UNIVERSITÉ Concordia UNIVERSITY

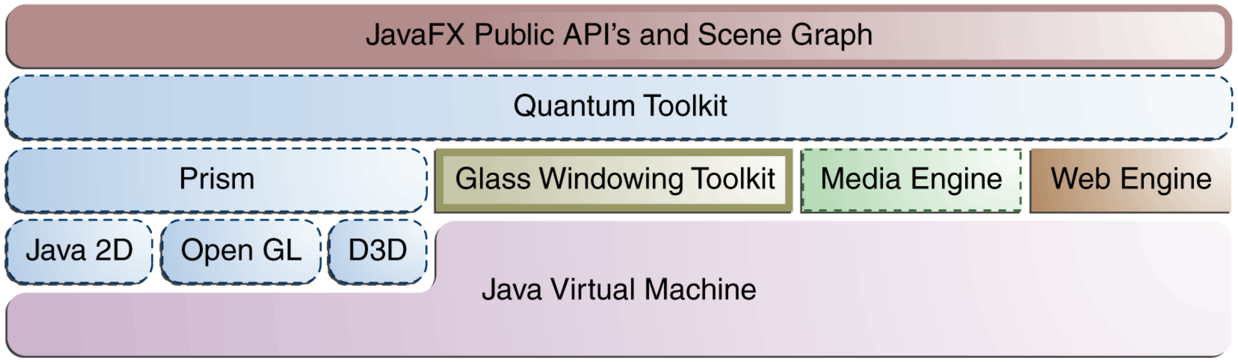
CENTRE FOR CONTINUING EDUCATION

DESKTOP APPLICATION DEVELOPMENT WITH JAVA – CEJV569

Lecture #7

- JavaFX
  - FXML
  - Scene Builder
- Internationalization

JavaFX Runtime High Level Architecture



JavaFX Glossary

- **Glass Windowing Toolkit**: Provides native operating services, such as managing the windows, timers, and surfaces
- **Prism**: Graphics pipeline that can run on hardware and software renderers
- **Quantum Toolkit**: Ties Prism and Glass together and makes them available to the JavaFX APIs

Java APIs and FXML

Java APIs for JavaFX	FXML
• End-to-end Java development • Java language features - generics, annotations, multi-threading • Fluent API for UI construction • Alternative JVM supported languages (e.g. Groovy, Scala) with JavaFX • Leverage sophisticated Java IDEs, debuggers and profilers • Java APIs preserve convenient JavaFX Script features (e.g., bind)	• Scriptable, XML-based markup language for defining UI • Convenient alternative to developing UI programmatically in Java • Easy to learn and intuitive for developers familiar with web technologies or other markup based UI technologies • Powerful scripting feature allows embedding scripts within FXML. Any JVM scripting language can be used, including JavaScript, Groovy, and Scala

Graphics and Media

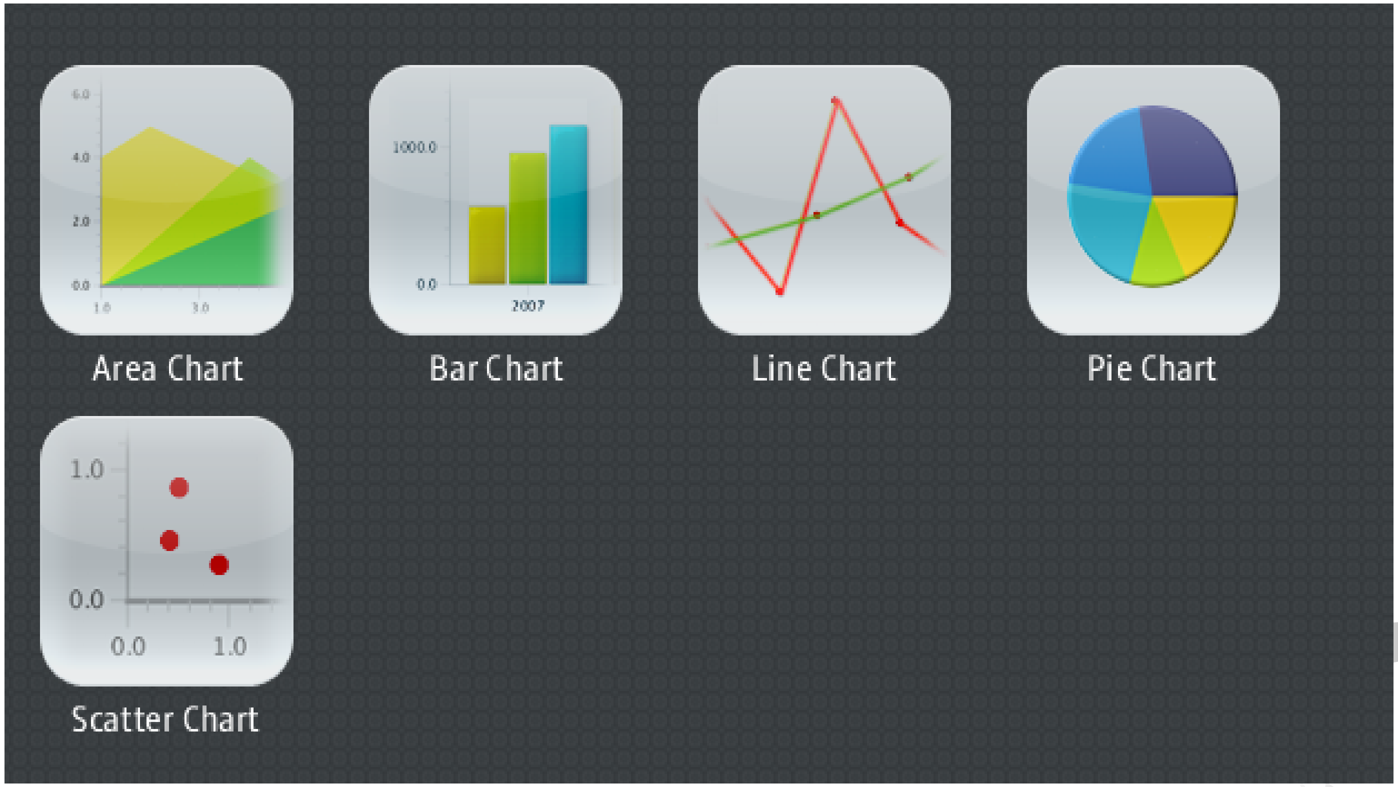
New Graphics Pipeline

- New hardware accelerated graphics pipeline (Prism)
- New windowing toolkit (Glass) for Prism
- Java2D software pipeline under Prism
- High-level support for making rich graphics simple
  - Shadows, Blurs, Reflections, Effects, 2D transforms
  - 3D Transforms today; Full 3D objects in future

Media

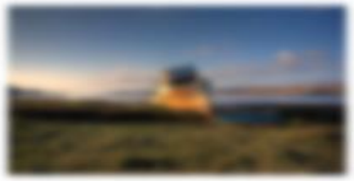
- Stable media framework based on GStreamer
- VP6, MP3 playback of Web multimedia content
- Low latency audio
- Performance improvements
- Full screen video

Charts



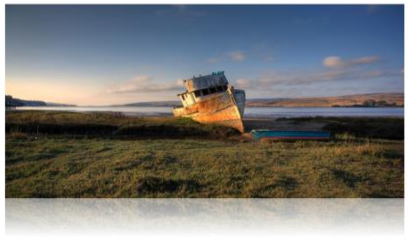
Effects...

GaussianBlur

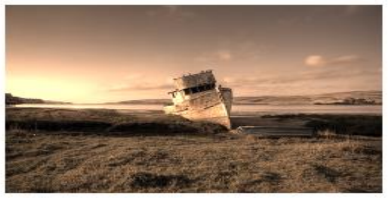


InnerShadow

Reflection



SepiaTone



Transforms

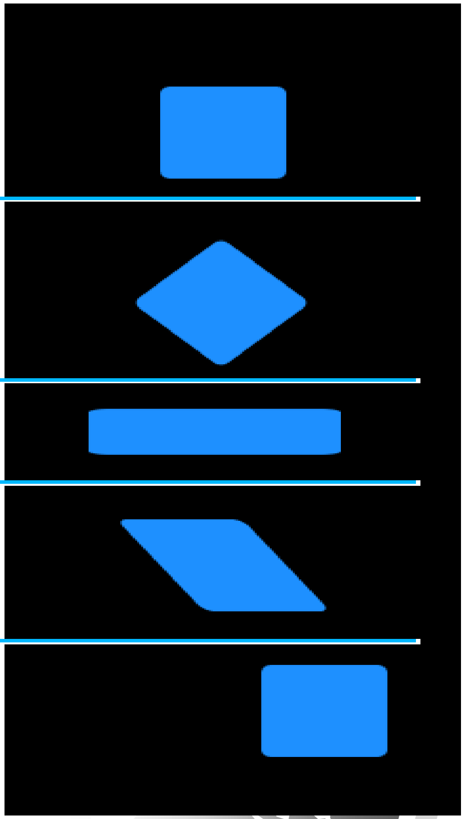
```
Rectangle rect=new Rectangle(0,0,60,60);
rect.setFill(Color.DODGERBLUE);
rect.setArcWidth(10);
rect.setArcHeight(10);
```

```
rect.setRotate(45);
```

```
rect.setScaleX(2);
rect.setScaleY(0.5);
```

```
Shear shear = new Shear(0.7, 0);
rect.getTransforms().add(shear);
```

```
rect.setTranslateX(40);
rect.setTranslateY(10);
```

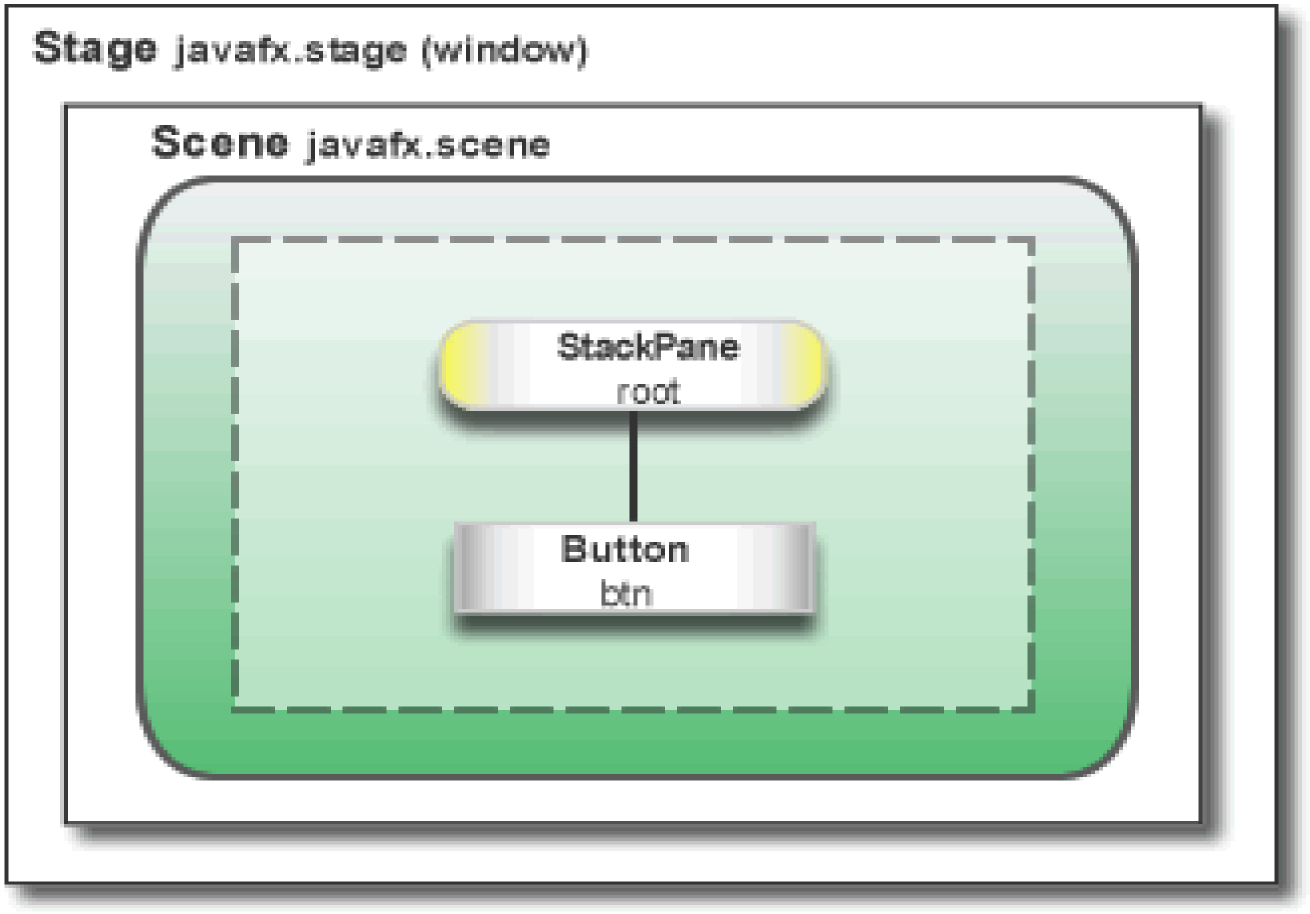


It begins on the Stage

- ☞ A Stage contains the UI of a JavaFX app
- ☞ A desktop Stage has its own top-level window that includes a border and title bar
- ☞ Initial stage is created by the JavaFX runtime
- ☞ Passed to the start() method
- ☞ Stage class has a set of properties and methods
  - `stage.setTitle("Hello World");`
  - `stage.show();`

Now its on to the Scene

- Scene is the top container in the JavaFX scene graph
- Holds the graphical elements that are displayed on the Stage
- Every element is a graphical node
  - Node is any class that extends javafx.scene.Node
- A scene graph is a hierarchical representation of the Scene
- Elements in the scene graph may contain child elements, and all of them are instances of the Node class
- Scene contains properties such as its width and height
- Scenes usually contain a root node which in turn contains all other nodes

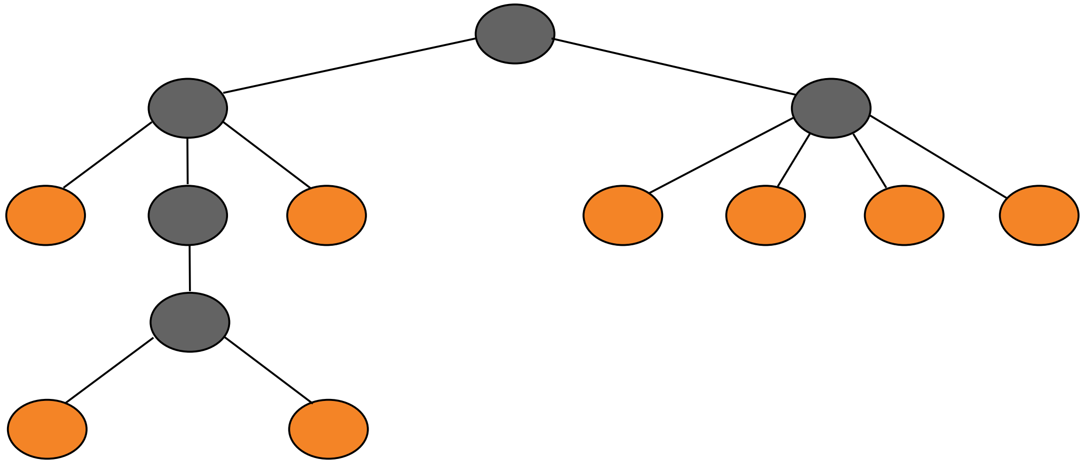

Stage javafx.stage (window)
Scene javafx.scene
StackPane
root
Button
btn

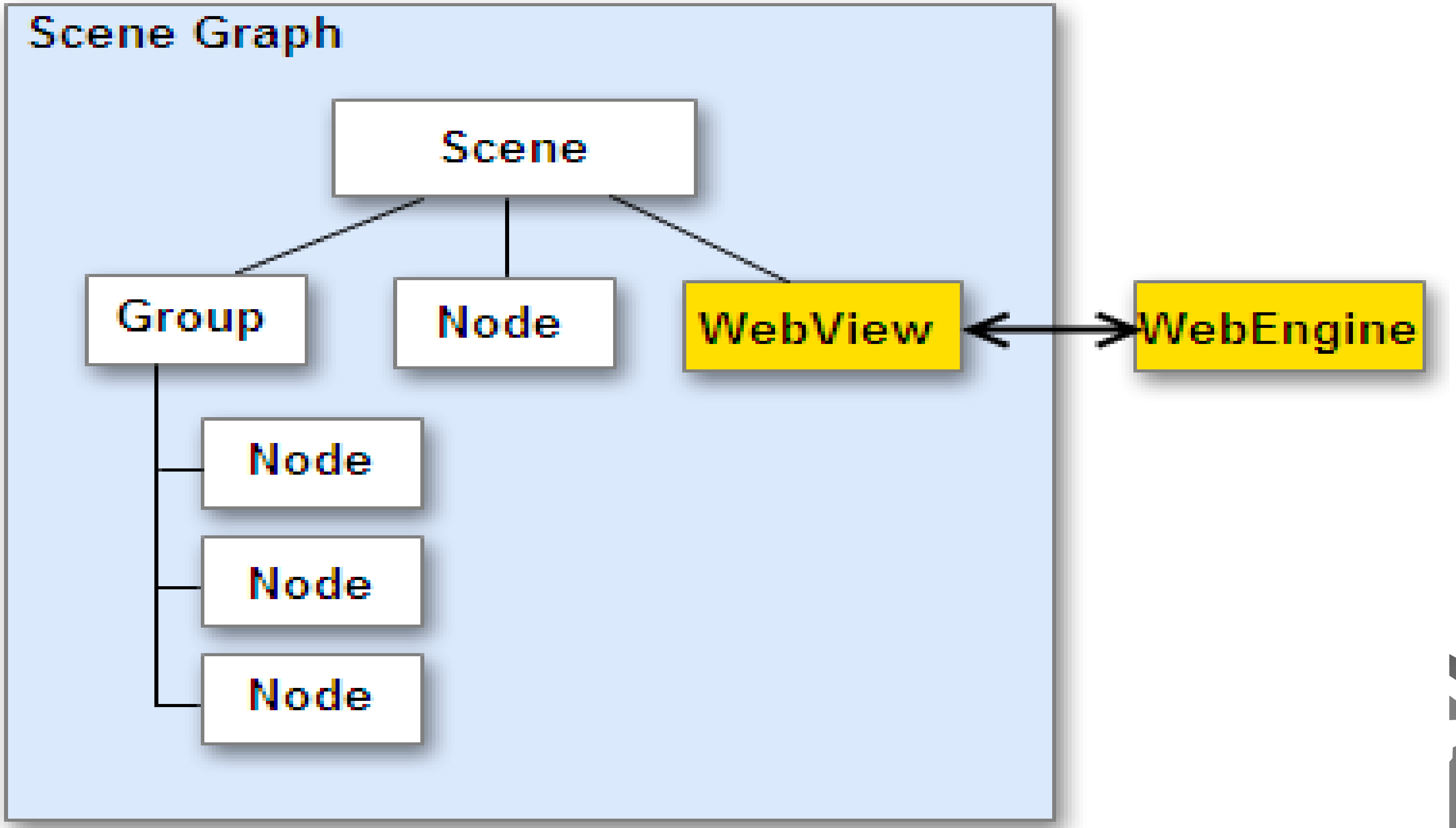
Scene Graph

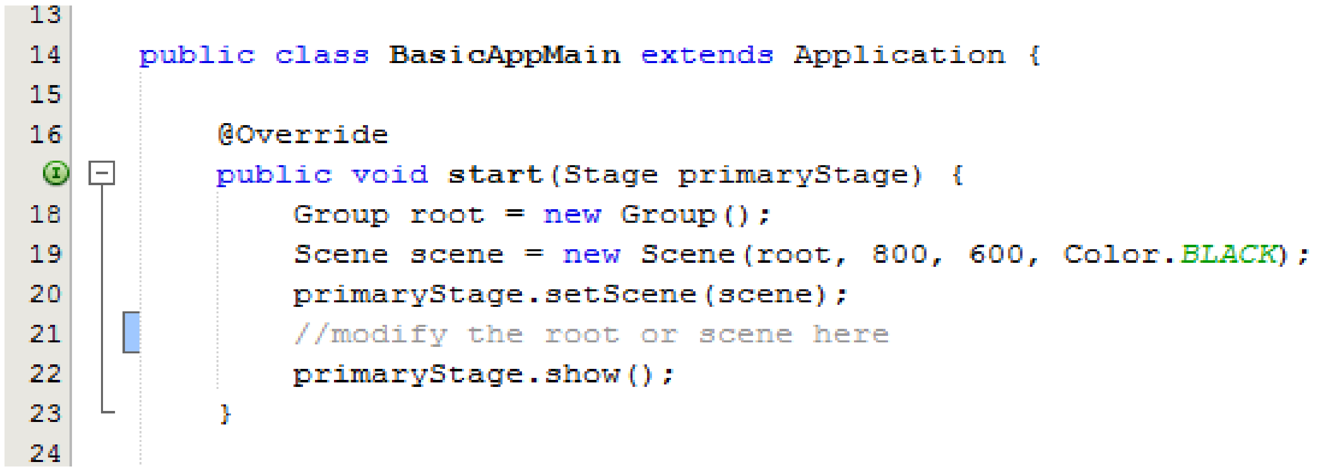
- AWT and Swing use a container/component hierarchy for organizing the GUI.
- Layout managers are fundamental to this, but can make development difficult and involved.
- JavaFX uses a scene graph which will be familiar to developers who have programmed in 3D.
- The concept is that all components in the GUI are represented by nodes.
- Each node can have one parent and groupings can be made by attaching multiple nodes (which may themselves be parents) to a parent.
- Applying effects to groups of nodes is simply a matter of applying the effect to the parent node.
- The ordering can also be altered within a group and for a group as a whole.

Scene Graph

- Directed Acyclic Graph
- Parents and children
- Representation of the GUI components



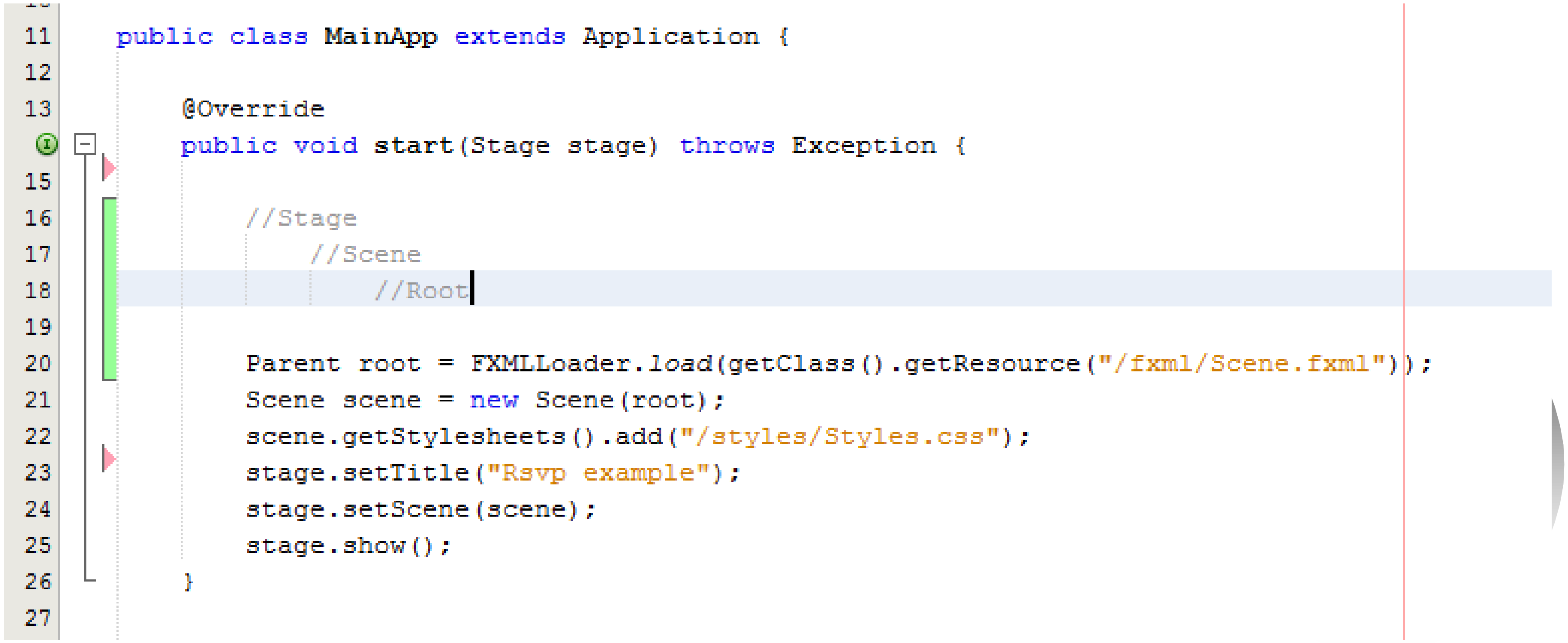

Scene Graph
Scene
Group
Node
WebView
WebEngine
Node
Node
Node



```
public class BasicAppMain extends Application {

    @Override
    public void start(Stage primaryStage) {
        Group root = new Group();
        Scene scene = new Scene(root, 800, 600, Color.BLACK);
        primaryStage.setScene(scene);
        //modify the root or scene here
        primaryStage.show();
    }

```



```
public class MainApp extends Application {

    @Override
    public void start(Stage stage) throws Exception {

        //Stage
            //Scene
                //Root

        Parent root = FXMLLoader.load(getClass().getResource("/fxml/Scene.fxml"));
        Scene scene = new Scene(root);
        scene.getStylesheets().add("/styles/Styles.css");
        stage.setTitle("Rsvp example");
        stage.setScene(scene);
        stage.show();
    }

```

Programmatic Versus Declarative Creation of the User Interface

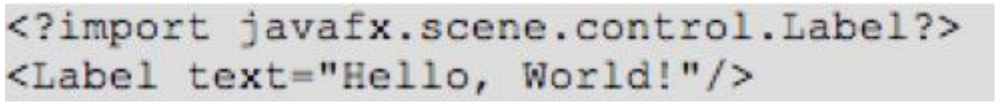
- JavaFX platform provides two complementary ways for creating a UI
  - Programmatic
    - User interface is created in code
  - Declarative
    - User interface is created with a tool that generates the code for you
    - JavaFX tool is Scene Builder

Scene Builder

- Given the recent announcement that Oracle will no longer be providing builds of the JavaFX Scene Builder tool, GluOn started to provide support for Scene Builder as an open source software.
- http://gluonhq.com/open-source/scene-builder/

FXML

- From 2.0 onwards, FXML has been introduced
  - More powerful and XML-based
  - All developers have knowledge of XML
- FXML – declarative XML-based language



```
<?import javafx.scene.control.Label?>
<Label text="Hello, World!"/>
```

- We can also use native Java code, but using FXML:
  - You will be forced to keep your presentation layer separate from the logic (business layer)
  - It is easier for you to maintain and edit the presentation layer
  - You can use JavaFx scenebuilder

Hello World Example (FXML)

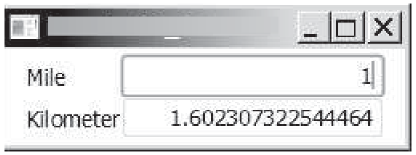
- FXML:
  - In your Netbeans create a new JavaFX FXML Application:
    - New Project > JavaFX (instead of Java) > JavaFX FXML Application

Code Structure

- FXMLExample.java:
  - This file takes care of the standard Java code required for an FXML application.
- FXMLDocument.fxml:
  - This is the FXML source file in which you define the user interface.
- FXMLDocumentController.java:
  - This is the controller file for handling the mouse and keyboard input.

Exercise 25

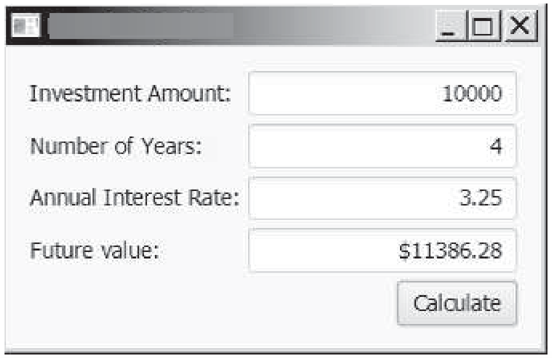
- Using Scene Builder
  - Create a miles/kilometers converter



Mile: 1

Kilometer: 1.602307322544464

  - Create an investment-value calculator



Investment Amount: 10000

Number of Years: 4

Annual Interest Rate: 3.25

Future value: $11386.28

Java’s International Support

1. Use *Unicode*

2. Provide the Locale class to encapsulate information about a specific locale. A Locale object determines how locale-sensitive information, such as date, time, and number, is displayed, and how locale-sensitive operations, such as sorting strings, are performed.

3. Use the ResourceBundle class to separate locale-specific information such as status messages and the GUI component labels from the program. The information is stored outside the source code and can be accessed and loaded dynamically at runtime from a ResourceBundle, rather than hard-coded into the program.

The `Locale` Class

A `Locale` object represents a specific geographical, political, or cultural region. An operation that requires a `Locale` to perform its task is called *locale-sensitive*. You can use `Locale` to tailor information to the user.

java.util.Local	
+Locale(language: String)	Constructs a locale from a language code.
+Locale(language: String, country: String)	Constructs a locale from language and country codes.
+Locale(language: String, country: String, variant: String)	Construct a locale from language, country, and variant codes.
+getCountry(): String	Returns the country/region code for this locale.
+getLanguage(): String	Returns the language code for this locale.
+getVariant(): String	Returns the variant code for this locale.
+getDefault(): Locale	Gets the default locale on the machine.
+getDisplayCountry(): String	Returns the name of the country as expressed in the current locale.
+getDisplayLanguage(): String	Returns the name of the language as expressed in the current locale.
+getDisplayName(): String	Returns the name for the locale. For example, the name is Chinese (China) for the locale Locale.CHINA.
+getDisplayVariant(): String	Returns the name for the locale's variant if exists.

Creating a Locale

To create a `Locale` object, you can use the following constructor in `Locale` class:

```
Locale(String language, String country)

Locale(String language, String country, String
variant)
```

```
Example:

new Locale("en", "US");

new Locale("fr", "CA");

Locale.CANADA

Locale.CANADA_FRENCH
```

The Locale-Sensitive Operations

An operation that requires a Locale to perform its task is called *locale-sensitive*. Displaying a number as a date or time, for example, is a locale-sensitive operation; the number should be formatted according to the customs and conventions of the user's locale.

Several classes in the Java class libraries contain locale-sensitive methods. Date, Calendar, DateFormat, and NumberFormat, for example, are locale-sensitive. All the locale-sensitive classes contain a static method, getAvailableLocales(), which returns an array of the locales they support. For example,

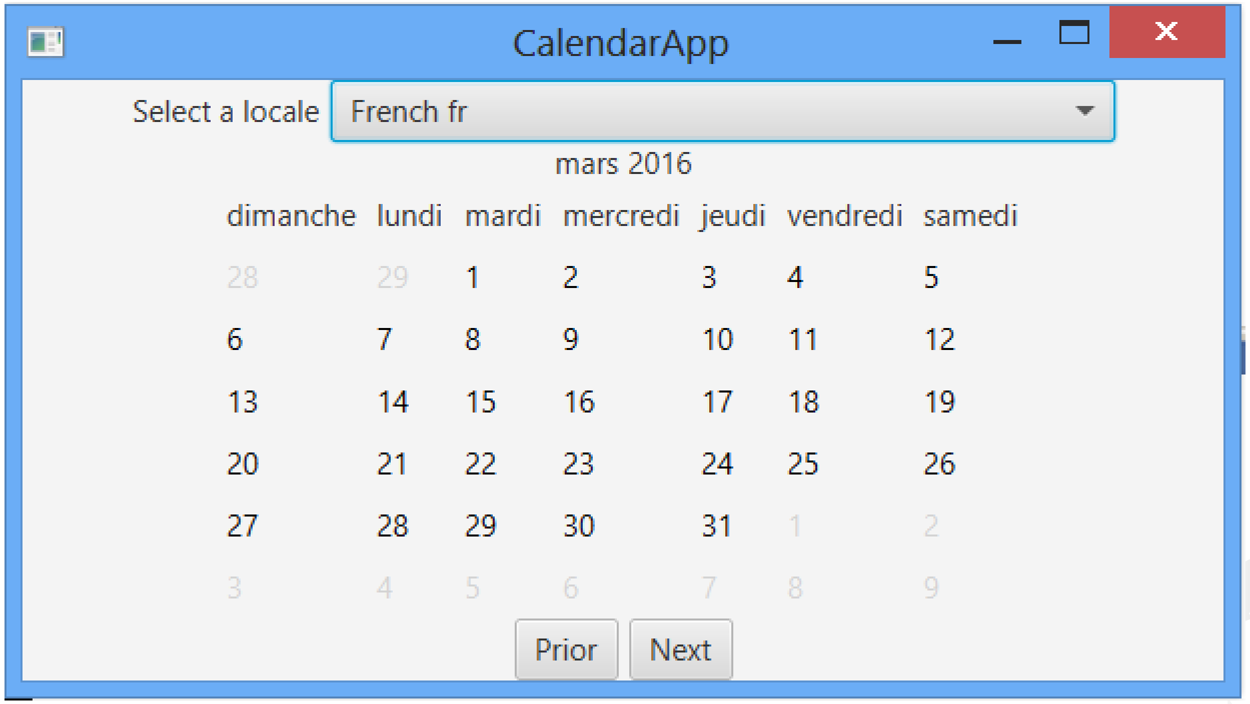
```
Locale[] availableLocales = Calendar.getAvailableLocales();
```

returns all the locales for which calendars are installed.

Example:
Displaying a Calendar

- Objective: Display the calendar based on the specified locale. The user can specify a locale from a combo box that consists of a list of all the available locales supported by the system.

Example, cont.

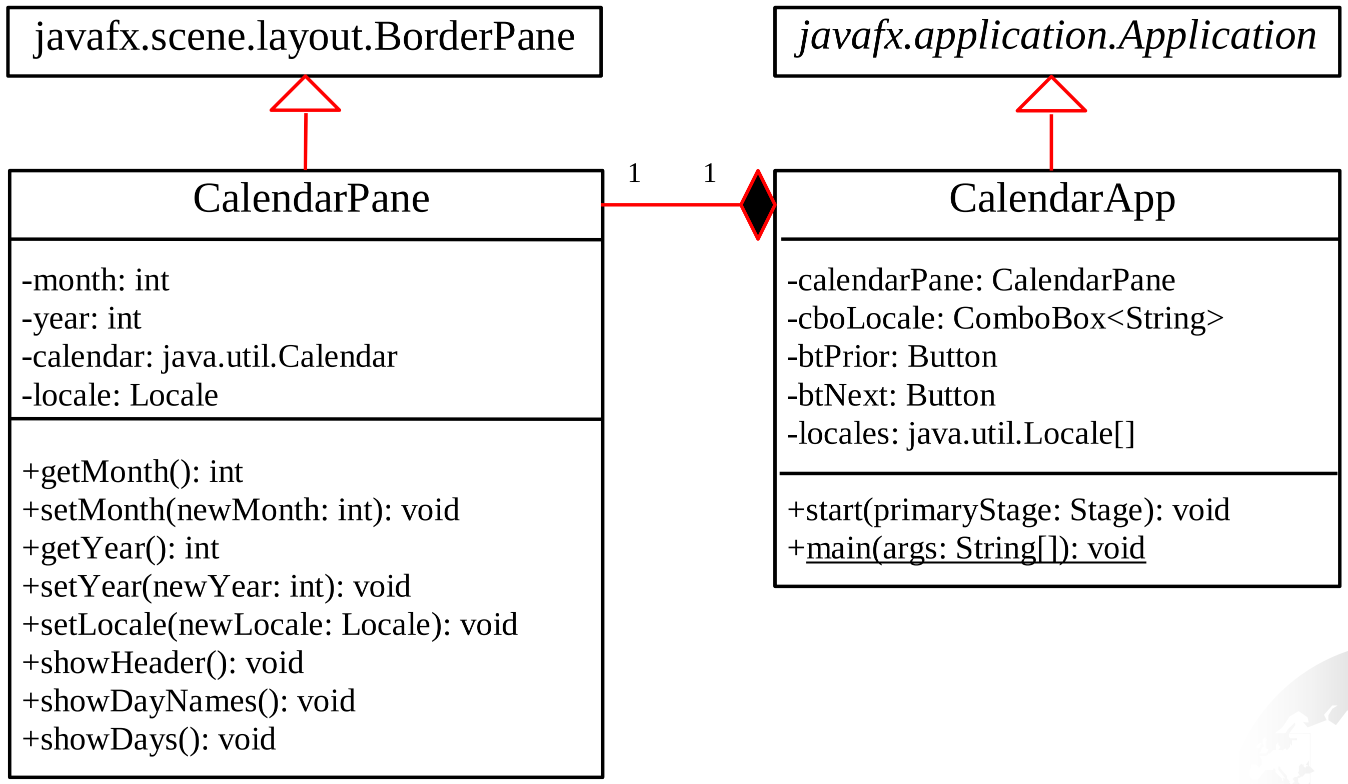


Select a locale: French fr

mars 2016

dimanche	lundi	mardi	mercredi	jeudi	vendredi	samedi
28	29	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Example, cont.



javafx.scene.layout.BorderPane
javafx.application.Application
CalendarPane
-month: int
-year: int
-calendar: java.util.Calendar
-locale: Locale
+getMonth(): int
+setMonth(newMonth: int): void
+getYear(): int
+setYear(newYear: int): void
+setLocale(newLocale: Locale): void
+showHeader(): void
+showDayNames(): void
+showDays(): void
1
1
CalendarApp
-calendarPane: CalendarPane
-cboLocale: ComboBox<String>
-btPrior: Button
-btNext: Button
-locales: java.util.Locale[]
+start(primaryStage: Stage): void
+main(args: String[]): void

CalendarPane

CalendarApp

Resource Bundles

A *resource bundle* is a Java class file or a text file that provides locale-specific information. This information can be accessed by Java programs dynamically.

When your program needs a locale-specific resource, a message string for example, your program can load the string from the resource bundle that is appropriate for the desired locale. In this way, you can write program code that is largely independent of the user's locale isolating most, if not all, of the locale-specific information in resource bundles.

Working with the ResourceBundle class

- MessageBundles are loaded by the `ResourceBundle` class

```
ResourceBundle rb =
      ResourceBundle.getBundle("MessagesBundle");
```

- Default location for bundles is in the root of the project
- Uses the default locale
- To use a locale that you define use the overloaded version of `getBundle`

```
ResourceBundle rb =
    ResourceBundle.getBundle("MessagesBundle",locale);
```

- where `locale` is an object of type `Locale`

MessagesBundle

- A properties text file
- Key is the tag that will be used in the code
- Value is what will be substituted for the tag

```
# English Canada                # French Canada
Welcome = Welcome               Welcome = Bienvenue
UserName = User Name            UserName = Nom d'utilisateur
Password = Password             Password = Mot de passe
SignIn = Sign In                SignIn = Connexion
SignInMsg = Button pressed      SignInMsg = Bouton enfoncé
```

Naming the Bundle

- May have any name you wish
- The Java language code is appended to the bundle name
- `MessagesBundle_en_CA.properties`
- `MessagesBundle_fr_CA.properties`
- List of codes:
- http://www.oracle.com/technetwork/java/javase/java8locales-2095355.html
- There should also be a default bundle without a country/language code that will be used if an appropriate bundle is not found
- `MessagesBundle.properties`

Working with the ResourceBundle class

- Once you have a reference to the ResourceBundle you use it like a property but with getString rather than getProperty

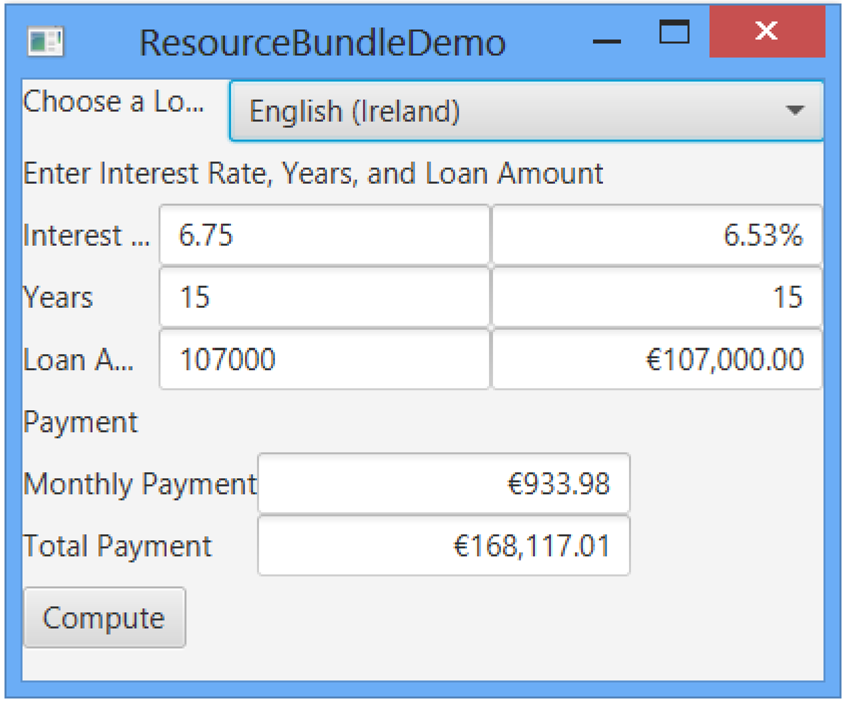
```
ResourceBundle rb =
       ResourceBundle.getBundle("MessagesBundle");
String s = rb.getString("Password");
```

- Any place that you need text in your code you should use the ResourceBundle
- Software is international and internationalization allows you to quickly prepare versions of your programs in any language
- Living here in Quebec this is effectively mandatory

Example: Using Resource Bundles

- Objective: This example modifies the NumberFormattingDemo program in the preceding example to display messages, title, and button labels in English, Chinese, and French languages.

Example, cont.


ResourceBundleDemo

Choose a Lo... English (Ireland)

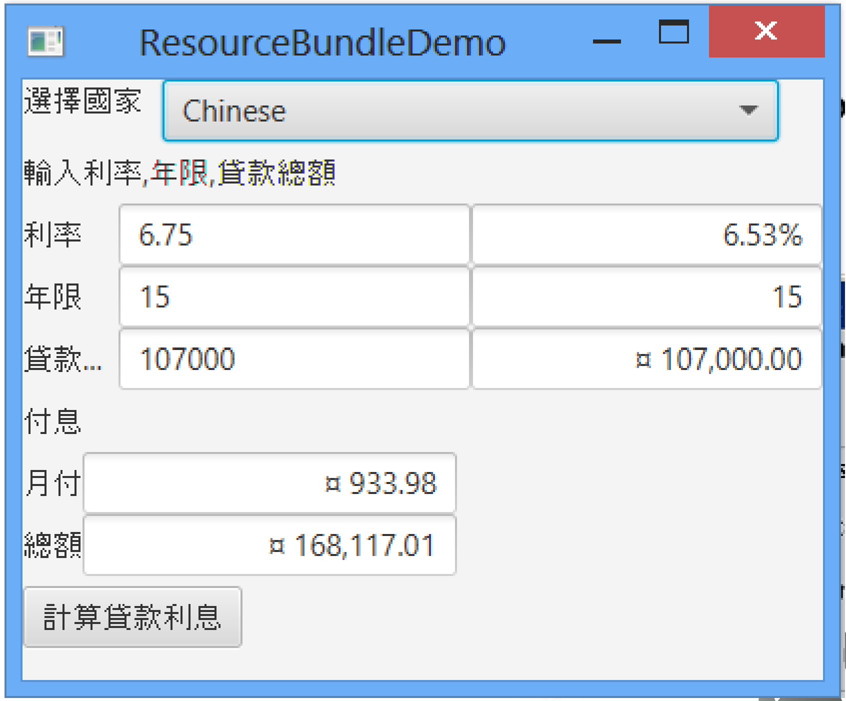
Enter Interest Rate, Years, and Loan Amount

Interest ...	6.75	6.53%
Years	15	15
Loan A...	107000	€107,000.00

Payment

Monthly Payment	€933.98
Total Payment	€168,117.01

Compute


ResourceBundleDemo

選擇國家 Chinese

輸入利率,年限,貸款總額

利率	6.75	6.53%
年限	15	15
貸款...	107000	¤ 107,000.00

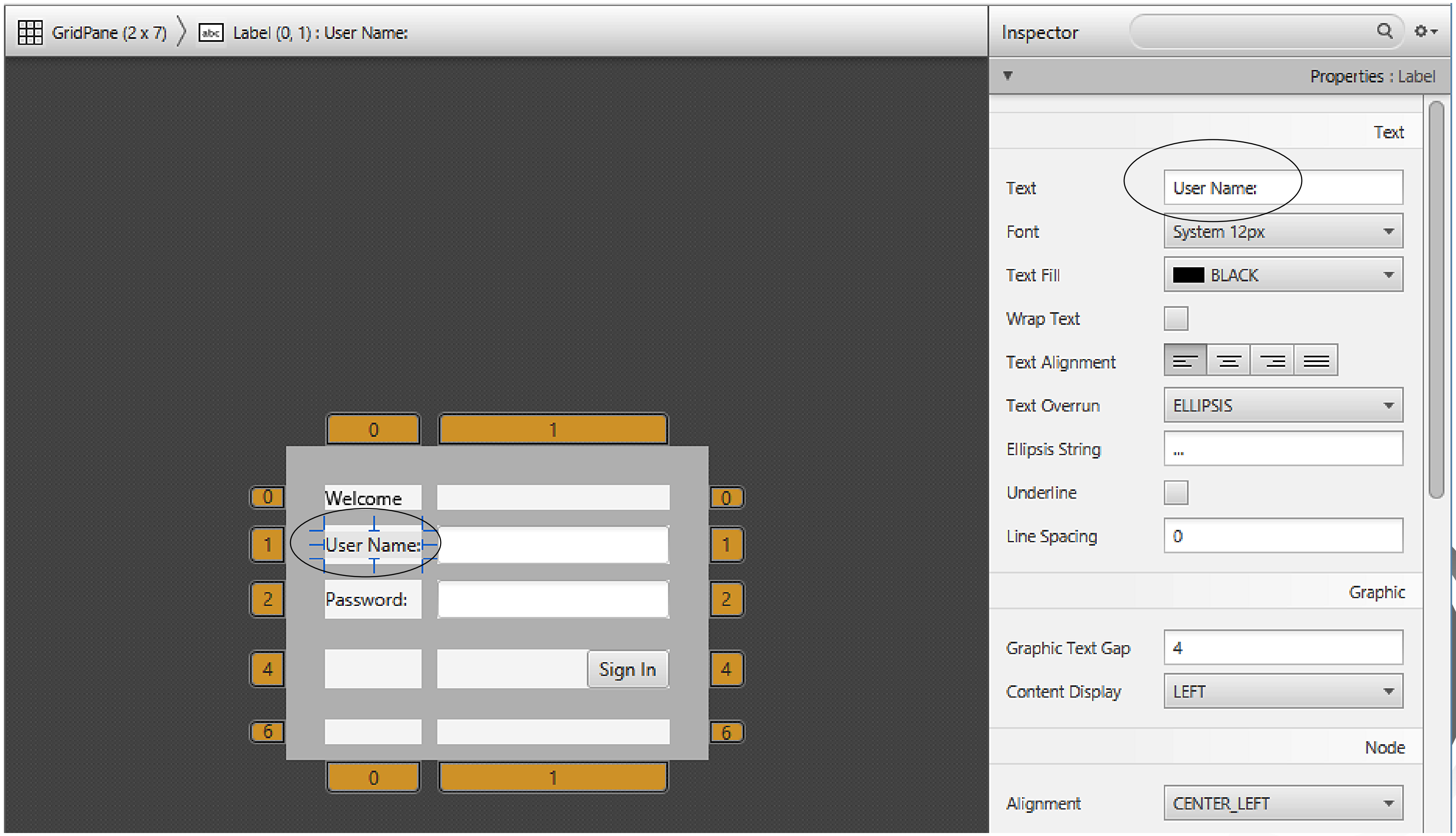
付息

月付	¤ 933.98
總額	¤ 168,117.01

計算貸款利息

ResourceBundleDemo

Hard Coding a String



GridPane (2 x 7) 〉 Label (0, 1) : User Name:

Inspector

Properties : Label

Text

Text	User Name:
Font	System 12px
Text Fill	■ BLACK
Wrap Text	☐
Text Alignment	
Text Overrun	ELLIPSIS
Ellipsis String	...
Underline	☐
Line Spacing	0

Graphic

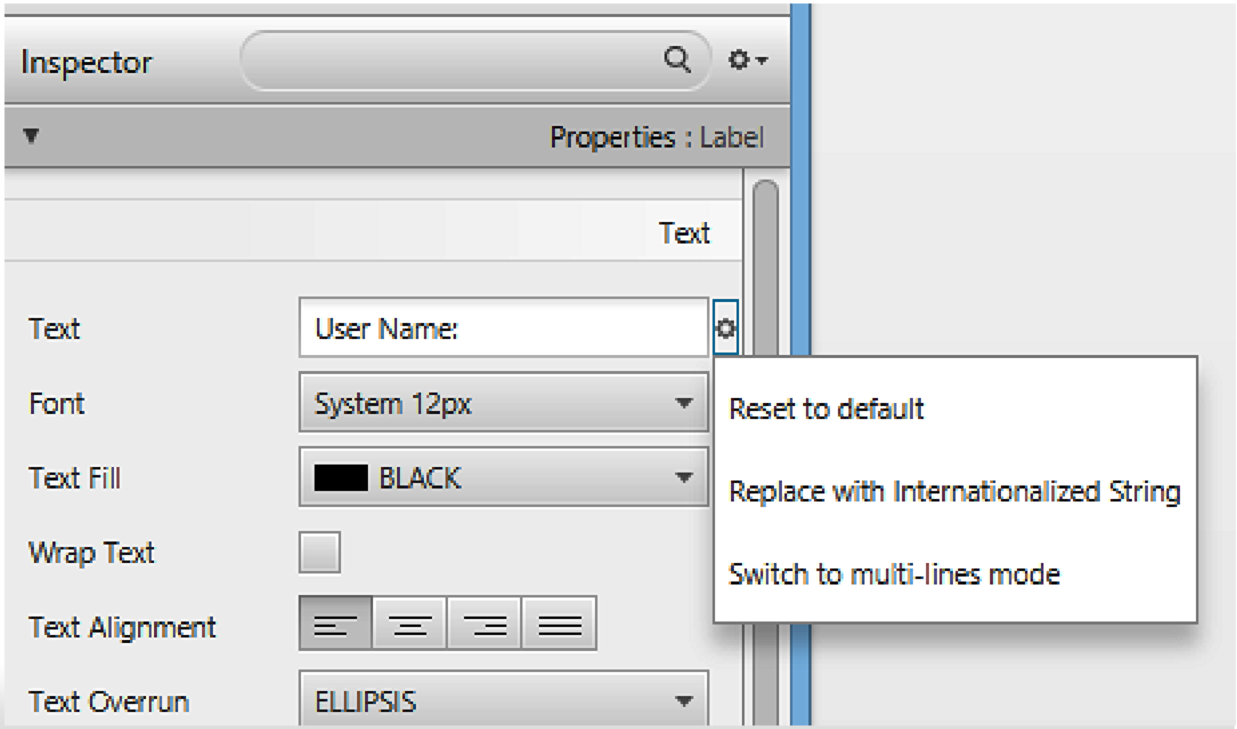
Graphic Text Gap	4
Content Display	LEFT

Node

Alignment	CENTER_LEFT

ResourceBundle and FXML

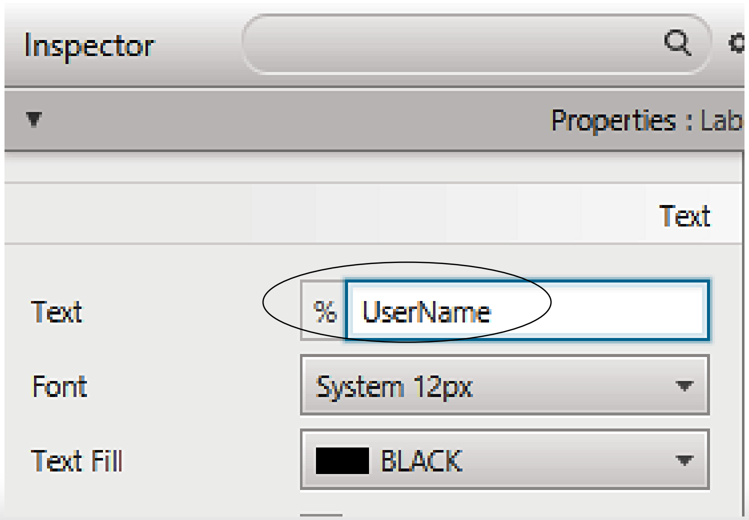
- Pointing next to theText input box reveals a gear icon
- Click on it and you will see


Inspector
Properties : Label
Text
Reset to default
Replace with Internationalized String
Switch to multi-lines mode

Text	User Name:
Font	System 12px
Text Fill	■ BLACK
Wrap Text	☐
Text Alignment	
Text Overrun	ELLIPSIS

ResourceBundle and FXML

- Select *Replace with Internationalized String*
- Replace the text in the input box with the key for the required text from the MessagesBundle.
- There is now a percent symbol (%) at the front of the input box
- This means that this is a key and not the actual text



Text % UserName
Font System 12px
Text Fill BLACK

ResourceBundle and FXML

- In the `start()` method you change this line:

```
Parent root = FXMLLoader.load(getClass().
            getResource("/fxml/Login.fxml"));
```

- To this line

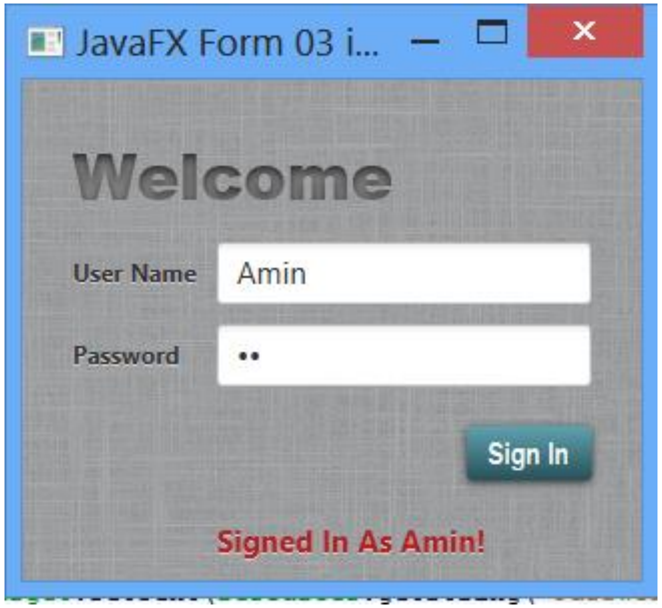
```
Parent root = FXMLLoader.load(getClass().
            getResource("/fxml/Login.fxml"),
            ResourceBundle.getBundle("MessagesBundle"));
```

- In the controller class you add the following class variable

```
@FXML
private ResourceBundle resources;
```

- Your fxml application is now internationalized

Internationalization

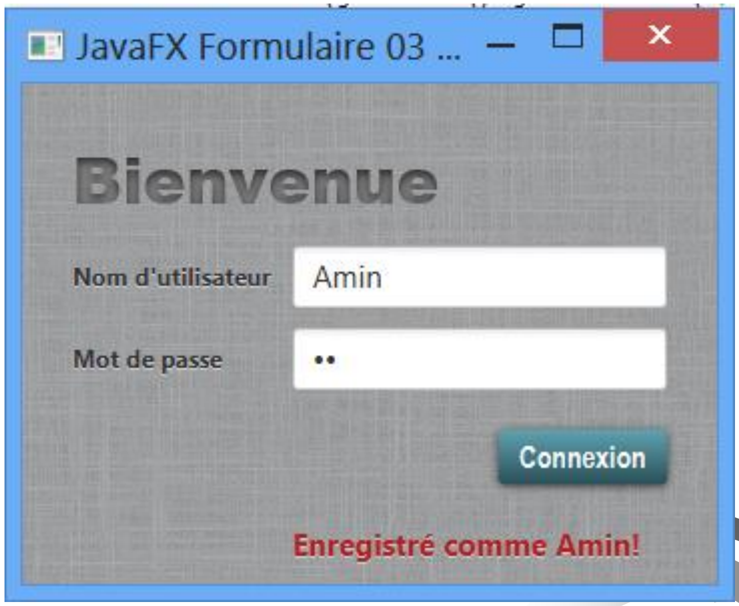

JavaFX Form 03 i...

Welcome

User Name: Amin

Password: ••

Signed In As Amin!


JavaFX Formulaire 03 ...

Bienvenue

Nom d'utilisateur: Amin

Mot de passe: ••

Enregistré comme Amin!

Editing an existing FXML

☞ Original fxml

```
<Text id="welcome-text" text="Welcome"
     GridPane.columnIndex="0" GridPane.columnSpan="2"
     GridPane.rowIndex="0" />
<Label text="User Name:" GridPane.columnIndex="0"
     GridPane.rowIndex="1" />
```

☞ Manual update by entering the % and the key name

```
<Text id="welcome-text" text="%Welcome"
     GridPane.columnIndex="0" GridPane.columnSpan="2"
     GridPane.rowIndex="0" />
<Label text= "%UserName" GridPane.columnIndex="0"
     GridPane.rowIndex="1" />
```

Exercise 26

- Make sure to use Scene Builder and Java FXML to create the following programs.
- Make sure to internationalize your code to at least support French and English. You can add another language of your choice.