Cartoon Help Slides
Topics

- Shape package
- Design
- Layout Panes
- Graphical and Logical Containment
- EventHandler
- Timeline
- Step by Step
- Helpful Documentation
To use a shape from the `shape` package type `import javafx.scene.shape.<shapename>;
- For example: `import javafx.scene.shape.Rectangle;

Ex. instantiating a new rectangle:
`Rectangle rectangle = new Rectangle(100, 200, Color.BLACK);`

Look at [Graphics II lecture](#) for reference on how to set properties of shapes

Also, check out the [JavaFX Shapes Documentation](#) on our website
Design (Part 1)

- What do you want your Cartoon to do?

- Draw a picture of what it will look like
  - This is part of the mini-assignment!

- Think about the classes you’ll need
  - which can just be JavaFX components?
  - which will you need to create?

- Consider what warrants public vs. private methods and classes
  - remember - split up your constructor into helper methods. Should these be public or private?
Design (Part 2)

- How will you contain your classes graphically?
  - remember the scene graph
- Determine what each of your classes should do
- Determine which classes need to know about other classes
  - this should impact your *logical* containment. More later...
Layout Panes (Part 1)

- `BorderPane borderPane = new BorderPane();`
  - arranges components in five regions
    - Top, Left, Center, Right, Bottom
    - unused regions are collapsed - don’t show up!
Layout Panes (Part 2)

- `HBox hBox = new HBox();`
  - horizontally aligns nodes

- `VBox vBox = new VBox();`
  - vertically aligns nodes

combined `HBox` and `VBox`
Layout Panes (Part 3)

- `GridPane gridPane = new GridPane();`
  - create a flexible grid of rows and columns in which to lay out nodes
You Can Contain Multiple Panes

Note: Unused regions of the BorderPane are not shown; they are automatically hidden!
Graphical and Logical Containment

- What’s the graphical containment of this program? How does this differ from logical containment? Consider scene graph!

- Reminder: Logical containment is like diagrams you’ve been making all along

- ex. what classes instantiate other classes (containment), and what classes know about other classes (association).
Logical Containment Diagram
The “Cartoon” Class

- In past projects, our “top-level” class has been LiteBrite or TASafeHouse.
- Now, our top-level class is the PaneOrganizer, but it shouldn’t be doing much more than organizing Panes.
- So we should put most of our animation & user input logic in a Cartoon class that we instantiate in the PaneOrganizer:
  - Organizing Panes → PaneOrganizer!
  - Quitting entire app → PaneOrganizer!
  - Moving shapes → Cartoon class!
  - KeyInput → Cartoon class!
EventHandler

- How do we make a Button print out something?
  - create a new Button
  - create one class that implements `javafx.event.EventHandler<ActionEvent>`
    - What method must this class define? What would the body be?
  - what should this class have in its parameters?
We don’t need full generality, and can do simple animation using a single KeyFrame that is repeated a fixed number of times or an indefinite number of times separated by a duration; each time EventHandler is called, it makes incremental changes to time-varying variables (e.g., (x, y) position of a shape)
Timeline (part 2)

1. Instantiate an instance of `KeyFrame`
   
   ```java
   KeyFrame kf = new KeyFrame(Duration.millis(1),
   new ShapeHandler());
   ```

2. Write a “`ShapeHandler`”, a class that implements `EventHandler<ActionEvent>`
   
   a. note that you don’t *have* to name it `ShapeHandler`
   b. what method must this class have?
3. Then we instantiate a `Timeline` and pass in our `KeyFrame` and set the `CycleCount` to `INDEFINITE`

```java
Timeline timeline = new Timeline(kf);
timeline.setCycleCount(Animation.INDEFINITE);
```
Timeline (part 4)

4. Start the Timeline!

timeline.play();
For More on Timelines

● See the Graphics II lecture
● Check out the Timeline section of the JavaFX Guide
● See the Javadoscs for Timeline
Step by Step (Part 1)

Review the Graphics I - III lecture slides, the JavaFX guide, and the JavaFX lab before starting!

- Fill in your App to set the title of the Stage
- Instantiate an instance of top-level class PaneOrganizer
- Create a Scene with a reference to the root Pane (which is created in PaneOrganizer)
- Don’t forget to set the Stage’s Scene!
- Show the Stage!
Step by Step (Part 2)

After each step, make sure your code compiles and runs!

- Instantiate an instance of your `Cartoon` class in `PaneOrganizer`
  - `Cartoon` can be totally empty at this point!

- Give `Cartoon` a root pane, and add it in the `PaneOrganizer`

- Get a quit button to work in `PaneOrganizer`
  - This should probably happen in a separate `VBox` or `HBox` contained in `PaneOrganizer`

- Add composite shape(s), label, and other necessary parts to `Cartoon`
Step by Step (Part 3)

After each step, make sure your code compiles and runs!

- In your Cartoon class, add a Timeline and EventHandler to get your composite shape to move at a constant speed

- Now that your composite shape is moving, get it to move at 20+ different speeds!

- In your Cartoon class, add an EventHandler to allow your user to change something visually based on key input

- Make sure you’ve met all of the requirements
  - e.g. does your Label update based on the Cartoon?

- Do a little dance cuz u earned it!!
Helpful Documentation

- We have prepared a lot of useful documentation for you!

- [JavaFX Shape Docs](#)
- [JavaFX Guide](#)

- Javadocs also contain lots of helpful information
  - you will learn more about these in lab next week!

- Whenever you want to find Javadocs of any class, we recommend to Google “<class> JavaFX Javadocs”
Good luck!

• Happy painting! Be creative, and as always…

Start early
Start today
Start yesterday!

“I f***ing love painting!”