Lecture 05A

Maintainable GUI Development (2/2)

JavaFX(ML) Controls and Styling

Horia A. Maior and Marjahan Begum

COMP2013

Developing Maintainable Software

Lecture 08A

Maintainable GUI Development (2/2)

JavaFX(ML) Controls and Styling

Horia A. Maior and Marjahan Begum

University of Nottingham
UK | CHINA | MALAYSIA

Topics for this Week

- Lecture 05A
  - Build Tools: Maven and Gradle (Not delivered last week)
  - JavaFX(ML) controls and styling

- Lab 05:
  - More complex GUI development

- Lecture 05B:
  - Coursework Q/A and support
  - Labsheet 5

Coursework Support

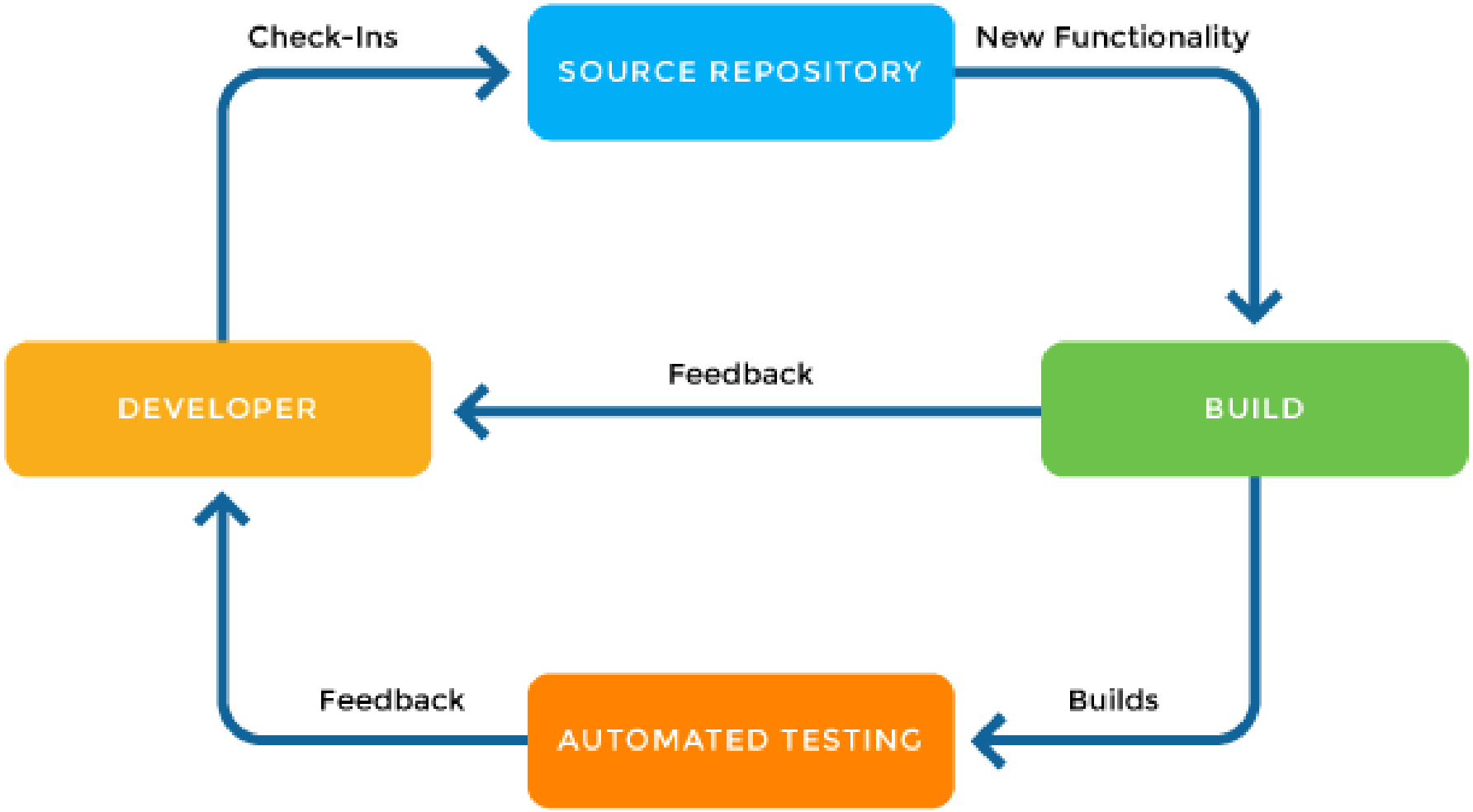
https://forms.office.com/e/grwgXdn0YF

COMP2013 Coursework Questions

University of Nottingham
UK | CHINA | MALAYSIA

Build Tools

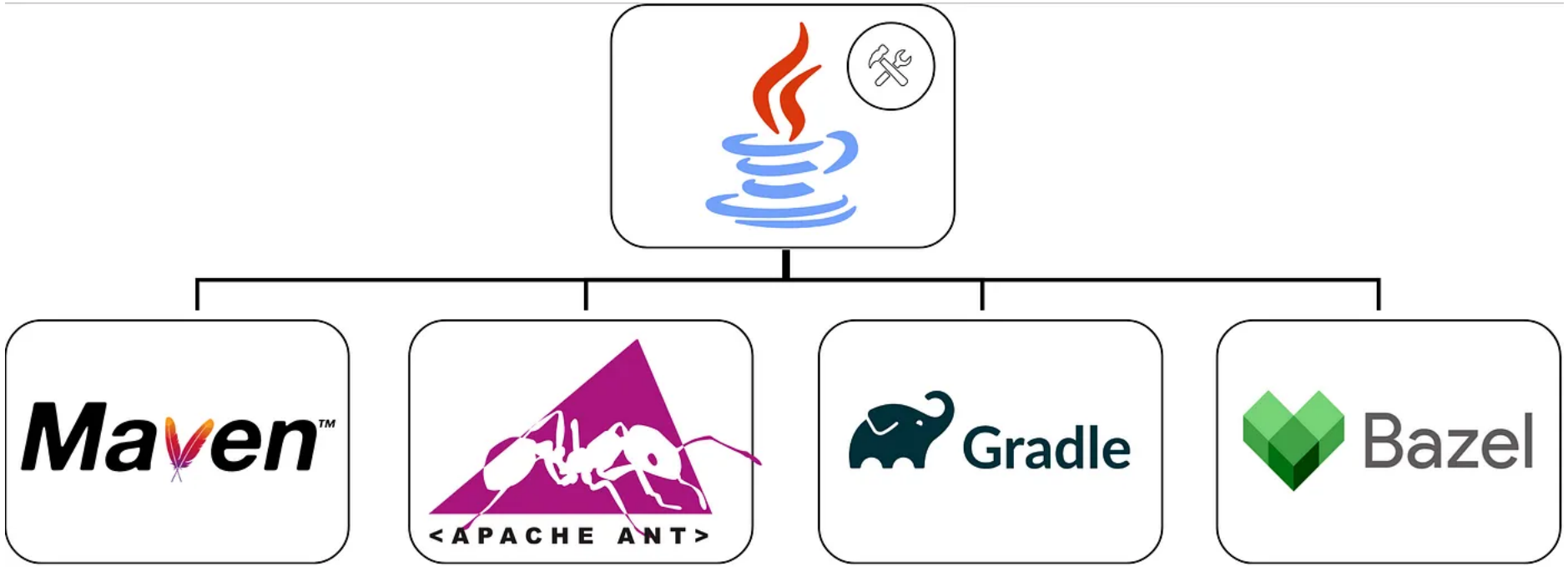
Question:
What are
Build Tools in
Programming
Languages?



https://www.plutora.com/ci-cd-tools/software-build-tools

Benefits of Build tools

- Project Compilation
- Reduces errors from manually running steps,
- Increases the consistency of the process.
- Dependencies
- Allow for better maintenance (keep a build log)
- Automated testing
- Deployment
- Supports documentation
- Repository management
- …

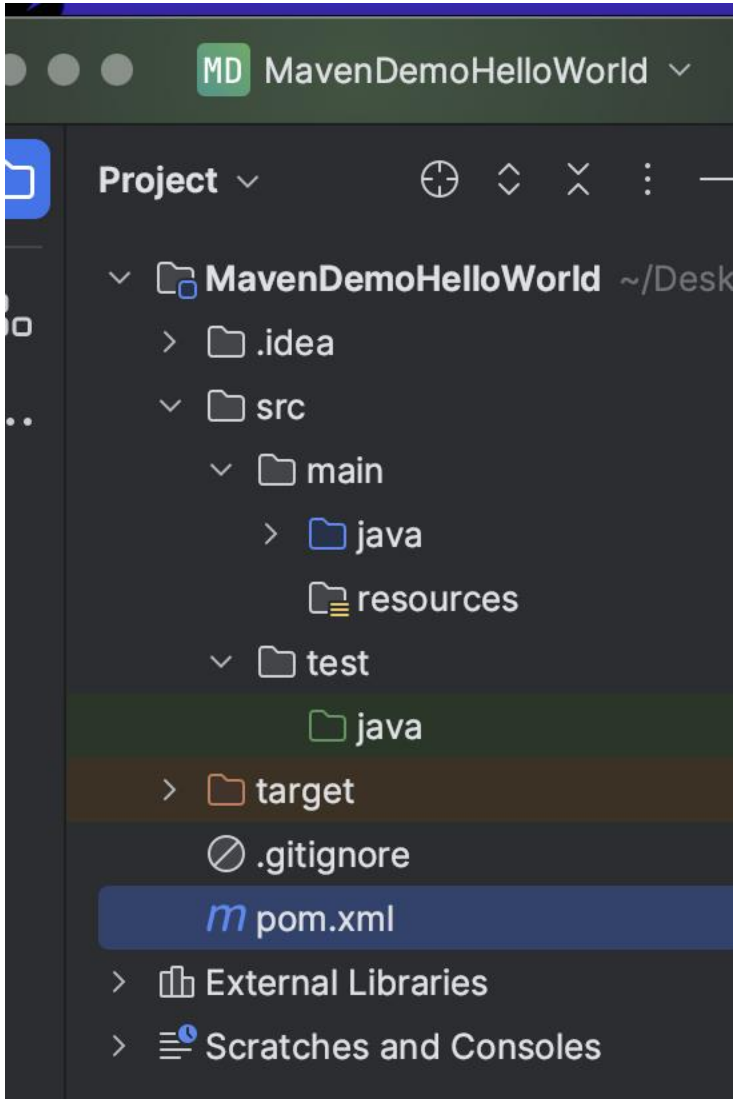

Maven™
<APACHE ANT>
Gradle
Bazel

What is Maven?

- A build tool

What is Maven?

- A build tool
- Maven builds are structural

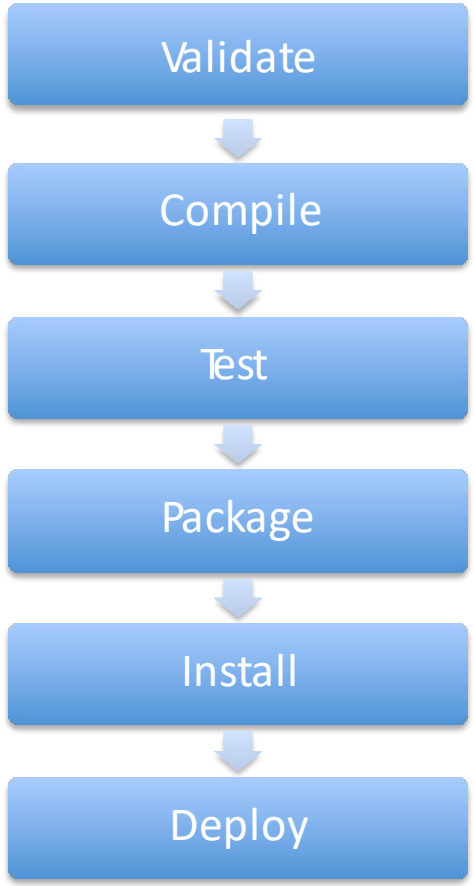


What is Maven?

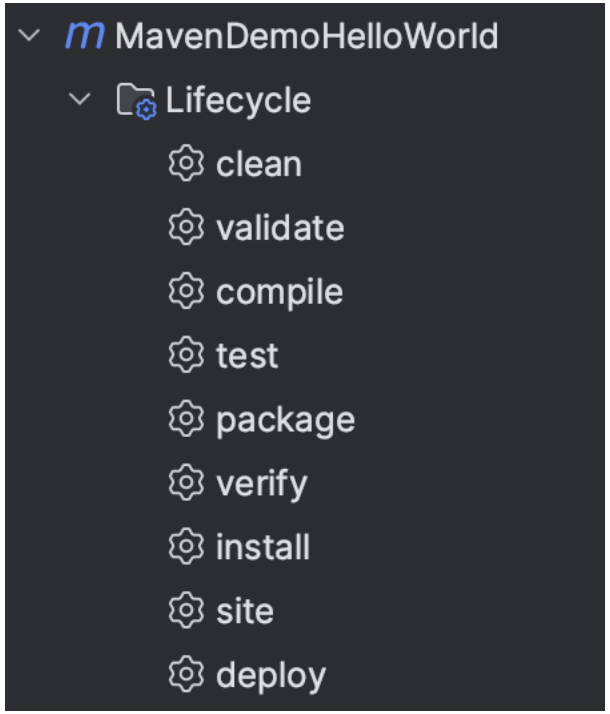
- A build tool
- Maven builds are structural
- Manages dependencies
  - You describe your dependencies, Maven downloads and includes all you need in your build path
- Manages repositories
  - Your projects share a common set of artifacts/jar files
  - No need to check in JAR files to version control

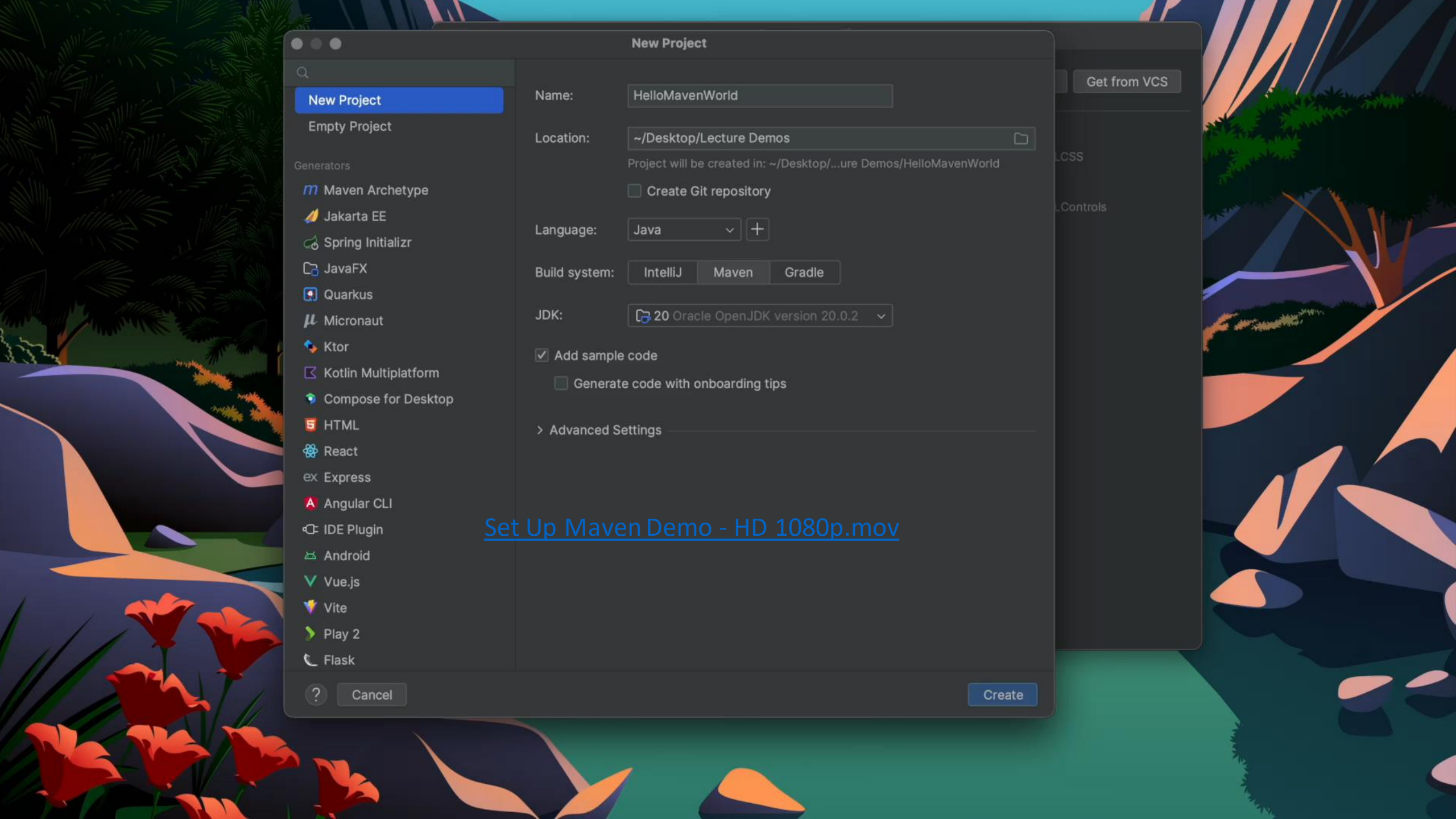
Maven Project Management

- Maven builds follows a lifecycle, using phases
  - Each lifecycle phase includes the ones above it
    - (Package starts with compile, then test, then package)
    - Plugins can customize or hook- in to the phases
  - Key Maven lifecycles
- **Default** – your normal lifecycle
- **Clean** – issued during the mvn cleanup command
- **Site** – issued during the mvn site command



The Default Lifecycle




New Project
New Project
Empty Project
Generators
Maven Archetype
Jakarta EE
Spring Initializr
JavaFX
Quarkus
Micronaut
Ktor
Kotlin Multiplatform
Compose for Desktop
HTML
React
Express
Angular CLI
IDE Plugin
Android
Vue.js
Vite
Play 2
Flask
Name:
HelloMavenWorld
Location:
~/Desktop/Lecture Demos
Project will be created in: ~/Desktop/...ure Demos/HelloMavenWorld
Create Git repository
Language:
Java
Build system:
IntelliJ
Maven
Gradle
JDK:
20 Oracle OpenJDK version 20.0.2
Add sample code
Generate code with onboarding tips
Advanced Settings
Cancel
Create
Get from VCS
Set Up Maven Demo - HD 1080p.mov

What is a POM file?

- Project Object Model (POM)
- Contains configuration information about your project
- Key items
  - Project Identification
  - Dependencies
  - Plug-Ins
  - Other Settings

pom.xml (MavenDemoHelloWorld) × Main.java

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apach
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.COMP2013</groupId>
    <artifactId>MavenDemoHelloWorld</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>20</maven.compiler.source>
        <maven.compiler.target>20</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

pom.xml (MavenDemoHelloWorld) × Main.java

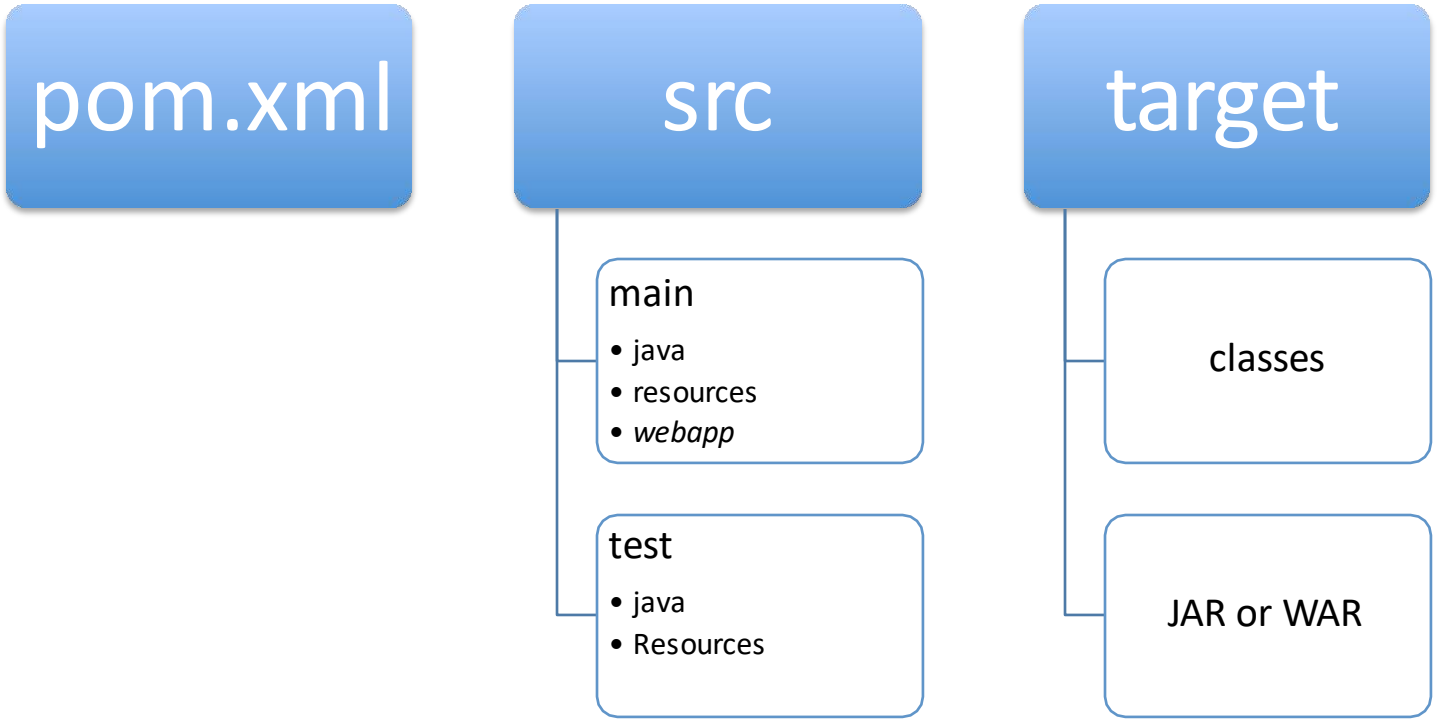
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apach
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.COMP2013</groupId>
    <artifactId>MavenDemoHelloWorld</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>20</maven.compiler.source>
        <maven.compiler.target>20</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
```

```
<artifactId>mavenDemoHelloWorld</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
    <maven.compiler.source>20</maven.compiler.source>
    <maven.compiler.target>20</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>

</project>
```

What happens when you Set Up a Maven Project?

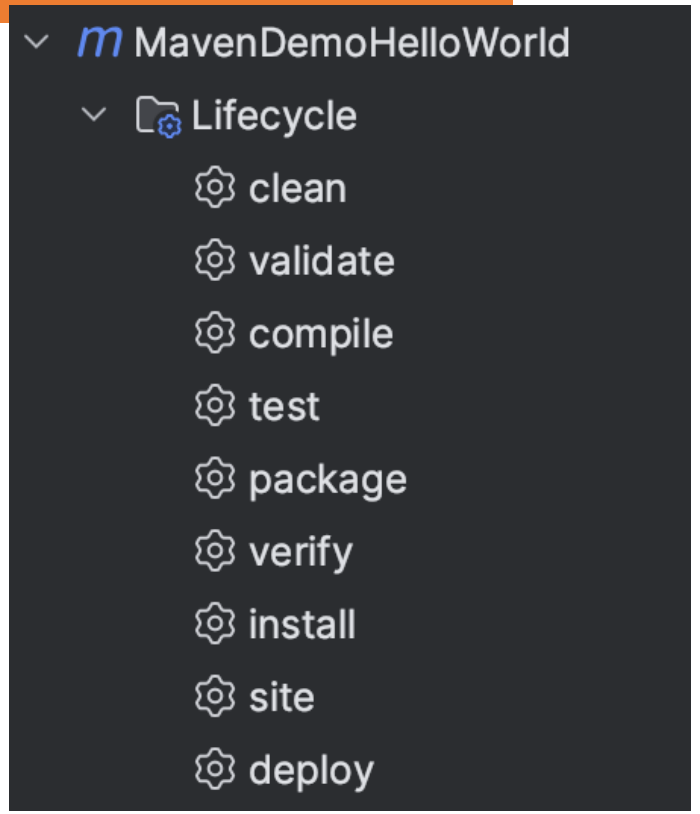
- Maven builds a project skeleton
  - Maven used its' conventions to place files in the right places
- Let's look at the directory structure…

Directory Structure, Basic Maven Project



Using Maven Commands

- The maven command (mvn)
  - mvn clean – removes files in target
  - mvn compile – compiles a project
  - mvn test – Runs all tests in the src/test directory
  - mvn package – builds the final target artifact (JAR, WAR)
  - mvn install – Installs the target artifact in your local repository
  - mvn deploy – Deploys the artifact (if configured)



Helpful Maven Reports

- **mvn site** -- Generates a web site report in target/site/index.html
- **mvn pmd:pmd** – runs a PMD (programming mistake detector) report, a source code analyzer for finding common flaws, output in target/site/pmd.html
- **mvn cobutura:cobertura** – runs a cobertura code test coverage report, the percentage of branches/lines accessed by unit tests, output in target/site/cobertura/index.html
- **mvn jdepend:generate** – runs a report on coupling between packages, output in target/site/jdepend-report.html
- **mvn checkstyle:checkstyle** – runs a checkstyle report
- **mvn javancss:javancss-report** – Runs a JavaNCSS (non commenting source statements) report to show method complexity, roughly equivalent to counting ";" and '{' characters in Java source file
- Note: all of these and more can be configured in the <site> section of the maven build… YMMV.

Managing Dependencies

- Key fact: Maven projects have ONE Artifact (JAR, WAR)
  - Can build projects that depend on other projects
  - Can build parent projects that build child projects
  - Can depend on other open source libraries and frameworks
- Manage all of this via the <dependencies> tags

- Adds dependency to jUnit
  - Downloads 3.8.1 of Junit and stores in your maven repository

```
<dependencies>
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
 </dependency>
</dependencies>
```

Dependency Tips…

- Look for public dependencies on www.mvnrepository.com
- Split out re-usable functionality into JAR projects
  - Create top-level projects with a “pom” target which coordinate the build of subordinate projects
  - Manage your own shared projects using a company repository (Archiva, others)

What we didn’t cover…

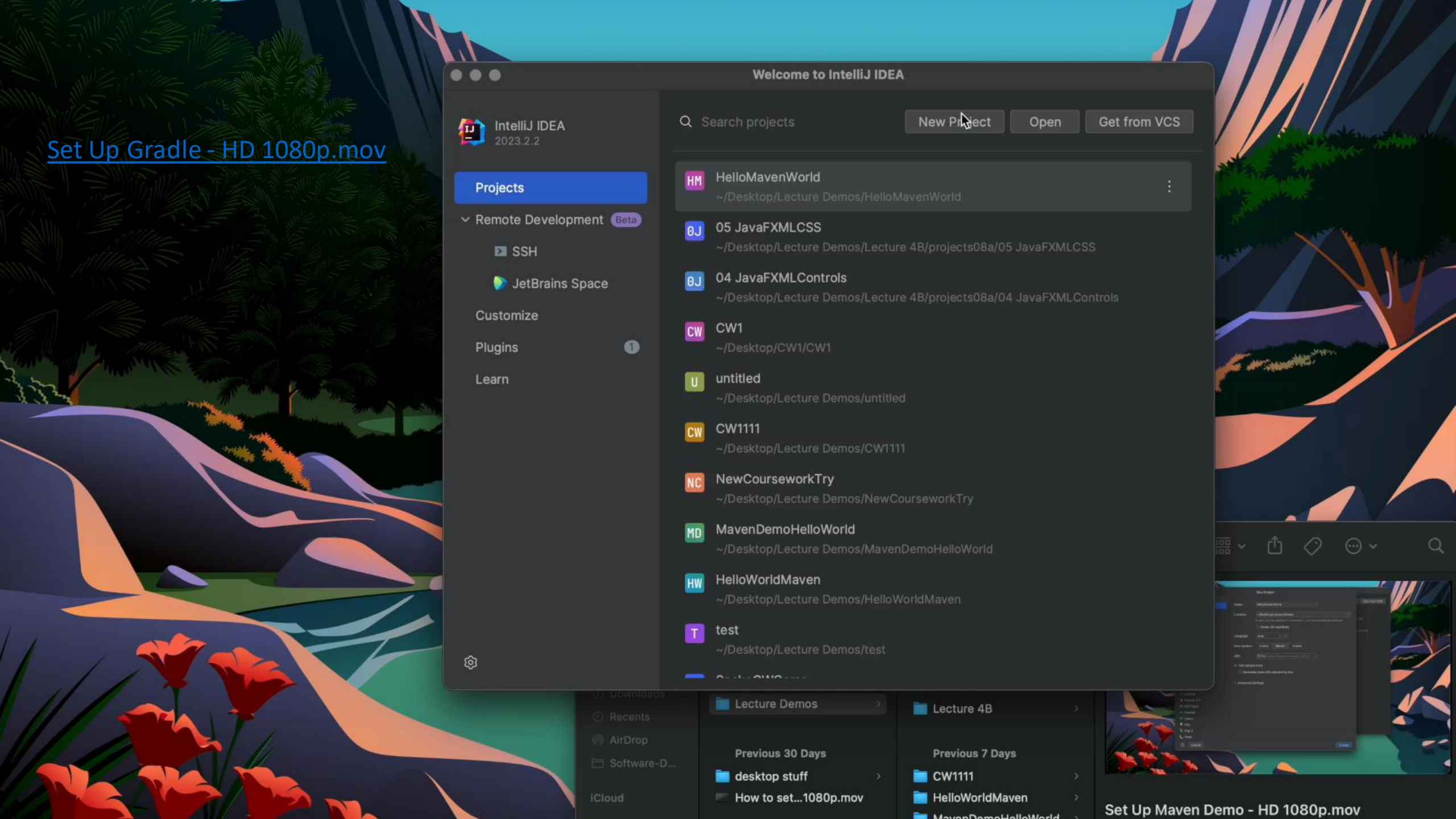
Resources

- Maven web site: http://maven.apache.org

Gradle

Gradle: A Contemporary Approach

- Replaces XML in favour of a domain specific language ‘groovy’
- Based on a programming language therefore can implement control flow
- Although it uses groovy itself, it can act as a build manager for any language
- Declarative: plug-ins add functionality
- More cleanly accomplishes required tasks of a typical development project, from compilation through testing and deployment.
- It’s the official build system for Android


Set Up Gradle - HD 1080p.mov
Welcome to IntelliJ IDEA
IntelliJ IDEA
2023.2.2
Search projects
New Project
Open
Get from VCS
Projects
Remote Development
Beta
SSH
JetBrains Space
Customize
Plugins
1
Learn
HelloMavenWorld
~/Desktop/Lecture Demos/HelloMavenWorld
05 JavaFXMLCSS
~/Desktop/Lecture Demos/Lecture 4B/projects08a/05 JavaFXMLCSS
04 JavaFXMLControls
~/Desktop/Lecture Demos/Lecture 4B/projects08a/04 JavaFXMLControls
CW1
~/Desktop/CW1/CW1
untitled
~/Desktop/Lecture Demos/untitled
CW1111
~/Desktop/Lecture Demos/CW1111
NewCourseworkTry
~/Desktop/Lecture Demos/NewCourseworkTry
MavenDemoHelloWorld
~/Desktop/Lecture Demos/MavenDemoHelloWorld
HelloWorldMaven
~/Desktop/Lecture Demos/HelloWorldMaven
test
~/Desktop/Lecture Demos/test
Recents
AirDrop
Software-D...
iCloud
Lecture Demos
Lecture 4B
Previous 30 Days
desktop stuff
How to set...1080p.mov
Previous 7 Days
CW1111
HelloWorldMaven
Set Up Maven Demo - HD 1080p.mov

Grade Build Cycle

- It launches as a new JVM process
  - It parses the *gradle.properties* file and configures Gradle accordingly
- Next, it creates a Settings instance for the build
- Then, it evaluates the *settings.gradle* file against the Settings object
- It creates a hierarchy of Projects, based on the configured Settings object
- Finally, it executes each *build.gradle* file against its project
  - These files must be precisely named
  - Automatic setup using a **Grade Wrapper**

Build: Tutorial Videos

- Tutorial series on Gradle
  - Designed for the more advanced of you
- Practical tutorials, running time about an hour (plus additional videos)
- Video series:
  - Introduction to Gradle (38 minutes)
  - Gradle Introduction on the Virtual Pair Programmers YT channel
  - https://www.youtube.com/watch?v=mPpncYETnTg&t=1021s

Reading Homework Comparing Build File Tools

- Interesting blog articles for you to read comparing Maven vs Gradle:
  - http://www.adam-bien.com/roller/abien/entry/maven_vs_ant
  - https://www.baeldung.com/ant-maven-gradle

Acknowledgements

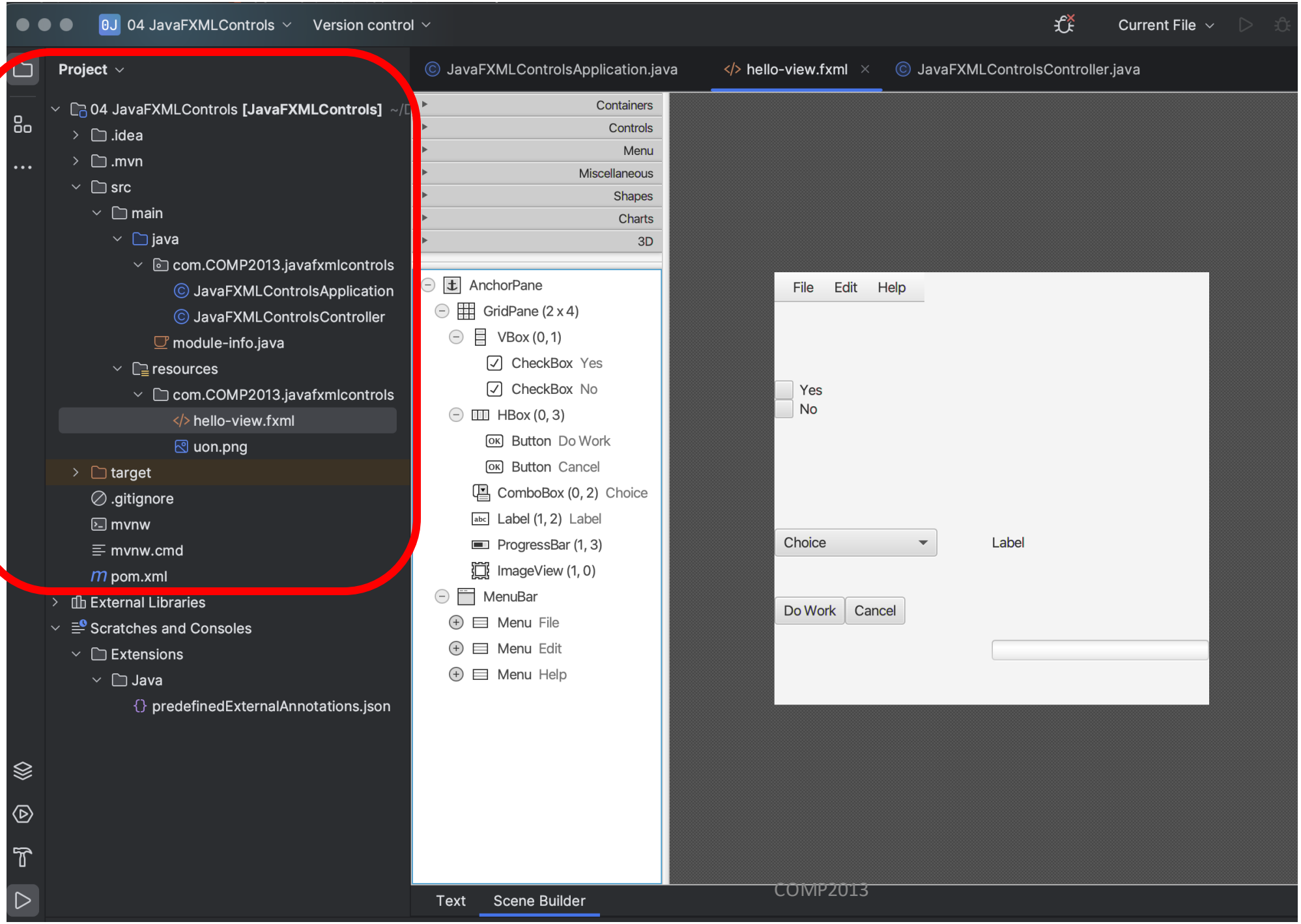
Thanks to:

Robert Laramee, Ken Rimple of Chariot Solutions and Julie Greensmith for the help with matterials for this.

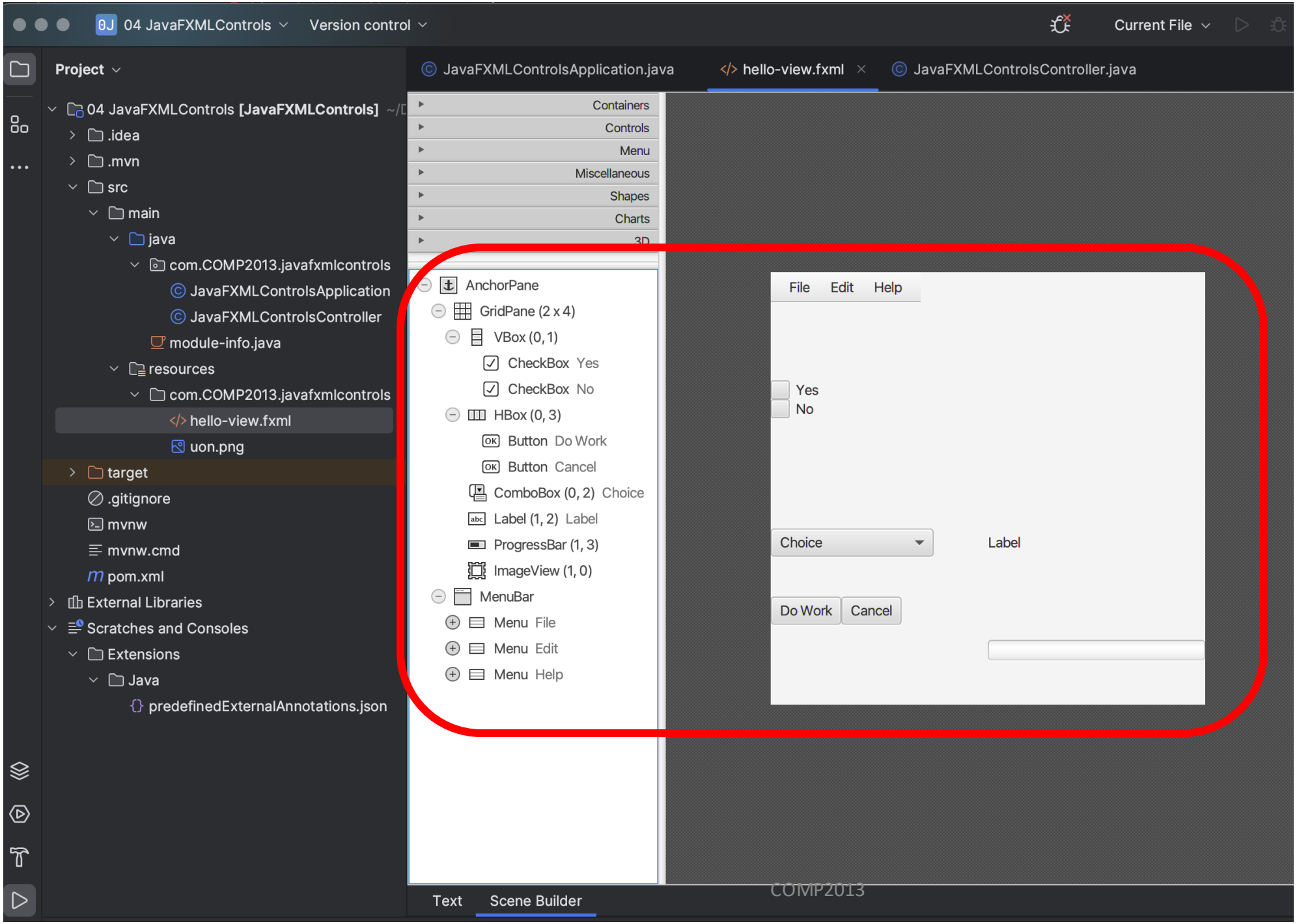
JavaFX advanced topics

FXML

Preparation


04 JavaFXMLControls
Version control
Current File
Project
04 JavaFXMLControls [JavaFXMLControls]
.idea
.mvn
src
main
java
com.COMP2013.javafxmlcontrols
JavaFXMLControlsApplication
JavaFXMLControlsController
module-info.java
resources
com.COMP2013.javafxmlcontrols
hello-view.fxml
uon.png
target
.gitignore
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
Extensions
Java
predefinedExternalAnnotations.json
JavaFXMLControlsApplication.java
hello-view.fxml
JavaFXMLControlsController.java
Containers
Controls
Menu
Miscellaneous
Shapes
Charts
3D
AnchorPane
GridPane (2 x 4)
VBox (0, 1)
CheckBox Yes
CheckBox No
HBox (0, 3)
Button Do Work
Button Cancel
ComboBox (0, 2) Choice
Label (1, 2) Label
ProgressBar (1, 3)
ImageView (1, 0)
MenuBar
Menu File
Menu Edit
Menu Help
File
Edit
Help
Yes
No
Choice
Label
Do Work
Cancel
Text
Scene Builder


DEMO


04 JavaFXMLControls
Version control
Current File
Project
04 JavaFXMLControls [JavaFXMLControls]
.idea
.mvn
src
main
java
com.COMP2013.javafxmlcontrols
JavaFXMLControlsApplication
JavaFXMLControlsController
module-info.java
resources
com.COMP2013.javafxmlcontrols
hello-view.fxml
uon.png
target
.gitignore
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
Extensions
Java
predefinedExternalAnnotations.json
JavaFXMLControlsApplication.java
hello-view.fxml
JavaFXMLControlsController.java
Containers
Controls
Menu
Miscellaneous
Shapes
Charts
3D
AnchorPane
GridPane (2 x 4)
VBox (0, 1)
CheckBox Yes
CheckBox No
HBox (0, 3)
Button Do Work
Button Cancel
ComboBox (0, 2) Choice
Label (1, 2) Label
ProgressBar (1, 3)
ImageView (1, 0)
MenuBar
Menu File
Menu Edit
Menu Help
File
Edit
Help
Yes
No
Choice
Label
Do Work
Cancel
Text
Scene Builder


DEMO

Preparation

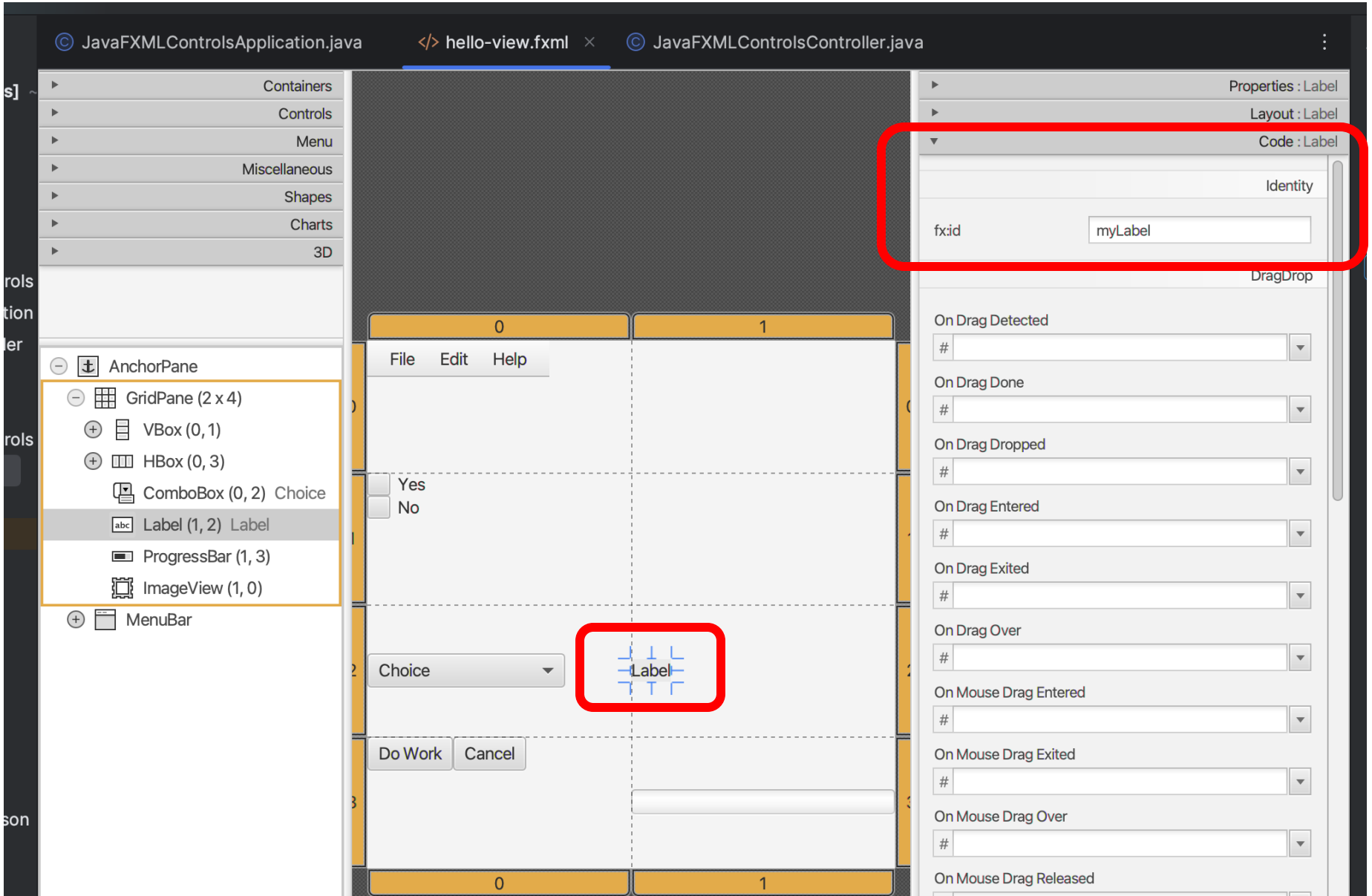
- The result

FXML

Controls

A selection of controls ... in more detail

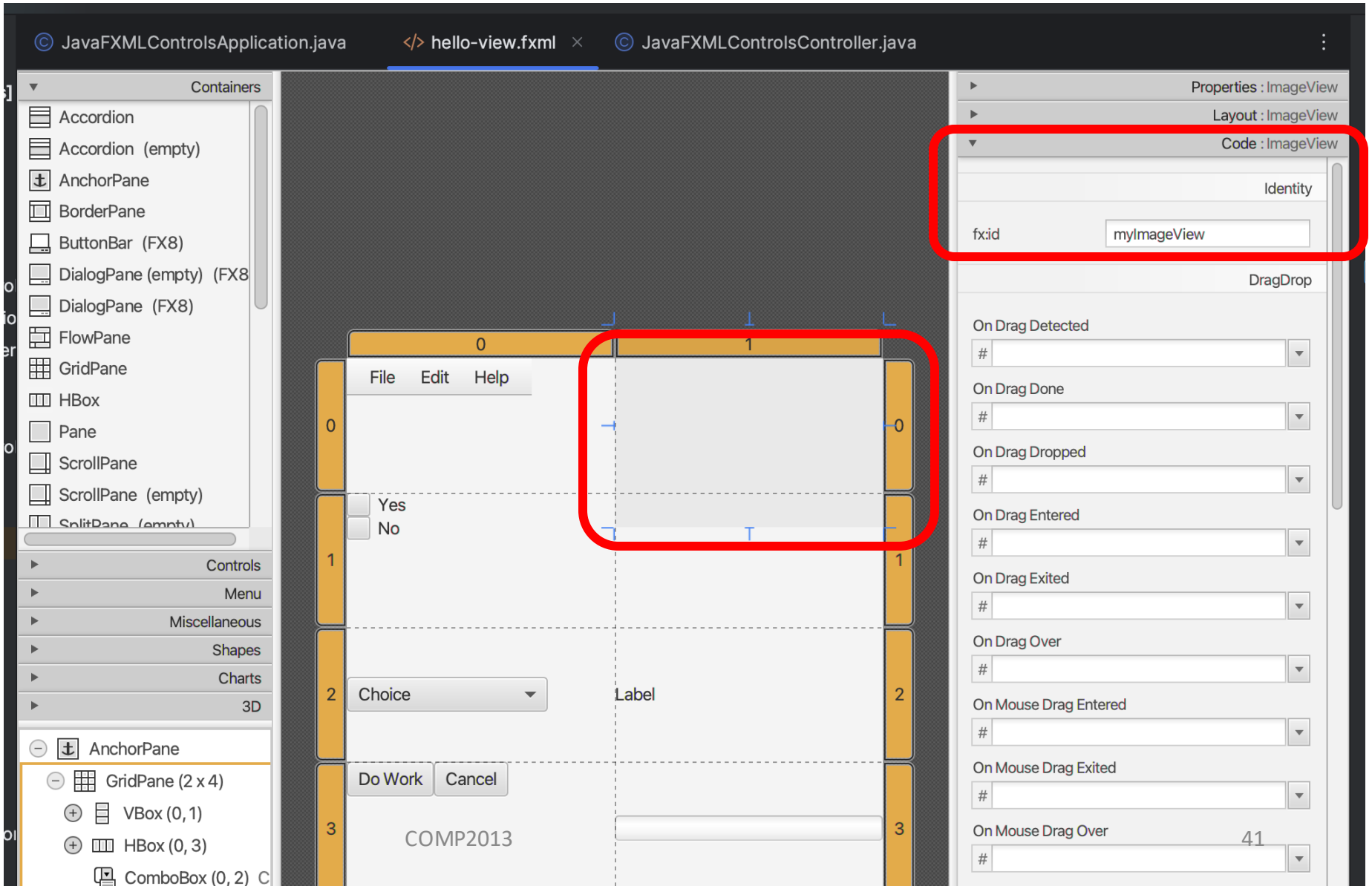
- My selection:
  - Label



University of Nottingham
UK | CHINA | MALAYSIA

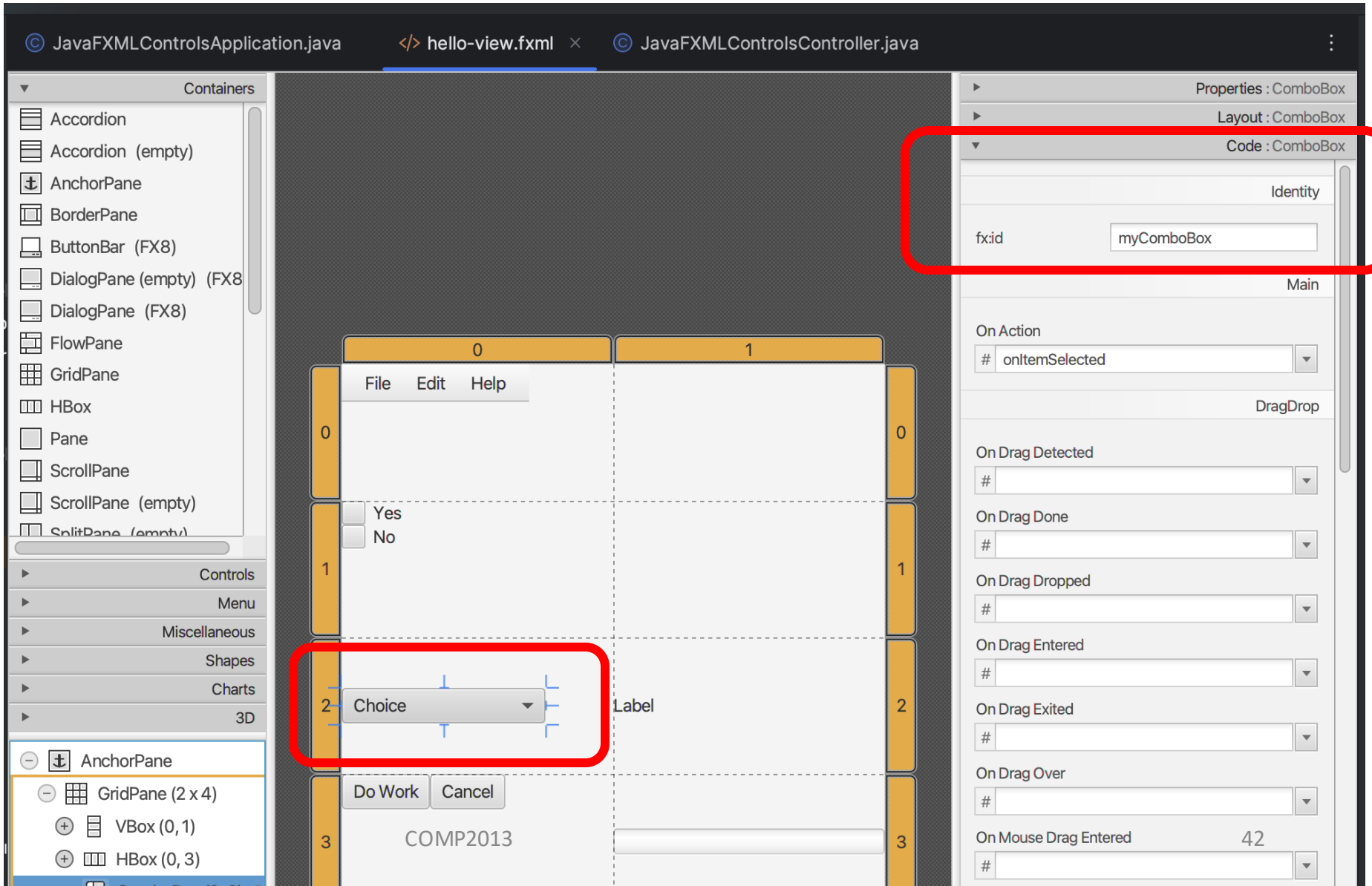
A selection of controls ... in more detail

- My selection:
  - Label
  - ImageView



A selection of controls ... in more detail

- My selection:
  - Label
  - ImageView
  - ComboBox



JavaFXMLControlsApplication.java | hello-view.fxml | JavaFXMLControlsController.java

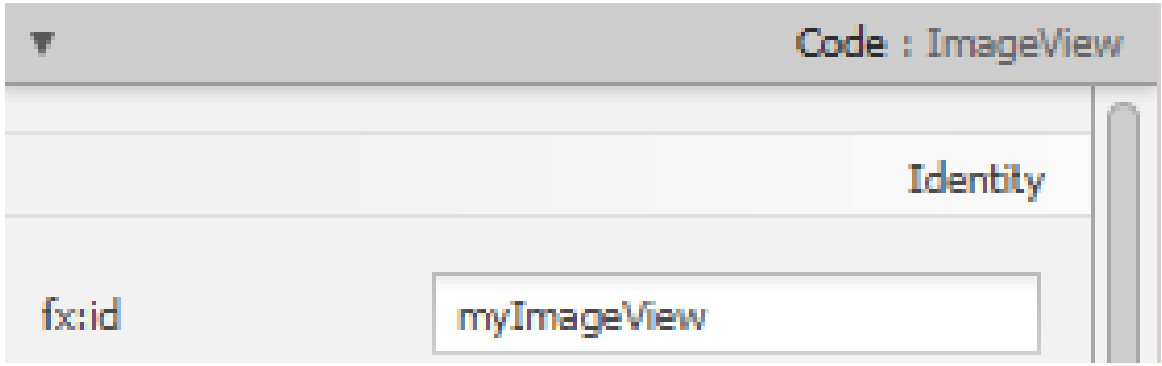
```
1   package com.COMP2013.javafxmlcontrols;
2
3   import ...
9
10  public class JavaFXMLControlsApplication extends Application {
11      @Override
12      public void start(Stage stage) throws IOException {
13          FXMLLoader fxmlLoader = new FXMLLoader(JavaFXMLControlsApplication.class.getResource( name: "hello-view.fxml"));
14          Scene scene = new Scene(fxmlLoader.load());
15          stage.setTitle("Hello!");
16          stage.setScene(scene);
17          stage.show();
18      }
19
20      public static void main(String[] args) { launch(); }
23  }
```

JavaFXMLControlsApplication.java | hello-view.fxml | JavaFXMLControlsController.java

```
3   > import ...
12
13  public class JavaFXMLControlsController {
14
15
16      @FXML private ComboBox<String> myComboBox;
17      @FXML private ImageView myImageView;
18      @FXML private Label myLabel;
19      @FXML private CheckBox myNoCheckBox;
20      @FXML private CheckBox myYesCheckBox;
21
22      @FXML
23      private void initialize(){
24          myYesCheckBox.setSelected(true);
25          ObservableList<String> options= FXCollections.observableArrayList( ...es: "Full","Half","Empty");
26          myLabel.textProperty().bind(myComboBox.getSelectionModel().selectedItemProperty());
27          myComboBox.setItems(options);
28          Image image=new Image(String.valueOf(JavaFXMLControlsApplication.class.getResource( name: "uon.png")));
29          myImageView.setImage(image);
30      }
```

ImageView

- Can be done with an ImageView Control
  - Give it an fx:id
  - Then in your code:



JavaFXMLControlsApplication.java | hello-view.fxml | JavaFXMLControlsController.java

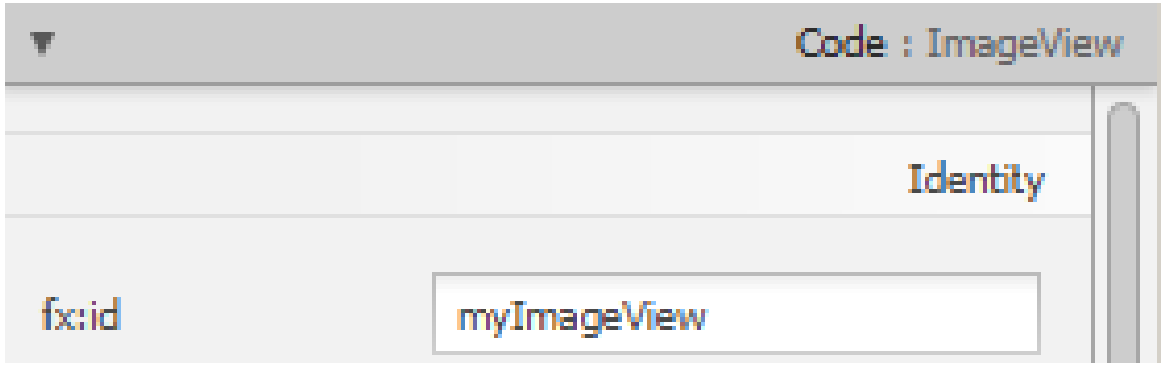
```
3   import ...
12
13  public class JavaFXMLControlsController {
14
15
16      @FXML private ComboBox<String> myComboBox;
17      @FXML private ImageView myImageView;
18      @FXML private Label myLabel;
19      @FXML private CheckBox myNoCheckBox;
20      @FXML private CheckBox myYesCheckBox;
21
22      @FXML
23      private void initialize(){
24          myYesCheckBox.setSelected(true);
25          ObservableList<String> options= FXCollections.observableArrayList( ...es: "Full","Half","Empty");
26          myLabel.textProperty().bind(myComboBox.getSelectionModel().selectedItemProperty());
27          myComboBox.setItems(options);
28          Image image=new Image(String.valueOf(JavaFXMLControlsApplication.class.getResource( name: "uon.png")));
29          myImageView.setImage(image);
30      }
```

JavaFXMLControlsApplication.java | hello-view.fxml | JavaFXMLControlsController.java

```
3   import ...
12
13  public class JavaFXMLControlsController {
14
15
16      @FXML private ComboBox<String> myComboBox;
17      @FXML private ImageView myImageView;
18      @FXML private Label myLabel;
19      @FXML private CheckBox myNoCheckBox;
20      @FXML private CheckBox myYesCheckBox;
21
22      @FXML
23      private void initialize(){
24          myYesCheckBox.setSelected(true);
25          ObservableList<String> options= FXCollections.observableArrayList( ...es: "Full","Half","Empty");
26          myLabel.textProperty().bind(myComboBox.getSelectionModel().selectedItemProperty());
27          myComboBox.setItems(options);
28          Image image=new Image(String.valueOf(JavaFXMLControlsApplication.class.getResource( name: "uon.png")));
29          myImageView.setImage(image);
30      }
```

ImageView

- Can be done with an ImageView Control
  - Give it an fx:id
  - Then in your code:

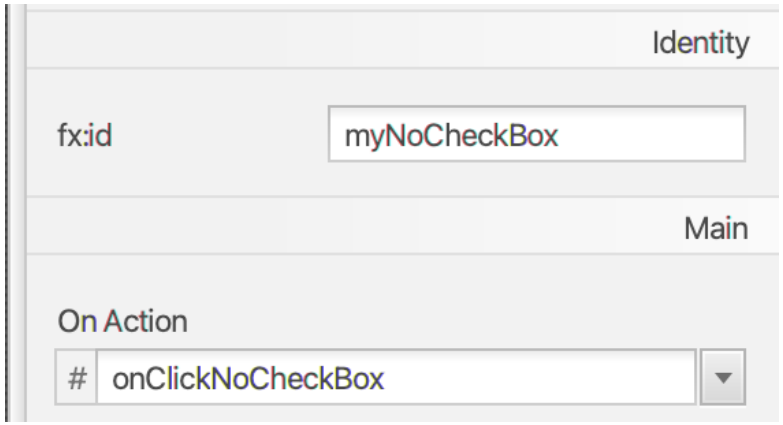


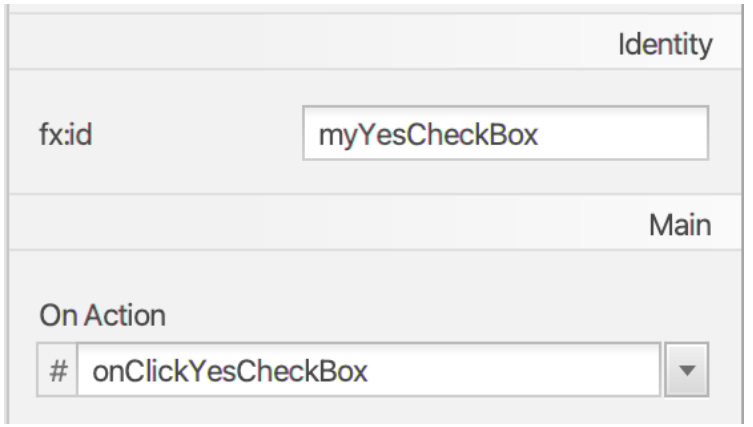
- Watch for working directories when loading files
  - My default is project\resources

CheckBox

- Check boxes have a boolean state, depending whether checked or not

- Steps
  - Add your "onClickYesCheckBox()" method to the corresponding Controller class
  - Give it an fx:id and listen for the action





JavaFXMLControlsApplication.java | hello-view.fxml | JavaFXMLControlsController.java

```
3   import ...
12
13  public class JavaFXMLControlsController {
14
15
16      @FXML private ComboBox<String> myComboBox;
17      @FXML private ImageView myImageView;
18      @FXML private Label myLabel;
19      @FXML private CheckBox myNoCheckBox;
20      @FXML private CheckBox myYesCheckBox;
21
22      @FXML
23      private void initialize(){
24          myYesCheckBox.setSelected(true);
25          ObservableList<String> options= FXCollections.observableArrayList( ...es: "Full","Half","Empty");
26          myLabel.textProperty().bind(myComboBox.getSelectionModel().selectedItemProperty());
27          myComboBox.setItems(options);
28          Image image=new Image(String.valueOf(JavaFXMLControlsApplication.class.getResource( name: "uon.png")));
29          myImageView.setImage(image);
30      }
```


```
32      @FXML
33      private void onClickYesCheckBox() {
34          if(myYesCheckBox.isSelected()){
35              myNoCheckBox.setSelected(false);
36          }else{
37              myNoCheckBox.setSelected(true);
38          }
39      }
```

```
41      @FXML
42      private void onClickNoCheckBox() {
43          if(myNoCheckBox.isSelected()){
44              myYesCheckBox.setSelected(false);
45          }else{
46              myYesCheckBox.setSelected(true);
47          }
48      }
```

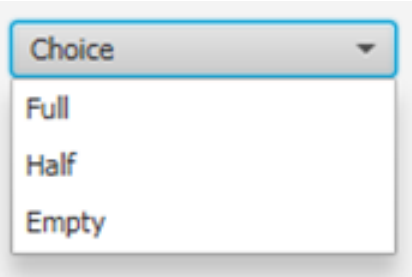

</>

- The result


Hello!
File
Edit
Help
University of
Nottingham
UK | CHINA | MALAYSIA
Yes
No
Choice
Do Work
Cancel

ComboBox

- This is the 'drop down box' style of control.
- It presents the user with a list of items, and the user can select one.
- Slightly different to other controls so far:
  - Need to get items into the list
  - Need to check when an item had been selected



JavaFXMLControlsApplication.java | hello-view.fxml | JavaFXMLControlsController.java

```
3   import ...
12
13  public class JavaFXMLControlsController {
14
15
16      @FXML private ComboBox<String> myComboBox;
17      @FXML private ImageView myImageView;
18      @FXML private Label myLabel;
19      @FXML private CheckBox myNoCheckBox;
20      @FXML private CheckBox myYesCheckBox;
21
22      @FXML
23      private void initialize(){
24          myYesCheckBox.setSelected(true);
25          ObservableList<String> options= FXCollections.observableArrayList( ...es: "Full","Half","Empty");
26          myLabel.textProperty().bind(myComboBox.getSelectionModel().selectedItemProperty());
27          myComboBox.setItems(options);
28          Image image=new Image(String.valueOf(JavaFXMLControlsApplication.class.getResource( name: "uon.png")));
29          myImageView.setImage(image);
30      }
```

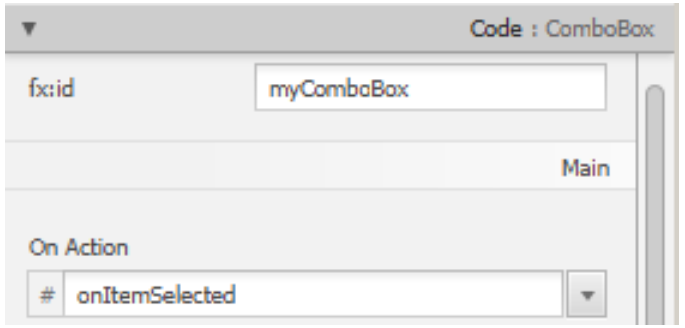
JavaFXMLControlsApplication.java | hello-view.fxml | JavaFXMLControlsController.java

```
3   import ...
12
13  public class JavaFXMLControlsController {
14
15
16      @FXML private ComboBox<String> myComboBox;
17      @FXML private ImageView myImageView;
18      @FXML private Label myLabel;
19      @FXML private CheckBox myNoCheckBox;
20      @FXML private CheckBox myYesCheckBox;
21
22      @FXML
23      private void initialize(){
24          myYesCheckBox.setSelected(true);
25          ObservableList<String> options= FXCollections.observableArrayList( ...es: "Full","Half","Empty");
26          myLabel.textProperty().bind(myComboBox.getSelectionModel().selectedItemProperty());
27          myComboBox.setItems(options);
28          Image image=new Image(String.valueOf(JavaFXMLControlsApplication.class.getResource( name: "uon.png")));
29          myImageView.setImage(image);
30      }
```

ObservableList: A list that allows listeners to track changes when they occur

ComboBox

- How do you get selections?
  - Using SceneBuilder and @FXML
    - Choose OnAction event in SceneBuilder
    - Then write a handler method



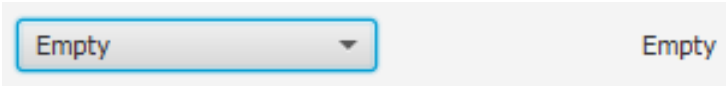

```
50      @FXML
51      private void onItemSelected() {
52          String s=myComboBox.getSelectionModel().getSelectedItem();
53          System.out.println("Choice: "+s);
54      }
```

ComboBox

- There is another way to get the choice from a combo box, and in fact to link other kinds of controls with variables: Binding

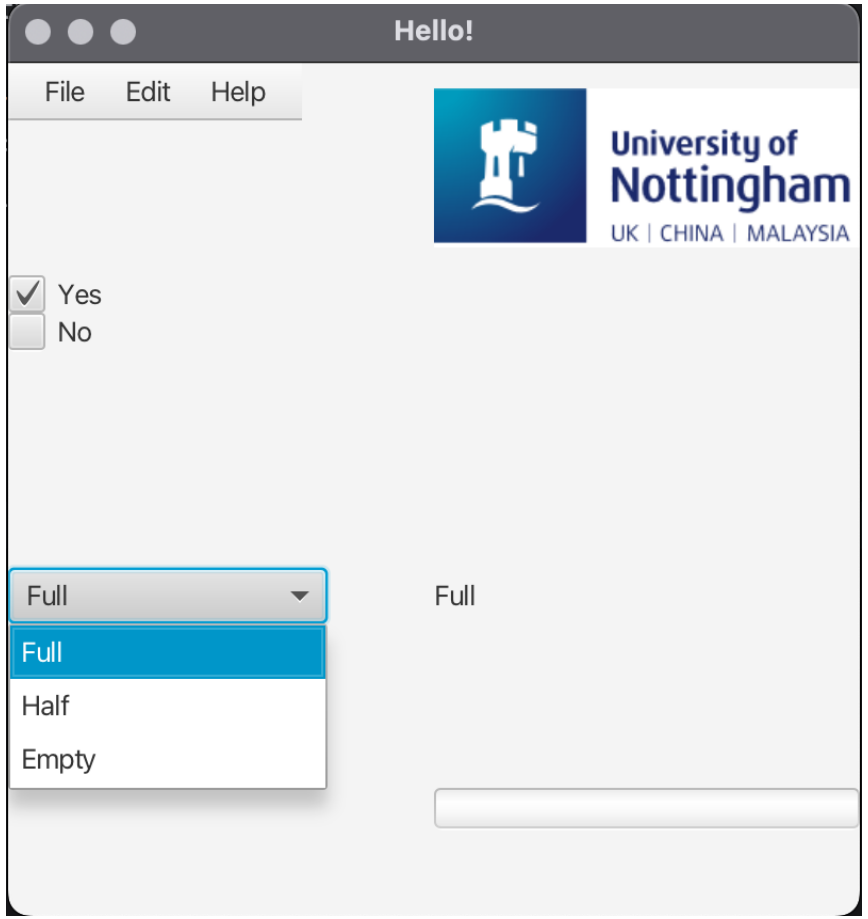
```
ObservableList<String> options= FXCollections.observableArrayList( ...es: "Full","Half","Empty");
myLabel.textProperty().bind(myComboBox.getSelectionModel().selectedItemProperty());
myComboBox.setItems(options);
```

- Whenever the value of the ComboBox property changes, the Label text changes automatically

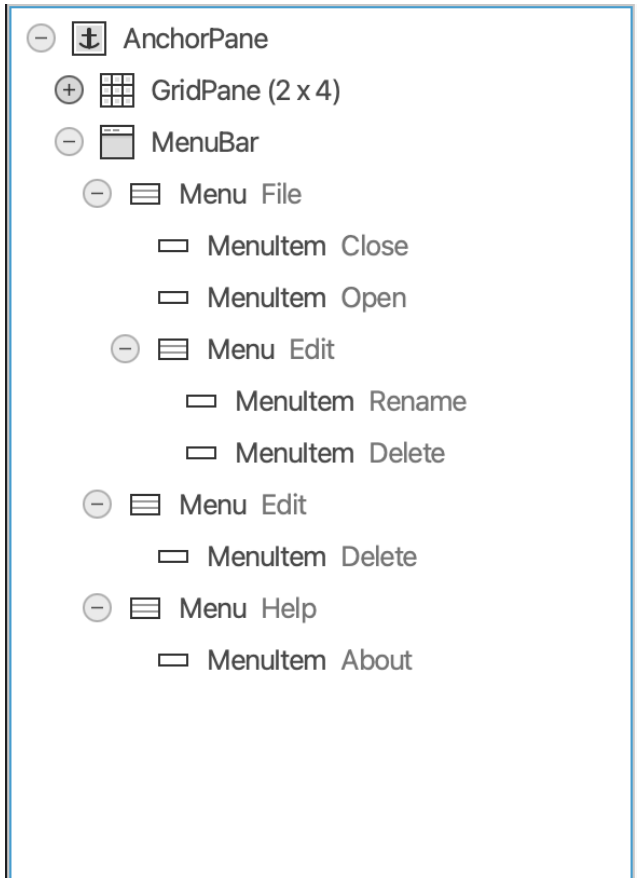


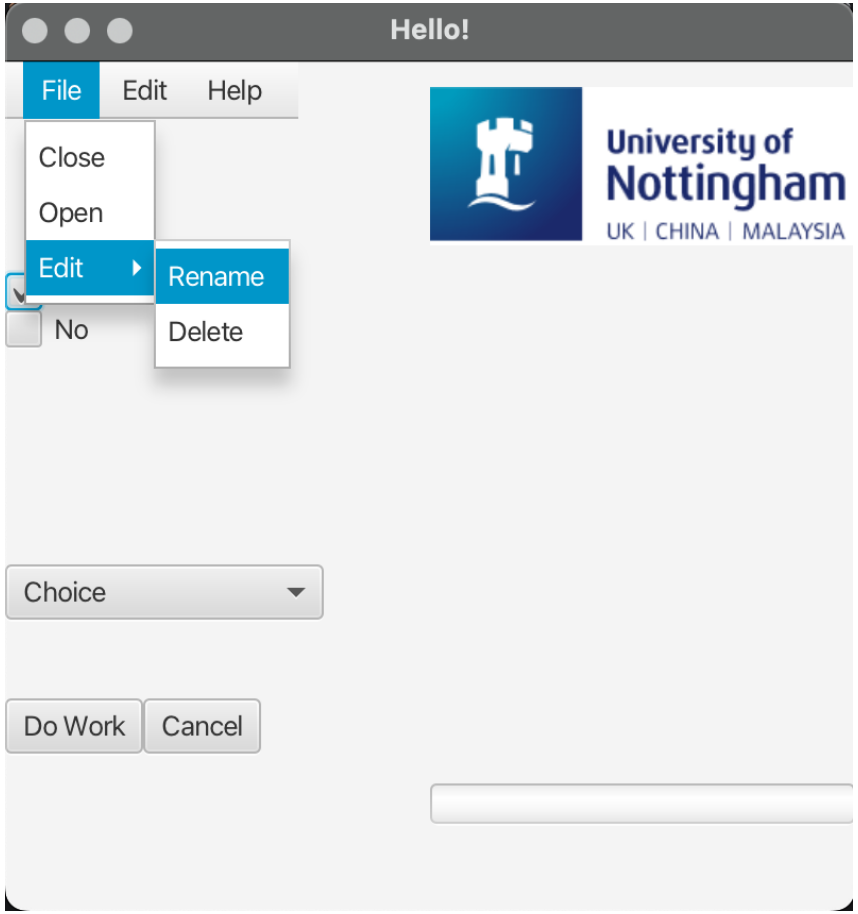
ComboBox

- The result

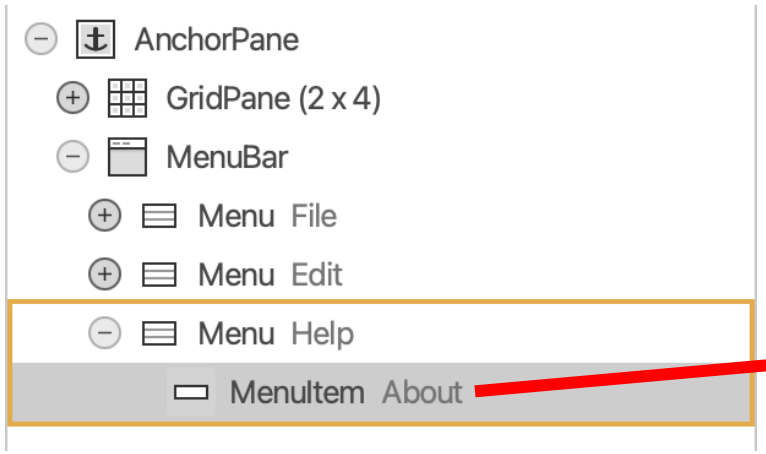


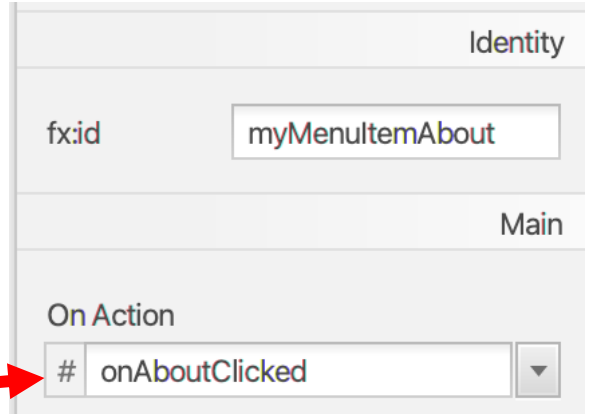
MenuBar


AnchorPane
GridPane (2 x 4)
MenuBar
Menu File
MenuItem Close
MenuItem Open
Menu Edit
MenuItem Rename
MenuItem Delete
Menu Edit
MenuItem Delete
Menu Help
MenuItem About


Hello!
File
Edit
Help
Close
Open
Edit
Rename
Delete
No
University of
Nottingham
UK | CHINA | MALAYSIA
Choice
Do Work
Cancel

MenuBar

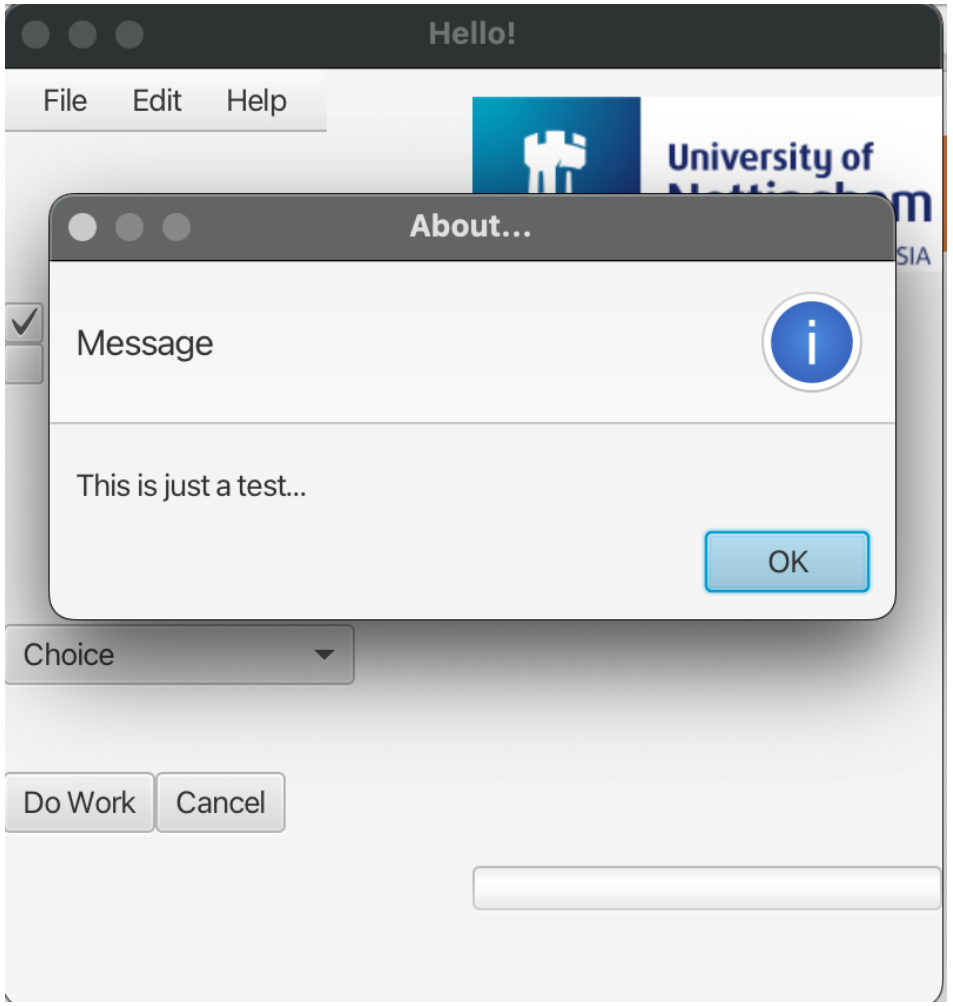




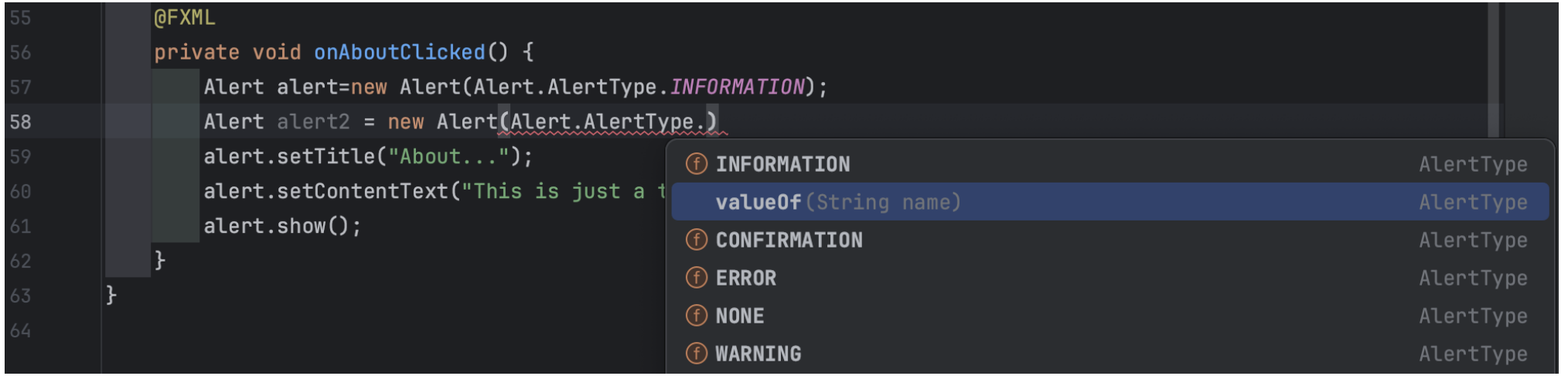

```
55      @FXML
56      private void onAboutClicked() {
57          Alert alert=new Alert(Alert.AlertType.INFORMATION);
58          alert.setTitle("About...");
59          alert.setContentText("This is just a test...");
60          alert.show();
61      }
```

MenuBar

- The result



Alert Types


@FXML
private void onAboutClicked() {
Alert alert=new Alert(Alert.AlertType.INFORMATION);
Alert alert2 = new Alert(Alert.AlertType.)
alert.setTitle("About...");
alert.setContentText("This is just a t
alert.show();
}
}
INFORMATION AlertType
valueOf(String name) AlertType
CONFIRMATION AlertType
ERROR AlertType
NONE AlertType
WARNING AlertType

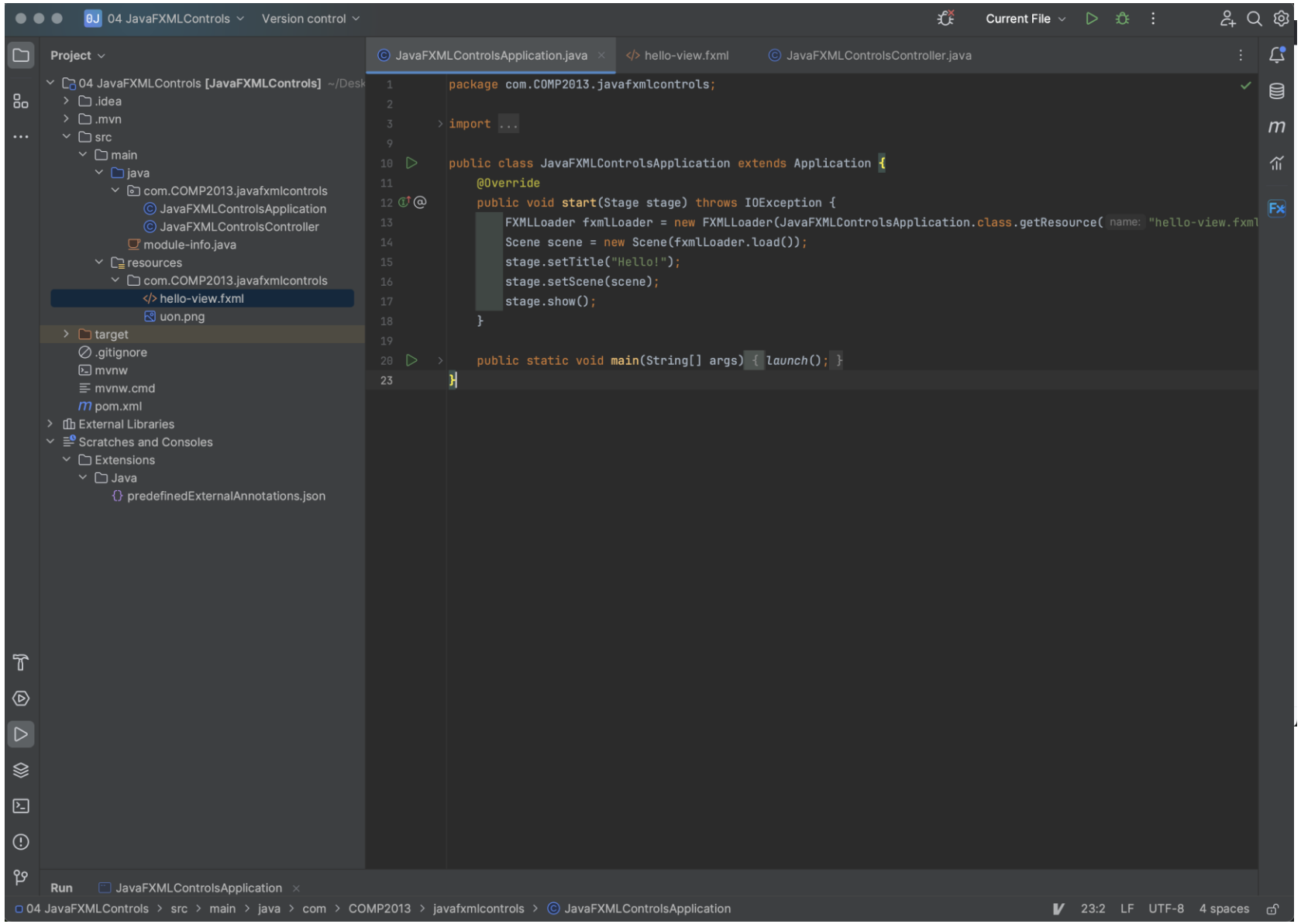
FXML

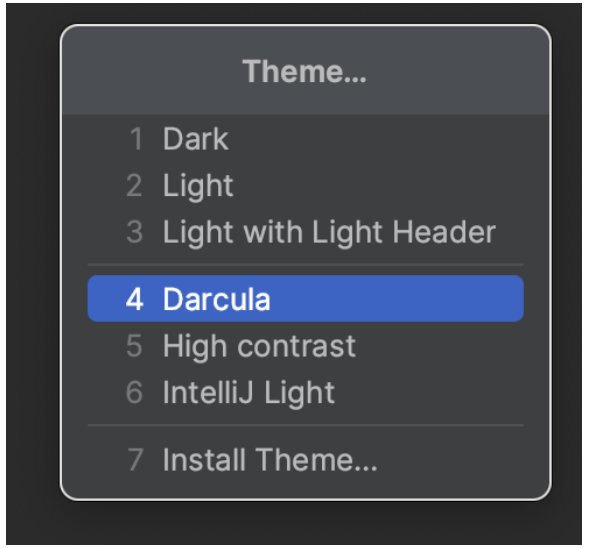
Styling

Customising GUI Appearance

- In addition to laying out the elements on the screen, we can also specify how they look
- Remember we want to keep appearance, design and code separated
- Here we will change the appearance without adjusting anything else.
  - Suppose your company acquires a software package
  - The first thing you might want to do is bring the appearance in line with the rest of your product range

Applications with Distinctive Themes





Cascading Style Sheet

- A Cascading Style Sheet (CSS) is a means to define the look (visual properties) of UI elements in a GUI application
  - Primary developed for use in web pages
  - Allows separation of presentation and content/behaviour

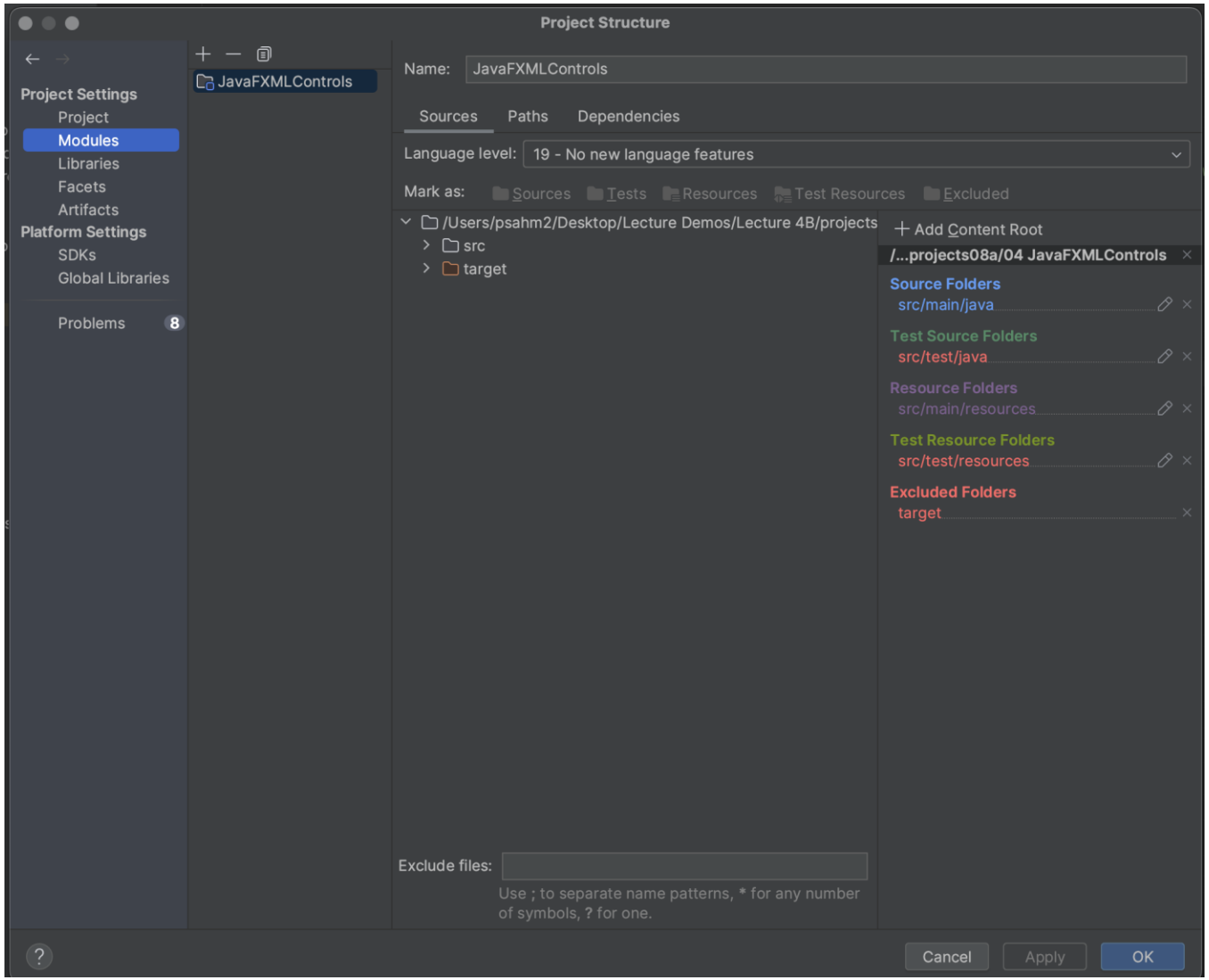
- CSS provides the syntax to define rules
  - A rule consists of selector and property/value pair(s)

```
.button {
        -fx-background-color: red;
        -fx-text-size: 22;
}
```

https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html

User Defined Style Sheets

- Create *.css style sheet in "resources" folder
  - Remember to add the resources folder to the "build-path“



User Defined Style Sheets

- JavaFX CSS naming conventions
  - All style class names are lowercase node names
  - If node name consists of multiple words style class names use hyphen
  - Property names start with "-fx-" followed by instance variables (lowercase + hypenated)

```
.check-box .box{
    -fx-background-color:white;
    -fx-border-color:red;
    -fx-border-radius:12px;
}
.check-box{
    -fx-font-size:16;
}
```

User Defined Style Sheets


```
30  public class JavaFXMLCSSApplication extends Application {
31
32      @Override
33      public void start(Stage primaryStage) throws Exception{
34          Parent root = FXMLLoader.load(getClass().getResource( name: "javafxmlcss-view.fxml"));
35          root.setStyle("-fx-font-size:20");
36          primaryStage.setTitle("Hello JavaFXMLCSS");
37          Application.setUserAgentStylesheet(Application.STYLESHEET_CASPIAN);
38          Scene scene=new Scene(root);
39          scene.getStylesheets().add(String.valueOf(getClass().getResource( name: "mystyles.css")));
40          primaryStage.setScene(scene);
41          primaryStage.show();
42      }
43
44
45      public static void main(String[] args) { launch(args); }
48  }
```

User Defined Style Sheets


```
30  public class JavaFXMLCSSApplication extends Application {
31
32      @Override
33      public void start(Stage primaryStage) throws Exception{
34          Parent root = FXMLLoader.load(getClass().getResource( name: "javafxmlcss-view.fxml"));
35          root.setStyle("-fx-font-size:20");
36          primaryStage.setTitle("Hello JavaFXMLCSS");
37          Application.setUserAgentStylesheet(Application.STYLESHEET_CASPIAN);
38          Scene scene=new Scene(root);
39          scene.getStylesheets().add(String.valueOf(getClass().getResource( name: "mystyles.css")));
40          primaryStage.setScene(scene);
41          primaryStage.show();
42      }
43
44
45      public static void main(String[] args) { launch(args); }
48  }
```

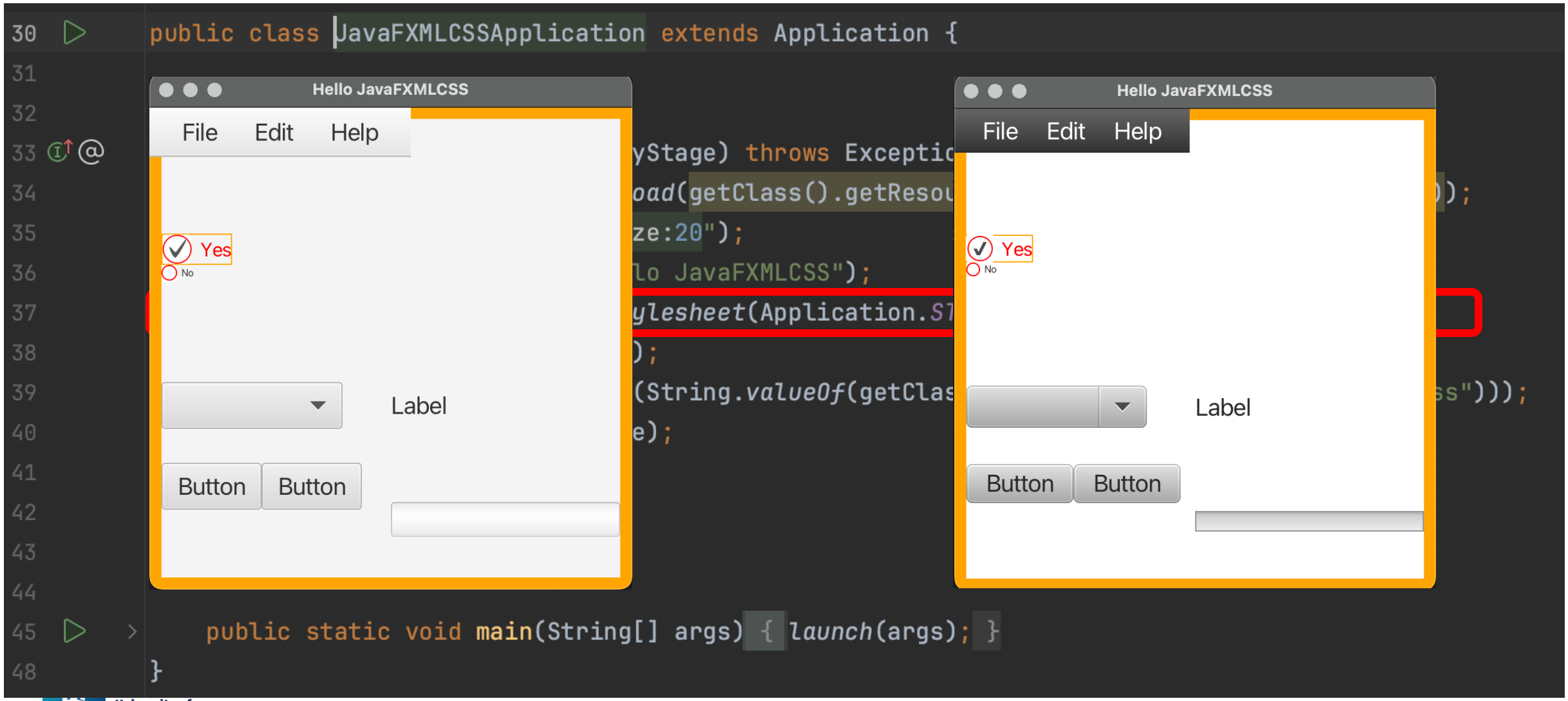
User Defined Style Sheets


```
30  public class JavaFXMLCSSApplication extends Application {
31
32      @Override
33      public void start(Stage primaryStage) throws Exception{
34          Parent root = FXMLLoader.load(getClass().getResource( name: "javafxmlcss-view.fxml"));
35          root.setStyle("-fx-font-size:20");
36          primaryStage.setTitle("Hello JavaFXMLCSS");
37          Application.setUserAgentStylesheet(Application.STYLESHEET_CASPIAN);
38          Scene scene=new Scene(root);
39          scene.getStylesheets().add(String.valueOf(getClass().getResource( name: "mystyles.css")));
40          primaryStage.setScene(scene);
41          primaryStage.show();
42      }
43
44
45      public static void main(String[] args) { launch(args); }
48  }
```

User Defined Style Sheets


```
30  public class JavaFXMLCSSApplication extends Application {
31
32      @Override
33      public void start(Stage primaryStage) throws Exception{
34          Parent root = FXMLLoader.load(getClass().getResource( name: "javafxmlcss-view.fxml"));
35          root.setStyle("-fx-font-size:20");
36          primaryStage.setTitle("Hello JavaFXMLCSS");
37          Application.setUserAgentStylesheet(Application.STYLESHEET_CASPIAN);
38          Scene scene=new Scene(root);
39          scene.getStylesheets().add(String.valueOf(getClass().getResource( name: "mystyles.css")));
40          primaryStage.setScene(scene);
41          primaryStage.show();
42      }
43
44
45      public static void main(String[] args) { launch(args); }
48  }
```

User Defined Style Sheets



User Defined Style Sheets

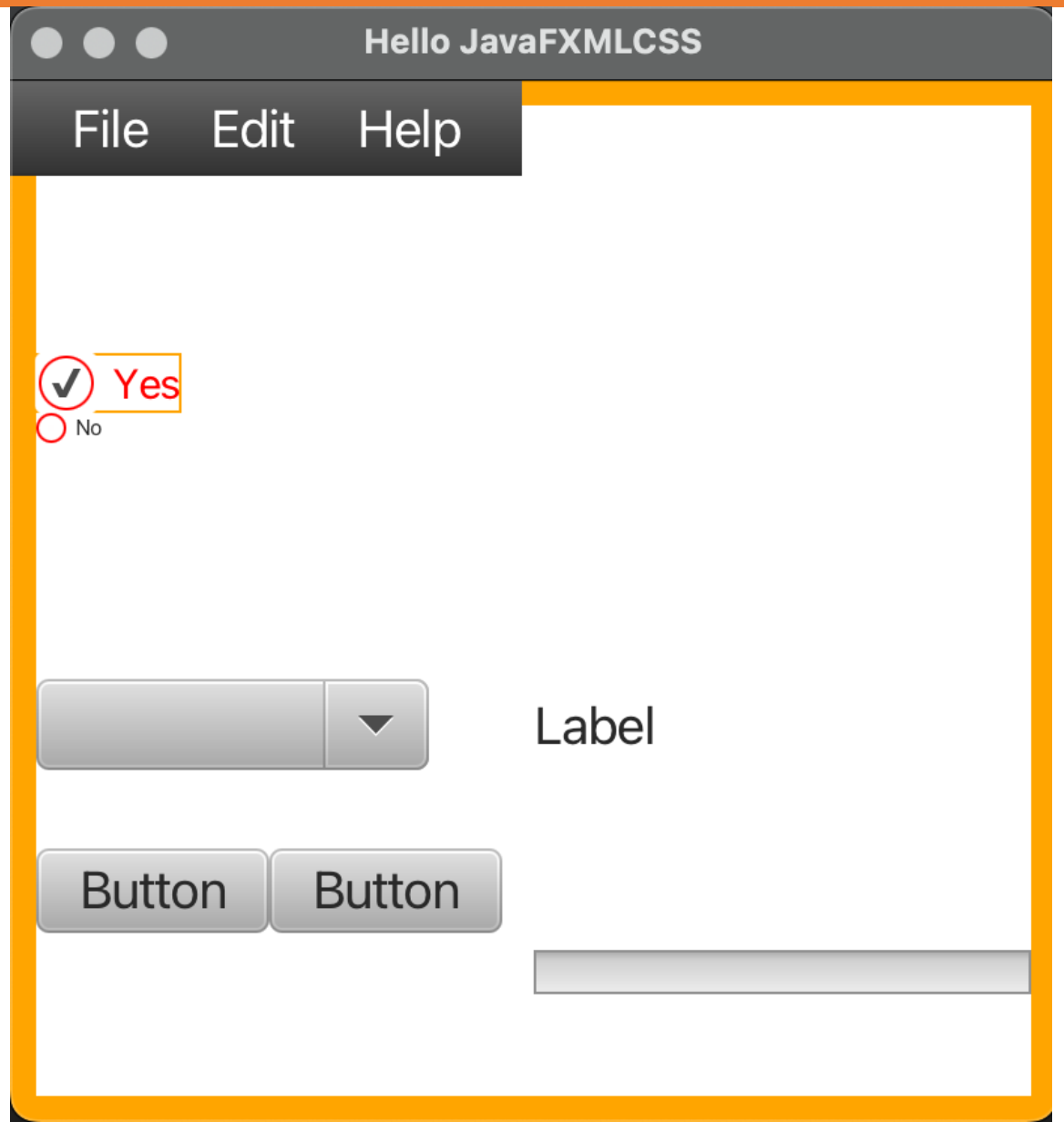

```
30  public class JavaFXMLCSSApplication extends Application {
31
32      @Override
33      public void start(Stage primaryStage) throws Exception{
34          Parent root = FXMLLoader.load(getClass().getResource( name: "javafxmlcss-view.fxml"));
35          root.setStyle("-fx-font-size:20");
36          primaryStage.setTitle("Hello JavaFXMLCSS");
37          Application.setUserAgentStylesheet(Application.STYLESHEET_CASPIAN);
38          Scene scene=new Scene(root);
39          scene.getStylesheets().add(String.valueOf(getClass().getResource( name: "mystyles.css")));
40          primaryStage.setScene(scene);
41          primaryStage.show();
42      }
43
44
45      public static void main(String[] args) { launch(args); }
48  }
```

Mystyles.css file


```
.check-box{
-fx-background-color:white;
-fx-border-color:red;
-fx-border-radius:12px;
}
.check-box{
-fx-font-size:16;
}
#myNoCheckBox{
-fx-font-size:8;
}
```

User Defined Style Sheets

- The result



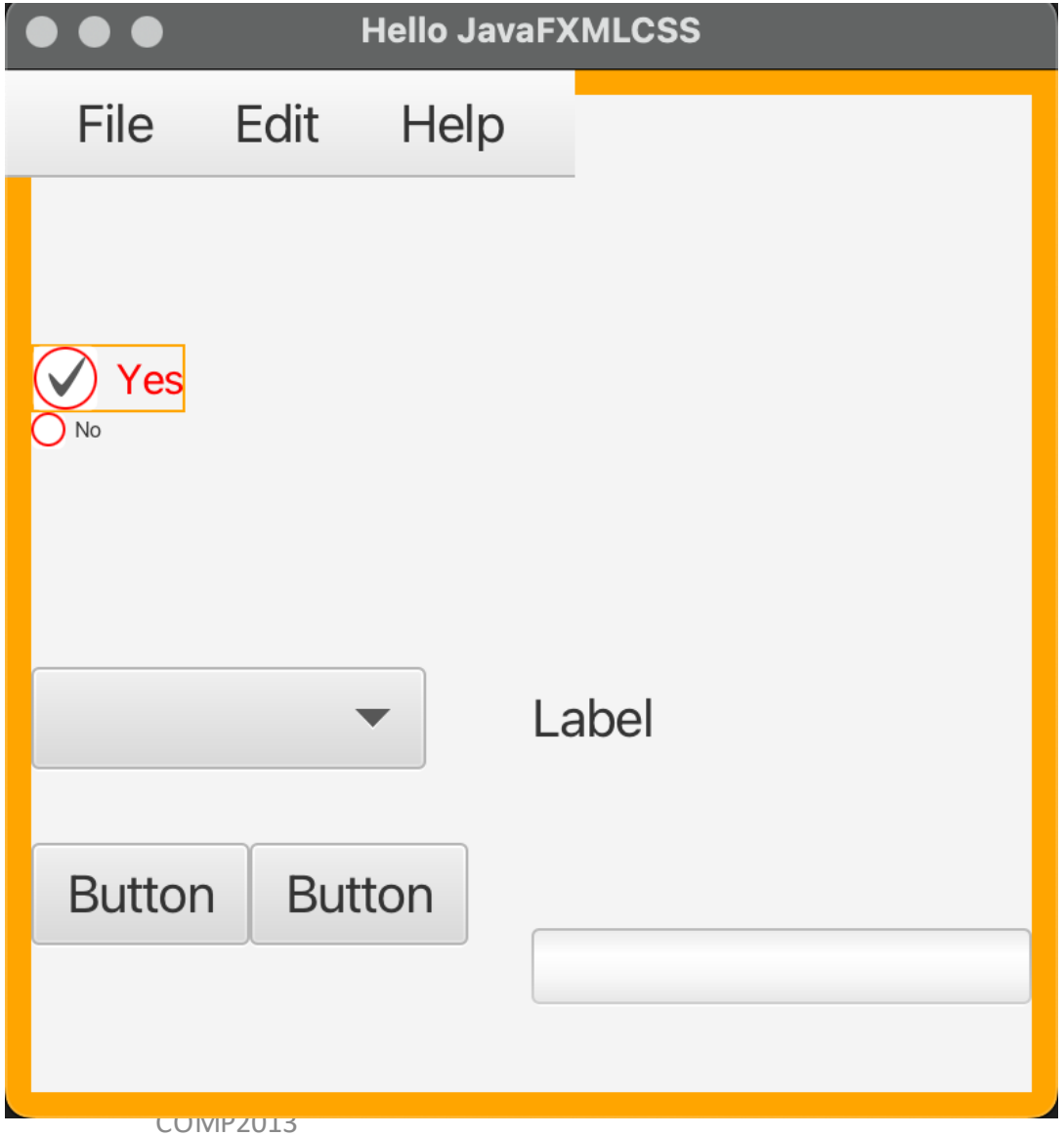
Default Style Sheets

- Default look that you get for JavaFX application is "modena.css“

```
Application.setUserAgentStylesheet(Application.STYLESHEET_CASPIAN);
primaryStage.setScene(scene);
primaryStage.show();
```

Default Style Sheets

- The result

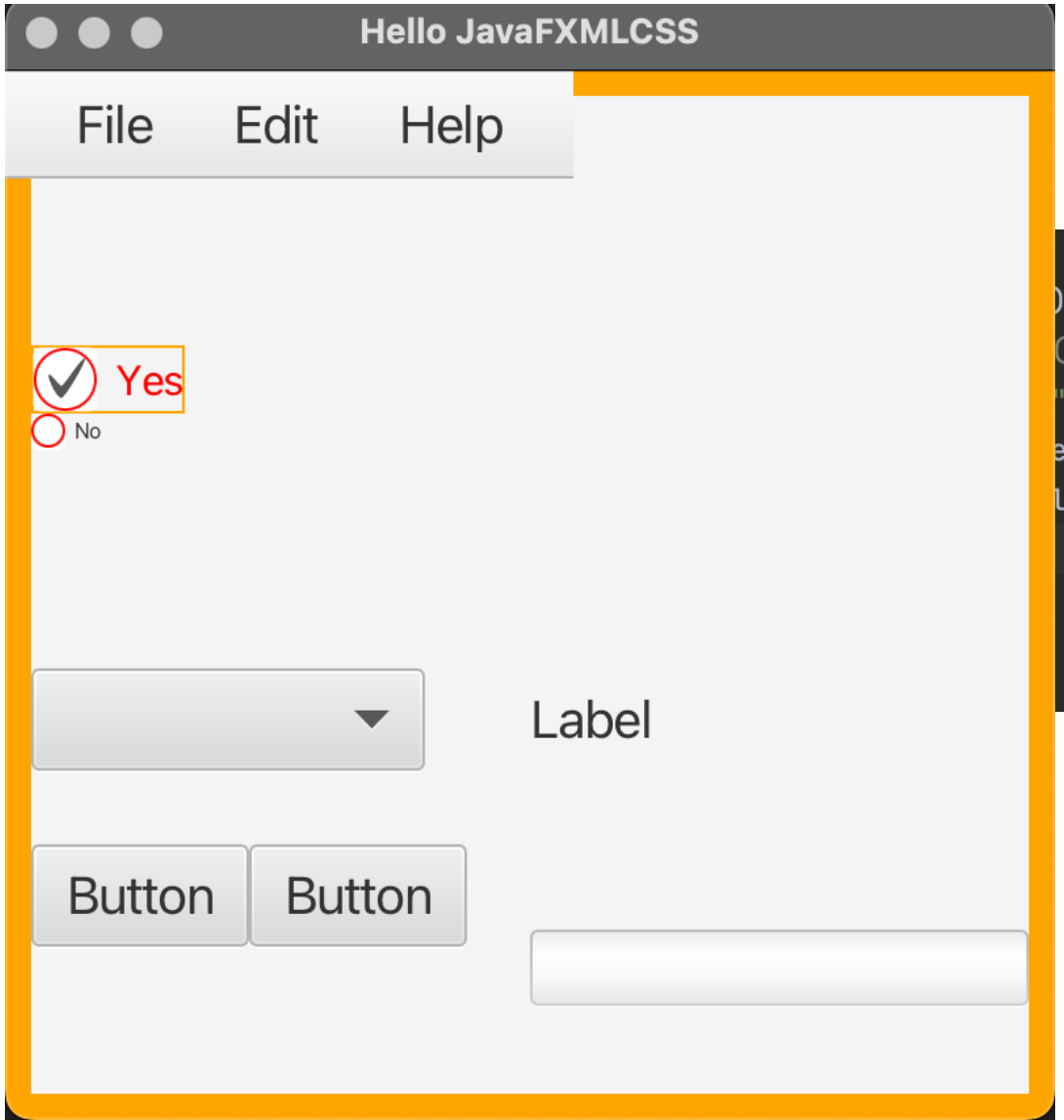


Adding Inline Styles

- You can add styles to individual nodes
  - Use: setStyle(String inlineStyle) to set the style and getStyle() to get the style of a node

```
public class Controller {
    @FXML private CheckBox myYesCheckBox;

    @FXML
    private void initialize(){
        myYesCheckBox.setSelected(true);
        myYesCheckBox.setStyle("-fx-text-fill:red;-fx-font-weight:bold");
    }
}
```

Adding Inline Styles




```
){
(myGridPane.getParent());
"-fx-border-width:10;-fx-border-color:orange");
ected(true);
le("-fx-text-fill:red;-fx-font-weight:bold;-fx-border-color:inherit");
```

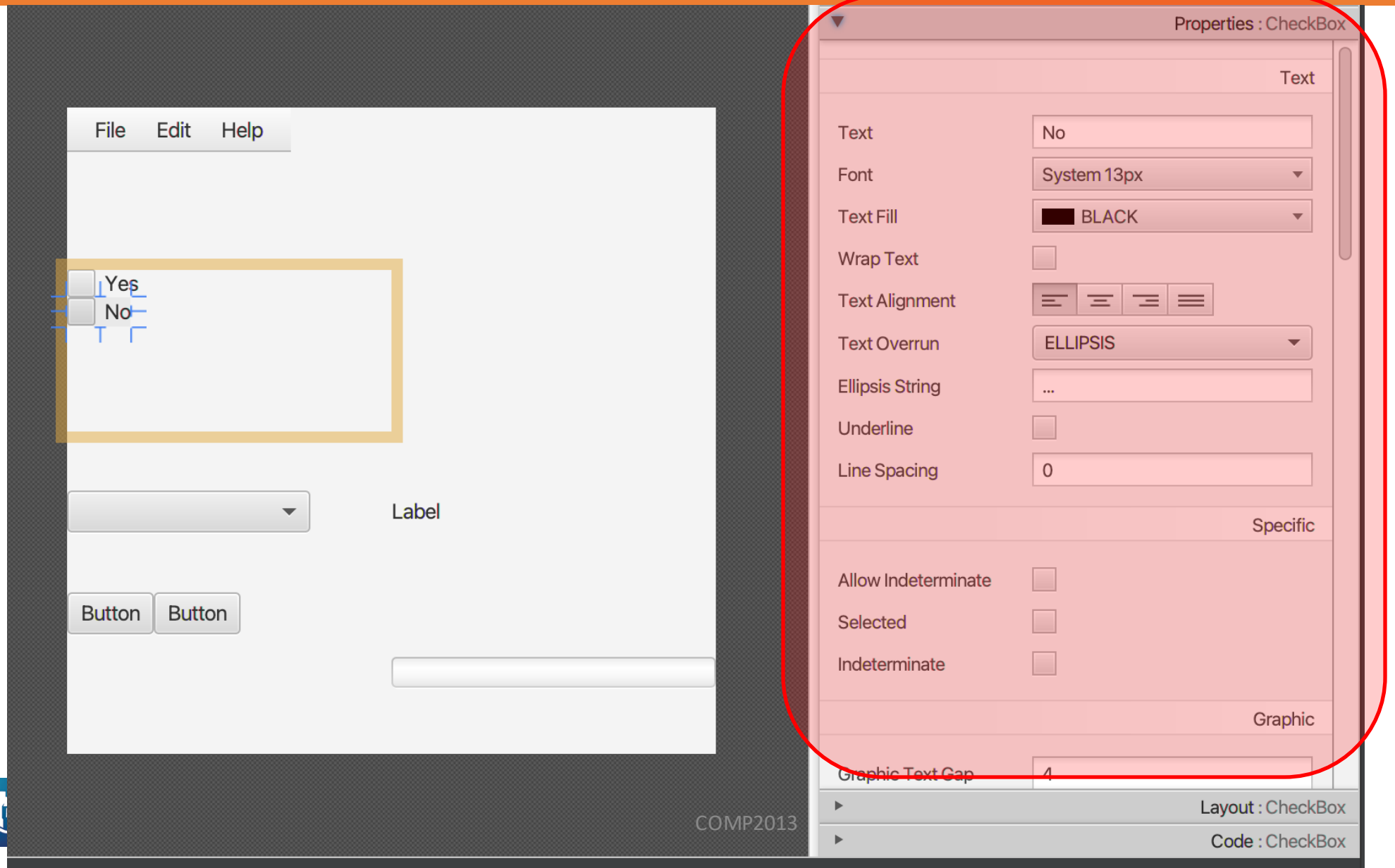
Priorities of Styles

- Ranking order (highest to lowest)
  - Inline style
  - Parent style sheet
  - Scene style sheet
  - Values set in the code using JavaFX API (e.g. using "setFont()")
  - User agent style sheet (JavaFX runtime)

Using ID selectors

- We can also use the IDs created in Scene Builder to set styles for individual nodes in the CSS style sheet

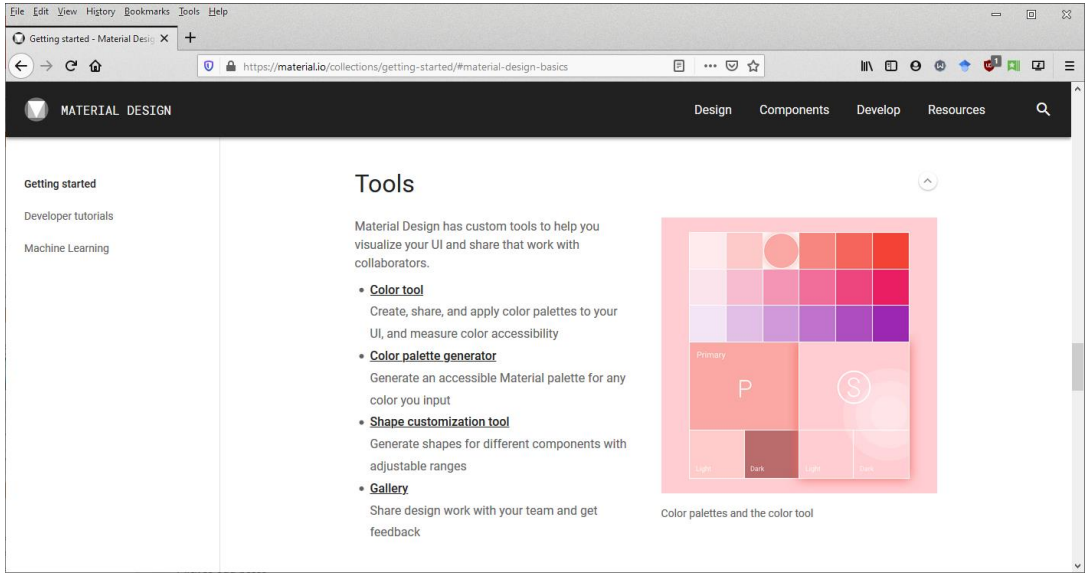
```
.check-box{
-fx-font-size:16;
}

#myNoCheckBox{
-fx-font-size:8;
}
```


File
Edit
Help
Yes
No
Label
Button
Button
Properties : CheckBox
Text
Text
No
Font
System 13px
Text Fill
BLACK
Wrap Text
Text Alignment
Text Overrun
ELLIPSIS
Ellipsis String
...
Underline
Line Spacing
0
Specific
Allow Indeterminate
Selected
Indeterminate
Graphic
Graphic Text Gap
4
Layout : CheckBox
Code : CheckBox

Resources

- General JavaFX
  - Comprehensive introduction to JavaFX
    - https://www3.ntu.edu.sg/home/ehchua/programming/java/Javafx1_intro.html

- Excellent material design guide
  - https://material.io/design/



some final remarks …