CS242
Advanced Programming
Concepts in Java
JavaFX
9-13-2013
Outline

- Graphical User Interfaces (GUIs)
  - JOptionPane
  - JavaFX
    - Content, Scene & Stage
    - Scene Graphs
- Deploying a Java Application

HW#2 posted; due 9/23/2013

Reading Assignment:
- Java Tutorial on GUIs: Creating a JavaFX GUI
JOptionPane

JOptionPane is a subclass of JComponent in the javax.swing package.

JOptionPane makes it easy to pop up a standard dialog box that prompts users for a value or informs them of something. For information about using JOptionPane, see How to Make Dialogs, a section in The Java Tutorial.

example: DialogTest
import javax.swing.JOptionPane;

public static void main(String[] args) {
    String name = JOptionPane.showInputDialog("What is your name?");
    String input = JOptionPane.showInputDialog("How old are you?");
    int age = Integer.parseInt(input);
    System.out.println("Hello, " + name + ". Next year, you'll be " + (age + 1));
}

Can something go wrong?
What if the user types “twenty”?
String name = JOptionPane.showMessageDialog("What is your name?");
String input = JOptionPane.showMessageDialog("How old are you?");

try {
    int age = Integer.parseInt(input); // possible error
    System.out.println("Hello, " + name + ". Next year, you'll be " + (age + 1));
} catch (NumberFormatException exc) {
    System.out.println("Input a valid integer for age");
}
boolean invalidAge = true;
while (invalidAge) {
    try {
        String input = JOptionPane.showInputDialog("How old are you?");
        int age = Integer.parseInt(input);
        invalidAge = false;
        System.out.println("Hello, "+name + ". Next year, you'll be "+(age + 1));
    } catch (NumberFormatException exc) {
        System.err.println("Try again, but this time input a valid integer for your age.");
    }
}
JavaFX in eclipse: e(fx)clipse

- website for installing JavaFX in eclipse:
  - www.eclipse.org/efxclipse/index.html
  - click on the Install tab

- Be sure you have the latest Eclipse IDE for Java EE Developers:
  - download from (Eclipse Kepler)

- Then follow the instructions on the webpage (it takes some time)

- There are some excellent tutorials on this webpage (click on the Tutorials tab)
Exercise: complete by 9/16 (Monday)

- Set up JavaFX for either NetBeans or Eclipse (your choice)
- If using NetBeans, view the video Building your first JavaFX application (this uses NetBeans 7.1, but is still very useful)
- If using Eclipse, follow the tutorials on the e(fx)clipse webpage
- Customize the first application in the above tutorial in some way
- Do the “Getting Started with JavaFX” tutorial on the Oracle webpage
The main class for a JavaFX application extends javafx.application.Application class. The start() method is the main entry point for all JavaFX applications.

A JavaFX application defines the user interface container by means of a stage and a scene.

- The JavaFX Stage class is the top-level JavaFX container (the window)
- The JavaFX Scene class is the container for all content
public class HelloJavaFXWorld extends Application {

@Override
public void start(Stage primaryStage) {

the stage is the top-level "container"

subclass of Application

override start()
public void start(Stage primaryStage) {

    Button btn = new Button(); // create a button
    btn.setText("Say 'Hello World'");
    btn.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            System.out.println("Hello World!");
        }
    });
}

creates a button and adds an action to be performed when the button is clicked
The JavaFX Scene class is the container for all content.

```java
// continue HelloJavaFXWorld class
StackPane root = new StackPane();
root.getChildren().add(btn);
```

- adds the button to the root of the content pane

```java
Scene scene = new Scene(root, 300, 250);
```

- creates a scene and adds the root of the content hierarchy to the scene

```java
primaryStage.setTitle("Hello World!");
primaryStage.setScene(scene);
primaryStage.show();
```
HelloJavaFXWorld: Scene

- The JavaFX Scene class is the container for all content.

```java
// continue HelloJavaFXWorld class
StackPane root = new StackPane();
root.getChildren().add(btn);

Scene scene = new Scene(root, 300, 250);
primaryStage.setTitle("Hello World!");
primaryStage.setScene(scene);
primaryStage.show();
```
In JavaFX, the content of the scene is represented as a hierarchical scene graph of nodes. In this example, the root node is a StackPane object, which is a resizable layout node. This means that the root node's size tracks the scene's size and changes when the stage is resized by a user.

The root node contains one child node, a button control with text, plus an event handler to print a message when the button is pressed.
Content: Scene graph

Figure 1–1 Hello World Scene Graph
main()

The main() method is not required for JavaFX applications when the JAR file for the application is created with the JavaFX Packager tool, which embeds the JavaFX Launcher in the JAR file. However, it is useful to include the main() method so you can run JAR files that were created without the JavaFX Launcher, such as when using an IDE in which the JavaFX tools are not fully integrated. Also, Swing applications that embed JavaFX code require the main() method.
Graphical User Interface (GUI)

To create some GUI components and display them in a window:

1. The stage is the argument to the start() method

2. Create content:
   a) Create a Pane as the root of the scene graph
   b) Create some components and add them to the root pane

3. Create a scene and add the content

4. Add the scene to the stage

5. Make the stage visible
Another JavaFX example

```java
import javafx.application.Application;
import javafx.scene.image.ImageView;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class NaoRobot extends Application {
    private ImageView nao1;

    @Override
    public void start(Stage primaryStage) {
        try {
            primaryStage.setTitle("Hello, Nao");
            nao1 = new ImageView(
                NaoRobot.class.getResourceAsStream("/images/nao.jpg"));
        }
    }
}
```
Another JavaFX example

// public void start(Stage primaryStage) continued
Pane root = new Pane();
root.getChildren().add(nao1);
primaryStage.setScene(new Scene(root, 600, 500));
primaryStage.show();
}
catch(Exception e) {
    e.printStackTrace();
}
}
Deploying an application

http://wiki.eclipse.org/Efxclipse/Tutorials/Tutorial1

1. Open the build.fxbuild file and fill in the following fields: Vendor name and Application version. Then, select the "Browse..." button next to the "Application class" entry.
Deploying an application

- Click on the "ant build.xml and run" link in the build section of the editor.
Deploying an application

- Expand the folders in the "Package Explorer" and double click your jar file.