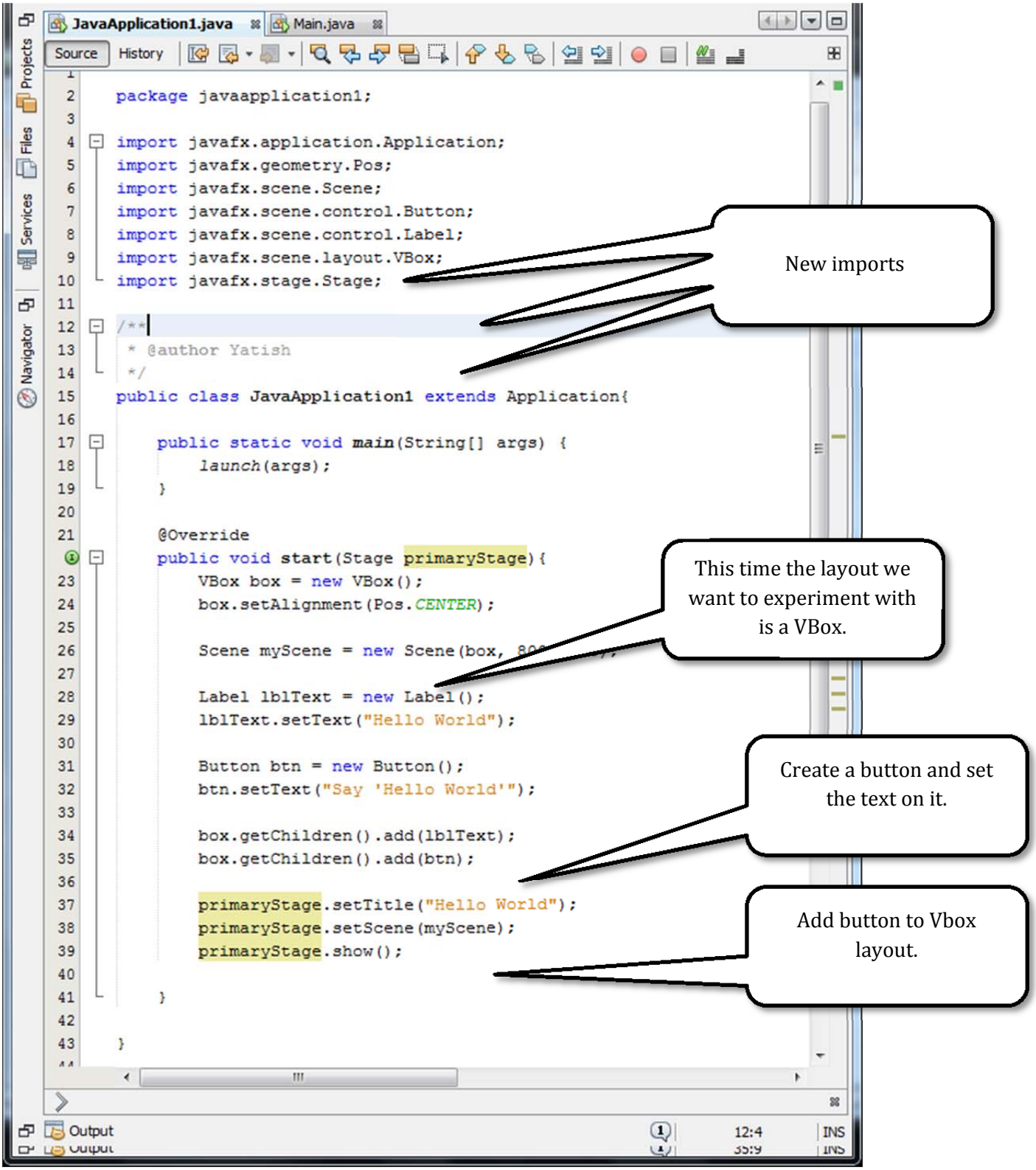
ADDING ACTION LISTENERS

SETTING UP THE BASIC WINDOW

You can modify the simple application that you made in the previous tutorial, or you can create a new one from scratch.



If you run the application, You should see the new button below the original "Hello world" label. However, nothing happens yet when you click the button.

From yatishparmar.com

ADDING AN ACTION LISTENER

```
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
```

We need to add two more imports to make the action listener work.

You can now add in the action listener for the button. The logical place to add it in your code is below where it is created and configured.

```
btn.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent event) {
        System.out.println("Hello World!");
    }
});
```

```
btn.setOnAction(new EventHandler<ActionEvent>() {
```

This creates an action listener. It defines a new anonymous inner class that implements EventHandler. An anonymous inner class is a class within a method that doesn't have a name.

```
    @Override
    public void handle(ActionEvent event) {
        System.out.println("Hello World!");
    }
```

EventHandler is an interface. This means it doesn't do anything, it provides a signature methods without implementation. It is like structure for programmers to follow when they are creating their own application. Here we override the empty handle method in EventHandler with our own functionality.

Run the application and click on the button. You should see a message in the output window in your IDE.

See if you can extend the application to output a message by adding a new label in the VBox when the button is clicked.