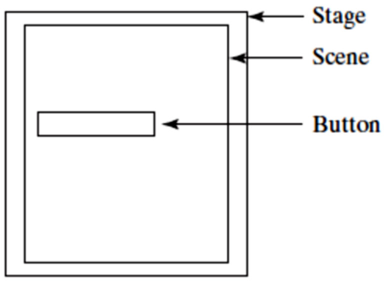
JavaFX Basics

Dr. Salma A. Mahmood

- ***JavaFX Basic Concepts***
  - ***Package :*** The JavaFX elements are contained in packages that begin with the **javafx** prefix. There are more than 30 JavaFX packages in its API library. for examples: **javafx.application**, **javafx.stage**, **javafx.scene**, and **javafx.scene.layout**. you must write import instruction to use them in top of any application. Use CTLR+SHIFT+I to insert missing packages to your program. For example:



import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout;
import javafx.scene.shape;
import javafx.scene.paint;

we can use $\longrightarrow$ ***import javafx.scene.*;***

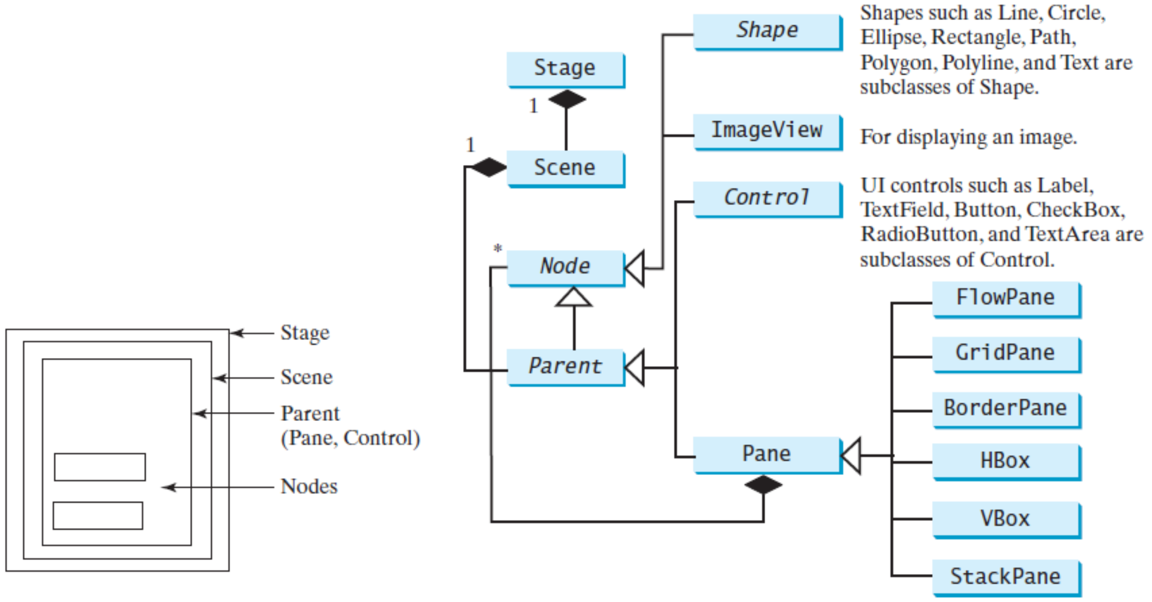
Stage and Scene :

- The central component implemented by JavaFX is the *stage*.
- A stage is a container for scenes and a scene is a container for the items that comprise the scene.
- As a result, all JavaFX applications have at least one stage and one scene.
- To create a JavaFX application, you will, at minimum, add at least one **Scene** object to a **Stage**.
- **Stage** is a top-level container. All JavaFX applications automatically have access to one **Stage**, called the *primary stage*.
- The primary stage is supplied by the run-time system when a JavaFX application is started. Although you can create other stages, for many applications, the primary stage will be the only one required
- **Scene** is a container for the items that comprise the scene. These can consist of controls, such as push buttons and check boxes, text, and graphics.
- To create a scene, you will add those elements to an instance of **Scene**.
- You can create instance of scene and add to the default stage primarystage as follow.

```
Button bt = new button("my button");          // create button instance named bt
Scene myscene = new Scene( bt , 300, 200);  // create scene instance named myscene with 300
                                               width and 200 length and connect with Node
primarystage.setscene( myscene);              // connect created scene with primarystage
primarystage.setTitle("my first program");    // put title to primarystage
primarystage.show();                          // primarystage
```

Nodes and Scene Graphs

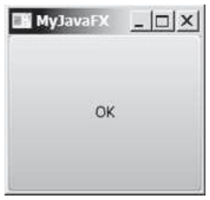
- The individual elements of a scene are called nodes. For example, a push button control is a node.
- Nodes can also consist of groups of nodes.
- A node with a child is called a parent node or branch node. Nodes without children are terminal nodes and are called leaves.
- The collection of all nodes in a scene creates what is referred to as a scene graph, which comprises a tree.
- There is one special type of node in the scene graph, called the root node. This is the top level node and is the only node in the scene graph that does not have a parent.
- The base class for all nodes is Node. There are several other classes that are, either directly or indirectly, subclasses of Node. These include Parent, Group, Region, and Control, to name a few.


Stage
Scene
Parent
(Pane, Control)
Nodes
Stage
1
1
Scene
*
Node
Parent
Shape
Shapes such as Line, Circle, Ellipse, Rectangle, Path, Polygon, Polyline, and Text are subclasses of Shape.
ImageView
For displaying an image.
Control
UI controls such as Label, TextField, Button, CheckBox, RadioButton, and TextArea are subclasses of Control.
Pane
FlowPane
GridPane
BorderPane
HBox
VBox
StackPane

Layouts

- JavaFX provides several layout panes that manage the process of placing elements in a scene. For example, the FlowPane class provides a flow layout and the GridPane class supports a row/column grid-based layout.
- The layout panes are packaged in javafx.scene.layout.

Example: The following example display one stage on screen as in figure :

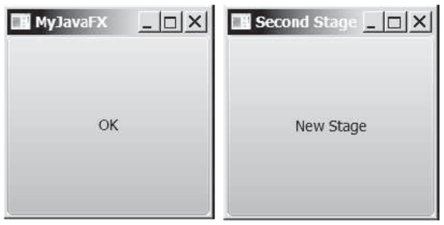


Package myJavaFX.java

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;

public class MyJavaFX extends Application {
        @Override                                    // Override the start method in the Application class
        public void start(Stage primaryStage) {      // Create a scene and place a button in the scene
                Button btOK = new Button("OK");
                Scene scene = new Scene(btOK, 200, 250);
                primaryStage.setTitle("MyJavaFX");   // Set the stage title
                primaryStage.setScene(scene);                  // Place the scene in the stage
                primaryStage.show();                           // Display the stage
        }
/* The main method is only needed for the IDE with limited  JavaFX support. Not needed for running from the command line.
*/
        public static void main(String[] args) {
                Application.launch(args);
        }
 } //end of Application
```

Another Example of multiple stages that can be displayed in a JavaFX program.



```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;


public class MultipleStageDemo extends Application {
        @Override                                  // Override the start method in the Application class
        public void start(Stage primaryStage) {
                // Create a scene and place a button in the scene
                Scene scene = new Scene(new Button("OK"), 200, 250);
                primaryStage.setTitle("MyJavaFX");       // Set the stage title
                primaryStage.setScene(scene);            // Place the scene in the stage
                primaryStage.show();                     // Display the stage

                Stage stage = new Stage();               // Create a new stage
                stage.setTitle("Second Stage");          // Set the stage title
                //Set a scene with a button in the stage
                stage.setScene(new Scene(new Button("New Stage"), 100, 100));
                stage.show();                            // Display the stage
           }
 }
```

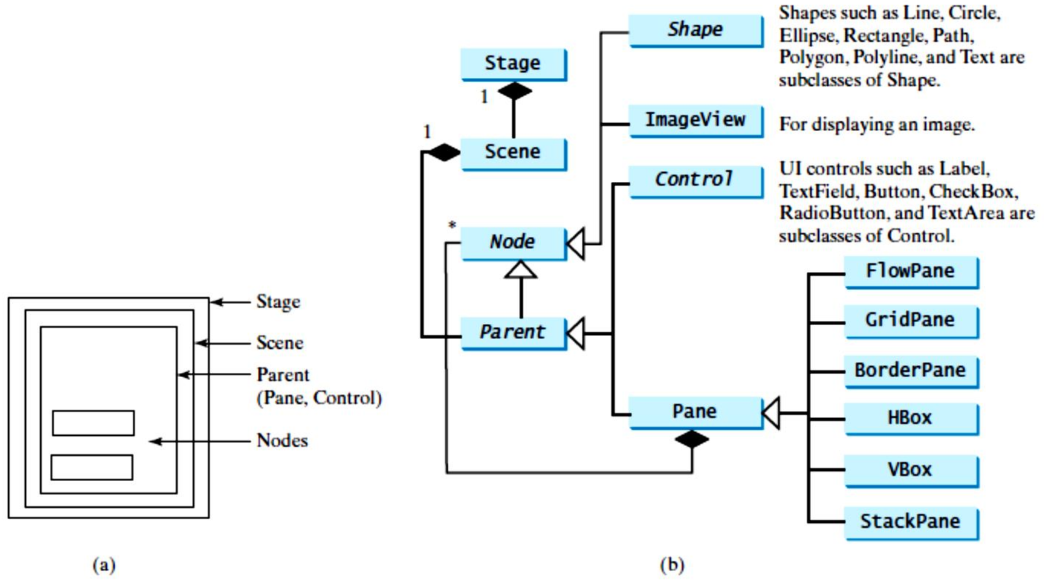
- ***Panes, UI controls, and shapes***

Panes, UI controls, and shapes are subtypes of **Node**.

A better approach is to use container classes, called ***panes***, for automatically laying out the nodes in a desired location and size. You place nodes inside a pane and then place the pane into a scene.

A ***node*** is a visual component such as a shape, an image view, a UI control, or a pane.
A *shape* refers to a text, line, circle, ellipse, rectangle, arc, polygon, polyline, etc.
A *UI* *control* refers to a label, button, check box, radio button, text field, text area, etc.

A scene can be displayed in a stage, the following figure shows the relationship among **Stage**, **Scene**, **Node**, **Control**, and **Pane.** Note that a **Scene** can contain a **Control** or a **Pane**, but not a **Shape** or an **ImageView**.
A **Pane** can contain any subtype of **Node**. You can create a **Scene** using the constructor **Scene(Parent, width, height)** or **Scene(Parent)**.

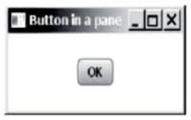

Stage
Scene
Parent
(Pane, Control)
Nodes
(a)
Stage
1
Scene
1
*
Node
Parent
Shape
Shapes such as Line, Circle, Ellipse, Rectangle, Path, Polygon, Polyline, and Text are subclasses of Shape.
ImageView
For displaying an image.
Control
UI controls such as Label, TextField, Button, CheckBox, RadioButton, and TextArea are subclasses of Control.
Pane
FlowPane
GridPane
BorderPane
HBox
VBox
StackPane
(b)

(a) Panes are used to hold nodes. (b) Nodes can be shapes, image views, UI controls, and panes.

Example: The following program places a button in a pane

```
package buttonapplication;

import javafx.application.Application;                    // for Application
import javafx.stage.Stage;                                      // for Stage
import javafx.scene.Scene;                                     // for Scene
import javafx.scene.layout.StackPane;            // for Panes
import javafx.scene.control.Button;                 // for Button control
//import javafx.scene.control.*;  can replace with above for all UI control
import javafx.event.ActionEvent;                               // for UI Action Handler
import javafx.event.EventHandler;                   // for UI Action Handler
//import javafx.event.*; can replace two above stataments
public class ButtonApplication extends Application {
   @Override
   public void start(Stage primaryStage) {
      StackPane root = new StackPane();
      root.getChildren().add( new Button("OK"));
      Scene scene = new Scene(root, 300, 250); // (pane, width, higth)
      primaryStage.setTitle("Hello World!");
      primaryStage.setScene(scene);
```



A button is placed in the center of the pane.

```
        primaryStage.show();
    }
}
```

The program creates a StackPane and adds a button as a child of the pane. The getChildren() method returns an instance of javafx.collections.ObservableList. ObservableList uses for storing a collection of elements. Invoking add(e) adds an element to the list. The StackPane places the nodes in the center of the pane on top of each other. Here, there is only one node in the pane. The StackPane respects a node's preferred size. So you see the button displayed in its preferred size.