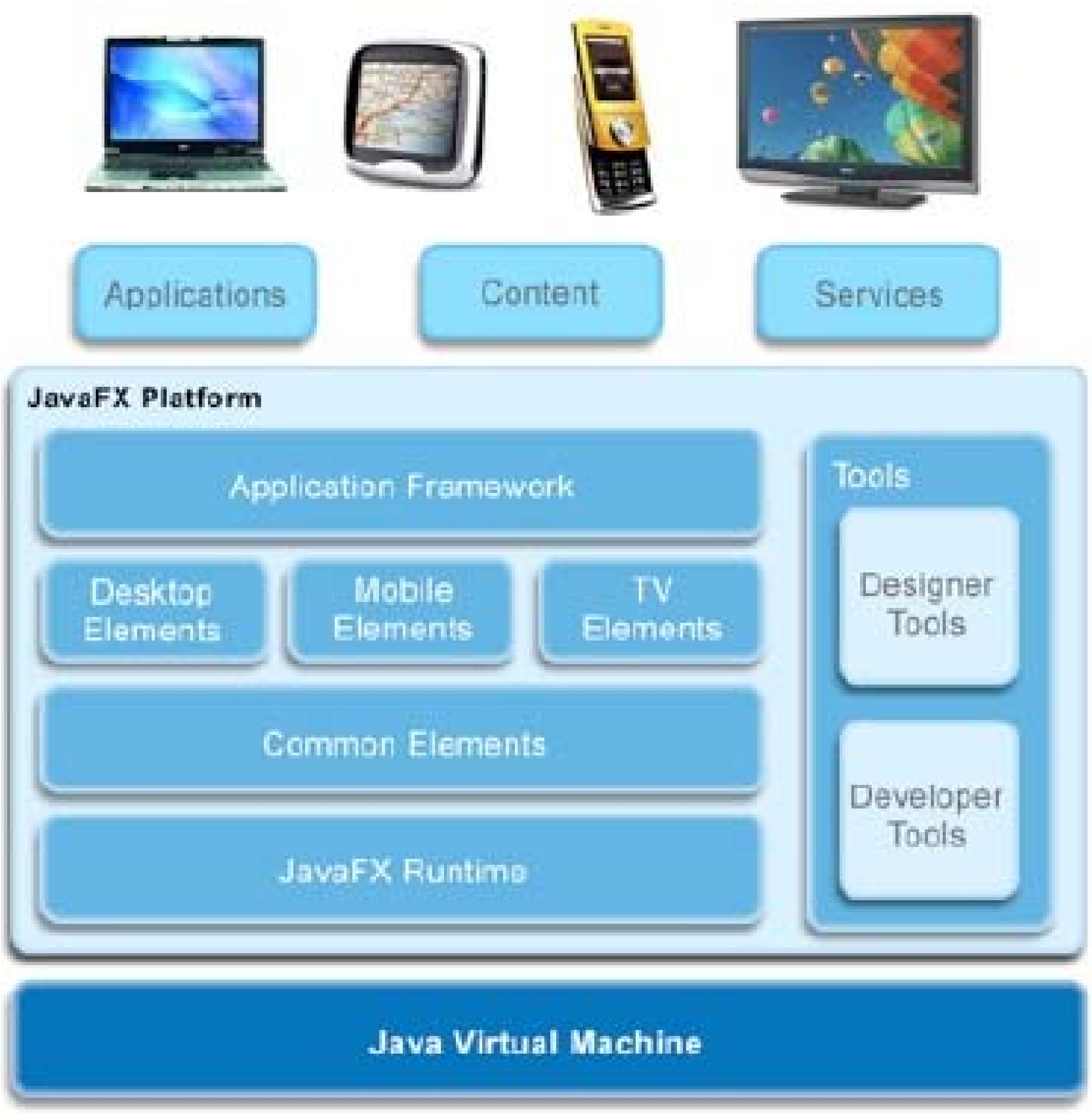
JavaFX Basics

What is JavaFX

- JavaFX is a new framework for developing Java GUI programs.
- JavaFX is used for creating desktop applications, as well as rich internet applications.
- JavaFX supports Microsoft Windows, Linux, and macOS, and all web browsers that run on them.
- JavaFX applications could run on any desktop that has Java SE, on any browser that could run Java EE, or on any mobile phone that could run Java ME.

What is JavaFX



JavaFX vs Swing and AWT

History:

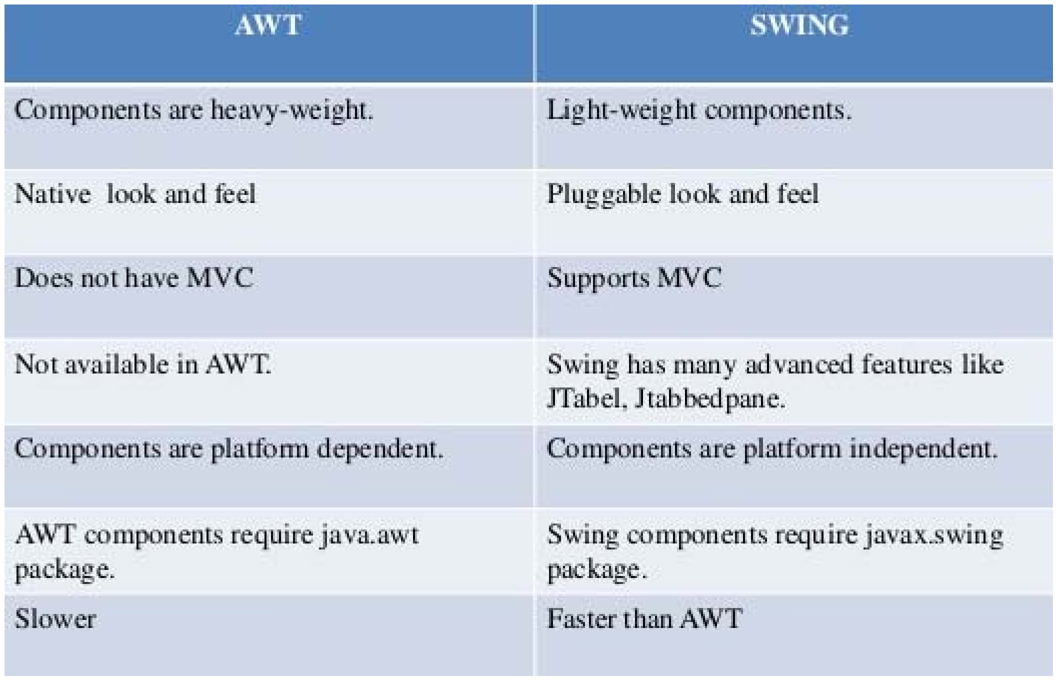
1. AWT (1996)
2. Swing (1998)
3. JavaFX (2008)

- When Java was introduced, the GUI classes were bundled in a library known as the *Abstract Windows Toolkit (AWT)*.
- AWT is fine for developing simple graphical user interfaces, but not for developing comprehensive GUI projects.
- In addition, AWT is prone to platform-specific bugs.

JavaFX vs Swing and AWT

History:

1. AWT (1996)
2. Swing (1998)
3. JavaFX (2008)



AWT	SWING
Components are heavy-weight.	Light-weight components.
Native look and feel	Pluggable look and feel
Does not have MVC	Supports MVC
Not available in AWT.	Swing has many advanced features like JTabel, Jtabbedpane.
Components are platform dependent.	Components are platform independent.
AWT components require java.awt package.	Swing components require javax.swing package.
Slower	Faster than AWT

- The AWT user-interface components were replaced by a more robust, versatile, and flexible library known as *Swing components*.
- Swing components depend less on the target platform and use less of the native GUI resource.
- Swing works only with desktop applications

JavaFX vs Swing and AWT

History:

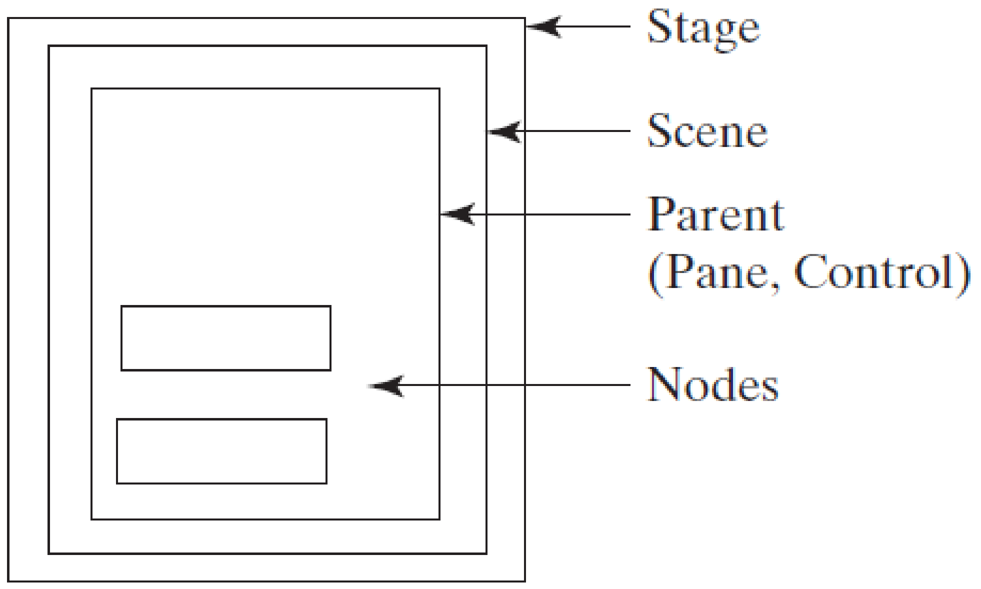
1. AWT (1996)
2. Swing (1998)
3. JavaFX (2008)

- With the release of Java 8, Swing is replaced by a completely new GUI platform known as *JavaFX*.
- With JavaFX you can control formatting with Cascading Style Sheets (CSS).
- JavaFX has more controls (collapsible, TitledPane and Accordion control)
- JavaFX has special effects (shadows, reflections, blurs, animations)

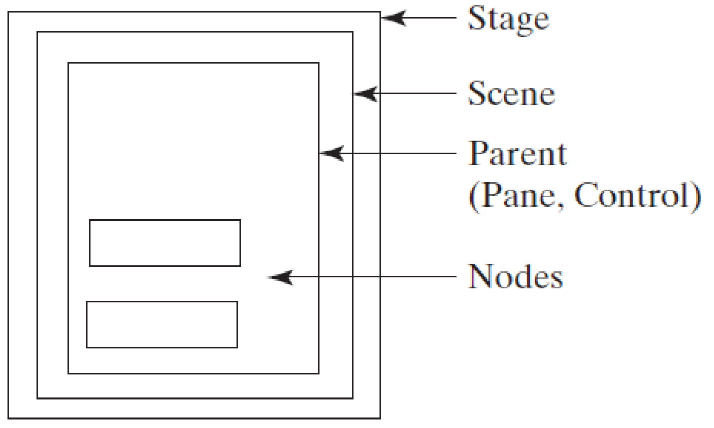
```
h1 { color: white;
  background: orange;
  border: 1px solid bla
  padding: 0 0 0 0;
  font-weight: bold;
}
/* begin: seaside-theme */

body {
  background-color:white;
  color:black;
  font-family:Arial,sans-serif;
  margin: 0 4px 0 0;
  border: 12px solid;
}
```

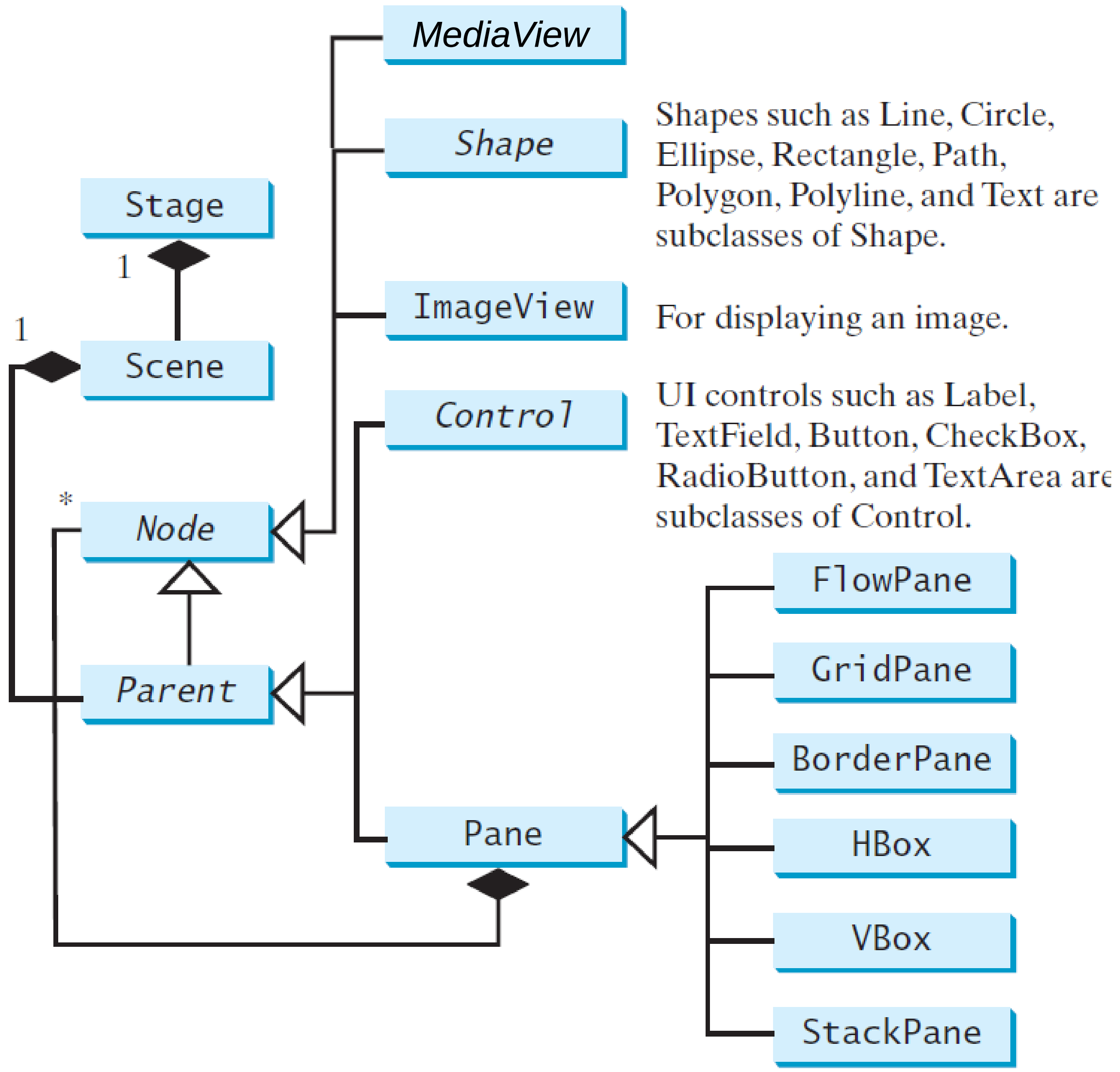
Basic Structure of JavaFX

- Application
- Override the start(Stage) method
- Stage, Scene, Node, and Parent



Panes, UI Controls, and Shapes





Shapes such as Line, Circle, Ellipse, Rectangle, Path, Polygon, Polyline, and Text are subclasses of Shape.

For displaying an image.

UI controls such as Label, TextField, Button, CheckBox, RadioButton, and TextArea are subclasses of Control.

Stage

- A stage is the "location" where graphic elements will be displayed.
- It corresponds to a window in a desktop environment.
- A stage is created by instantiating a new instance of javafx.stage.Stage class.
- The "primary stage" (which corresponds to the main window of an application) is created automatically by the JavaFX framework when an application is launched, and is supplied as argument to the start() method of the Application class.

Scene

- A scene is a “container” for the set of graphic elements to be displayed on a given stage.
- The graphic elements are called “nodes” and are organized in a scene according to a hierarchical tree structure, with one root node and a set of (direct or indirect) children of the root node.
- This tree structure is referred to as the scene graph.
- The Java class representing a scene is javafx.scene.Scene.
- When a scene is assigned to a stage, all the scene contents (defined by the scene graph) are displayed on the stage.

Node and Parent

- Any graphic element is a subclass of the abstract class javafx.scene.Node.
- A scene graph must contain at least one node (the root node).
- Parent is the base class for all nodes that have children in the scene graph.

First Example

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

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);
  }
}
```

Multi-Stage Example

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

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
  }
}
```