CS 1181
Week Five

Reese Hatfield

Custom Swing Graphics

Reese Hatfield

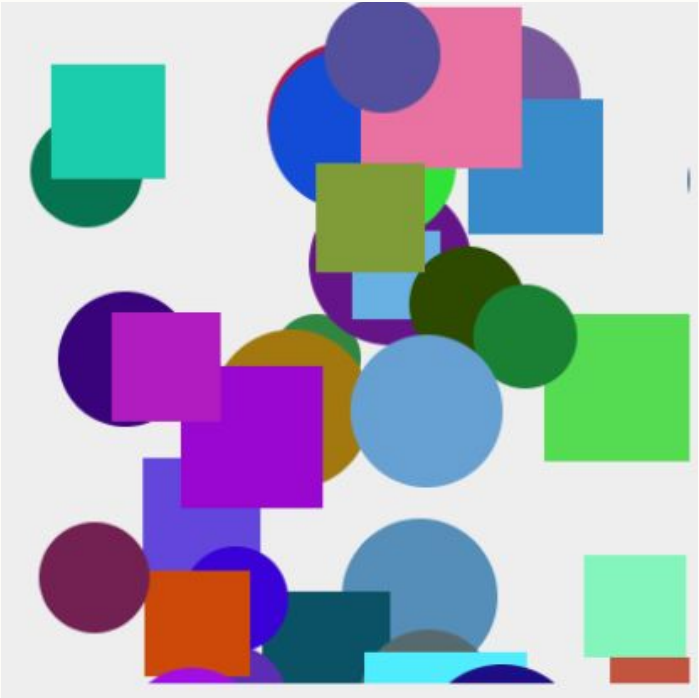
Boring Graphics

- We haven't *really* made our own graphics

- Just wrapping other peoples JComponents
- How do we make more interesting stuff?

Cool Graphics

- How do we get stuff like this?
- Actually custom graphics?



Cool Graphics

- Let’s look at how JComponent works
- paintComponent(Graphics g)
- We can use this method to make our own graphics

Cool Graphics

- paintComponent gives us a Graphics object
- super().paintComponent()
  - renders the default component to the screen (background)

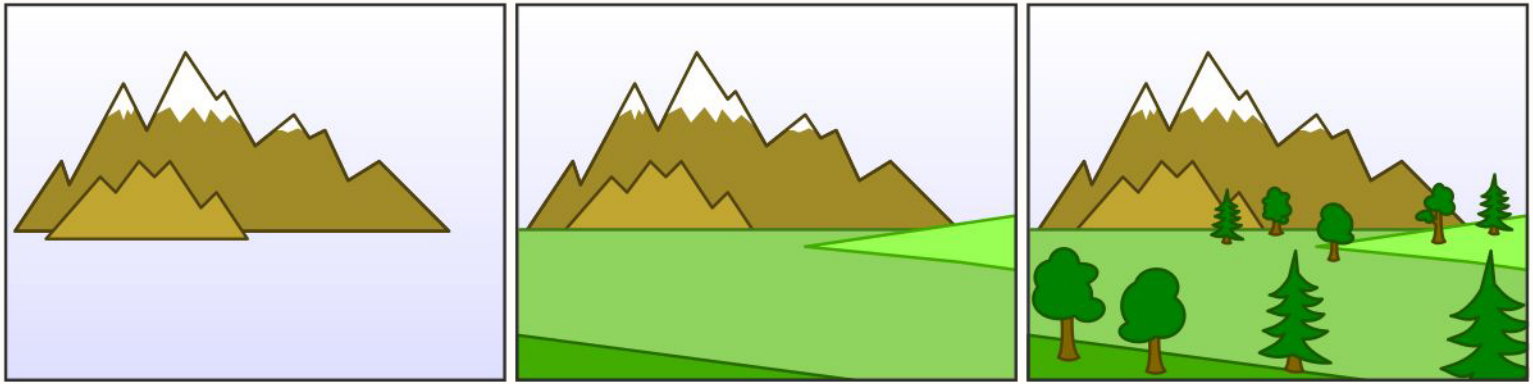
- “Graphics g” is kinda like a paintbrush

Cool Graphics

- Let’s make our own Circle object
  - Red
  - Round
  - Draw it anywhere
- What should we extend?

Painter's Algorithm

- "Must honor opaque property"
- Why?
- Painter's Algorithm



Cool Graphics

- We can call repaint() to draw a component *again*
- Draw at position
- Change position
- Repeat

Cool Graphics

- If we change where we draw an object repeatedly
- We can animate objects on the screen
- Let’s make our Circle move!

Cool Graphics

- Timer class for animations
- new Timer(
    int delay,
    ActionListener listener
)
- Let’s put repeat code in the listener

Cool Graphics

- If we parametrize our movements:
  - We can make the ball bounce
  - Made position a field
  - Effectively a “framerate”

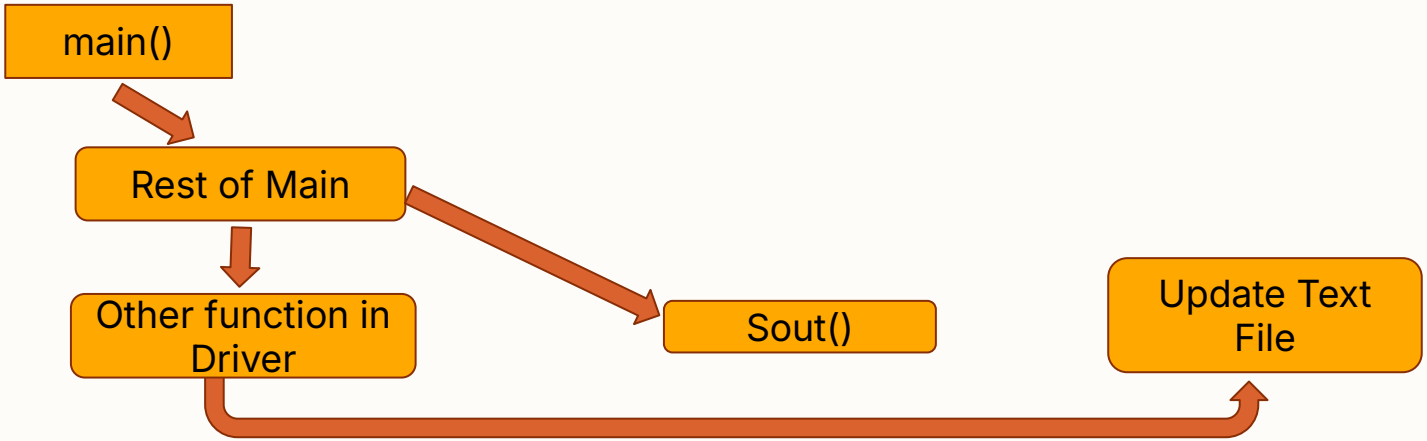
- Limitless possibilities for object animation

Separation of Concern

Reese Hatfield

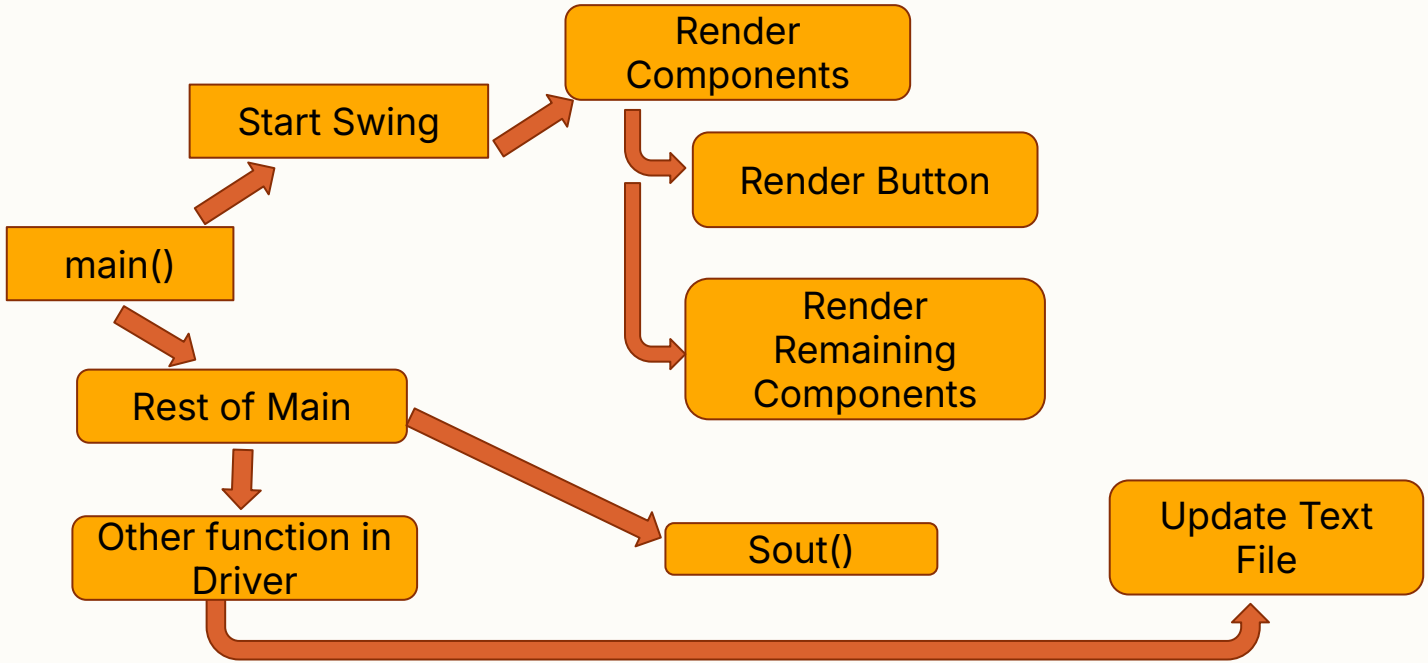
Messy

- Spaghetti code



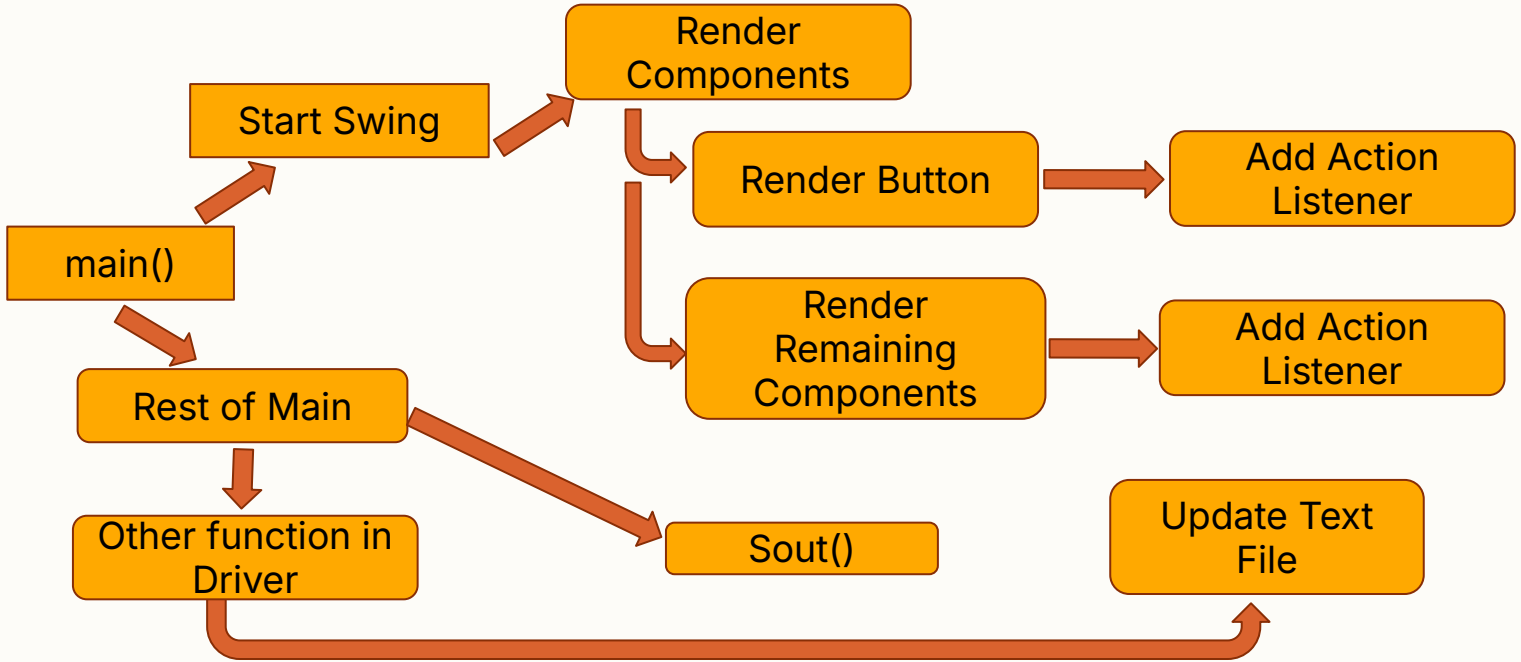
Messy

- Spaghetti code



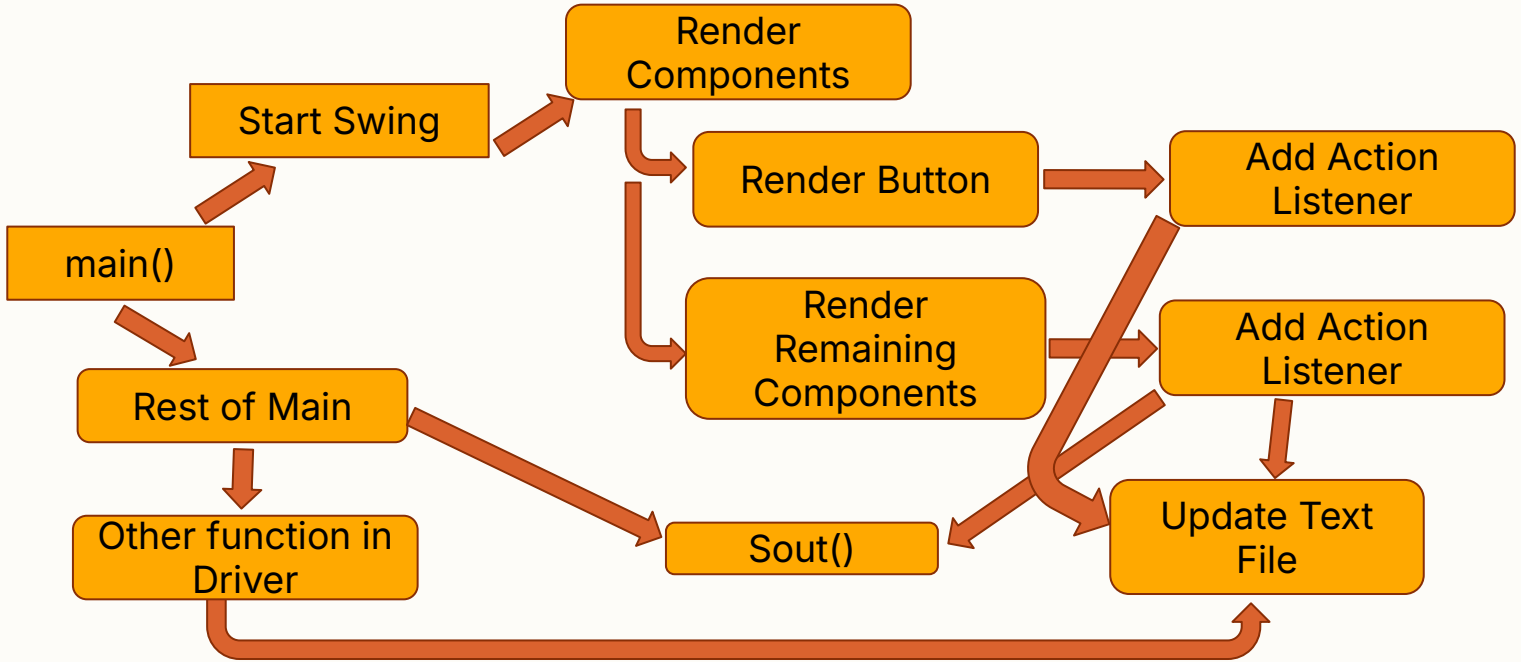
Messy

- Spaghetti code



Messy

- Spaghetti code



How do we fix this?

- We've talked about
  - Encapsulation
  - ActionListener implemented on a button
  - Declaration/Implementation separation

- Why did we talk about all this?

Eradicating the Spaghetti

- All of those are fundamentally the same idea
- Separation of concern

- This idea scales alongside your code
- Break code out into pieces
  - More maintainable

Separation of Concern

- We can use these principles to our advantage
- Separate our application into layers

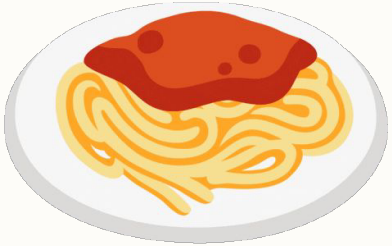
- Many different “layer” models
- Let’s look at a few

Separation of Concern

- Three Tiered Applications
  - Presentation Layer
    - GUI components
  - Business Layer
    - Fn calls and logic
  - Data Layer
    - Writing to files, etc

Separation of Concern

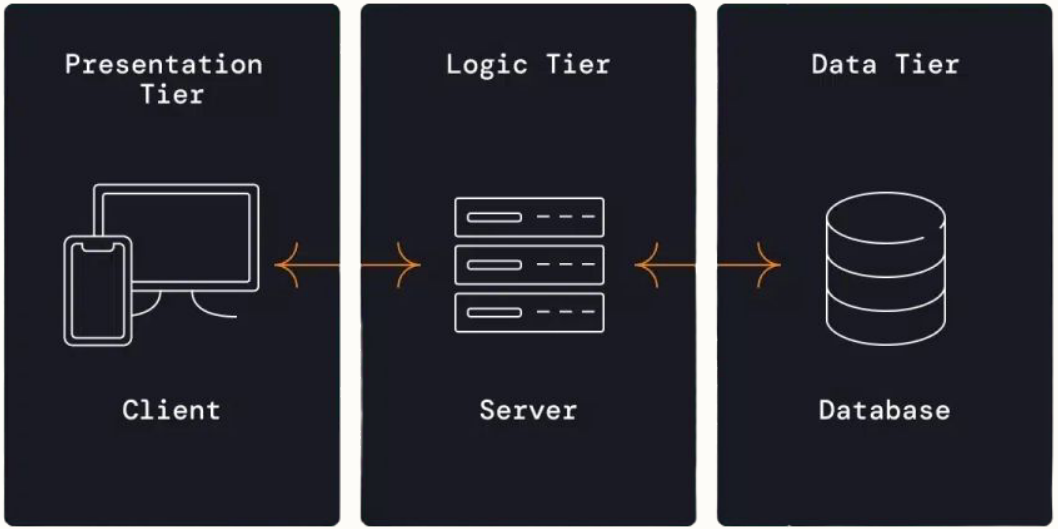
- Java Swing makes this really easy to manage
- Prevent spaghetti code



- This makes Swing scale as your code grows in complexity

Separation of Concern

- Let’s write a Swing app with this model
- Favorite Color Storage App



Favorite Color Storage

- I want to store favorite colors in a text file
  - Limited amount of people
  - Limited amount of colors

- I want to be the only person who can store a color
  - Password “protected”

Favorite Color Storage

- Useful View Components
  - JPasswordField
  - JComboBox
  - JButton
- Controller Layer
  - Checks the password
- Data Layer
  - Side Effects (what does this mean?)

Favorite Color Storage

- Let’s make this more robust
- Where should we deal with exceptions?
  - Custom Exceptions?
- How should we convey to the user something went wrong?

Dealing with Exceptions

- Handle this in which layer?
- Why?

Dealing with Exceptions

- Handle this in which layer?
- Why?

- Controller Layer
  - Makes it simple to deal with in the view
    - Return a boolean
  - Deal with robust data exceptions

When something goes wrong

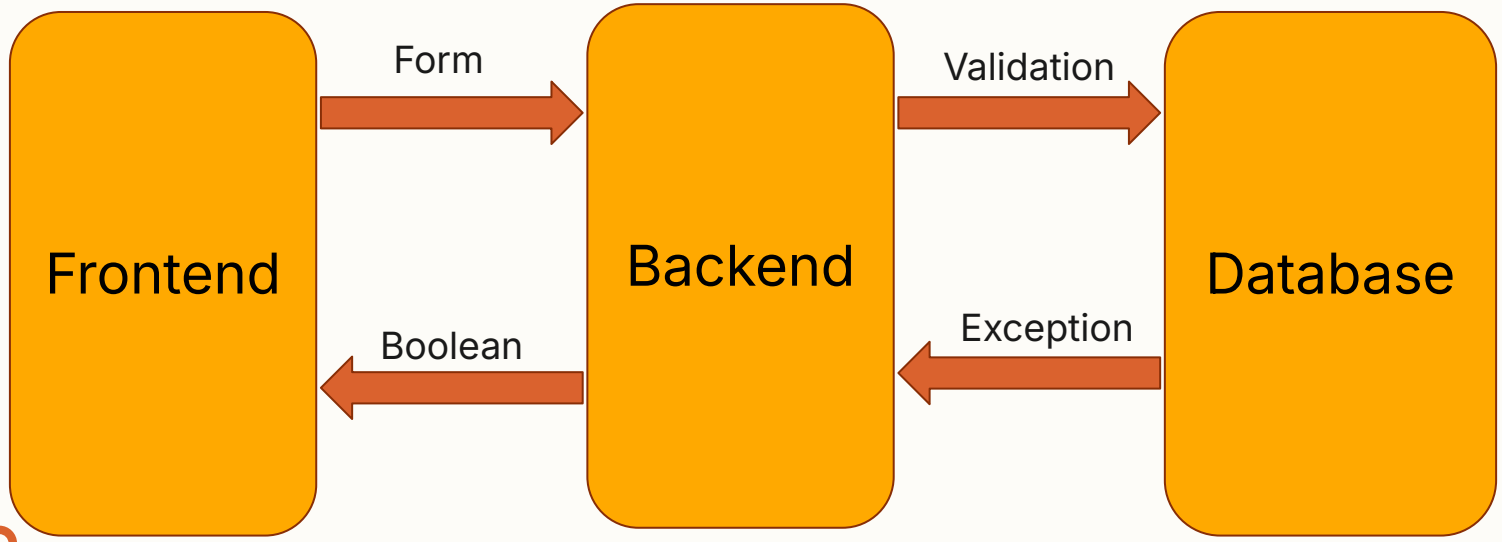
- JOptionPane
  - Used statically (usually)
  - Pop-up alert

- Since our controller returns a boolean

- Easy to check

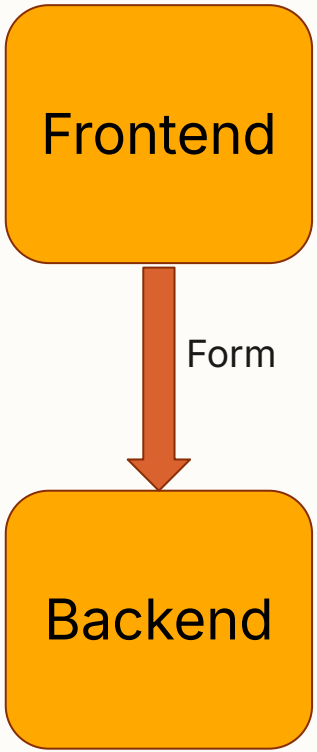
Aside: Data Transfer

- Look at the flow of our data



Aside: Data Transfer

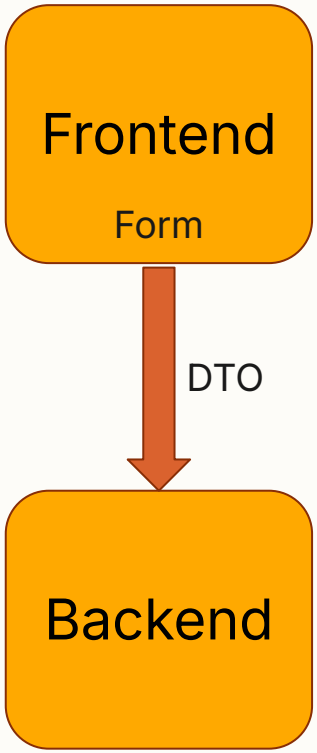
- Passing data from frontend to backend
- Find that line of code
- Often times, this will pass a wrapper object instead



Aside: Data Transfer

- Data Transfer Object (DTO)
- Plain old java object (POJO)

- Let’s do it
- Useful for more robust backends



Review

- We built a “full stack” application
- Side effects the file system

- Robust Exceptions
- Easy to interact with

Key Listeners

- Let’s revisit an old example
- Moving Red Circle

- How could we make it move on our own?
  - With arrow keys for instance

Key Listeners

- KeyListener interface
  - Has multiple methods
  - Cannot use a lambda
- Check for getKeyCode() == KeyEvent.VK_DOWN

- Repeat

Key Listeners

- Let’s see it in actions
- Commonly used for your projects

- Check boundary conditions
    - For invalid moves

Review

- Midterm June 24th

- We have covered a lot of material
  - Still more to cover
  - Will spend next week doing review examples

Review

- Inheritance
- Abstract Classes
- Interfaces
- Dynamic Dispatch
- Implementation Separation
- Data Modeling
- Type Composition
- Casting
- Comparable/ator sorting
- Copy Constructors
- Checked/Unchecked Exceptions
- try/catch/finally
- Swing
- JButtons, Input methods
- ActionListener
- Event Dispatch Thread
- Layout Managers
- JPanels as containers
- Custom graphics (paintComponent(g))
- Separation of Concern

Review