Introduction to JavaFX

JavaFX is a graphics framework for creating desktop and mobile apps.

Unlike previous Java graphics frameworks, JavaFX interfaces can be defined declaratively (in XML) instead of Java code.

Example of both ways shown later.

Some Concepts

A JavaFX application contains a Window

that contains a Stage

that contains a graph (tree) of Components

and Layouts

appearance is controlled by properties you can set:

size, color, background, spacing (social distancing), ...

U.I. is Presented on a Stage

The Stage Contains a Scene

```
stage.setScene( scene );
```
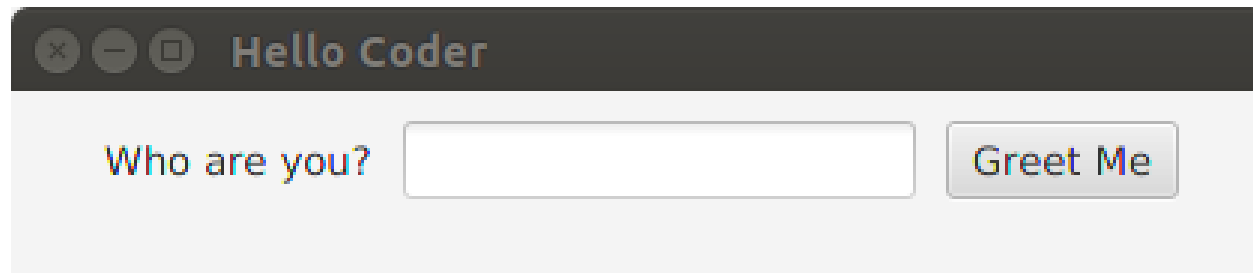
Scene has Components & Layouts

```
scene.setLayout( livingRoomLayout );
scene.getChildren().add( Sofa );
```

What You Need to Know

1. What are the Layouts and Containers?

2. What are the Components?

3. What Properties can I set? (appearance)

4. How to respond to Events?

Create this UI in Code
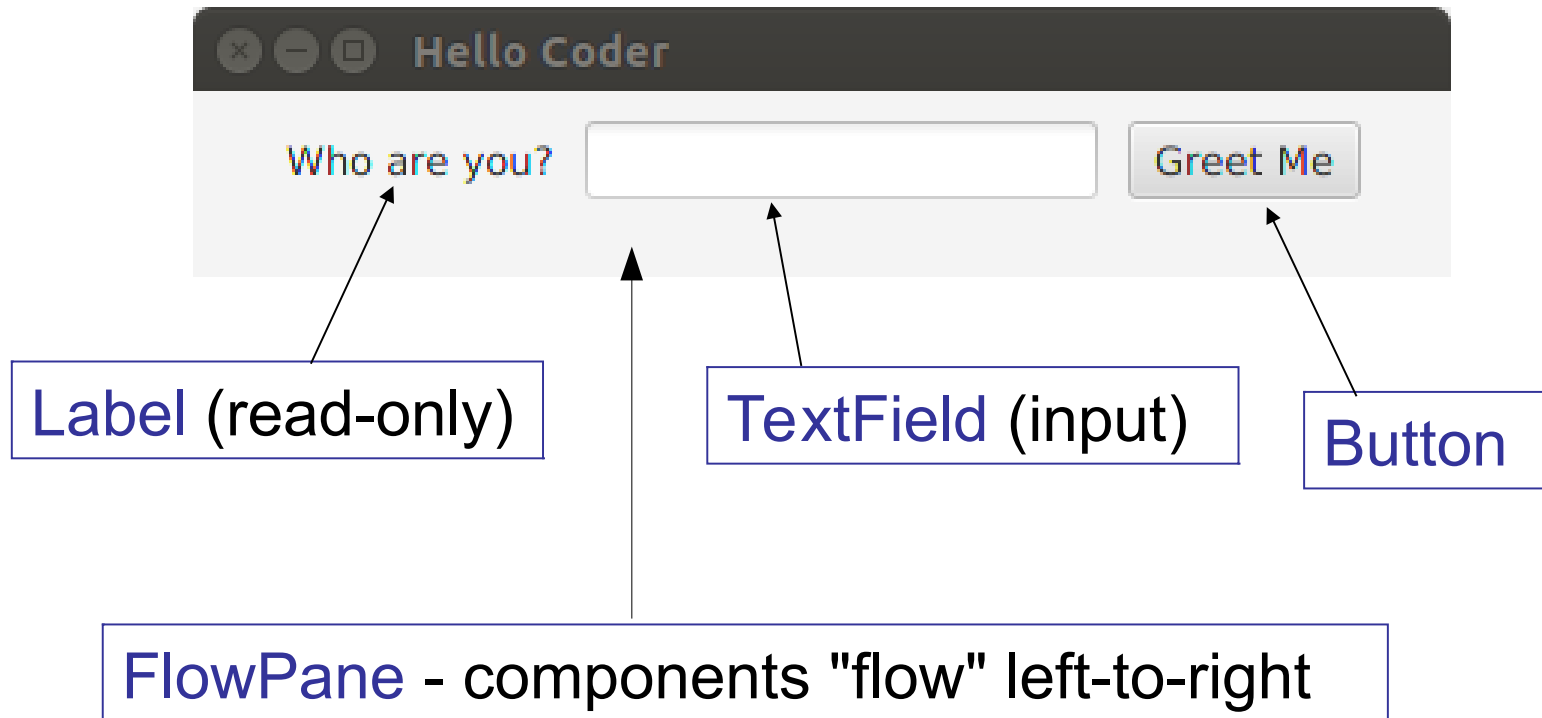
Structure of JavaFX App (main)

```java
public class HelloFX extends Application {
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        // Create a container as root node in the Scene
        FlowPane root = new FlowPane();
        // Set appearance of container (spacing, alignment)

        // Add components to the container)

        // Show the scene graph on the Stage
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }
```
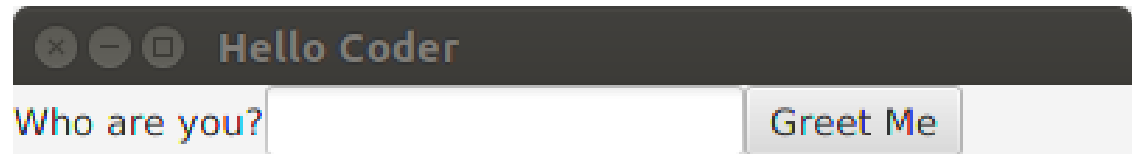
Define Container & Components

Hello Coder

Who are you? [_____] [Greet Me]

Label (read-only)

TextField (input)

Button

FlowPane - components "flow" left-to-right

Add Components

```java
public void start(Stage primaryStage) {
  FlowPane root = new FlowPane();
  // Set appearance of container

  // Add components to the container)
  Label prompt = new Label("Who are you?");
  TextField nameField = new TextField();
  Button button = new Button("Greet Me");
  root.getChildren.add(prompt);
  root.getChildren.add(nameField);
  root.getChildren.add(button);
```

View It

Looks ugly.

Run-time Annoyance

When you run a JavaFX application with Java 11 you may this message:

```
Error: JavaFX runtime components are
   missing, and are required to run this...
```

This relates to *modules* in Java 9+. Here's a fix:

Cmd line:

java --module-path /path/to/javafx/lib

**   --add-modules javafx.base,javafx.controls**

IDE: Add --module-path and --add-modules to **VM args.**

Java 8 - Retrograde Solution

Java 8 includes JavaFX in the JDK (no external Jars) and does not use modules.

You can add JDK 8 to your system and configure it in Eclipse or IntelliJ, and maybe in VS Code.

You choose which IDE (JDK8, JDK11, etc.) for each project.

You must be careful to run from command line using Java 8 "java" command, too.

Otherwise JavaFX classes will not be found.

Properties for Appearance

Every control has properties you can set that effect its appearance. Modify the FlowPane:

```java
FlowPane root = new FlowPane();
// Set appearance of container
root.setAlignment(Pos.CENTER);
root.setHgap(10.0);
root.setPadding(new Insets(10.0));
```
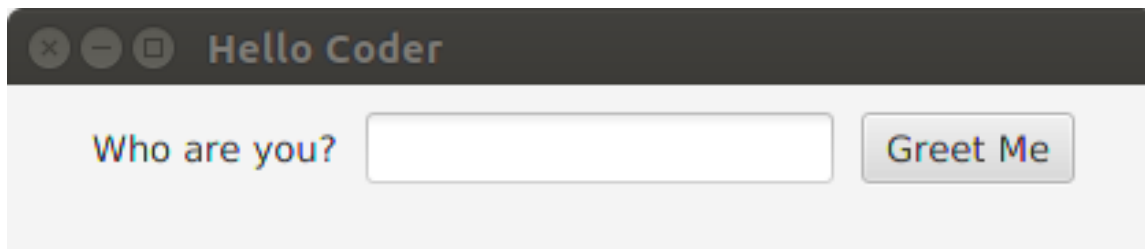
Where to learn properties?

The Oracle JavaFX Tutorial gives many examples of setting properties of components.

Oracle has downloadable PDF and ePub for…

Getting Started with JavaFX

JavaFX Layouts

JavaFX UI Controls

Using SceneBuilder (visual layout) its even easier.

Modularize

start() method is getting long.

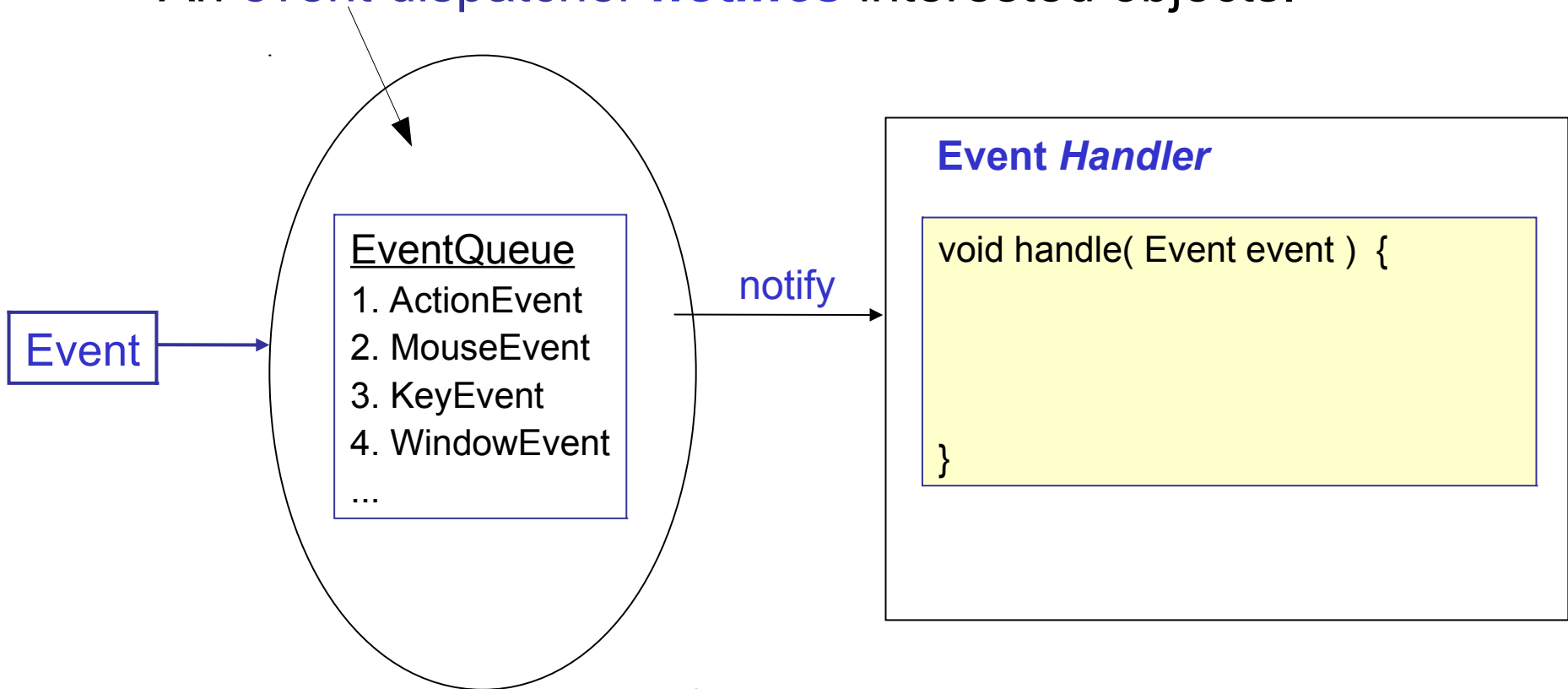Separate component creation to its own method.

```java
public void start(Stage primaryStage) {
    FlowPane root = initComponents();

    // Show the scene graph on the Stage
    primaryStage.setScene(new Scene(root));
    primaryStage.show();
```

Add Behavior

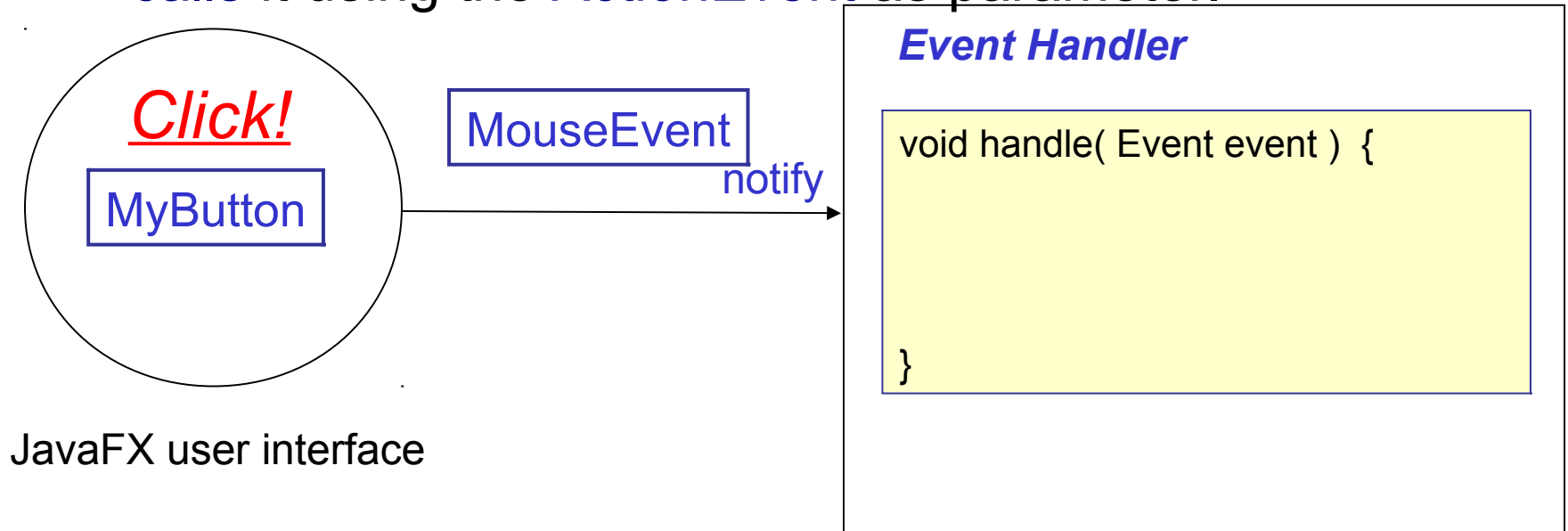UI should respond to click on "Greet Me" button.

Events

- Graphics applications use *events*.
- Event is caused by user actions.
- An event dispatcher *notifies* interested objects.

```
EventQueue
1. ActionEvent
2. MouseEvent
3. KeyEvent
4. WindowEvent
...
```

Event

notify

Event *Handler*

```
void handle( Event event )  {



}
```

Events

1. User clicks mouse on a button -- that's an *Event*.
2. JavaFX creates a MouseEvent object.
   - the MouseEvent describes what happened: which component? which mouse button?
3. JavaFX looks for a registered "*Event Handler*", and calls it using the ActionEvent as parameter.

Click!

MyButton

MouseEvent

notify

Event Handler

```
void handle( Event event ) {



}
```

JavaFX user interface

Adding Event Handlers

You tell JavaFX what events you want to handle, and which code to invoke:

button.setOnAction(EventHandler<ActionEvent>)

== or ==

button.addEventHandler(eventType, eventHandler)

Write an EventHandler

This example uses an *inner class*.
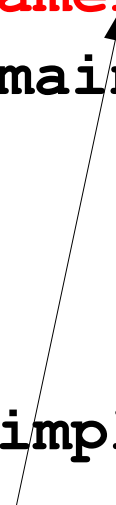
Many examples use *anonymous class* or *lambda.*

```
class ClickHandler
        implements EventHandler<ActionEvent> {
    public void handle(ActionEvent event) {
        String name = nameField.getText().trim();
        if (name.isEmpty()) {
            nameField.setPromptText(
                    "Please enter a name");
        }
        else showDialog("Hello, "+name);
    }
}
```

Access the TextField

EventHandler needs access to the `nameField`.

Define it as an attribute instead of a local variable.

```java
public class HelloFX extends Application {
    private TextField nameField;
    public static void main(String[] args) {
        launch(args);
    }

    class ClickHandler implements ... {
        // inner class can access outer class
    }
}
```

Attach Event Handler

```
private void initComponents() {

    Button button = new Button("Greet me");

    button.setOnAction(new ClickHandler());
```

showDialog

Instead of printing on *boring* System.out,
pop-up an Alert box to greet user.

```java
public void showDialog(String message) {
    Alert alert = new
            Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("Greetings");
    alert.setHeaderText( message );
    // wait for user to dismiss dialog
    alert.showAndWait();
}
```
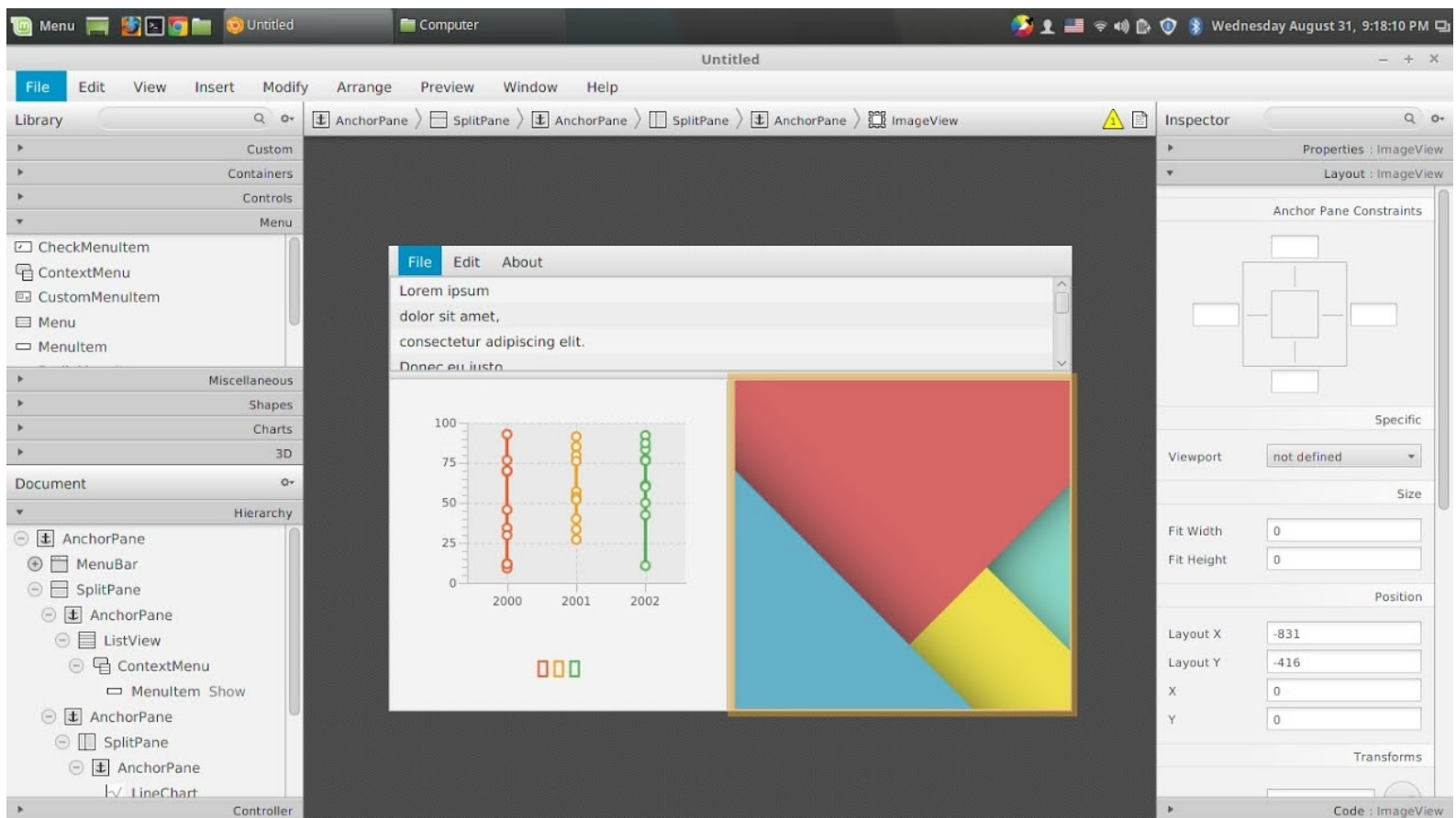
Run it
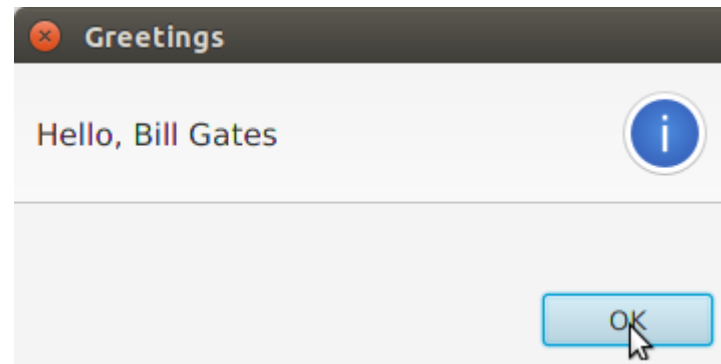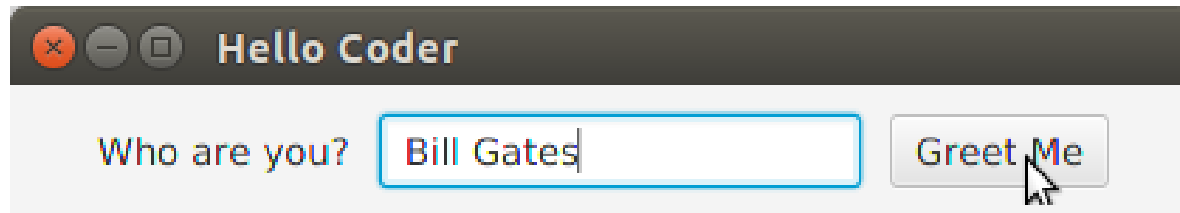
After greeting the person, clear the text from nameField

SceneBuilder

Visual tool for creating graphical UI.   But first...

Writing a UI in Code is Good

Good to learn the concepts and components first.

For a *dynamic UI*, it may be necessary to add components using code.

Good Tutorials

Oracle's JavaFX Tutorial - uses code only. Lots of info about components.

code.makery - SceneBuilder tutorial, 7 parts.

https://code.makery.ch/library/javafx-tutorial/part1/

Vojtech Ruzicka JavaFX & SceneBuilder tutorial

https://www.vojtechruzicka.com/javafx-getting-started/

also 7 parts. Instructions for IntelliJ.

Suggest a Good Tutorial?

If you find a good tutorial site or video(s), post the links on Google classroom.

Or send to me. Posting for everyone is better.