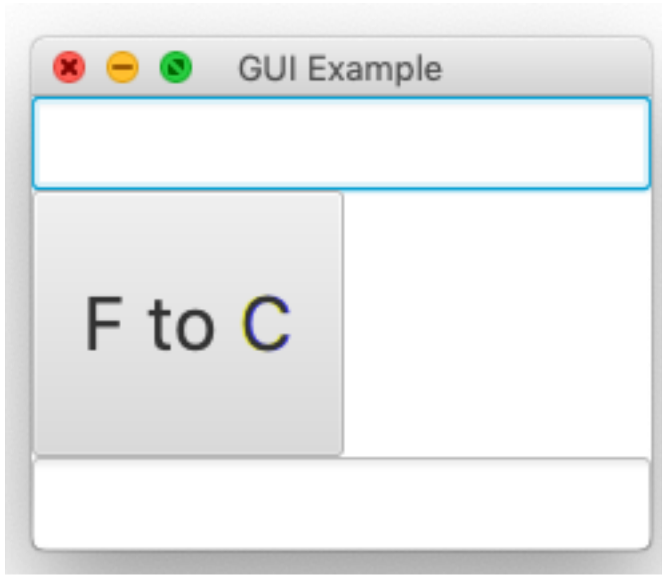
GUI

The Library

- ScalaFX
  - An interface for JavaFX
  - Allows Scala specific features to be used with JavaFX
- Find the xml for the library and add it to your pom.xml

GUI by Example

- Let's build a degrees converter
- Convert degrees Fahrenheit to degrees Celsius



```
object SampleGUI extends JFXApp {

  val inputDisplay: TextField = new TextField {
    style = "-fx-font: 18 ariel;"
  }

  val outputDisplay: TextField = new TextField {
    editable = false
    style = "-fx-font: 18 ariel;"
  }

  val button: Button = new Button {
    minWidth = 100
    minHeight = 100
    style = "-fx-font: 28 ariel;"
    text = "F to C"

  }

  this.stage = new PrimaryStage {
    title = "GUI Example"
    scene = new Scene() {
      content = List(
        new VBox() {
          children = List(inputDisplay, button, outputDisplay)
        }
      )
    }
  }

}
```

GUI by Example

- Extend JFXApp from ScalaFX (JavaFX)
- JFXApp has a state variable named stage
  - Set stage to the GUI elements we want

```
object SampleGUI extends JFXApp {

  val inputDisplay: TextField = new TextField {
    style = "-fx-font: 18 ariel;"
  }

  val outputDisplay: TextField = new TextField {
    editable = false
    style = "-fx-font: 18 ariel;"
  }

  val button: Button = new Button {
    minWidth = 100
    minHeight = 100
    style = "-fx-font: 28 ariel;"
    text = "F to C"

  }

  this.stage = new PrimaryStage {
    title = "GUI Example"
    scene = new Scene() {
      content = List(
        new VBox() {
          children = List(inputDisplay, button, outputDisplay)
        }
      )
    }
  }

}
```

GUI by Example

- **New syntax incoming!**
- Create new instances of the GUI elements we'll use
- Instead of calling a constructor with (parens) we use {braces} to execute code
- All code in braces in executed after the object is created

```
object SampleGUI extends JFXApp {

  val inputDisplay: TextField = new TextField {
    style = "-fx-font: 18 ariel;"
  }

  val outputDisplay: TextField = new TextField {
    editable = false
    style = "-fx-font: 18 ariel;"
  }

  val button: Button = new Button {
    minWidth = 100
    minHeight = 100
    style = "-fx-font: 28 ariel;"
    text = "F to C"

  }

  this.stage = new PrimaryStage {
    title = "GUI Example"
    scene = new Scene() {
      content = List(
        new VBox() {
          children = List(inputDisplay, button, outputDisplay)
        }
      )
    }
  }

}
```

GUI by Example

- TextField has a state variable named style
- Set it to the style you want
- A new TextField is created on the heap
- It's style variable is then overwritten to this String
- A reference to the new object is stored in inputDisplay

```
object SampleGUI extends JFXApp {

  val inputDisplay: TextField = new TextField {
    style = "-fx-font: 18 ariel;"
  }

  val outputDisplay: TextField = new TextField {
    editable = false
    style = "-fx-font: 18 ariel;"
  }

  val button: Button = new Button {
    minWidth = 100
    minHeight = 100
    style = "-fx-font: 28 ariel;"
    text = "F to C"

  }

  this.stage = new PrimaryStage {
    title = "GUI Example"
    scene = new Scene() {
      content = List(
        new VBox() {
          children = List(inputDisplay, button, outputDisplay)
        }
      )
    }
  }

}
```

GUI by Example

- Use the same syntax to setup a button

```scala
object SampleGUI extends JFXApp {

  val inputDisplay: TextField = new TextField {
    style = "-fx-font: 18 ariel;"
  }

  val outputDisplay: TextField = new TextField {
    editable = false
    style = "-fx-font: 18 ariel;"
  }

  val button: Button = new Button {
    minWidth = 100
    minHeight = 100
    style = "-fx-font: 28 ariel;"
    text = "F to C"

  }

  this.stage = new PrimaryStage {
    title = "GUI Example"
    scene = new Scene() {
      content = List(
        new VBox() {
          children = List(inputDisplay, button, outputDisplay)
        }
      )
    }
  }

}
```

GUI by Example

- Use the same syntax to setup the stage
- Triple nested object creation!
- Create a PrimaryStage, Scene, and VBox (Vertical Box)

```
object SampleGUI extends JFXApp {

  val inputDisplay: TextField = new TextField {
    style = "-fx-font: 18 ariel;"
  }

  val outputDisplay: TextField = new TextField {
    editable = false
    style = "-fx-font: 18 ariel;"
  }

  val button: Button = new Button {
    minWidth = 100
    minHeight = 100
    style = "-fx-font: 28 ariel;"
    text = "F to C"

  }

  this.stage = new PrimaryStage {
    title = "GUI Example"
    scene = new Scene() {
      content = List(
        new VBox() {
          children = List(inputDisplay, button, outputDisplay)
        }
      )
    }
  }

}
```

But the button doesn't do anything

Scala Functions - Detour

- Define a Scala function (not method)
  - (parameter_list) => function_body

```
(x: Int, y: Int) => x + y
```

- Can define a code block for larger functions

```
(obj: PhysicalObject, dt: Double) => {
  val newX = obj.location.x + dt * obj.velocity.x
  val newY = obj.location.y + dt * obj.velocity.y
  val newZ = 0.0.max(obj.location.z + dt * obj.velocity.z)
  new PhysicsVector(newX, newY, newZ)
}
```

Scala Functions - Detour

- Functions can be stored in variables
    - Functions are "first-class citizens" in Scala
    - Function is just another Scala type
    - Variable types must define the types of the function

```scala
val addFunction: (Int, Int) => Int = (x: Int, y: Int) => x + y
```

```scala
val computePotentialLocation: (PhysicalObject, Double) => PhysicsVector = (obj: PhysicalObject, dt: Double) => {
  val newX = obj.location.x + dt * obj.velocity.x
  val newY = obj.location.y + dt * obj.velocity.y
  val newZ = 0.0.max(obj.location.z + dt * obj.velocity.z)
  new PhysicsVector(newX, newY, newZ)
}
```

Back to Buttons

- The button class has a state variable onAction
    - onAction can be set to a function (technically an EventHandler)

```
onAction = (event: ActionEvent) => game.clickGold()
```

- Set onAction to define the behavior of a button
- Sometimes removing the ActionEvent type fixes errors (?)

```
onAction = event => buttonPressed()
```

Lecture Question

Question: Make a GUI for a number guessing game. The game should

- Store a random number from 1-100
- Allow the user to guess the number and display higher, lower, or correct according to the guess
- Display the number of guesses made

* This question will be open until midnight

Lecture Question

There is no testing or greater for this question. Submit whatever you design for your game to earn full credit