public void actionPerformed(ActionEvent e) {
    if (e.getActionCommand().equals("Add Light")) {
        GOval bulb = new GOval(2*BULB_H_RADIUS, 2*BULB_V_RADIUS);
        bulb.setFilled(true);
        bulb.setColor(Color.RED);

        GRect base = new GRect(BULB_BASE_SIZE, BULB_BASE_SIZE);
        base.setFilled(true);
        base.setColor(Color.BLACK);

        int x = rgen.nextInt(0, getWidth());
        int y = rgen.nextInt(0, getHeight());
        add(bulb, x – BULB_H_RADIUS, y);
        add(base, x - BULB_BASE_SIZE / 2.0, y + 2*BULB_V_RADIUS);
        bulbs.add(bulb);
    }
}
```

```
public void actionPerformed(ActionEvent e) {
      if (e.getActionCommand().equals("Add Light")) {
            GOval bulb = new GOval(2*BULB_H_RADIUS, 2*BULB_V_RADIUS);
            bulb.setFilled(true);
            bulb.setColor(Color.RED);

            GRect base = new GRect(BULB_BASE_SIZE, BULB_BASE_SIZE);
            base.setFilled(true);
            base.setColor(Color.BLACK);

            int x = rgen.nextInt(0, getWidth());
            int y = rgen.nextInt(0, getHeight());
            add(bulb, x – BULB_H_RADIUS, y);
            add(base, x - BULB_BASE_SIZE / 2.0, y + 2*BULB_V_RADIUS);
            bulbs.add(bulb);
      }
}
```

```
public void actionPerformed(ActionEvent e) {
      if (e.getActionCommand().equals("Add Light")) {
            GOval bulb = new GOval(2*BULB_H_RADIUS, 2*BULB_V_RADIUS);
            bulb.setFilled(true);
            bulb.setColor(Color.RED);

            GRect base = new GRect(BULB_BASE_SIZE, BULB_BASE_SIZE);
            base.setFilled(true);
            base.setColor(Color.BLACK);

            int x = rgen.nextInt(0, getWidth());
            int y = rgen.nextInt(0, getHeight());
            add(bulb, x – BULB_H_RADIUS, y);
            add(base, x - BULB_BASE_SIZE / 2.0, y + 2*BULB_V_RADIUS);
            bulbs.add(bulb);
      }
}
```

```
public void actionPerformed(ActionEvent e) {
      if (e.getActionCommand().equals("Add Light”)) {
            GOval bulb = new GOval(2*BULB_H_RADIUS, 2*BULB_V_RADIUS);
            bulb.setFilled(true);
            bulb.setColor(Color.RED);

            GRect base = new GRect(BULB_BASE_SIZE, BULB_BASE_SIZE);
            base.setFilled(true);
            base.setColor(Color.BLACK);

            int x = rgen.nextInt(0, getWidth());
            int y = rgen.nextInt(0, getHeight());
            add(bulb, x – BULB_H_RADIUS, y);
            add(base, x - BULB_BASE_SIZE / 2.0, y + 2*BULB_V_RADIUS);
            bulbs.add(bulb);
      }
}
```

```
public void actionPerformed(ActionEvent e) {
      if (e.getActionCommand().equals("Add Light”)) {
            GOval bulb = new GOval(2*BULB_H_RADIUS, 2*BULB_V_RADIUS);
            bulb.setFilled(true);
            bulb.setColor(Color.RED);

            GRect base = new GRect(BULB_BASE_SIZE, BULB_BASE_SIZE);
            base.setFilled(true);
            base.setColor(Color.BLACK);

            int x = rgen.nextInt(0, getWidth());
            int y = rgen.nextInt(0, getHeight());
            add(bulb, x – BULB_H_RADIUS, y);
            add(base, x - BULB_BASE_SIZE / 2.0, y + 2*BULB_V_RADIUS);
            bulbs.add(bulb);
      }
}
```

```
public void actionPerformed(ActionEvent e) {
      if (e.getActionCommand().equals("Add Light”)) {
            GOval bulb = new GOval(2*BULB_H_RADIUS, 2*BULB_V_RADIUS);
            bulb.setFilled(true);
            bulb.setColor(Color.RED);

            GRect base = new GRect(BULB_BASE_SIZE, BULB_BASE_SIZE);
            base.setFilled(true);
            base.setColor(Color.BLACK);

            int x = rgen.nextInt(0, getWidth());
            int y = rgen.nextInt(0, getHeight());
            add(bulb, x – BULB_H_RADIUS, y);
            add(base, x - BULB_BASE_SIZE / 2.0, y + 2*BULB_V_RADIUS);
            bulbs.add(bulb);
      }
}
```

```
public void actionPerformed(ActionEvent e) {
      if (e.getActionCommand().equals("Add Light")) {
            GOval bulb = new GOval(2*BULB_H_RADIUS, 2*BULB_V_RADIUS);
            bulb.setFilled(true);
            bulb.setColor(Color.RED);

            GRect base = new GRect(BULB_BASE_SIZE, BULB_BASE_SIZE);
            base.setFilled(true);
            base.setColor(Color.BLACK);

            int x = rgen.nextInt(0, getWidth());
            int y = rgen.nextInt(0, getHeight());
            add(bulb, x – BULB_H_RADIUS, y);
            add(base, x - BULB_BASE_SIZE / 2.0, y + 2*BULB_V_RADIUS);
            bulbs.add(bulb);
      }
}
```