GUIs and Event Handling: Part Two

Images

Transformations

Radio Buttons

Check Boxes

More Controls

Drawing

Mouse Events

Dialog Boxes

MVC Pattern

IMAGES

The ImageView Class

- An ImageView node can be used to display an Image object.
  - Images can be bmp, jpeg, gif, png
  - You must use an ImageView- you cannot add an Image directly to a container.

1. Create your image
   - Constructor: Image(String filename)
   - Constructor: Image(String filename, double width, double height, boolean preserveRatio, boolean smooth)
2. Create your ImageView
3. Add your ImageView to your scene graph

```
Image image = new Image("picture.jpg");
ImageView imageView = new ImageView(image);
pane.getChildren().add(imageView);
```

- In eclipse, image files go in the bin folder!

Practice

- Create a GUI that displays a picture.
- Add text to the GUI to describe the picture.

Transformations

- An effect applied to a node
- Translation: slide
  - setTranslateX(double), setTranslateY(double)
- Scaling: enlarge or shrink
  - setScaleX(double), setScaleY(double)
- Rotation: turn
  - setRotate(double) // degrees can be positive (clockwise) or negative (counterclockwise)

- Applying a transformation does **not** change the underlying characteristics of the node!
- You can apply a transformation to an entire group!

- Review the transformations example.

MORE CONTROLS

CheckBox

- A *check box* can be toggled on or off (yes/no, selected/unselected, etc.)
- Constructor: takes the text to be displayed next to the check box
- Methods:
  - setOnAction- specifies the *event handler*
  - setSelected(boolean)
  - isSelected

Practice

- Modify the picture program so that the user can decide whether or not to display the picture label.

RadioButton

- Radio buttons are used for a set of mutually exclusive options
  - Like multiple choice
  - **Each** choice is a radio button
- Radio buttons are grouped together into a *toggle group*
  - The toggle group is not visual- it's a logical grouping
- Constructor: takes the text to be displayed next to the radio button
- Methods:
  - setOnAction- specifies the *event handler*
  - setSelected(boolean)
  - isSelected
  - setToggleGroup(ToggleGroup)

Radio Button Steps

- Create **each** button (e.g., each choice)
- Create a ToggleGroup
- Set the toggle group of *each* button (not visual)
- Add **each** button to your scene graph(visual)
- Set the action of **each** button

Practice

- Modify the picture program so that the user can display either the dog picture or the cat picture.

EVEN MORE CONTROLS!

TextArea

- Used to display output or to read input that spans multiple lines
- Automatically scrolls
- Methods:
  - setEditable(boolean)
  - getText()
  - setText(String)
  - append(String)
- Use a button (or other control) to retrieve the text entered

Sliders

- Choice along a continuum of values
- Methods:
  - setMin
  - setMax
  - setValue
  - setShowTickLabels, setShowTickMarks, setMajorTickUnit
  - valueProperty().getValue().doubleValue();
    - // get the value that is currently selected
  - valueProperty().addListener(listenerMethod);
    - // listen and react AS the user is changing the value
    - listener method header: private void method(ObservableValue<? extends Number> ov, Number oldValue, Number newValue)

ComboBox

- A drop-down menu
- Constructor: takes an ObservableList<T> as parameter
  - Can hold String, Enum, or any type!

```
ObservableList<String> options =
    FXCollections.observableArrayList("A", "B","C", "D");
ObservableList<MyEnumType> options2 =
    FXCollections.observableArrayList(
    Arrays.asList(MyEnumType.values()));
ComboBox comboBox = new ComboBox(options);
```

- Methods:
  - setValue()
  - getValue() // just get the value that is currently selected
  - getSelectionModel().selectedItemProperty().addListener(listenerMethod)
    - // listen and react AS the user is changing the value
    - listener method header: private void method(ObservableValue<? extends String> ov, String oldVal, String newVal)

ComboBox- Windows 10

- Note! ComboBox crashes some programs run on Windows 10. To prevent this, add the following to your start method:

```
System.setProperty("glass.accessible.force",
"false");
```

Menus

- MenuBar, Menu, MenuItem classes
  - MenuBar menuBar = new MenuBar();
    - Menu fileMenu = new Menu(“File”);
      - MenuItem newItem = new MenuItem(“New”);
      - MenuItem openItem = new MenuItem(“Open”);
      - MenuItem closenItem = new MenuItem(“Close”);
    - fileMenu.getItems().addAll(newItem, openItem, closeItem);
  - menuBar.getMenus().add(fileMenu);
- Set the action of a menu **item**
  - When this item is chosen, the code will execute
  - Use setOnAction method

ListView

- ListView<String> list = new ListView<String>();
- Methods
  - list.setItems(FXCollections.observableArrayList(“a”, “b”, “c”));
  - setPrefWidth(pixels)
  - setPrefHeight(pixels)
  - setOrientation(Orientation.HORIZONTAL);
  - getSelectionModel().selectedItemProperty().addListener(listenerMethod)
    - listener method header: private void method(ObservableValue<? extends String> ov, String oldVal, String newVal)

Practice

- Review the ComponentsDisplay example.

On Your Own Practice

- Choose a different control not covered in the notes/video.
  - Perhaps from this list here: https://docs.oracle.com/javafx/2/ui_controls/jfxpub-ui_controls.htm
- Create a program that uses it!

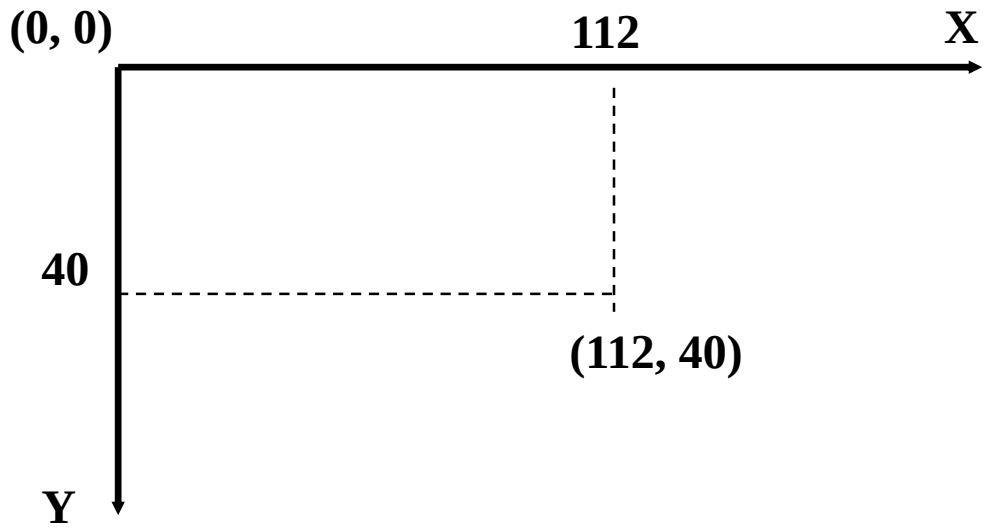
OR

- Go more in-depth with one of the covered controls!

DRAWING

Java Coordinate Space

- Use a coordinate space
  - Origin is in the top-left corner.
  - x-values increase to the right
  - y-values increase down



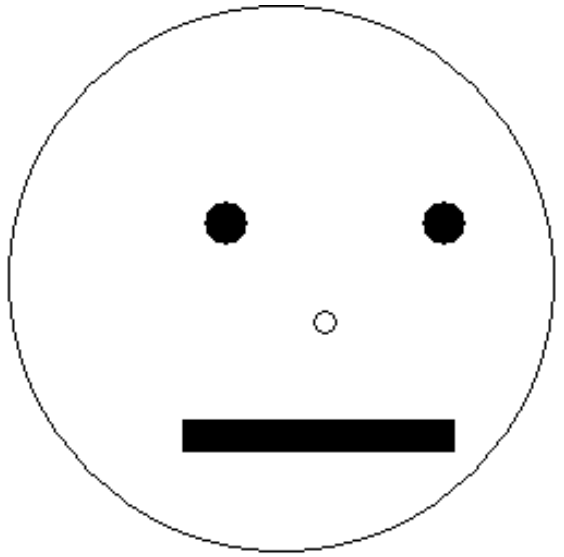
Shape Class Constructors

- Drawing is accomplished by adding shapes to the scene graph.
- When adding to the scene graph, shapes are drawn in the order in which they are added.
- The Pane layout is good to use with shapes because it uses absolute positioning. (You might nest a Pane inside of another layout- such as BorderPane, VBox, or HBox.)

- Line(startX, startY, endX, endY)
- Rectangle(upperLeftX, upperLeftY, width, height)
- Circle(centerX, centerY, radius)
- Ellipse(centerX, centerY, radiusX, radiusY)

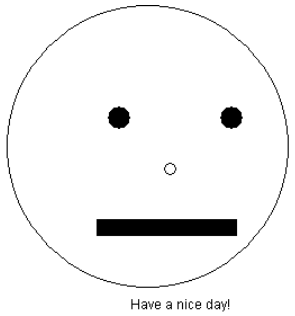
Practice

- Create a GUI that draws a face with a happy message.



Practice

- Modify the face program. Allow the user change the color of the eyes to a new color.
- Create a GUI that draws 50 circles of random color and size.
  - Add a redraw button.
  - Add a checkbox so the user can select filled or unfilled.
  - Add radio buttons for the circles to be drawn in random colors or just a single color.



MOUSE EVENTS

Mouse Events

- When you click the mouse over a node/scene, three events occur:
  - mouse pressed
  - mouse released
  - mouse clicked (a press and release on the same node)
- When you move the mouse, four events might occur:
  - mouse entered
  - mouse exited
  - mouse moved
  - mouse dragged (moved while the button is pressed)

Mouse Events

- Scene and Node objects have a method for handling each of these events:
  - setOnMouseClicked
  - setOnMouseMoved
  - setOnMouseDragged
  - etc.
- The method you pass in (this::handleMouseEvents) takes one parameter of type MouseEvent
  - MouseEvent methods getPoint() or getX() and getY() return the location of the mouse when the event occurred
  - Make sure you import the MouseEvent class from JavaFX!!

Practice

- Write a GUI that counts the number of mouse clicks. Print the number of clicks and the integer coordinates of each click.
- Write a “rubber line” GUI.
- Write a program to draw a “tail” on the cursor as it moves.

DIALOG BOXES

Dialog Boxes

- A dialog box is a pop-up window
- Dialog boxes:
  - convey information
  - confirm an action
  - allow the user to enter data
  - allow the user to make a choice
- Dialog boxes have a single, specific purpose
  - User interactions with dialog boxes are usually brief

Dialog Class

- Methods to display information:
  - setHeaderText(String)
  - setTitle(String)
  - setContentText(String)
- Method to get information: showAndWait()
  - Returns an Optional<Type> (e.g., Optional<String> or Optional<ButtonType>)
    - Optional<Type> method:
      - isPresent()
      - get()

Dialog Child Classes

- Alert
  - Types of Alerts:
    - AlertType.INFORMATION
    - AlertType.CONFIRMATION
    - AlertType.WARNING
    - AlerType.ERROR
  - Constructor: send in the type (from the list above)
- TextInputDialog
- ChoiceDialog

Basic Approach

- Create and set the display text (header, title, content text)
- Show and wait
  - Don’t forget this!
- Retrieve the information
  - If it’s present, get it!

Practice

- Write a program that uses dialog boxes.
  - Obtain two integer values.
  - Display the sum and product of the two values.
  - Ask whether the user wants to process another pair of values.
- Modify the drawing circle program to allow the user to input the number of circles to draw.

Common Package Imports for FX Programs

```
import javafx.application.*;
import javafx.event.*;
import javafx.geometry.*;
import javafx.scene.*;
import javafx.scene.control.*;
import javafx.scene.image.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.scene.paint.*;
import javafx.scene.shape.*;
import javafx.scene.text.*;
import javafx.stage.*;
import java.util.*;
```

MVC Pattern

Model-View-Controller

- A design pattern!
- Separate the data from the interface.
  - They do not know about each other!

The Model

- The *model* stores the *content*
  - Data and methods
  - In Java (and other object-oriented languages) this is often a class (with instance data variables and methods)
  - Often called the *business logic* or the *data structures*
- The model has no idea how it's being used.
  - No user interface!
  - Non visual!

The View

- The user interface
  - Displays content to the user
  - Provides a way for the user to provide content
- The view doesn’t know how to process content

The Controller

- Links together the model and the view
  - Gets data from the view and passes it to the model for processing
  - Passes results back to the view for display

Practice

- Review the Account example.
  - The user enters in a username. The program repeatedly generates a password.
  - What is the model?
  - Review the GUI view and controller.
  - Review the text-based (console) view and controller.
- Note that the model and the view have no knowledge of each other!
- Note that the model can be used with different views!

- Review the Employee example.

On Your Own Practice

- Use the MVC pattern and JavaFX to create a GUI where the user can enter information about Students.
  - Create a simple Student class (the model).
    - Use only one or two variables- such as name, id, tuitionOwed, etc.
    - Create one class-specific method- such as register, pay tuition, etc.
  - Create the view, which allows:
    - users to enter information about a student or multiple students
    - users to display created student(s)
    - users to process the student(s) using the method you created
  - Create the controller, which links together the model and view, and:
    - creates the objects using data from the view
    - defines the methods that display and process the students