

# CREATING A JAVAFX APPLICATION

JavaFX is a set of graphics and media packages provided by Oracle. It allows developers to create applications that make use of a GUI. JavaFX has superseded Swing.

## SETTING UP THE BASIC APPLICATION STRUCTURE

Although you can generate the basic structure of a JavaFX project using a wizard, it is more useful to create an application from scratch.

Create a standard Java application in your IDE.

Add in the code annotated below.

The screenshot shows the NetBeans IDE interface with the title bar "JavaApplication1 - NetBeans IDE 8.0". The menu bar includes File, Edit, View, Navigator, Source, Refactor, Run, Debug, Profile, Team, Tool, Window, Help, and Search (Ctrl+I). The toolbar has icons for file operations like New, Open, Save, and Build. The Projects, Files, and Services panes are visible on the left. The Navigator pane is open at the bottom. The main editor window displays JavaApplication1.java:

```
1 package javaapplication1;
2
3
4 import javafx.application.Application;
5 import javafx.stage.Stage;
6
7 /**
8 * 
9 * @author Yatish
10 */
11 public class JavaApplication1 extends Application {
12
13
14     public static void main(String[] args) {
15
16         }
17     }
18
19     @Override
20     public void start(Stage primaryStage) {
21
22     }
23
24 }
25
```

Annotations in the code:

- A callout bubble points to the imports:

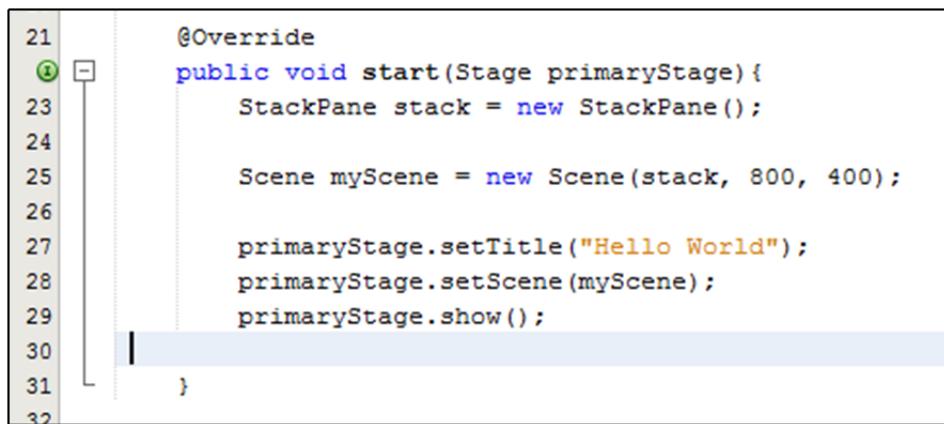
These imports allow the application to run.
- A large callout bubble points to the class definition:

Your application is a subclass of the JavaFX Application.  
This means that it will inherit all the members of its superclass. This is useful as you do not need to code, test and debug everything.
- A callout bubble points to the start method override:

We want to define our own start, so we override the member (i.e. original start method) from the

## SETTING UP THE WINDOW

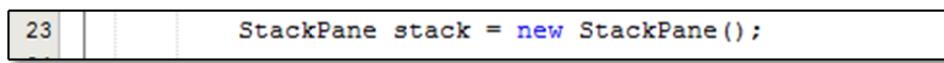
This is our new start method.



```
21  @Override
22  public void start(Stage primaryStage) {
23      StackPane stack = new StackPane();
24
25      Scene myScene = new Scene(stack, 800, 400);
26
27      primaryStage.setTitle("Hello World");
28      primaryStage.setScene(myScene);
29      primaryStage.show();
30  }
31
32 }
```

In order to remove errors you will have to import `javafx.scene.Scene` and `javafx.stage.Stage`.

## JAVAFX LAYOUTS

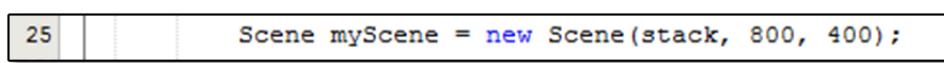


```
23 StackPane stack = new StackPane();
```

Every stage will use a root node to manage its content. Here we will use a StackPane to organise our content. There are eight built-in layouts than can be used instead of Group in JavaFX:

- 1) BorderPane used to create a window divided into top, bottom, left, right and centre areas.
- 2) HBox used to line up content horizontally.
- 3) VBox used to line up content vertically.
- 4) StackPane used to stack content items on top of each other.
- 5) GridPane used to create a tabular layout.
- 6) FlowPane used to create a layout that can flow either horizontally or vertically.
- 7) TilePane is similar to FlowPane, but all nodes are the same dimensions.
- 8) AnchorPane used to create a layout where nodes are anchored to the sides or centre of the layout.

## THE SCENE



```
25 Scene myScene = new Scene(stack, 800, 400);
```

The Scene class is the container for all content in the scene on screen. There are different constructors available. You must specify the root node. This can be a group, or one of the eight layouts discussed above. The 800 and 400 refer to the width (x) and height (y) of the scene.

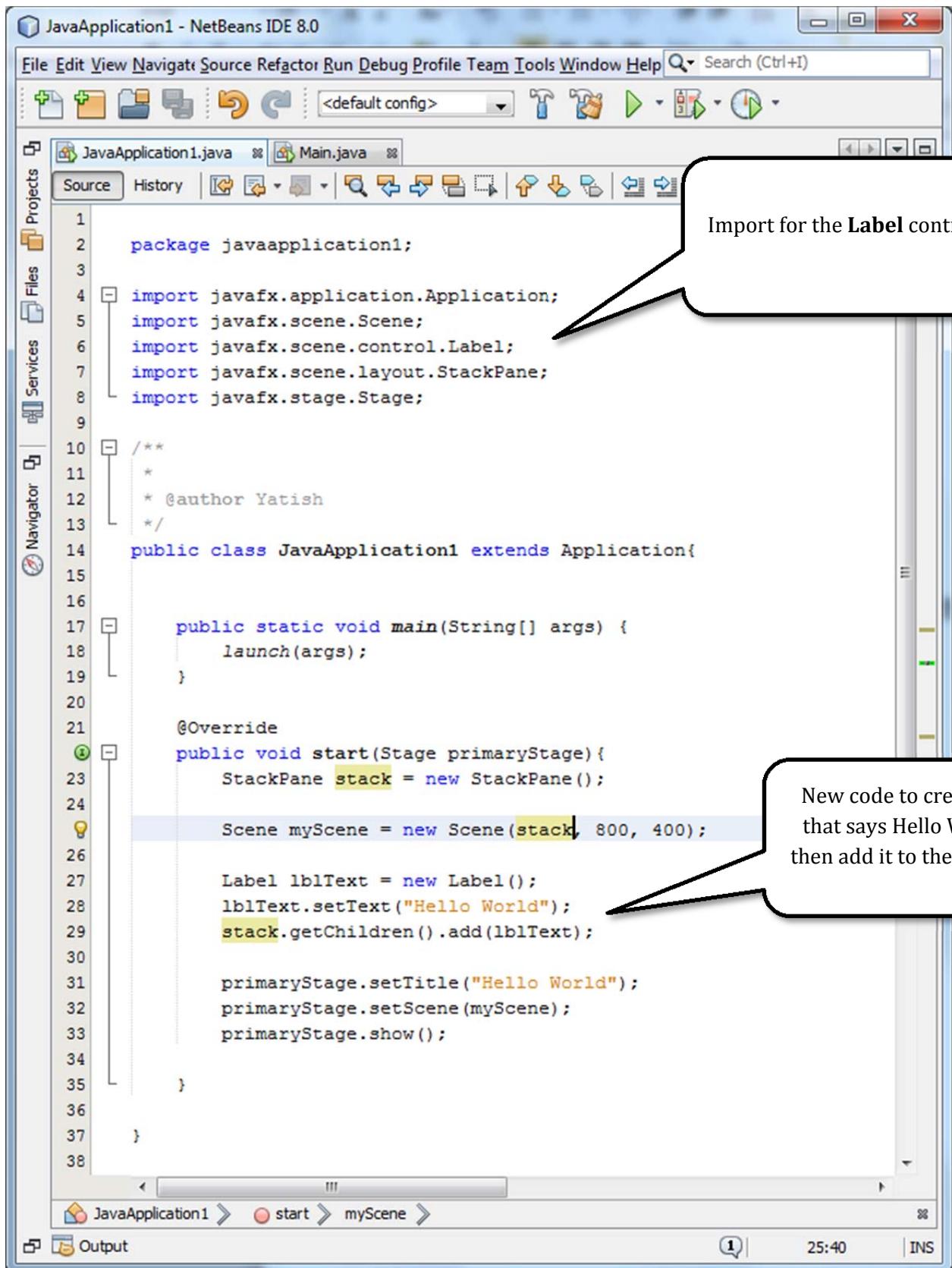
Try running the application now, nothing happens. That's because the main method needs to be told to launch the application.

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

This passes the command-line arguments when the application is run to the JavaFX launch method.

If you run the application now, you should see a blank window called Hello World.

## ADDING CONTENT



JavaApplication1 - NetBeans IDE 8.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Search (Ctrl+I)

JavaApplication1.java Main.java

Source History

```
1 package javaapplication1;
2
3
4 import javafx.application.Application;
5 import javafx.scene.Scene;
6 import javafx.scene.control.Label;
7 import javafx.scene.layout.StackPane;
8 import javafx.stage.Stage;
9
10 /**
11 *
12 * @author Yatish
13 */
14 public class JavaApplication1 extends Application{
15
16
17     public static void main(String[] args) {
18         launch(args);
19     }
20
21     @Override
22     public void start(Stage primaryStage){
23         StackPane stack = new StackPane();
24
25             Scene myScene = new Scene(stack, 800, 400);
26
27             Label lblText = new Label();
28             lblText.setText("Hello World");
29             stack.getChildren().add(lblText);
30
31             primaryStage.setTitle("Hello World");
32             primaryStage.setScene(myScene);
33             primaryStage.show();
34     }
35
36
37 }
38
```

Import for the **Label** control.



New code to create a label that says Hello World and then add it to the StackPane.

Run your application to see what happens.