Application Program Development

APD545

Instructor: Maryam Sepehrinour
Email: Maryam.Sepehrinour@SenecaPolytechnic.ca

Outcomes

- ✓ Understanding User Interfaces
- ✓ What is a model?
- ✓ A GUI (Graphical User Interface)
- ✓ An Application
- ✓ Window Component
- ✓ Container

Understanding User Interface

- Allows the user to interact with the underlying program/software.
- Has the ability to take-in information from the user and also to provide visual or audible information back to the user. (Often has physical interactive components)

What is a Model?

Model

- *The **model** of an application consists of all classes that represent the "business logic" part of the application ... the underlying system on which a user interface is attached.*
  - The model is always developed separately from the user interface.
  - It should not assume any knowledge about the user interface at all (e.g., model classes should not assume that **System.out.println()** is available).

User Interface

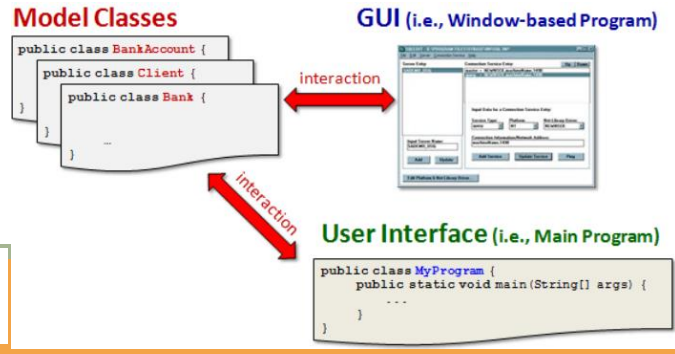
- *The **user interface** is the part of the application that is attached to the model which handles interaction with the user and does NOT deal with the business logic.*
  - The user interface always makes use of the model classes and often causes the model to change according to user interaction.
  - The changes to the model are often reflected back (visually) on the user interface as a form of immediate feedback.

GUI

- *A **graphical user interface (GUI)** is a user interface that makes use of one or more windows to interact with the user.*

What is GUI?

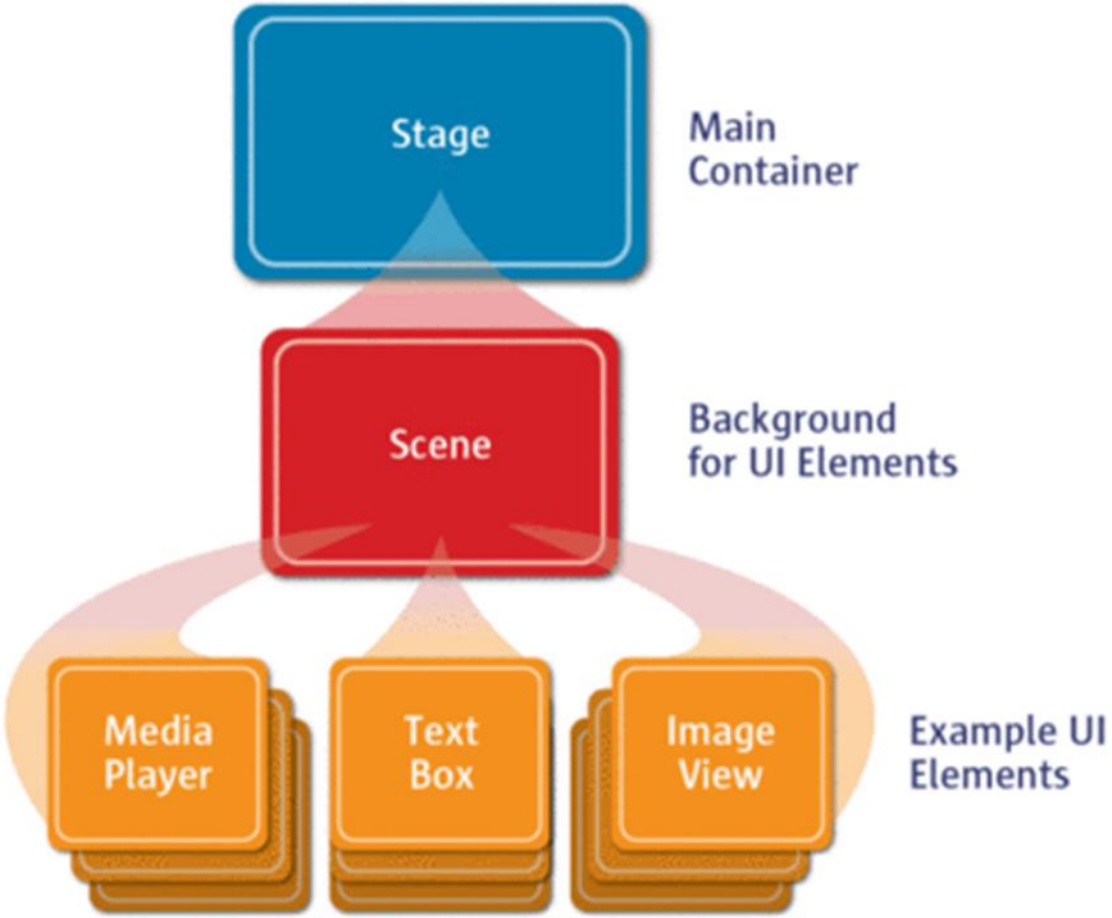
- A graphical user interface (**GUI**) presents a user-friendly mechanism for interacting with an app.
- GUIs are built from GUI components—also called controls or widgets (short for window gadgets).
- A GUI component is an object with which the user interacts via the mouse, the keyboard or another form of input, such as voice recognition.
- An **application** (or app) is a computer program with a graphical user interface which can interact with user to perform tasks & calculations, obtain & visualize information, and potentially interact with the real world through sensors and hardware.
- it is important to understand that there should always be a separation in your code between the model classes and the user interface classes. (Re-usability)

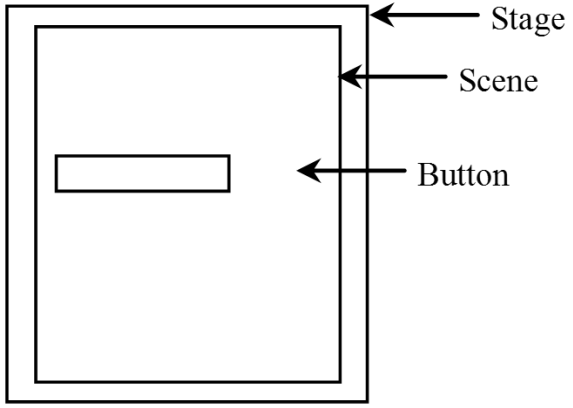


What is JavaFx?

- **JavaFX** is an open source Java-based framework for developing rich client applications.
- The **Scene Builder** tool is a standalone JavaFX GUI visual layout tool that can also be used with various IDEs.
- JavaFX Scene Builder generates **FXML** (FX Markup Language)—an XML vocabulary for defining and arranging JavaFX GUI controls. JavaFX Scene Builder completely hides the FXML details from you, so you can focus on defining what the GUI should contain without specifying how to generate it — this is an example of declarative style programming.
-

JavaFx Windows Structure





Components of JavaFx

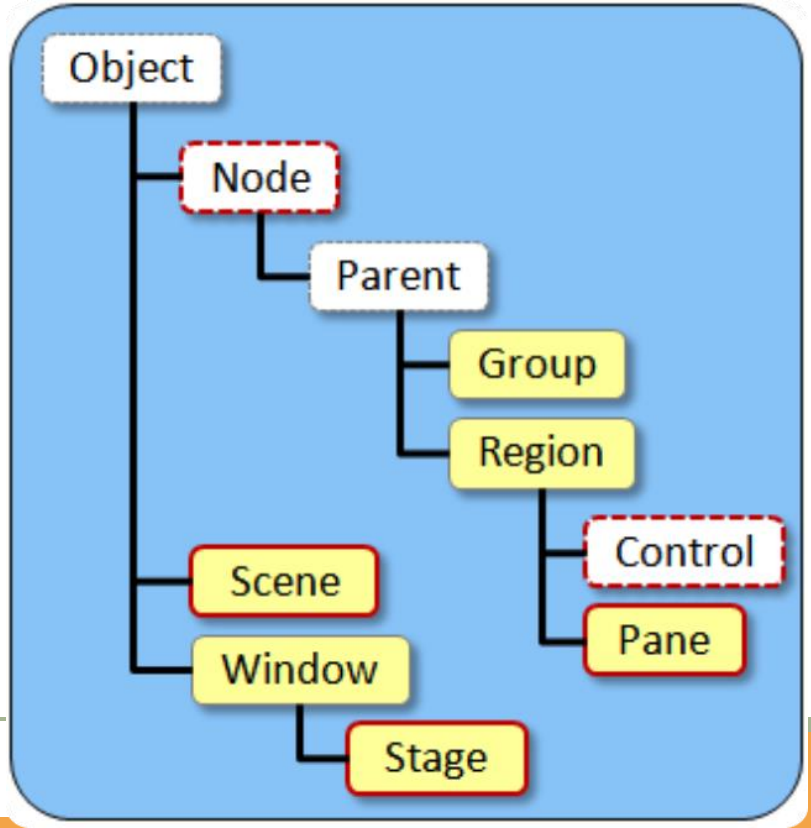
- **Stage**
  - The window in which a JavaFX app's GUI is displayed is known as the stage and is an instance of class Stage (package javafx.stage).
- **Scene**
  - The stage contains one active scene that defines the GUI as a scene graph
  - A tree data structure of an app's visual elements, such as GUI controls, shapes, images, video, text and more. The scene is an instance of class Scene (package javafx.scene).
- **Window**
  - A window component is an object with a visual representation that is placed on a window and usually allows the user to interact with it.
    - In JavaFX, typical window components (e.g., buttons, textFields, lists) are Control objects ... and they represent the various components of the window.
    - Components are often grouped together, much like adding elements to an array.
- **Node**
  - Each visual element in the scene graph is a node.
  - An instance of a subclass of Node (package javafx.scene)

Components of JavaFx

- **Container**
  - Is an object that contains components and/or other containers.
  - The most easily understood container in JavaFX is the Pane class.
  - It is the base class for its various subclasses that are used to automatically manage the layout of the window components.
  - Its variety of subclasses allow the components to be resized and repositioned automatically, while maintaining a particular arrangement/layout on the window.
- **Layout Containers**
  - Nodes that have children are typically layout containers that arrange their child nodes in the scene.
- **Controls**
  - Are GUI components, such as Labels that display text, TextFields that enable a program to receive text typed by the user, Buttons that initiate actions and more..

Components of JavaFx

- **Control class**
  - An FXML GUI's event handlers are define in a so-called controller class.
- **Event handler**
  - Is a method that responds to a user interaction.



Basic Structure of JavaFx – Example

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.stage.Stage;
5
6  public class JavaFXBasic extends Application {
7    @Override // Override the start method in the Application class
8    public void start(Stage primaryStage) {
9      // Create a button and place it in the scene
10     Button btOK = new Button("OK");
11     Scene scene = new Scene(btOK, 200, 250);
12     primaryStage.setTitle("JavaFX Basic Structure"); // Set the stage title
13     primaryStage.setScene(scene); // Place the scene in the stage
14     primaryStage.show(); // Display the stage
15   }
16
17   /**
18    * The main method is only needed for the IDE with limited
19    * JavaFX support. Not needed for running from the command line.
20    */
21   public static void main(String[] args) {
22     launch(args);
23   }
24 }
```

Display a simple shape in JavaFx – Example

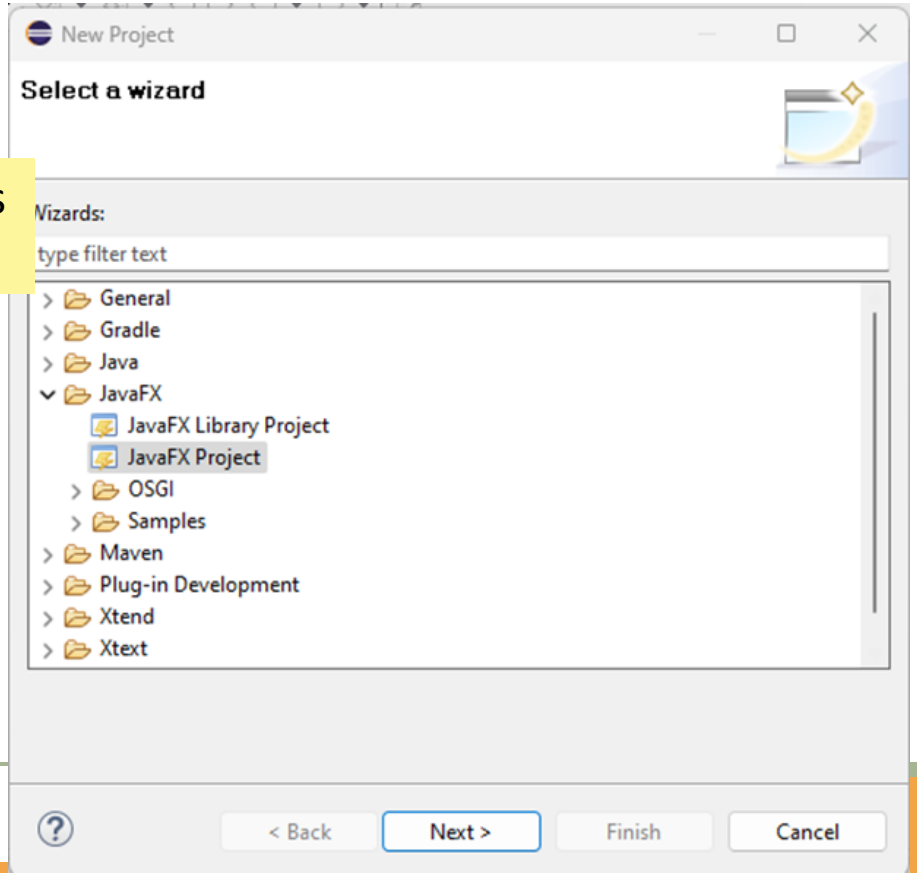
```
 1  import javafx.application.Application;
 2  import javafx.scene.Scene;
 3  import javafx.scene.layout.Pane;
 4  import javafx.scene.paint.Color;
 5  import javafx.scene.shape.Circle;
 6  import javafx.stage.Stage;
 7
 8  public class ShowCircle extends Application {
 9    @Override
10    public void start(Stage primaryStage) {
11      // Create a circle and set its properties
12      Circle circle = new Circle();
13      circle.setCenterX(100);
14      circle.setCenterY(100);
15      circle.setRadius(50);
16      circle.setStroke(Color.BLACK); // Set circle stroke color
17      circle.setFill(Color.WHITE);
18
19      // Create a pane to hold the circle
20      Pane pane = new Pane();
21      pane.getChildren().add(circle);
22
23      // Create a scene and place it in the stage
24      Scene scene = new Scene(pane, 200, 200);
25      primaryStage.setTitle("ShowCircle");
26      primaryStage.setScene(scene);
27      primaryStage.show();
28    }
```

```
29
30    /**
31     * The main method is only needed for the
IDE with limited
32     * JavaFX support. Not needed for running
from the command line.
33     */
34    public static void main(String[] args) {
35      launch(args);
36    }
37  }
```

Welcome Application

- Let us build our first Welcome to JavaFx example using eclipse.
- Create a new JavaFx project using eclipse.
- File -> New -> Project

Please Follow the steps, in the original slides (Week#3- Segment#1)



Thank you!