CS242
Advanced Programming Concepts in Java
Exception Handling
Introduction to JavaFX
9–11–2013
Outline

- Summary of Exception Handling
- Graphical User Interfaces (GUIs)
  - Introduction to JavaFX

**HW#2** posted; due 9/23/2013

Reading Assignment:
- Java Tutorial on GUIs: [Creating a JavaFX GUI](#)
First consider a classic D6
Recap: Exceptions

- Item 40: Use checked exceptions for recoverable conditions and run-time exceptions for programming errors

- Item 42: Favor the use of standard exceptions
Exception handling is not supposed to replace a simple test.

```
try {
    s.pop();
} catch(EmptyStackException es) {...}
```

```
if (!s.empty()) s.pop(); // is much better
```
Things to remember

- Item 39: Use exceptions only for exceptional conditions

A well-designed API must not force its client to use exceptions for ordinary control flow.
Things to remember

- Do not micromanage exceptions.
  
  ```java
  try { s1 } catch(Exception e) {...};
  try { s2 } catch(Exception e) {...};
  etc.
  ```

- Item 43: Throw exceptions appropriate to the abstraction.
Things to remember

- Do not squelch exceptions.

```java
Image loadImage(String s) {
    try {
        // lots of stuff
    } catch(Exception e) {} // so there!
}
```

Item 47: Don’t ignore exceptions.
Graphical User Interfaces: JavaFX
What is JavaFX?

- JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms.
  - visual effects, animations, sound
  - mobile phones, consumer, television, desktop
- scene-graph-based programming model
- The JavaFX SDK is included in JDK 7
Java Technology

Java Language
- java
- javac
- javadoc
- jar
- javap
- JPDA
- JConsole
- Java VisualVM
- Java DB
- Security
- Int'l
- RMI
- IDL
- Deploy
- Monitoring
- Troubleshoot
- Scripting
- JVM TI
- Web Services

Deployment
- Java Web Start
- Applet / Java Plug-in

User Interface
- Swing
- Java 2D
- AWT
- Accessibility
- Drag and Drop
- Input Methods
- Image I/O
- Print Service
- Sound
- IDL
- JDBC
- JNDI
- RMI
- RMI-IIOP
- Scripting
- Beans
- Int'l Support
- Input/Output
- JMX
- JNI
- Math
- Networking
- Override Mechanism
- Security
- Serialization
- Extension Mechanism
- XML JAXP
- lang and util
- Collections
- Concurrency Utilities
- JAR
- Base Libraries
- Management
- Preferences API
- Ref Objects
- Reflection
- Regular Expressions
- Versioning
- Zip
- Instrumentation

Java Virtual Machine
- Java HotSpot Client and Server VM
JavaFX resources

- Overview of JavaFX
- The Java Tutorial on GUIs: Creating Java Graphical User Interfaces:
  - Creating a JavaFX GUI
- JavaFX Demos and Samples
  - Scroll to bottom, then download the zip file
- Javadoc for JavaFX
- Eclipse & Netbeans have good tutorials on using JavaFX
Getting Started with JavaFX

1. Get set up
2. Create the Framework
3. Define the UI
4. Lay out the UI controls
5. Deploy
1. Get set up

- The first step in getting started with JavaFX is to download and install the Java SE 7 JDK, which includes the JavaFX runtime libraries and utilities. See the [JDK 7 and JRE 7 Installation Guide](#) for instructions.

- JavaFX Scene Builder provides a simple way to build the UI for your application and connect it to the application logic. To install this tool, see [Installing JavaFX Scene Builder](#).
Each JavaFX application has a similar framework. The main class for the application extends the Application class. The start() method sets up and shows the stage, which contains the UI for the application. See Hello World, JavaFX Style for information on setting up a NetBeans project and creating a simple application.
3. Define the UI

- The stage for the application interface contains one or more scenes, which in turn contain the controls, shapes, and images that make up your user interface. See Using JavaFX UI Controls for information on defining controls such as buttons, labels, and text fields for your UI.

- The JavaFX Scene Builder's Library panel provides the same UI controls that are available in the SDK. See Library Panel for more information.
After you decide what controls you want in your UI, use one of the built-in layout panes to manage the size and position of the controls. See Using Built-in Layout Panes for information on setting up your UI controls to create the screens for your UI.

An easier way to lay out your UI is available with the Library panel in JavaFX Scene Builder. You can also manage layouts in JavaFX Scene Builder using the Inspector panel. See Inspector Panel for information.
JavaFX applications run on a desktop, in a browser, and by using Web Start. When your application is ready, create the files that your users will run. See Getting Started in Deploying JavaFX Applications for information on packaging and deploying your application.
Example: Hello, JavaFX World

1. Set up for NetBeans (Eclipse is similar)

<table>
<thead>
<tr>
<th>Software or Resource</th>
<th>Version Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NetBeans IDE</strong></td>
<td>7.3</td>
</tr>
<tr>
<td><strong>Java Development Kit (JDK)</strong></td>
<td>7 update 10 or more recent</td>
</tr>
<tr>
<td><strong>JavaFX SDK (cobundled with JDK)</strong></td>
<td>2.2.4 or more recent</td>
</tr>
</tbody>
</table>

- **Download** NetBeans IDE bundled with JavaEE and install
Example: Hello, JavaFX World

2. Create a JavaFX project in NetBeans
   - File | New | Project & select JavaFX, next
   - Give the project a name, and select finish

Demonstrated in class
package hellojavafxworld;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
/** The main() method is ignored in correctly deployed JavaFX application. main() serves only as fallback in case the application can not be launched through deployment artifacts, e.g., in IDEs with limited FX support. NetBeans ignores main().
@param args the command line arguments */

    public static void main(String[] args) {
        launch(args);
    }
}
public class HelloJavaFXWorld extends Application {
    @Override
    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!");
            }
        });
    }
    // continued on next slide
// continue HelloJavaFXWorld class

StackPane root = new StackPane();
root.getChildren().add(btn); // add the button

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

primaryStage.setTitle("Hello World!");
primaryStage.setScene(scene);
primaryStage.show();
}
Exercise: complete by 9/16 (Monday)

- Setup 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)

- Customize the default application in some way