

Interactors

Brahm Capoor
brahm@stanford.edu

Learning goals for today

To learn how to use **interactors** in our programs

To go **under the hood** of a program

To see how we can use Computer Science to **understand people**

‘...a device that [...] had about a hundred tiny flat press buttons and a screen about four inches square on which any one of a million “pages” could be summoned at a moment’s notice. It looked insanely complicated, and [...] had the words **DON’T PANIC** printed on it in large friendly letters’

- Douglas Adams, *The Hitchhiker’s Guide to the Galaxy*

How do we interact with programs?

As Console Programs

```
public class myProgram extends ConsoleProgram {...}
```

Using our Mouse and Keyboard

```
public void mouseMoved(MouseEvent e){...}
```

Using UI Elements (buttons, sliders, text fields)

// ^_(ツ)_/^-

Using the tools we already have,
how could we make a button?

What's wrong with this approach to making buttons?

Not a separate part of the interface

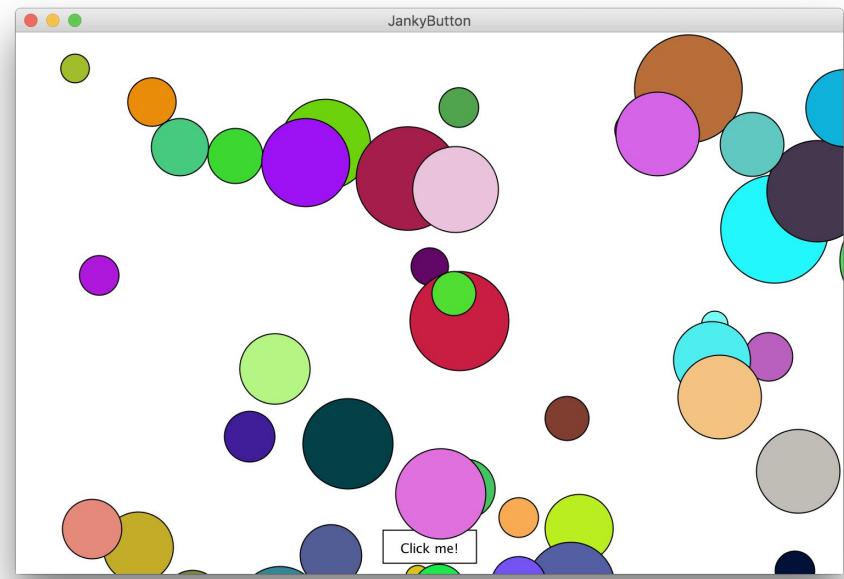
Doesn't give any indication that it was clicked

Looks pretty bad

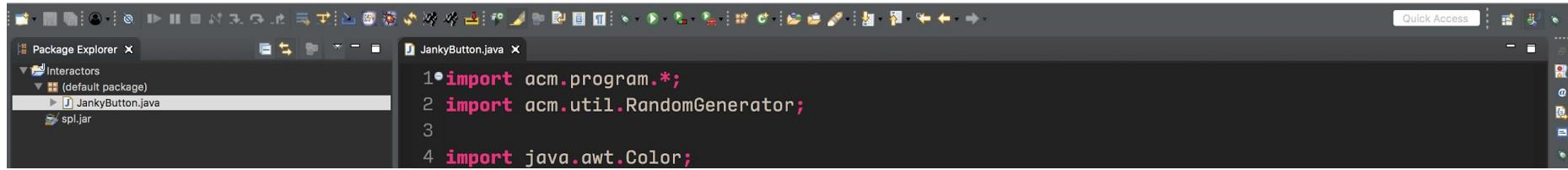
Inconsistent with other programs

Can't use it in ConsolePrograms

Lots of work to create

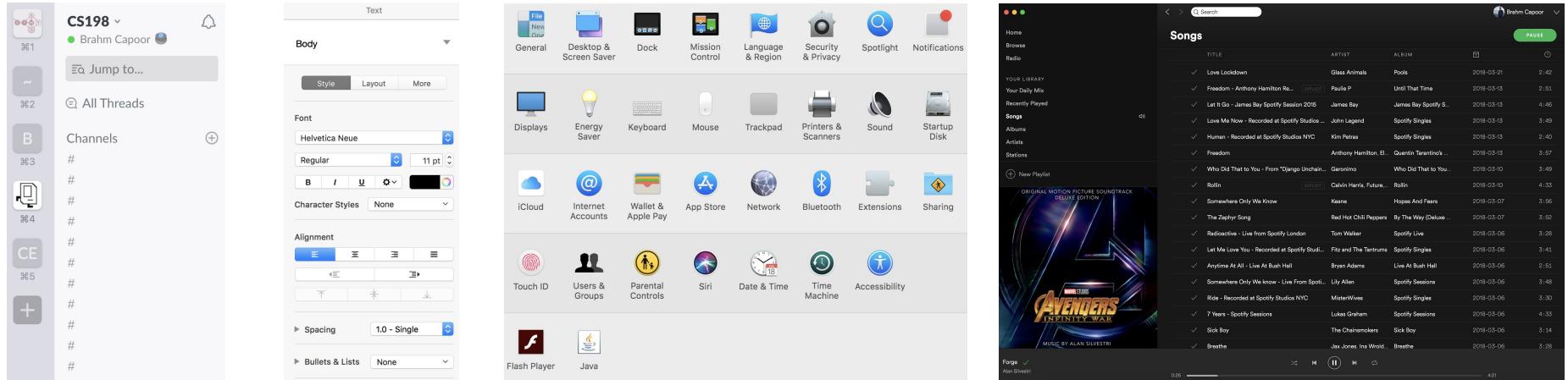


Making these interfaces would be devastating



A screenshot of an IDE interface. On the left is the 'Package Explorer' showing a project named 'Interactors' containing a file 'JankyButton.java'. The main area displays the following Java code:

```
1•import acm.program.*;
2 import acm.util.RandomGenerator;
3
4 import java.awt.Color;
```

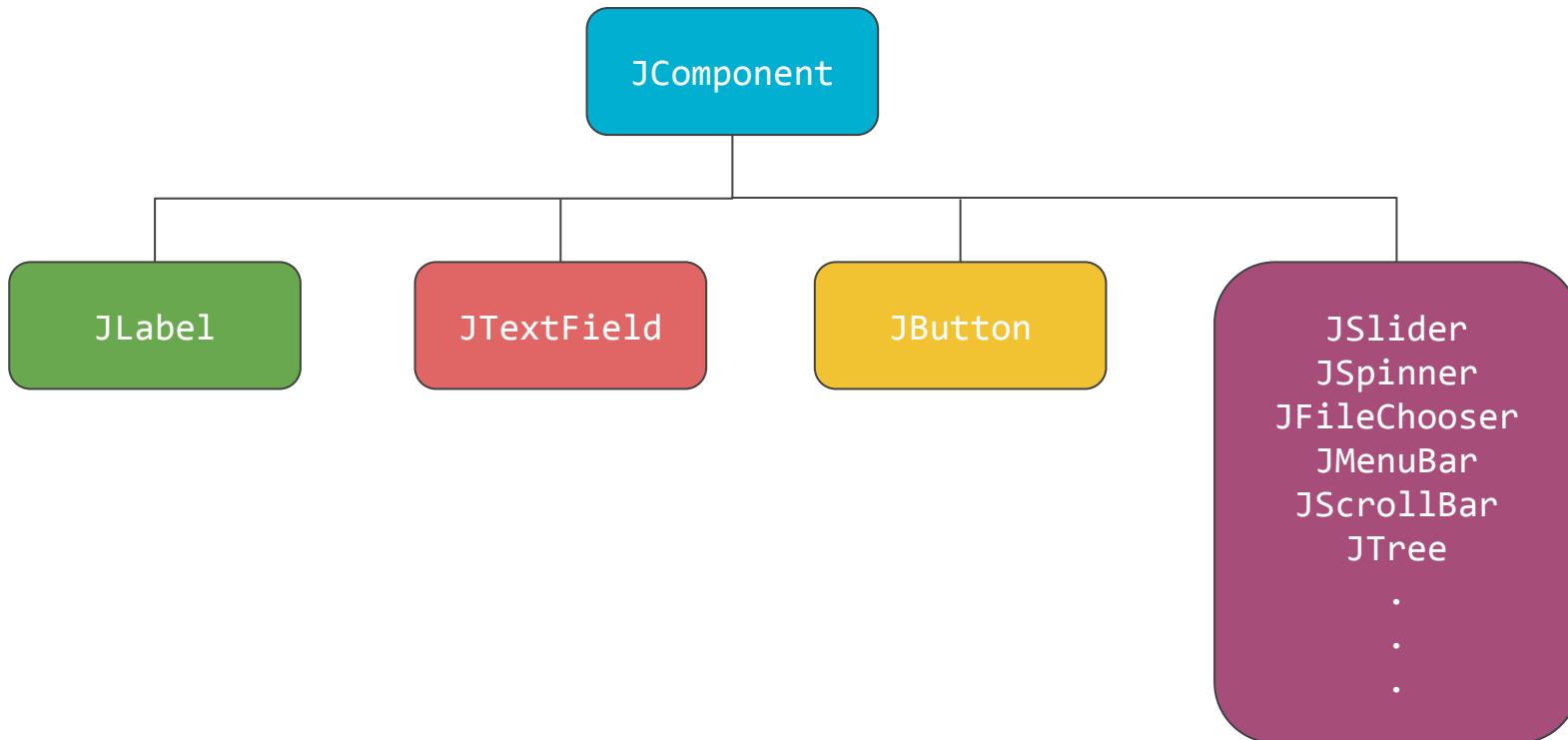


Three screenshots illustrating user interface design:

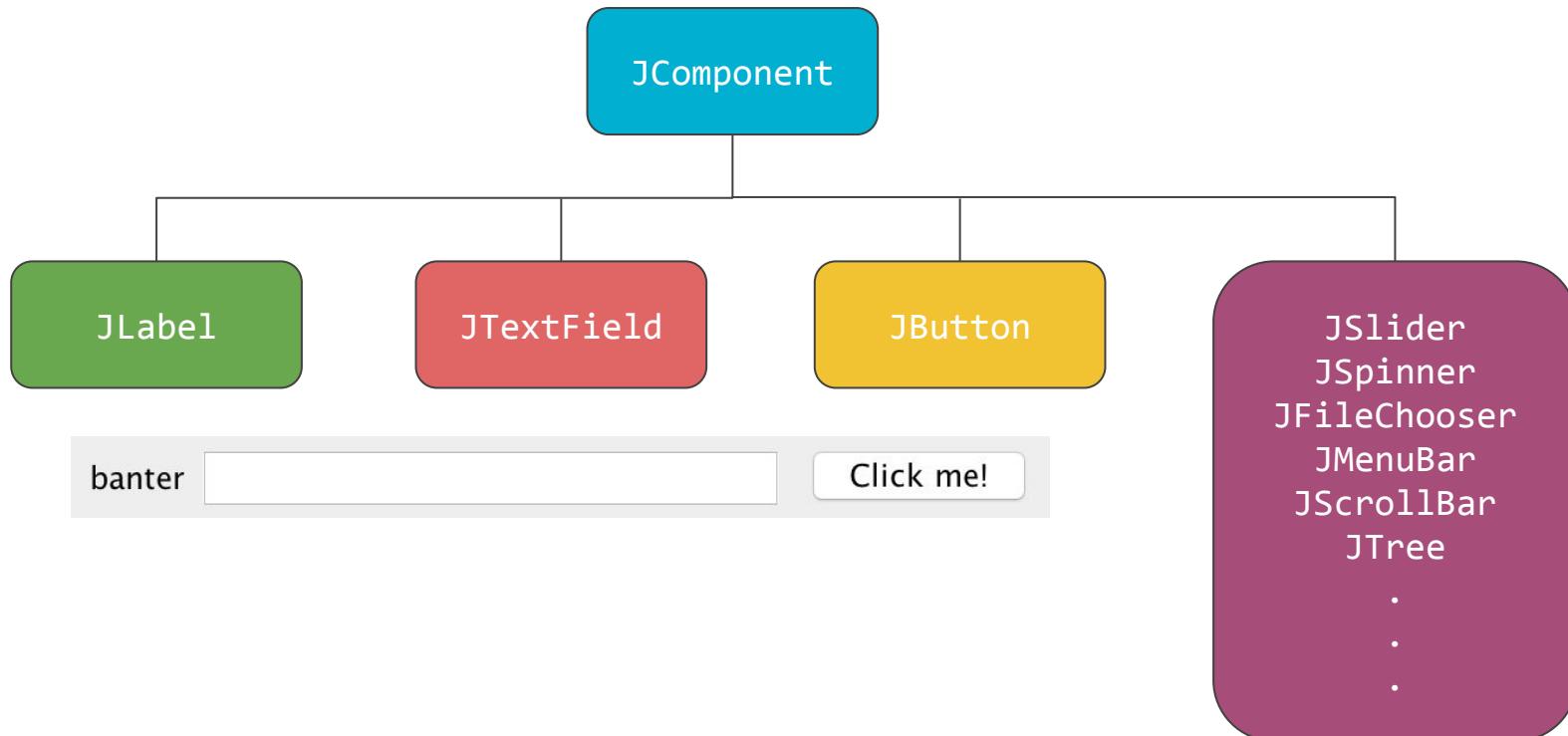
- Left Screenshot:** A messaging sidebar for 'CS198'. It shows a list of channels (#1 through #6) and a list of messages. One message is highlighted with a grey background and the text 'Jump to...'. Other messages include 'All Threads' and 'B'.
- Middle Screenshot:** A 'Text' editor interface. It includes a toolbar with 'Style', 'Layout', and 'More' buttons. Below the toolbar are sections for 'Font' (set to 'Helvetica Neue'), 'Character Styles' (None), 'Alignment' (centered), 'Spacing' (1.0 - Single), and 'Bullets & Lists' (None). To the right is a large grid of icons representing various system settings and services, such as General, Desktop & Screen Saver, Dock, Mission Control, Language & Region, Security & Privacy, Spotlight, Notifications, Displays, Energy Saver, Keyboard, Mouse, Trackpad, Printers & Scanners, Sound, Startup Disk, iCloud, Internet Accounts, Wallet & Apple Pay, App Store, Network, Bluetooth, Extensions, Sharing, Touch ID, Users & Groups, Parental Controls, Siri, Date & Time, Time Machine, Accessibility, Flash Player, and Java.
- Right Screenshot:** A media player interface for Spotify. It shows a library of songs, including 'Love Lockdown' by Glass Animals, 'Freedom' by Anthony Hamilton, 'Let It Go' by James Bay, 'Love Me Now' by John Legend, 'Human' by Kim Petras, 'Freedom' by Anthony Hamilton, 'Who Did That To You' by Calvin Harris, 'Rollin' by Calvin Harris, 'Somewhere Only We Know' by Keane, 'The Zephyr Song' by Red Hot Chili Peppers, 'Radioactive' by Tom Walker, 'Let Me Love You' by Pitbull, 'Anytime At All' by Bush Hall, 'Somewhere Only We Know' by Uly Allen, 'Ride' by Lukas Graham, '7 Years' by Lukas Graham, and 'Breathe' by Alan Jackson. The current track is 'Avengers: Infinity War' by Alan Silvestri. The interface includes a search bar, a song list, and playback controls.

Programming is about standing on the shoulders
of giants

Meet today's giant



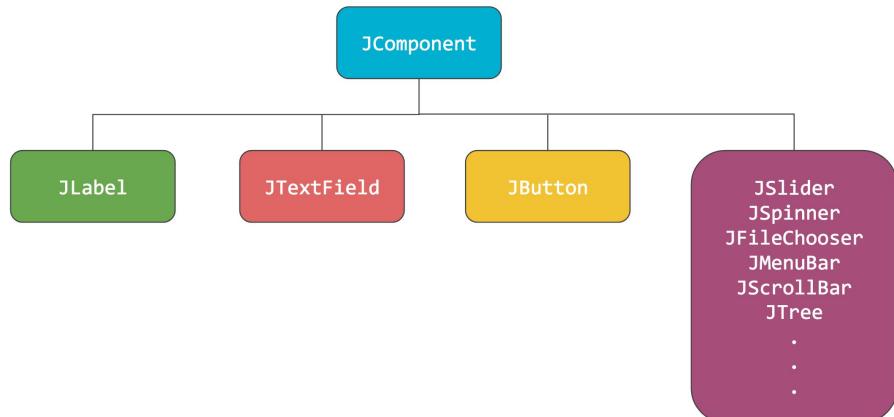
Meet today's giant



JComponents

Java handles how they look

You handle how they work



Our first JComponents

```
JLabel label = new JLabel("banter");

JTextField field = new JTextField(20);      // 20 characters wide

JButton button = new JButton("Click me");

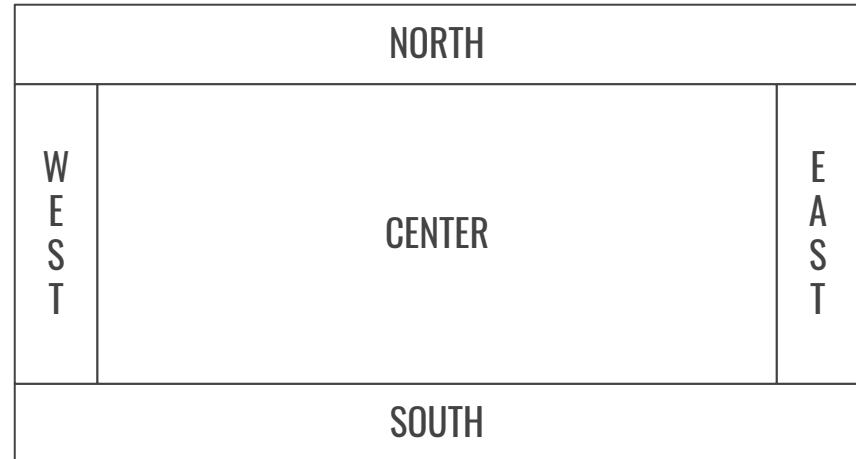
// how do we add these to the window?
```

Regions in a window

Java divides every window into 5 regions

Center: your `ConsoleProgram` or
`GraphicsProgram`

The other regions only show up when you **add things** to them



Putting JComponents on the window

```
JLabel label = new JLabel("banter");

JTextField field = new JTextField(20);      // 20 characters wide

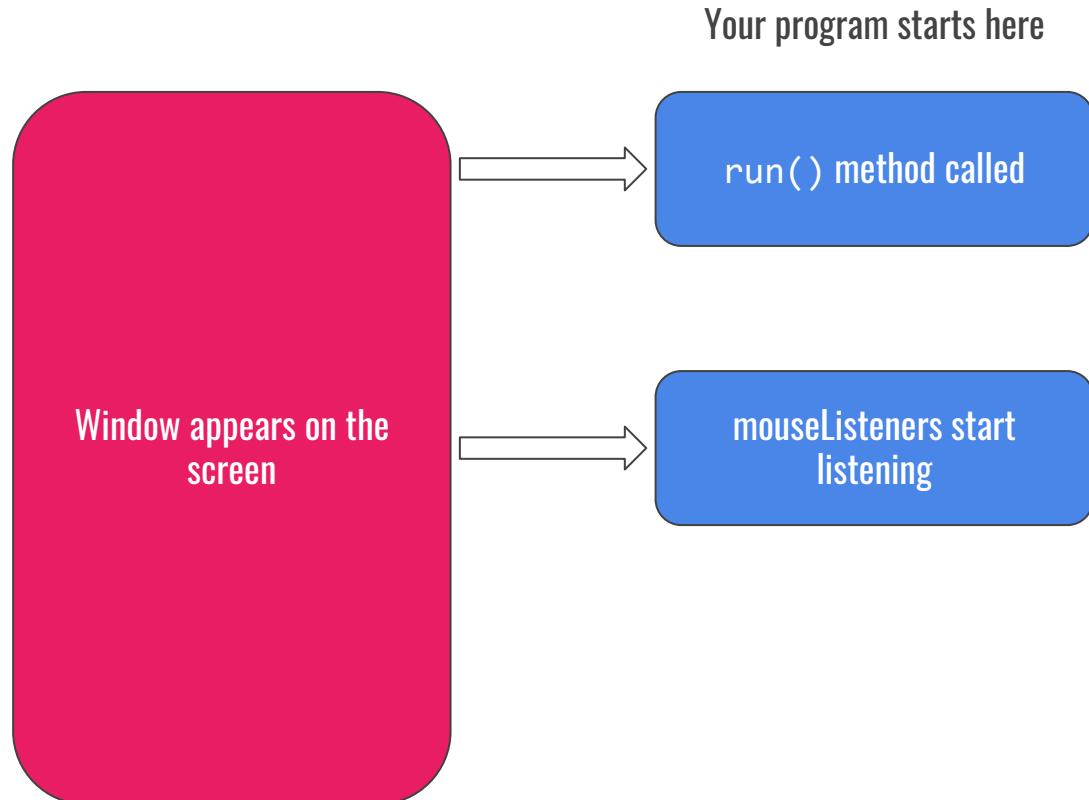
JButton button = new JButton("Click me");

add(label, SOUTH);
add(field, SOUTH);
add(button, SOUTH); }
```

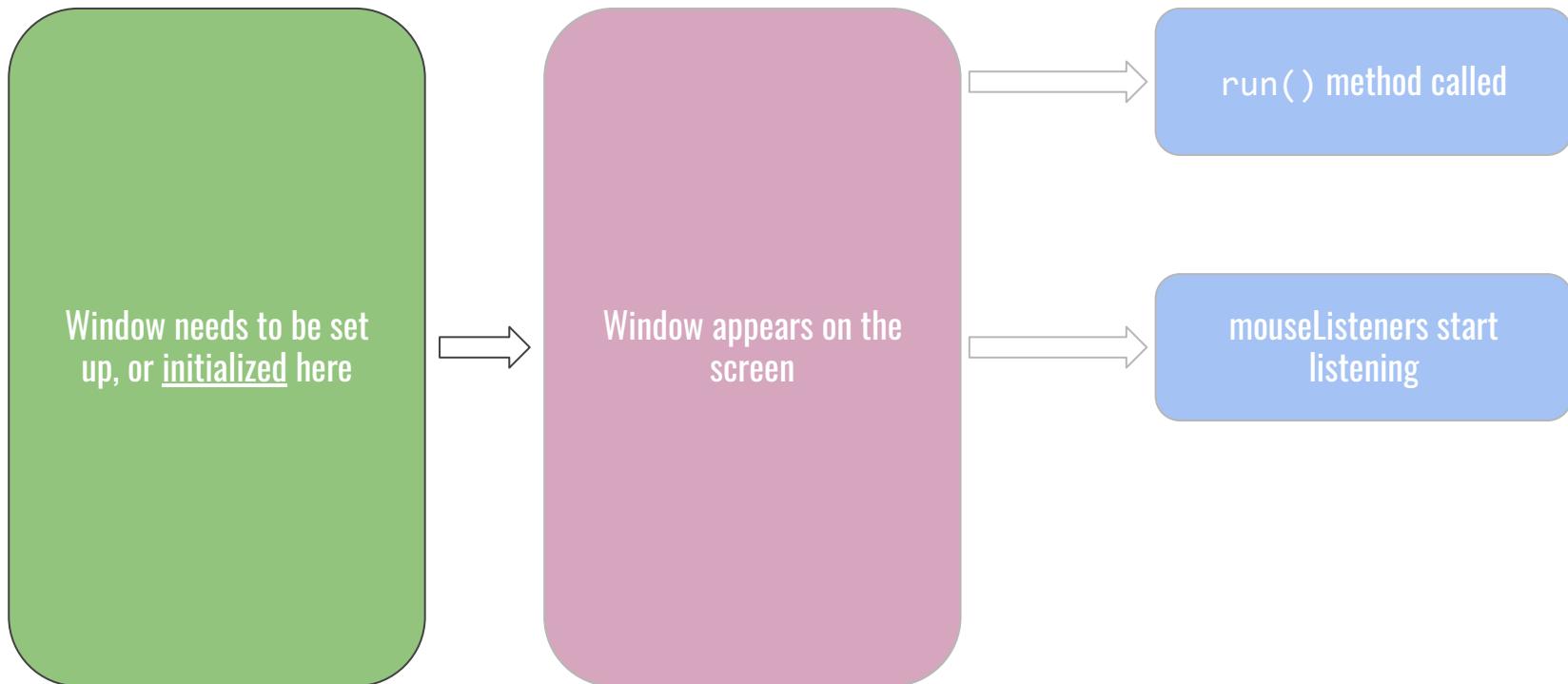
Java automatically arranges the components in the SOUTH region for you

Let's run() with it

What we know so far



Diving under the hood

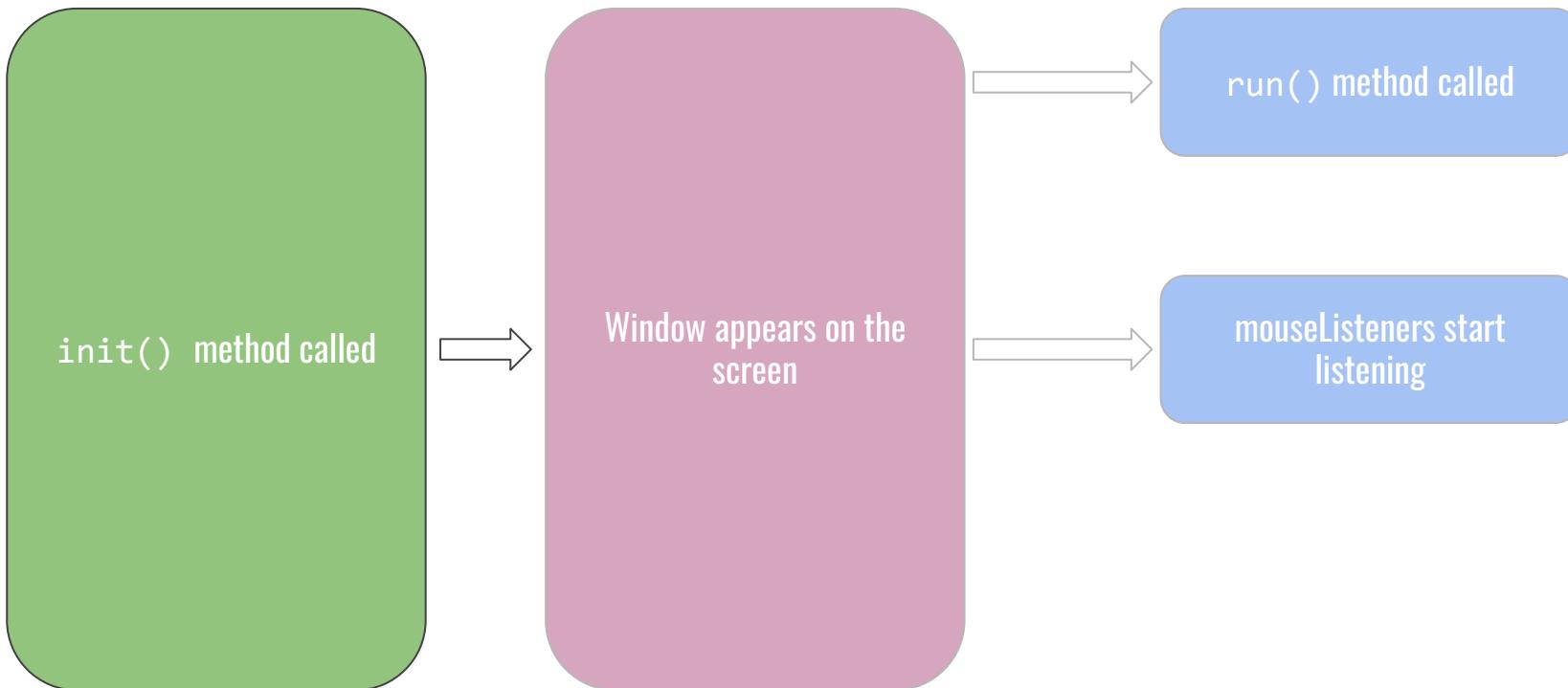


Diving under the hood

Your program starts here

```
Window window;
public void init() {
    // Set up the window here
}
```

Diving under the hood



Let's run with init()

// (sorry)

The takeaway: add JComponents in init()

How to use JComponents

Where else have our programs had to respond to user actions that could happen anytime?

Like MouseListeners, using components requires **Event-Driven programming**

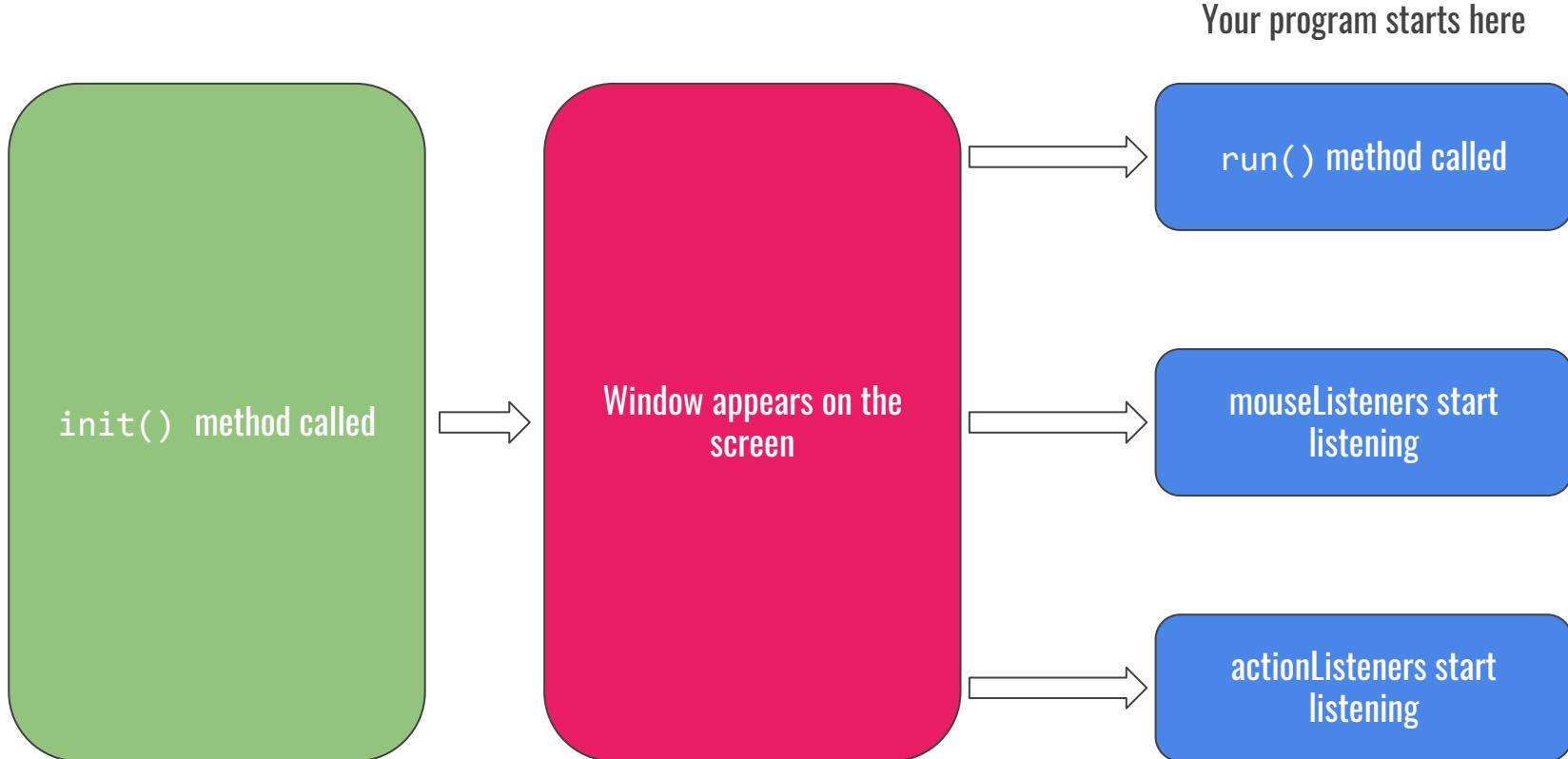
```
public void actionPerformed(ActionEvent e){  
    String command = e.getActionCommand();  
    // Process command  
}
```

```
public void init() {
    JButton button = new JButton("Click me!");
    field.addActionListener(this);      // enable pressing enter
    field.setActionCommand("Typed");   // set the field's action command

    add(button, SOUTH);
    add(field, SOUTH);
    addActionListeners();           // start listening for user actions
}

public void actionPerformed(ActionEvent e){
    String command = e.getActionCommand();
    if (command.equals("Click me!")) {
        println("Button clicked!");
    }
    if (command.equals("Typed")) {
        println(field.getText());     // needs to be an instance variable
    }
}
```

Diving under the hood



Let's make something cool!

The [xkcd](#) Color survey: what names do people give colors?

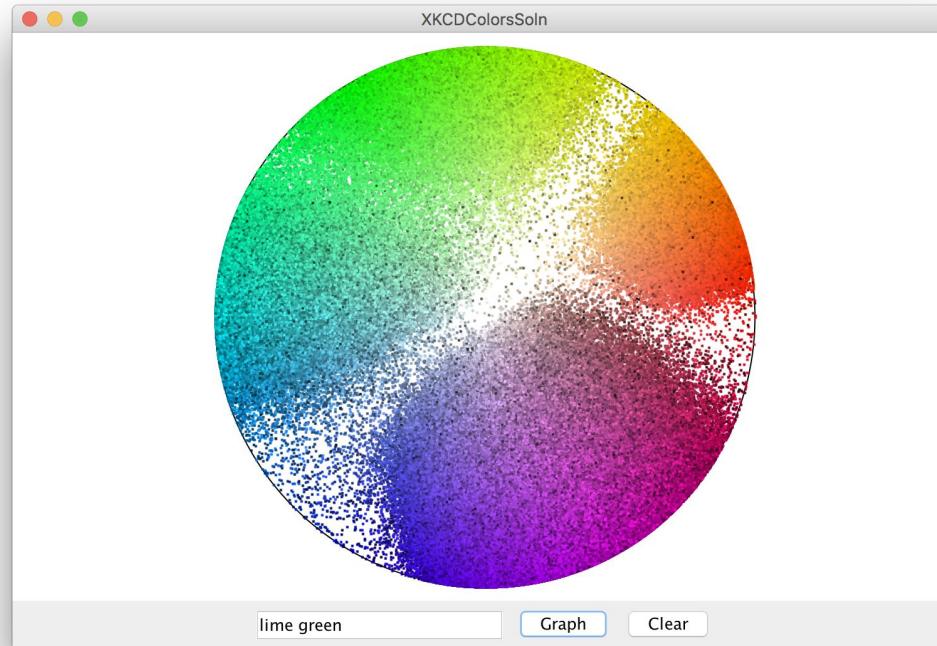
Dataset: 2.85 million RGB colors + names

navy blue	
27	
34	
98	
blue	
41	
201	
234	
lime green	
99	
212	
32	
red brown	
160	
89	
66	
.	
.	
.	

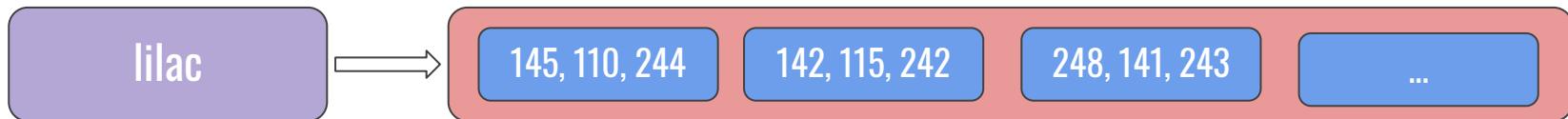
This is a lot of data

11401401	midnight	blue
11401402	19	
11401403	14	
11401404	78	

A cool visualization!

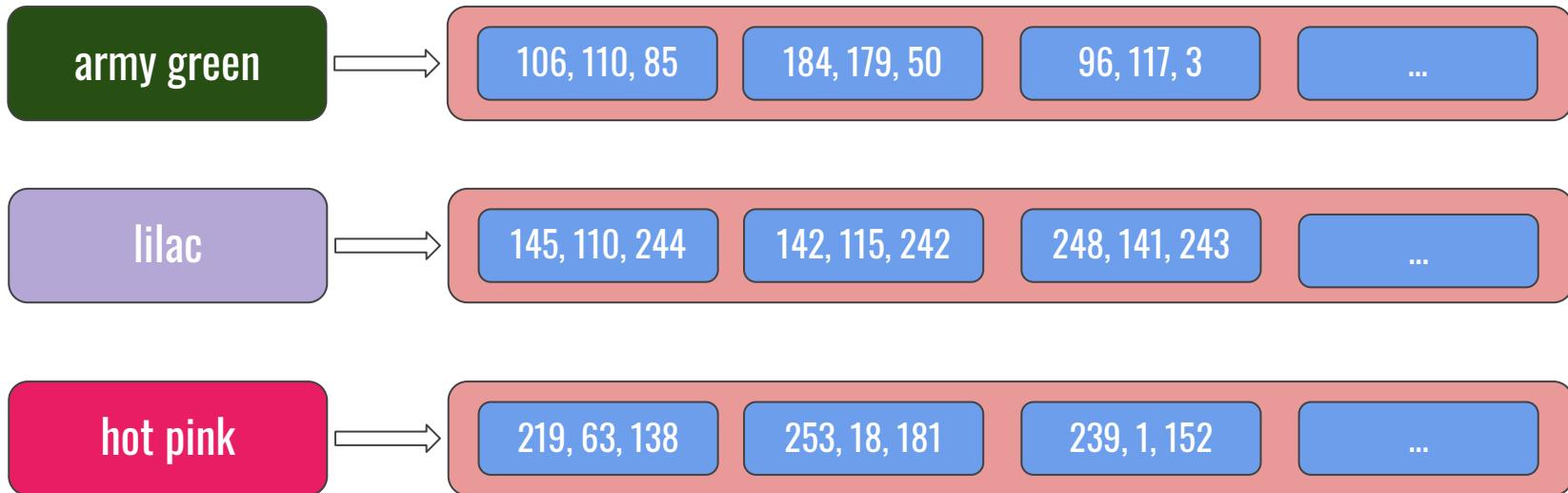


Milestone 1: How do we represent this data?



Hint: each of these is a Color

Milestone 1: How do we represent this data? ✓



```
HashMap<String, ArrayList<Color>> colorMap;
```

Milestone 2: How do we load data from the file?

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

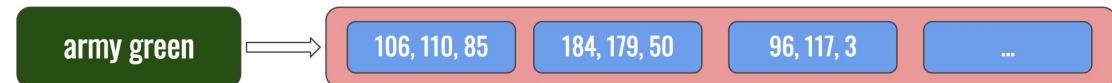
89

66

.

.

.



lilac

hot pink

106, 110, 85

184, 179, 50

96, 117, 3

145, 110, 244

142, 115, 242

248, 141, 243

219, 63, 138

253, 18, 181

239, 1, 152

```
private HashMap<String, ArrayList<Color>> readFile() {  
    // fun Scanner shenanigans  
}
```

Milestone 2: How do we load data from the file? ✓

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

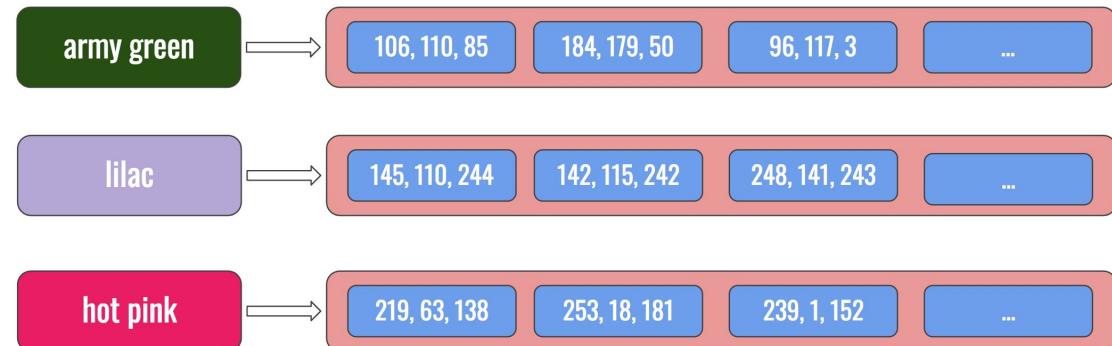
89

66

.

.

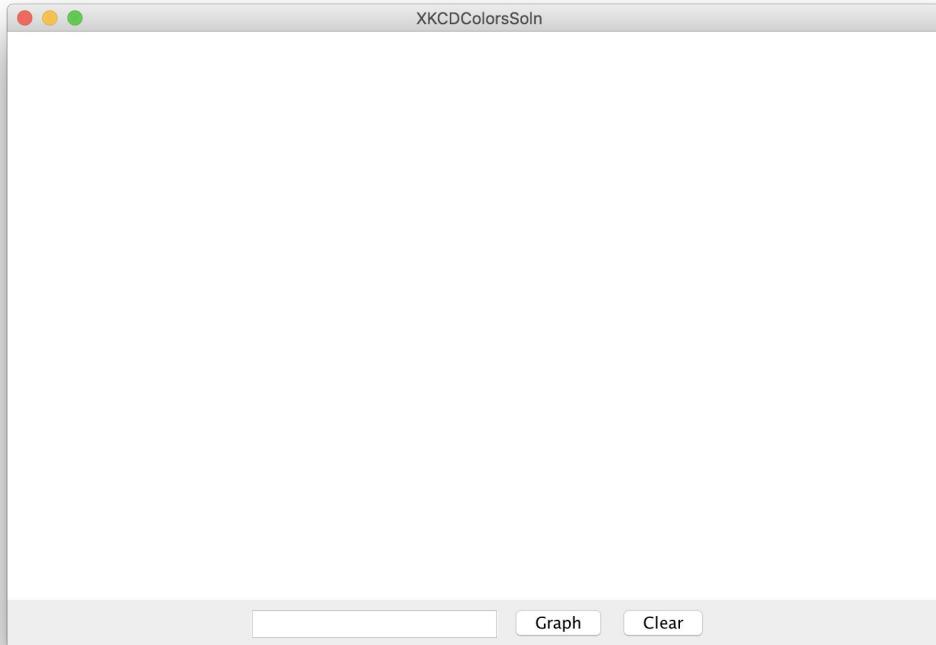
.



```
private HashMap<String, ArrayList<Color>> readFile() {  
    // fun Scanner shenanigans  
}
```

[Click here for an animation of the file reading](#)

Milestone 3: How do we set up the interactors?



Milestone 4: How do we put all the pieces together?

Suppose you have a method

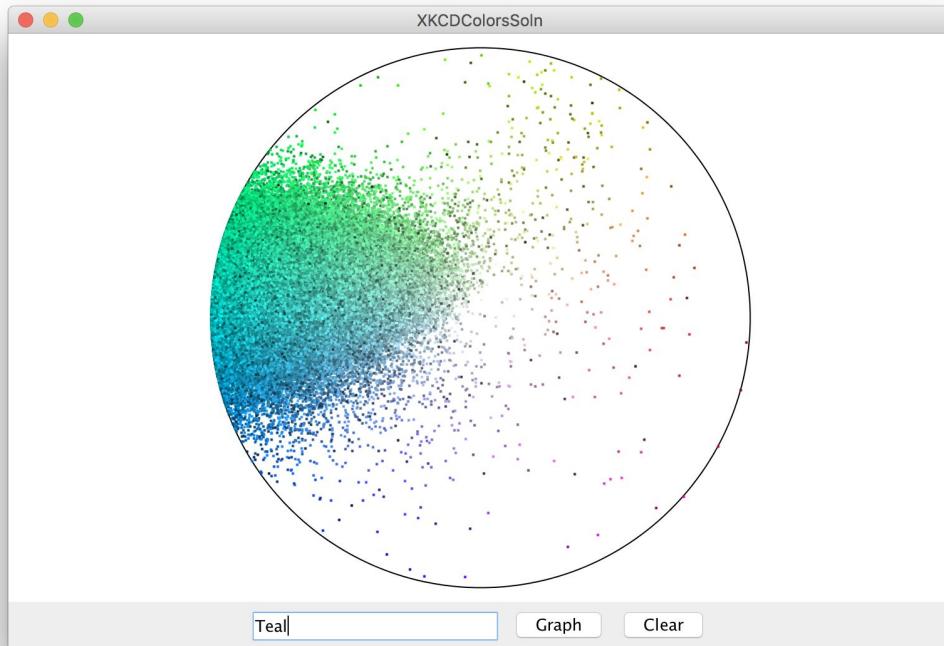
```
private void plotColor(Color color)
```

that puts a single color dot on the screen in the correct place

and another method

```
private void clearAll()
```

that removes all the colored dots



Computer Science helps us learn about people

xkcd's analysis of the results

Overflow slides

1. How do we plot a color?
2. A file reading demo

How do we plot a color in xkcd colors?

The HSB Color Space

// (you're not required to know this)

How do we plot a color?

We normally break colors into a **red**, **green** and **blue** component

You can think of each color as a point on a 3d graph with a **red**, **green** and **blue** axis

This graph is called the **RGB Color Space**

Each axis on the graph goes from 0 to 255



[RGB Color Space Atlas](#)

How do we plot a color?

We don't **have** to break a color into RGB values

RGB is easy for computers to understand, but **not** for humans

What does it **mean** for a color to be 127 / 255 red? 

We tend to think of a color in terms of its **general color range**, how **vivid** it is, and how **bright** it is.

How do we plot a color?

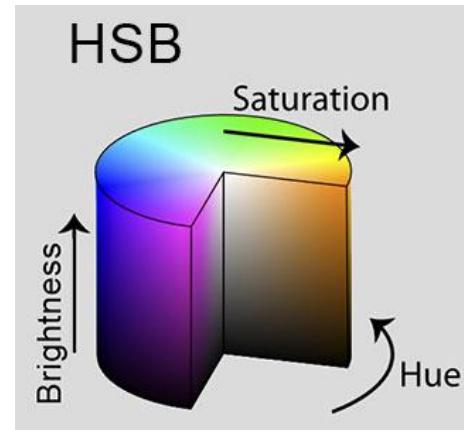
We can also break a color into these three components:

The color's **hue** represents its general color range (its location on the color wheel)

The color's **saturation** represents how vivid it is

The color's **brightness** represents how bright it is

Each color is now a point in the **HSB Color Space**



[source](#)

How do we plot a color?

Java gives us a method to break a color up into HSB Components:

```
float[] HSBComponents = Color.RGBtoHSB(color.getRed(), color.getGreen(), color.getBlue(), null);
```

HSBComponents has **three** elements, which are in order:

1. color's hue
2. color's saturation
3. color's brightness

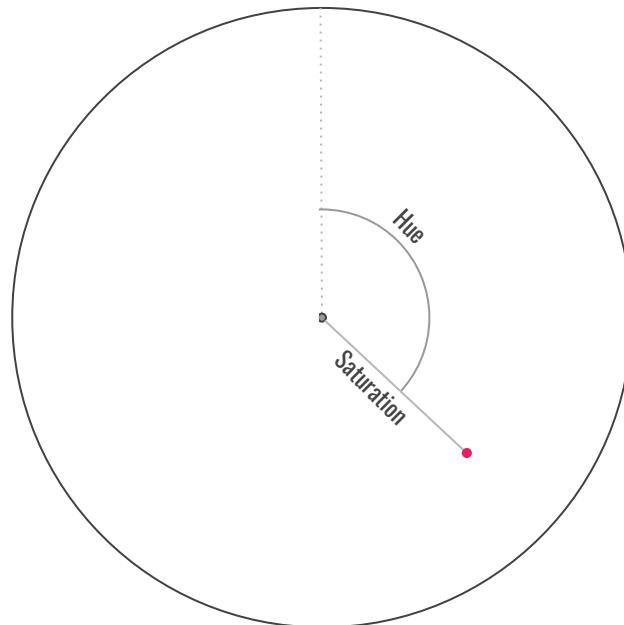
don't worry about the
float data type!

(it's like a double, but for
smaller numbers)

How do we plot a color?

We use the color's **hue** and **saturation** to figure out where in the circle the color's point goes

We color the point with its **corresponding color**, which captures brightness



How do we plot a color?

```
float[] components = Color.RGBtoHSB(color.getRed(),
                                      color.getGreen(),
                                      color.getBlue(),
                                      null);

double radius = getRadius() * components[1];
double theta = components[0] * Math.PI * 2.0;

double x = getWidth() / 2.0 + radius * Math.cos(theta);
double y = getHeight() / 2.0 - radius * Math.sin(theta);

GRect pt = new GRect(x, y, 1, 1);
pt.setFilled(true);
pt.setColor(color);
```

Get HSB Components from
color

How do we plot a color?

```
float[] components = Color.RGBtoHSB(color.getRed(),
                                      color.getGreen(),
                                      color.getBlue(),
                                      null);

double radius = getRadius() * components[1];
double theta = components[0] * Math.PI * 2.0;

double x = getWidth() / 2.0 + radius * Math.cos(theta);
double y = getHeight() / 2.0 - radius * Math.sin(theta);

GRect pt = new GRect(x, y, 1, 1);
pt.setFilled(true);
pt.setColor(color);
```

radius is based on saturation (as a fraction of 100) and angle is based on hue (as a fraction of 360)

How do we plot a color?

```
float[] components = Color.RGBtoHSB(color.getRed(),
                                      color.getGreen(),
                                      color.getBlue(),
                                      null);

double radius = getRadius() * components[1];
double theta = components[0] * Math.PI * 2.0;

double x = getWidth() / 2.0 + radius * Math.cos(theta);
double y = getHeight() / 2.0 - radius * Math.sin(theta);

GRect pt = new GRect(x, y, 1, 1);
pt.setFilled(true);
pt.setColor(color);
```

Calculate exact position of the
point on the screen using
trigonometry

How do we plot a color?

```
float[] components = Color.RGBtoHSB(color.getRed(),
                                      color.getGreen(),
                                      color.getBlue(),
                                      null);

double radius = getRadius() * components[1];
double theta = components[0] * Math.PI * 2.0;

double x = getWidth() / 2.0 + radius * Math.cos(theta);
double y = getHeight() / 2.0 - radius * Math.sin(theta);

GRect pt = new GRect(x, y, 1, 1);
pt.setFilled(true);
pt.setColor(color);
```



Plot the point on the screen

How colorMap is made

A more detailed animation

[back to main slides](#)

```
navy blue
27
34
98
blue
41
201
234
lime green
99
212
32
red brown
160
89
66
.
.
.
```

```
Scanner sc = new Scanner(new File(COLORS_FILE));
HashMap<String, ArrayList<Color>> result =
    new HashMap<String, ArrayList<Color>>();

while (sc.hasNextLine()) {
    String colorName = sc.nextLine();
    String red = sc.nextLine();
    String green = sc.nextLine();
    String blue = sc.nextLine();

    int r = Integer.parseInt(red);
    int g = Integer.parseInt(green);
    int b = Integer.parseInt(blue);

    Color color = new Color(r, g, b);

    if (!result.containsKey(colorName)) {
        result.put(colorName, new ArrayList<Color>());
    }

    result.get(colorName).add(color);
}
```

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
while (sc.hasNextLine()) {
```

result

NO KEYS

```
navy blue ←  
27  
34  
98  
blue  
41  
201  
234  
lime green  
99  
212  
32  
red brown  
160  
89  
66  
.  
.  
.
```

```
String colorName = sc.nextLine();
```

while loop variables

colorName: "navy blue"\

result

NO KEYS

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
String red = sc.nextLine();
```

while loop variables

colorName: "navy blue"
red: "27"

result

NO KEYS

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
String green = sc.nextLine();
```

while loop variables

colorName: "navy blue"

red: "27"

green: "34"

result

NO KEYS

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
String blue = sc.nextLine();
```

while loop variables

colorName: "navy blue"

red: "27"

green: "34"

blue: "98"

result

NO KEYS

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
int r = Integer.parseInt(red);
```

while loop variables

colorName: "navy blue"

red: "27"

green: "34"

blue: "98"

r: 27

result

NO KEYS

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
int g = Integer.parseInt(green);
```

while loop variables

colorName: "navy blue"

red: "27"

green: "34"

blue: "98"

r: 27

g: 34

result

NO KEYS

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
int b = Integer.parseInt(blue);
```

while loop variables

colorName: "navy blue"

red: "27"

green: "34"

blue: "98"

r: 27

g: 34

b: 98

result

NO KEYS

```
navy blue  
27  
34  
98  
blue  
41  
201  
234  
lime green  
99  
212  
32  
red brown  
160  
89  
66  
. . .
```

```
Color color = new Color(r, g, b);
```

while loop variables

colorName:	“navy blue”
red:	“27”
green:	“34”
blue:	“98”
r:	27
g:	34
b:	98

color: 

result

NO KEYS

```
navy blue  
27  
34  
98  
blue  
41  
201  
234  
lime green  
99  
212  
32  
red brown  
160  
89  
66  
.  
.  
.
```

```
if (!result.containsKey(colorName)) {  
    result.put(colorName, new ArrayList<Color>());  
}
```

while loop variables

colorName:	“navy blue”
red:	“27”
green:	“34”
blue:	“98”
r:	27
g:	34
b:	98

color: 

result

“navy blue”: {}

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
result.get(colorName).add(color);
```

while loop variables

colorName: "navy blue"

red: "27" color: 

green: "34"

blue: "98"

r: 27

g: 34

b: 98

result

"navy blue": {  }

navy blue

27

34

98



blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
while (sc.hasNextLine()) {
```

result

“navy blue”: {█████}

navy blue

27

34

98

blue



41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
String colorName = sc.nextLine();
```

while loop variables

colorName: "blue"

result

"navy blue": { } []

navy blue

27

34

98

blue

41



201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
String red = sc.nextLine();
```

while loop variables

colorName: "blue"

red: "41"

result

"navy blue": { }

navy blue

27

34

98

blue

41

201



234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
String green = sc.nextLine();
```

while loop variables

colorName: "blue"

red: "41"

green: "201"

result

"navy blue": { } []

navy blue

27

34

98

blue

41

201

234



lime green

99

212

32

red brown

160

89

66

.

.

.

```
String blue = sc.nextLine();
```

while loop variables

colorName: "blue"

red: "41"

green: "201"

blue: "234"

result

"navy blue": { [] }

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
int r = Integer.parseInt(red);
```

while loop variables

colorName: "blue"

red: "41"

green: "201"

blue: "234"

r: 41

result

"navy blue": { [] }

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
int g = Integer.parseInt(green);
```

while loop variables

colorName: "blue"

red: "41"

green: "201"

blue: "234"

r: 41

g: 201

result

"navy blue": { [] }

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
int b = Integer.parseInt(blue);
```

while loop variables

colorName: "blue"

red: "41"

green: "201"

blue: "234"

r: 41

g: 201

b: 234

result

"navy blue": { [] }

```
navy blue  
27  
34  
98  
blue  
41  
201  
234  
lime green  
99  
212  
32  
red brown  
160  
89  
66  
. . .
```

```
Color color = new Color(r, g, b);
```

while loop variables

colorName:	“blue”	color:	[]
red:	“41”		
green:	“201”		
blue:	“234”		
r:	41		
g:	201		
b:	234		

result

```
“navy blue”: { [ ] }
```

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
if (!result.containsKey(colorName)) {  
    result.put(colorName, new ArrayList<Color>());  
}
```

while loop variables

colorName: "blue" color: [REDACTED]

red: "41"

green: "201"

blue: "234"

r: 41

g: 201

b: 234

result

"navy blue": {[REDACTED]}

"blue": {}

navy blue

27

34

98

blue

41

201

234

lime green

99

212

32

red brown

160

89

66

.

.

.

```
result.get(colorName).add(color);
```

while loop variables

colorName: "blue" color: 

red: "41"

green: "201"

blue: "234"

r: 41

g: 201

b: 234

result

"navy blue": {}

"blue": {}

[back to main slides](#)