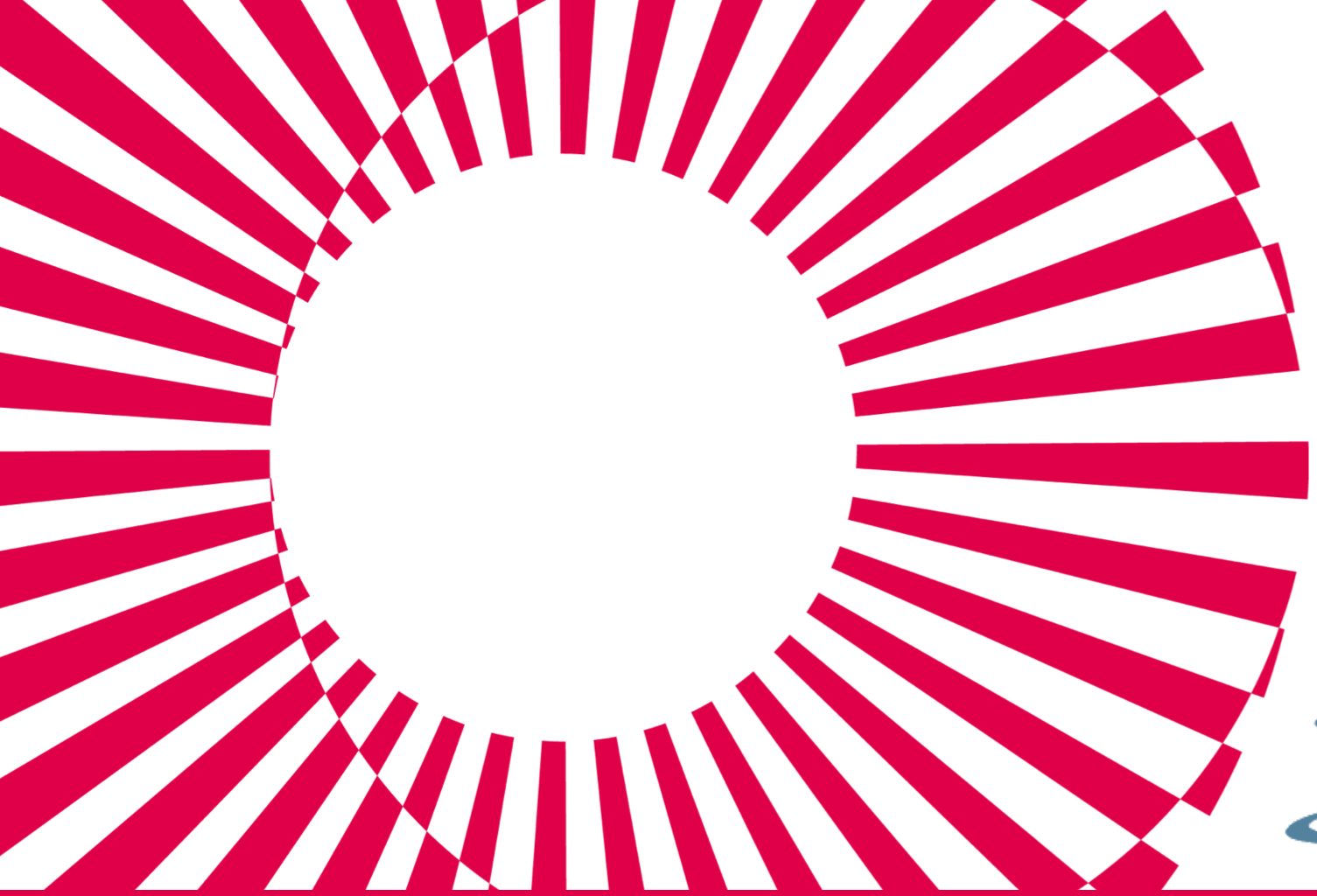


JavaFx™

UC Leuven Limburg

MOVING MINDS

Intro

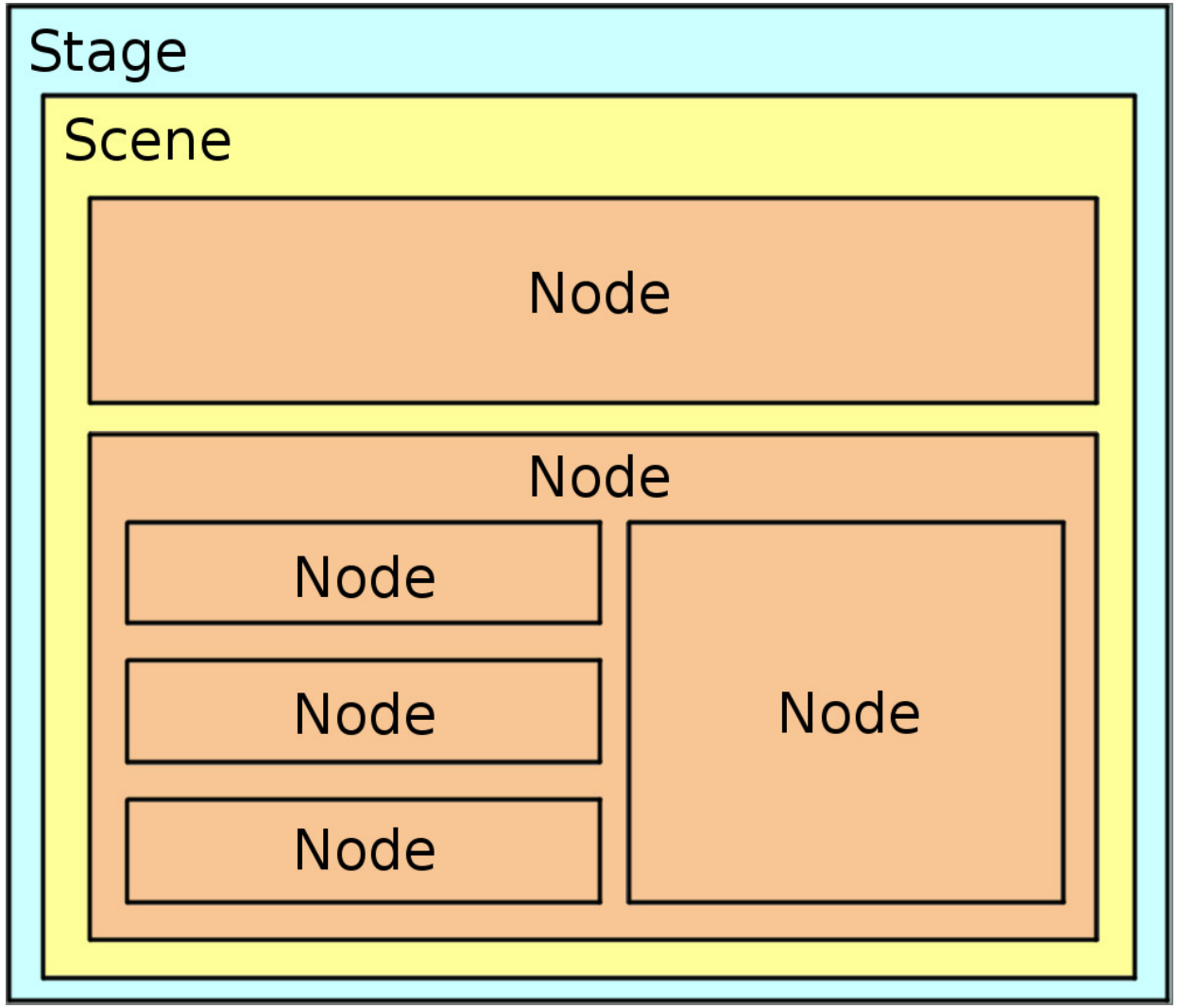
JavaFX

- The UI Toolkit for Java-Based Client Applications
- Part of the JDK 8
- Successor of Swing

- UI controls, layouts, and charts
- Accelerated 2D and 3D graphics
- Audio and video support
- Effects and animations
- HTML5 support
- Bindings, CSS, FXML, ...

Major components

- Stage
  - Main container
- Scene
  - Background for UI components
- Node
  - textbox,
  - button,
  - image view,
  - media player,
  - …



Before you begin

- Install de JavaFx - plugin voor Eclipse:
  - Kies in het menu *Help* | *Install New Software …*
  - Voeg de update-site voor de plugin toe: http://download.eclipse.org/efxclipse/updates-released/3.0.0/site
  - Kies *e(fx)clipse - install* en ga verder met de installatie

- Wanneer de installatie is voltooid, maak je geen gewoon Java project aan, maar een JavaFX-project:
  - In Eclipse, kies *File* > *New* > *Other... JavaFX* > *JavaFX Project*

Entry point: `Application`

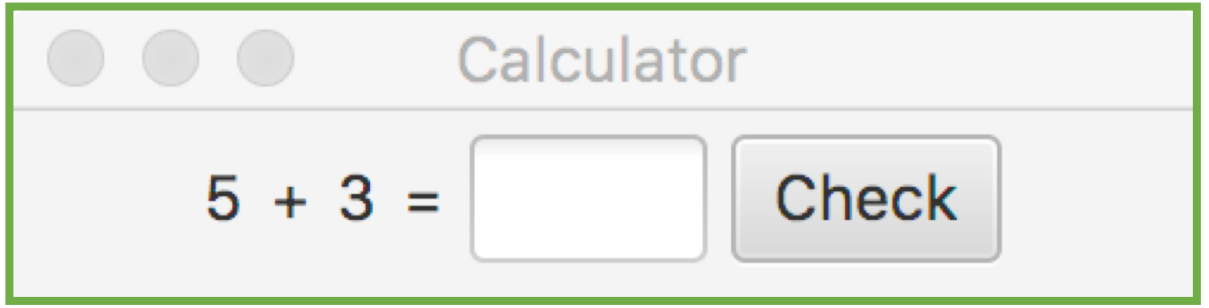
```
public class MyCounterApp extends Application {


    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        //code of your application
    }
}
```

create subclass of *Application* class

launch() internally calls the *start()* method of the Application class

start() method receives a *stage* as parameter

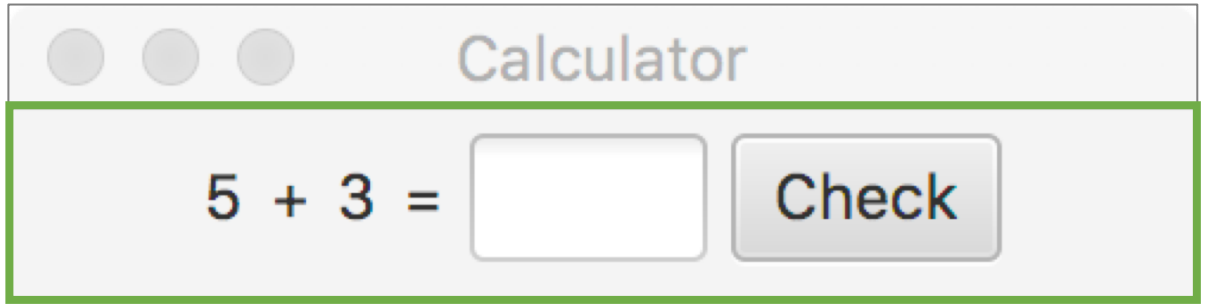
Main Window: `Stage`




```
@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle(”Calculator");
    primaryStage.setScene(mainScene);
    ...
}
```

inside the *stage* you add a *scene*

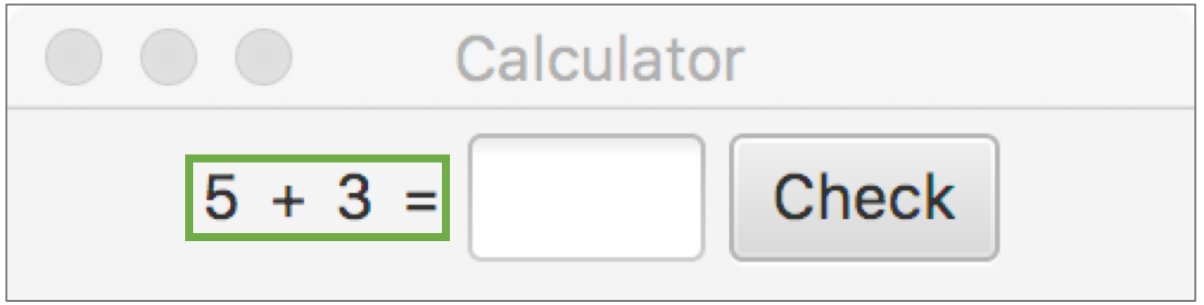
Background for nodes:
Scene




```
@Override
public void start(Stage primaryStage) {
    ...
    Scene mainScene = new Scene(root, 250, 40);
    primaryStage.setScene(mainScene);
    ...
}
```

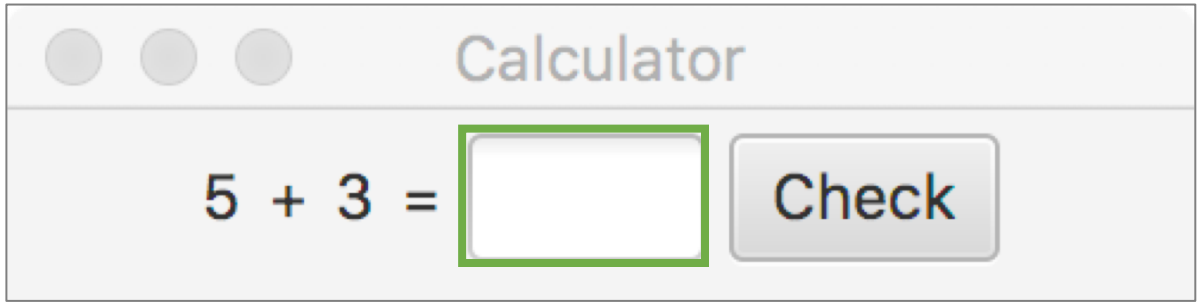
inside the *scene* the actual nodes are added

Show text: `Label` node



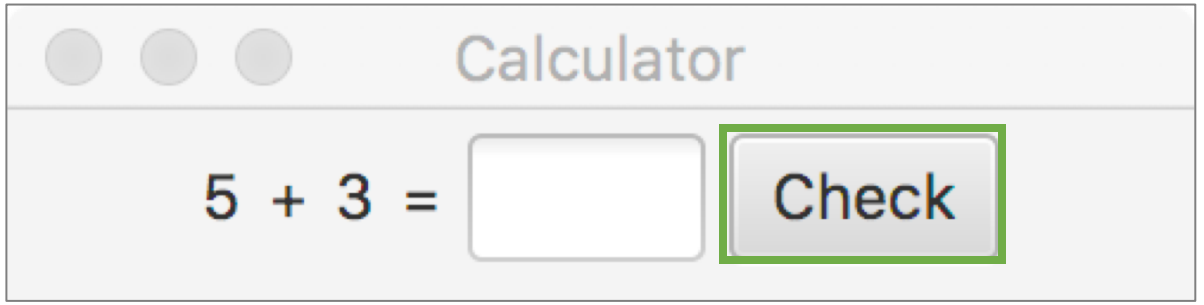

```
Label numberLabel = new Label("5 + 3 = ");
```

Enter text: `TextField` node




```
TextField numberTextField = new TextField();
```

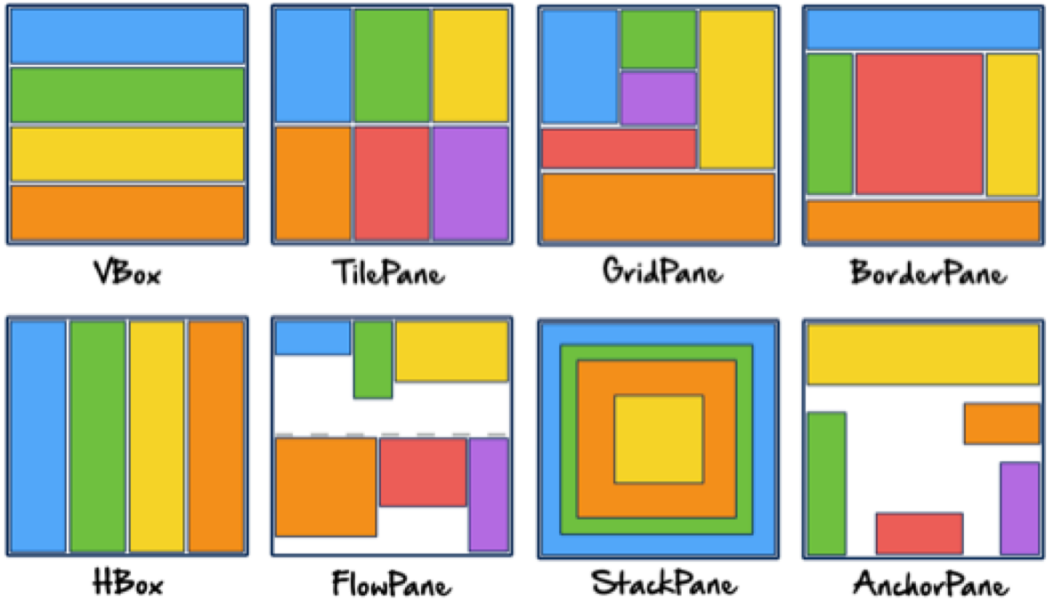
Enter text: `Button` node




```
Button setNumberButton = new Button(”Check");
```

Layout nodes: `Pane` node

- The arrangement of the nodes is called the ***Layout***
- Several predefined layouts:
  - HBox,
  - VBox,
  - Border Pane,
  - Stack Pane,
  - Flow Panel,
  - …



- The class named ***Pane*** is the base class of all the layouts in JavaFX.

Example: `FlowPane` node

- Nodes within a FlowPane
  - are laid out consecutively
  - wrap at the boundary set for the pane.

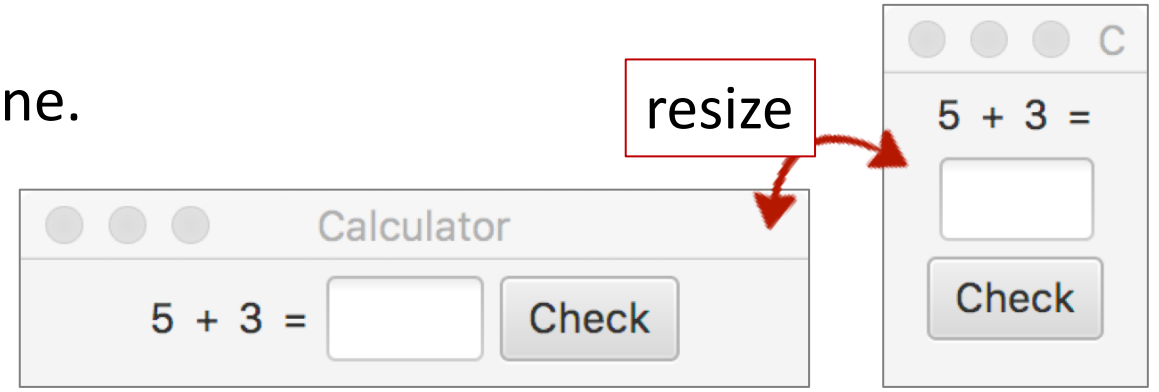
FlowPane is the parent node, other nodes will be added to it

```
FlowPane root = new FlowPane();
root.setAlignment(Pos.BASELINE_CENTER);
root.setVgap(5);
root.setHgap(5);

Scene mainScene = new Scene(root, 250, 40);
```

properties of a *pane* can be changed

Example: `FlowPane` node

- Nodes within a FlowPane
  - are laid out consecutively
  - wrap at the boundary set for the pane.



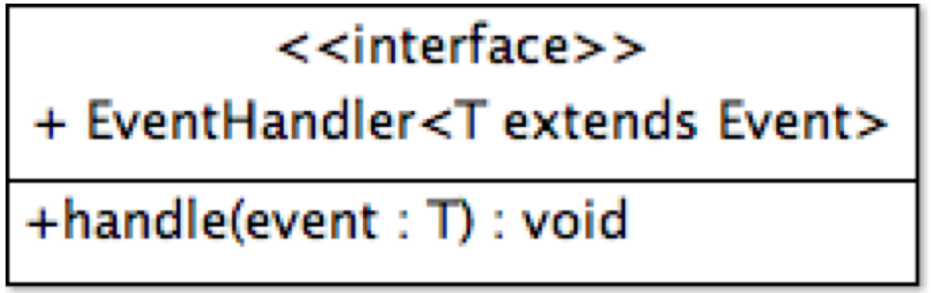

```
FlowPane root = new FlowPane();
...
root.getChildren().add(exerciseLabel);
root.getChildren().add(resultTextField);
...
```

Child nodes are added to the parent node

Show view

```
@Override
public void start(Stage primaryStage) {
    FlowPane root = new FlowPane();
    //add labels, buttons, ...

    Scene mainScene = new Scene(root, 250, 40);
    primaryStage.setScene(mainScene);

    primaryStage.show();
}
```

Perform actions

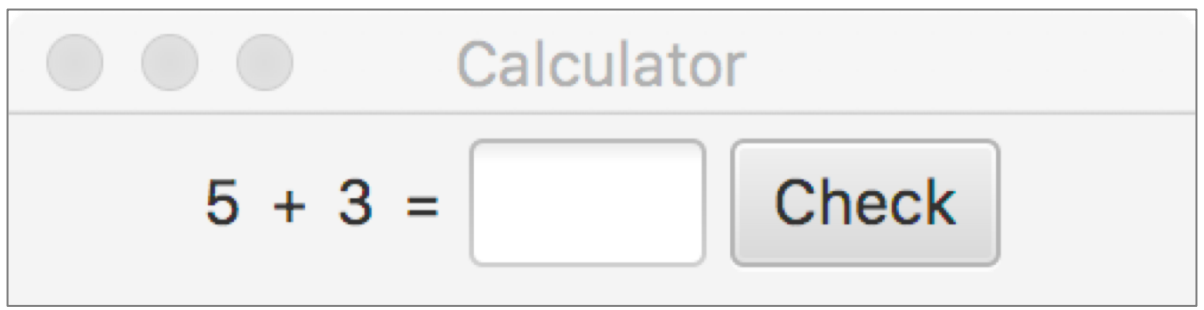


- How can you know when a user has pushed a button?
  - Java provides an interface *EventHandler*
  - Button has a method *setOnAction(eventHandler<ActionEvent> : handler)*

```
checkButton.setOnAction(new CheckResultHandler());
```

```
class CheckResultHandler implements EventHandler<ActionEvent> {
    @Override
    public void handle(ActionEvent event) {
      getCalculator().setValue(resultTextField.getText());
      ...
    }
}
```

?


Calculator
5 + 3 =
Check

If you really want to do JavaFX...

- FXML
  - XML-based language designed
    to build the user interface for JavaFX applications
- Css
  - Use CSS to create a custom look for your application
- Binding
  - allows you to synchronize the value of two properties
    so that whenever one of the properties changes,
    the value of the other property is updated automatically
- Scene Builder
  - easily layout JavaFX UI controls, charts, shapes, ...

```
<?xml version="1.0" encoding

<?language JavaScript?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox fx:id="vbox" layoutX="10.0" layoutY=
    prefWidth="300.0" spacing="10" xmlns:f
    xmlns="http://javafx.com/javafx/2.2">
    <style>
        -fx-padding: 10;
        -fx-border-style: solid inside;
        -fx-border-width: 2;
        -fx-border-insets: 5;
        -fx-border-radius: 5;
        -fx-border-color: blue;
    </style>
    <children>
        <Label fx:id="inputLbl" alignm
            cacheHint="SCALE" prefHeig
            text="Please insert Your
        <TextField fx:id="inputText"
        <Button fx:id="okBtn" alignme
            mnemonicParsing="false" t
        <Label fx:id="outputLbl" ali
            cacheHint="SCALE" prefHe
            textAlignment="LEFT" />
        <TextArea fx:id="outputText
            wrapText="true" />
    </children>
</VBox>
```