Graphical User Interfaces in Java

CSC02A2

Outline

Outline

Outline

GUI Frameworks
Abstract Windowing Toolkit
Swing
JavaFX

Java GUI Classes
Java GUI Classes
Layout Nodes

Low-level Rendering
The GraphicsContext class
GUI Coordinate System
The redrawCanvas method

More JavaFX Code
Custom JavaFX Canvas
Properties
File chooser

GUI Frameworks

Abstract Windowing Toolkit

Initial Java GUI framework. GUI components are mapped directly to the operating system (peer model). Due to this direct mapping AWT components are referred to as heavyweight components. Many problems with this framework:

- Due to the many layers of abstraction between a `Java` application and the host operating system, heavyweight components are slow to create and manipulate.
- It is not possible to create new components as they require an operating system component to map to.

Swing

Newer Java GUI framework. Swing components are painted directly on canvases using Java code. Swing components depend less on the operating system and use less of the native GUI resources. Mostly for desktop applications. Referred to as lightweight components.

Although the rendering aspect of AWT was replaced by Swing, many helper classes from AWT are still widely used.

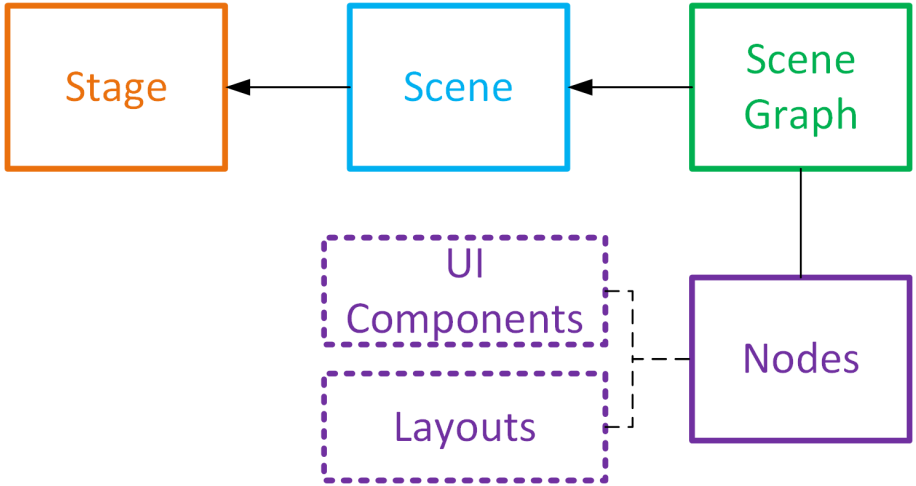
JavaFX

JavaFX incorporates modern GUI technologies to enable you to develop rich Internet applications. A rich Internet application (RIA) is a Web application designed to deliver the same features and functions normally associated with desktop applications. A JavaFX application can run seemlessly on a desktop and from a Web browser.

Additionally, JavaFX provides a multi-touch support for touch-enabled devices such as tablets and smart phones. JavaFX has a built-in 2D, 3D, animation support, video and audio playback, and runs as a stand-alone application or from a browser.

Java GUI Classes

Java GUI Classes I



Java GUI Classes II

JavaFX makes use of multiple classes to construct the GUI.

The following are used to construct a JavaFX application:

- `javafx.application.Application` - hosts the Stage.
- `javafx.stage.Stage` - displays one Scene at a time.
- `javafx.scene.Scene` - holds one Scene Graph.
- `Scene Graph` - a hierarchy of Nodes descending from the root node.
- `javafx.scene` - package of Nodes to construct a Scene Graph.
- `javafx.scene.control` - control Nodes for user interaction.
- `javafx.scene.layout` - Nodes to layout other nodes.

Java GUI Classes III

GUI helper classes that enable further functionality out of JavaFX applications:

- `javafx.scene.canvas` - a canvas for rendering.
- `javafx.scene.canvas.GraphicsContext` - allows simple items to be drawn (strings, lines, shapes).
- `javafx.scene.paint.Color` - colour class used to render components with colour.
- `javafx.stage.FileChooser` - selects a file to be opened by the application.
- `javafx.beans.property` - special variables which can be observed for changes.
- `javafx.beans.value.ChangeListener` - a listener for changes in property values.
- `javafx.beans.value.ObservableValue` - a property value that is changed (passed to event handlers).

Layout Nodes I

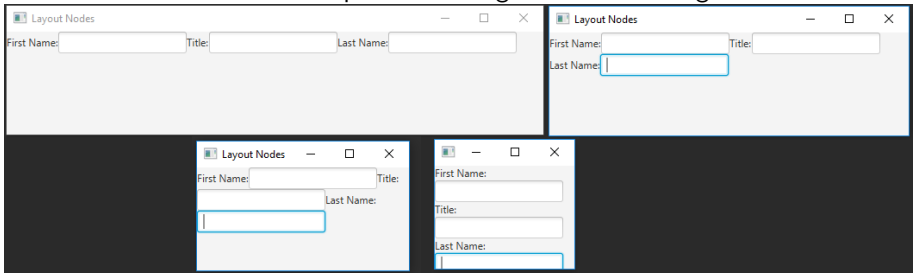
Nodes cannot act independently of one another. Nodes need to be constructed into a Scene Graph. In order to make applications more consistent (and reduce fine tuning by the programmer), JavaFX has special nodes which store and manage the layout of other nodes. These are called layout nodes.

When placing nodes in the GUI, hard-coded pixel measurements might look fine on one system but be unusable on another (think desktop vs mobile phone).

Java's layout nodes provide a level of abstraction that automatically maps your user interface on all window systems.

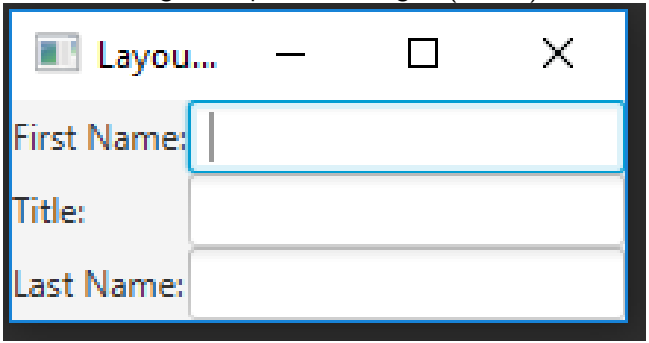
Layout Nodes II

`FlowPane` - components arranged from left to right.



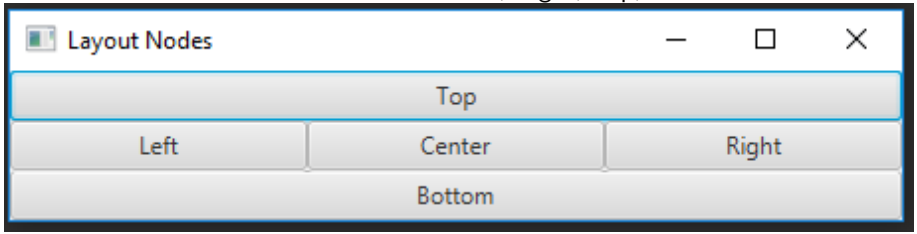
Layout Nodes III

`GridPane` - arranges components in a grid (matrix) formation.



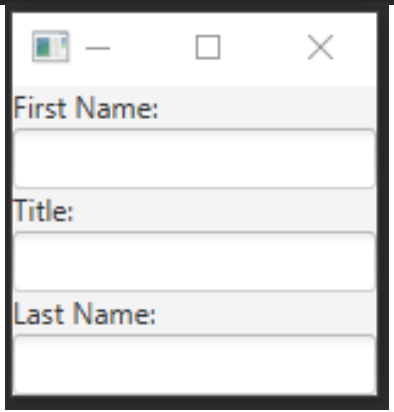
Layout Nodes IV

`BorderPane` - divides a container into Left, Right, Top, Bottom and Center.



Layout Nodes V

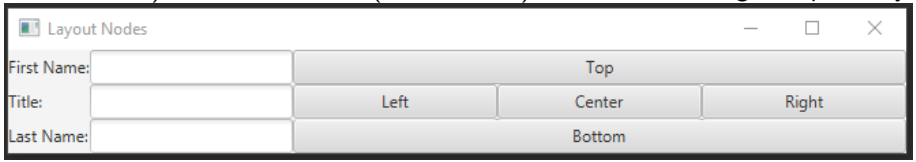
`HBox` and `VBox` - stack nodes horizontally or vertically.



Layout Nodes VI

Layouts nodes can be combined by adding layout nodes to other nodes.

In this case an `HBox` has two children; a GridPane (with labels and textfields) and a BorderPane (with buttons) - on the left and right respectively.



Low-level Rendering

JavaFX Canvas

In JavaFX we have access to a very special node called a `Canvas` from the `javafx.scene.canvas` package.

The `Canvas` provides us with the ability to draw shapes and text onto the node itself in a programmatic fashion.

The `Canvas` makes use of a `GraphicsContext` class to perform the actual draw operations and a `redrawCanvas` method to initiate the draw operations.

The `GraphicsContext` class

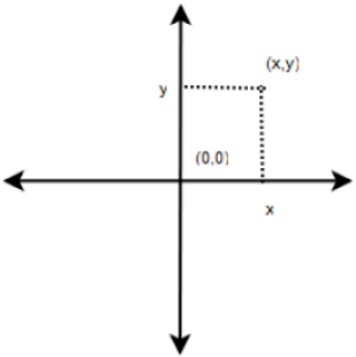
The `GraphicsContext` class is an abstract class that provides a device independent graphics context way of drawing text, basic shapes and images. The `Canvas` is responsible for managing its own `GraphicsContext` to render the canvas.

We can use the `getGraphicsContext2D` method to access the `GraphicsContext` instance of the current canvas. This is the only way to obtain a reference to a `GraphicsContext` instance so that we can use its draw operations.

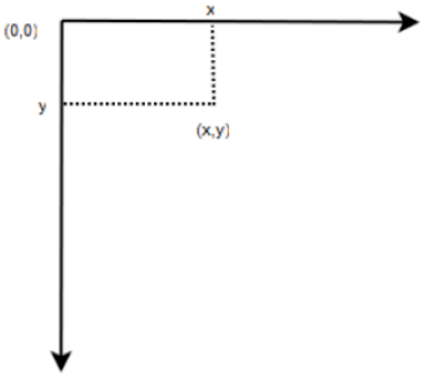
Custom drawing is possible using the `redrawCanvas` method with a `GraphicsContext`. Extra elements can be drawn or the look of the component can be completely changed using this method.

Note: The onus is on the programmer to call this method when something needs to be drawn.

GUI Coordinate System



The cartesian plane



The Java coordinate system

`GraphicsContext` Drawing methods

Most drawing methods in the `GraphicsContext` class require coordinates to draw elements. The following methods can be used to draw different elements:

- `clearRect` - Clear part of the component to the background colour.
- `setStoke` - Sets the colour of stroke operations.
- `strokeText` - Draw text.
- `strokeLine` - Draw a line.
- `strokeOval` - Draw an oval.
- `strokeRect` - Draw a rectangle.
- `setFill` - Sets the colour of fill operations.
- `fillOval` - Fills an oval.
- `fillRect` - Fills a rectangle.
- `fillPolygon` - Draw an arbitrary shape using arrays or ordered x and y coordinates.
- `drawImage` - Draw an image.

More JavaFX Code

A Custom JavaFX Canvas

```
1  public class MyCanvas extends Canvas{
2    public MyCanvas() {
3      //Constructor code
4    }
5    public void manuallyRedrawCanvas(){
6      redrawCanvas(); //We can call the redrawCanvas method manually when
       ↪  data changes
7    }
8    //RedrawCanvas method
9    public void redrawCanvas() {
10     //Get the GraphicsContext
11     GraphicsContext gc = this.getGraphicsContext2D();
12     //Clear canvas
13     gc.clearRect(0, 0, this.getWidth(), this.getHeight());
14     //Set Fill colour
15     gc.setFill(Color.GRAY);
16     //Draw a filled rectangle
17     gc.fillRect(0, 0, 10, 55);
18   }
19 }
```


Using Properties to Interact with the Canvas

We can create a Property and bind a listener to it to perform actions when the data in the property variable changes.

```
1  //Create a property using one of the Property Classes
2  BooleanProperty dataChangedProperty = new SimpleBooleanProperty(false);
3  //Implement an anonymous listener for changes in the property
4  dataChangedProperty.addListener(new ChangeListener<Boolean>() {
5    //Implement the appropriate method to handle the event
6    @Override
7    public void changed(ObservableValue<? extends Boolean> observable,
     ↪  Boolean oldValue, Boolean newValue) {
8      //Do something when property changes
9      canvas.manuallyRedrawCanvas();
10     //(Why not redraw the Canvas with the new data)
11   }
12 });
```


File chooser

```
1  //Create a file chooser
2  FileChooser fc = new FileChooser();
3  //Give it a title
4  fc.setTitle("Choose the file");
5  //Set the starting directory
6  fc.setInitialDirectory(new File("."));
7  //Show the fileChooser using the primaryStage as the parent
8  File file = fc.showOpenDialog(primaryStage);
9  //Check to see if a valid file was returned
10 if(file != null) {
11   //Do something with the file
12   //Or access the file's path with: file.getAbsolutePath();
13 }
```