Applications Programming

Lab - GUIs

Program steps:

1. Create the leaves.
2. Create the branches.
3. Set the event handlers.
4. Set the scene and show the stage.

NOTE: If NetBeans fails to start or run on the lab machines, it can be fixed with the following steps:

1. Exit NetBeans
2. Open a terminal from RedHat - System Tools - Terminal
3. Type the command: `rm -rf /tmp/yourstudentnumber`
4. Restart NetBeans
5. Re-open your project

Tutor demo

Main class: `IncDec`

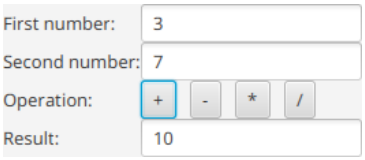
Make an application that increments and decrements a number.

Your tutor will code the solution.

Student Specification #1

Main class: `Calculator`

Make a simple calculator application.

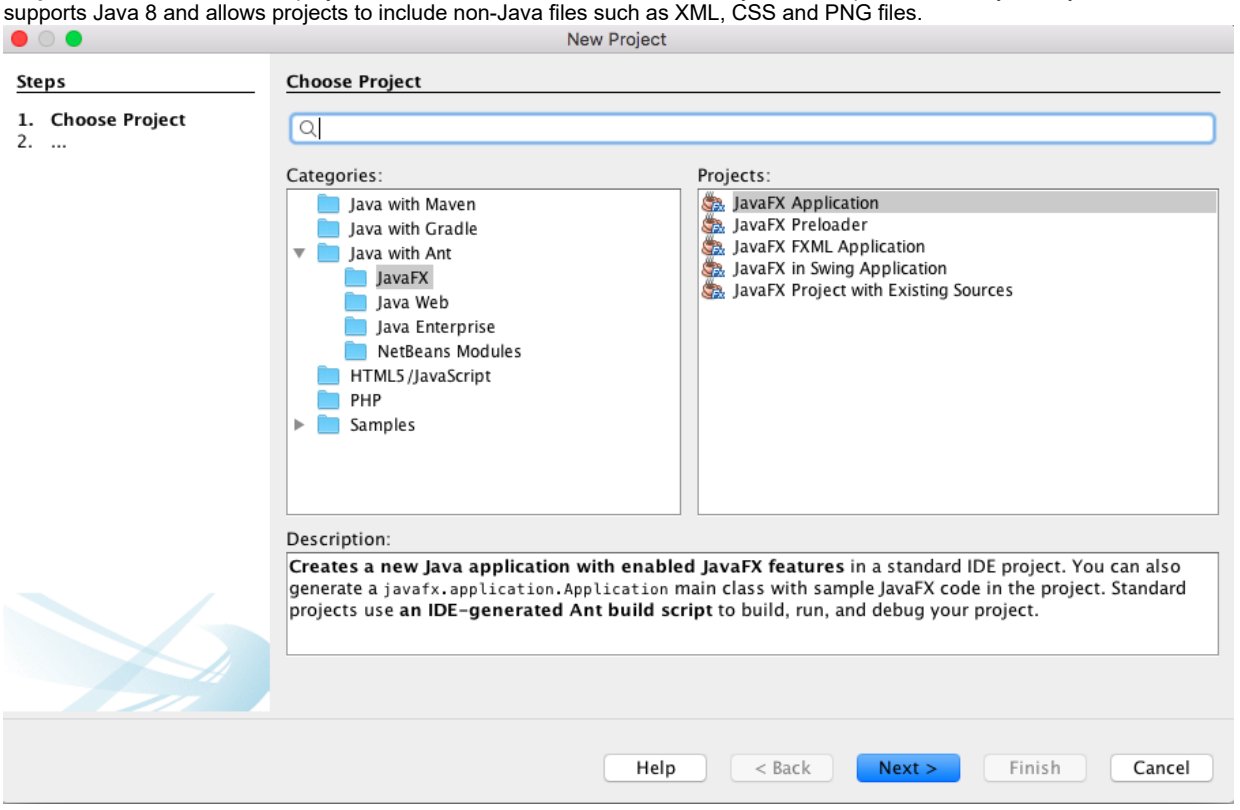


The user can input 2 numbers. The program can add, subtract, multiply or divide the two numbers and display the result.

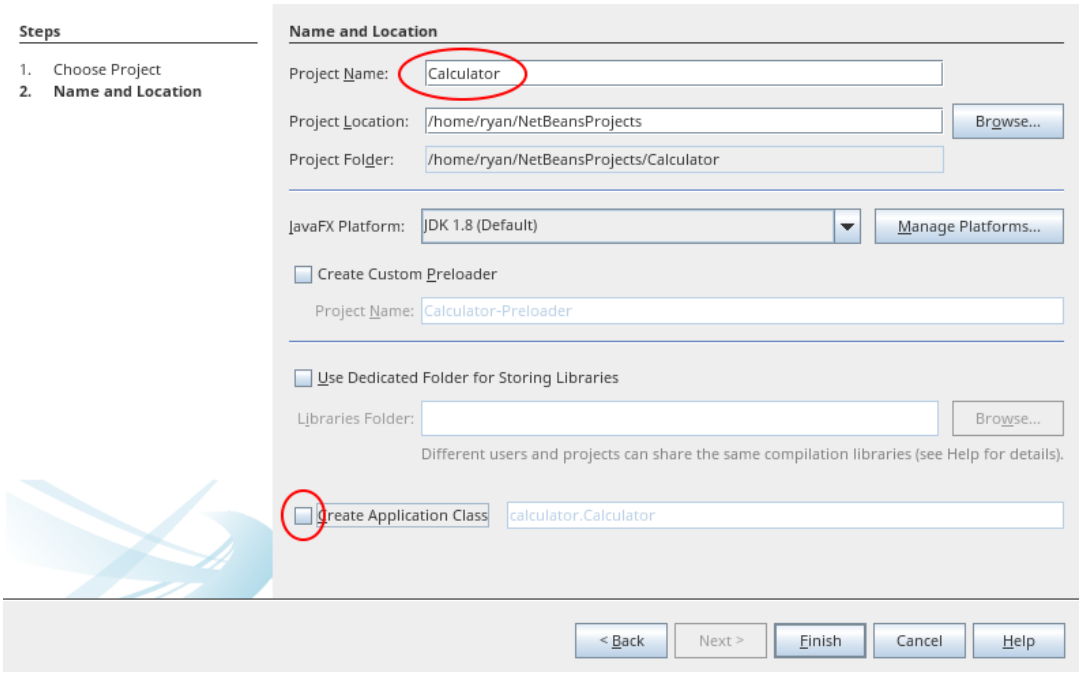
Steps

Step 1. If you didn't write down the patterns, open the pattern page. You will need them as a reference. You may also find it helpful to refer to the lecture code in the Study module for this week.

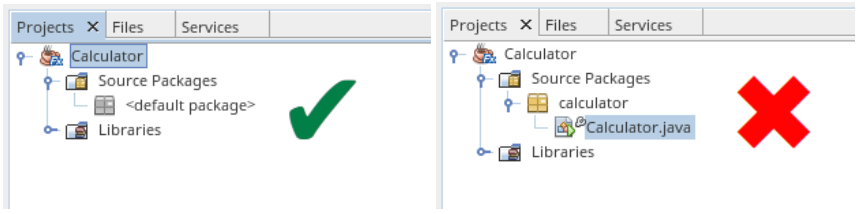
Step 2. Create a new JavaFX project in NetBeans IDE called "Calculator". If you have a preferred IDE, you may use it instead, as long as it supports Java 8 and allows projects to include non-Java files such as XML, CSS and PNG files.



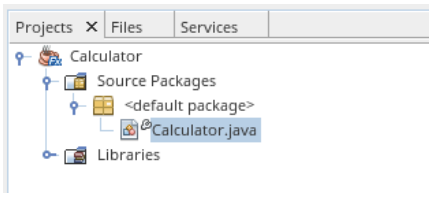
If you use NetBeans, enter the project name as shown below and deselect the "Create Application Class" checkbox.



After you create the project, double check that it was created correctly as follows:



Step 3. Create a new Java class called `Calculator` inside the default package:



Step 4. Copy and paste the following code as a template for your Calculator class:

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

    private Label firstNumberLbl;
    private TextField firstNumberTf;
    private Label secondNumberLbl;
    private TextField secondNumberTf;
    private Label operationLbl;
    private Button addBtn;
    private Button subBtn;
    private Button mulBtn;
    private Button divBtn;
    private Label resultLbl;
    private TextField resultTf;

    @Override
    public void start(Stage stage) throws Exception {
        // 1. create the leaf nodes

        // 2. create the branch node

        // 3. set the scene, show the stage
    }
}
```

Step 5. Notice that certain words in your program are underlined in red. These are compile errors which you need to fix. In this case, the issue is that you need to import these classe before you can use them. In NetBeans, you can automatically fix most compile errors by clicking on the code that is underlined in red and pressing Alt-Enter. Using this technique, import all the required classes. **HOWEVER**, make sure you import classes from the javafx package, **NOT** the awt package.

Correct	Incorrect
`import javafx.scene.control.Button;`	`import java.awt.Button;`

Step 6. Write some code to create the leaf nodes. Place this code after `// 1. create the leaf nodes`

Step 7. For the first version of this application, you will create the simpler screen layout shown below. To do this, create a `VBox` branch node with 10 pixels of spacing and add all of the leaf nodes to it:

Step 8. Set the scene and show the stage.
Use your VBox as the root node of the scene. Run your program to see if the GUI looks as in the screenshot.

AT THIS POINT, run your code to see if you're on the right track.

Step 9. Define getters/setters for your text fields that convert between String and int. For this application, you only need to "get" the first two numbers and "set" the result:

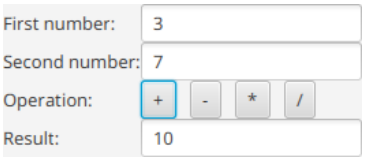
```
public class Calculator extends Application {
    public int getFirstNumber() { ... }
    public int getSecondNumber() { ... }
    public void setResult(int value) { ... }
}
```

Step 10. Define the event handlers. e.g. addBtn.setOnAction(...); for the 4 buttons. You need to learn all the different techniques, so:

1. Define an event handler for addBtn using an *inner class* named `AddHandler`. It should set the result to: the first number + the second number.
2. Define an event handler for subBtn using an *anonymous inner class*. It should set the result to: the first number - the second number.
3. Define an event handler for mulBtn using a *lambda expression*. It should set the result to: the first number * the second number.
4. Define an event handler for divBtn using a *lambda expression*. It should set the result to: the first number / the second number.

Test that your application works in NetBeans.

Step 11. Finally, modify your GUI layout to use a GridPane as shown in the following screenshot:

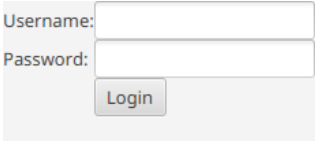


Step 12. Add a title of "xxx(zzz)'s Calculator " at the top of the GridPane. "xxx" the student name, "zzz" is the student's ID. Then submit the finished application to ED!

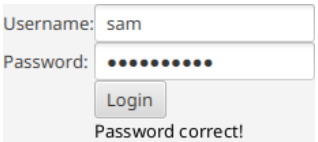
Student Specification #2

Main class: `LoginForm`

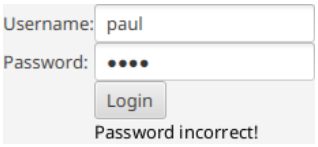
In the same project, create another class called `LoginForm`. This application should show the following scene on the primary stage with the stage title "Login":



If the user inputs username "sam" and password "mypassword" and then presses the Login button, you show the following message:



If the user inputs any other username or password and then presses the Login button, you show you show the following message:



Hint: The message is not a Label. Labels should only appear before form inputs/controls to label the value that the user inputs. The message is a `Text`. Create it with `resultTxt = new Text()`. The Text node starts off empty, showing no message. When you want to display a message, you call the `setText` method on the text.

Hint: The `LoginForm` class should also be created in the default package alongside your existing Calculator class. In NetBeans, you can run the LoginForm class by right-clicking on it and selecting "Run File".

Add a title of "xxx(zzz)'s Login" at the top of the GridPane. "xxx" the student name, "zzz" is the student's ID. Submit your code to ED.