More GUIs
Ch 6.1-6.3

14) Show that all of the zeros lie between [-3, 3] for
   \[ f(x) = 2x^3 - 13x^2 + 2x - 5 \]

15) List all possible rational roots for
   \[ f(x) = 2x^3 - 13x^2 + 2x - 5 \]
   \[ \pm 1 \pm 2 \]
   \[ \pm 1 \pm 5 \]

A WHALE is fine too
Highlights

- Drawing stuff

- Colors
MouseListener

ActionListeners work when you press a button, but it is much more fun to work with a mouse.

You can implement MouseListener to get information from the mouse:

```java
public class MouseThingy implements MouseListener
```

This works very similar to ActionListener, except with more methods.
MouseListener

The methods you need to override are:

- `mousePressed()` - when you press down
- `mouseReleased()` - when you let go of click
- `mouseClicked()` - a full click (no drag)
- `mouseEntered()` - mouse goes on top of your window (from outside the window)
- `mouseExit()` - when the mouse cursor leaves your window

(See: TellMeMouseCoordinates.java) (and see: RunAway.java)
Layouts

Brief overview... for details see:
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

Boarder:
5 sections: top, bot, left, center, right

Flow:
left to right(ish) (default)

Grid:
grid cells
MouseListener

The methods you need to override are:

mousePressed() - when you press down
mouseReleased() - when you let go of click
mouseClicked() - a full click (no drag)
mouseEntered() - mouse goes on top of your window (from outside the window)
mouseExit() - when the mouse cursor leaves your window

(See: TellMeMouseCoordinates.java)
(and see: RunAway.java)
Previously we used JLabels to hold text

We can also use JLabel to hold an image:

```java
ImageIcon pic = new ImageIcon("C:/funnyPic.jpg");
funnyPicture = new JLabel();
funnyPicture.setIcon(pic);
```

1. Make an ImageIcon object
2. Make a JLabel object
3. Link them together
   (See: DrawImages.java)
Extending JFrame

To do some more advanced drawing, we will have to extend JFrame (to override methods):

```java
public class DrawCircle extends JFrame {
    // DrawCircle is just an example name
}
```

Then we need to override the paint() method:

```java
@Override
public void paint(Graphics g) {
    super.paint(g);
}
```
Graphics class

Now to draw, we simply have to use the Graphics instance in paint():

```java
@override
public void paint(Graphics g) {
    super.paint(g);
    g.drawOval(50, 50, 20, 20);
}
```

(See: DrawCircle.java)
When to repaint?

Java normally updates your GUI automatically.

Sometimes you need to update manually:
- repaint - redraw the whole window
- pack - resize the window
- revalidate - layout is redone (in case you removed any objects)

If your window seems to not be correct (or not updating) try running one/all of these.
We have seen how to detect when you click the mouse, so let's use that to draw!

To do this we need to make a MouseListener:
- When press the mouse down, mark this as one corner of the rectangle
- When we release the mouse, mark this as the other end of the rectangle

(mini-paint!)
(See: MouseDraw.java)
Color

To draw in color, we simply need to change the Graphics class instance with:

```
setColor() - sets the drawing color
```

```
g.setColor(Color.red);
```

`g` is an instance of Graphics

After running the command above, anything drawn with the Graphics class will be red.

(See: RedMouseDraw.java)
The JColorChooser is a pre-built class that allows you to choose a color (from a GUI).

To run the GUI, use `showDialog()` (returns `Color`):

```java
Color temp = new JColorChooser().showDialog(originalFrame, "JColorChooser", rectangleColor);
```

(See: RainbowMouseDraw.java)
drawString()

You can also use the Graphics class to draw a more customizable String than JLabel

To do need to simply call drawString():

```java
g.drawString("All your base are belong to us!", 20, 70);
```

starting x pixel
starting y pixel

Can also change Font with:

```java
g.setFont(new Font("Arial", Font.ITALIC, 30));
```

(See: DrawStringExample.java) point size