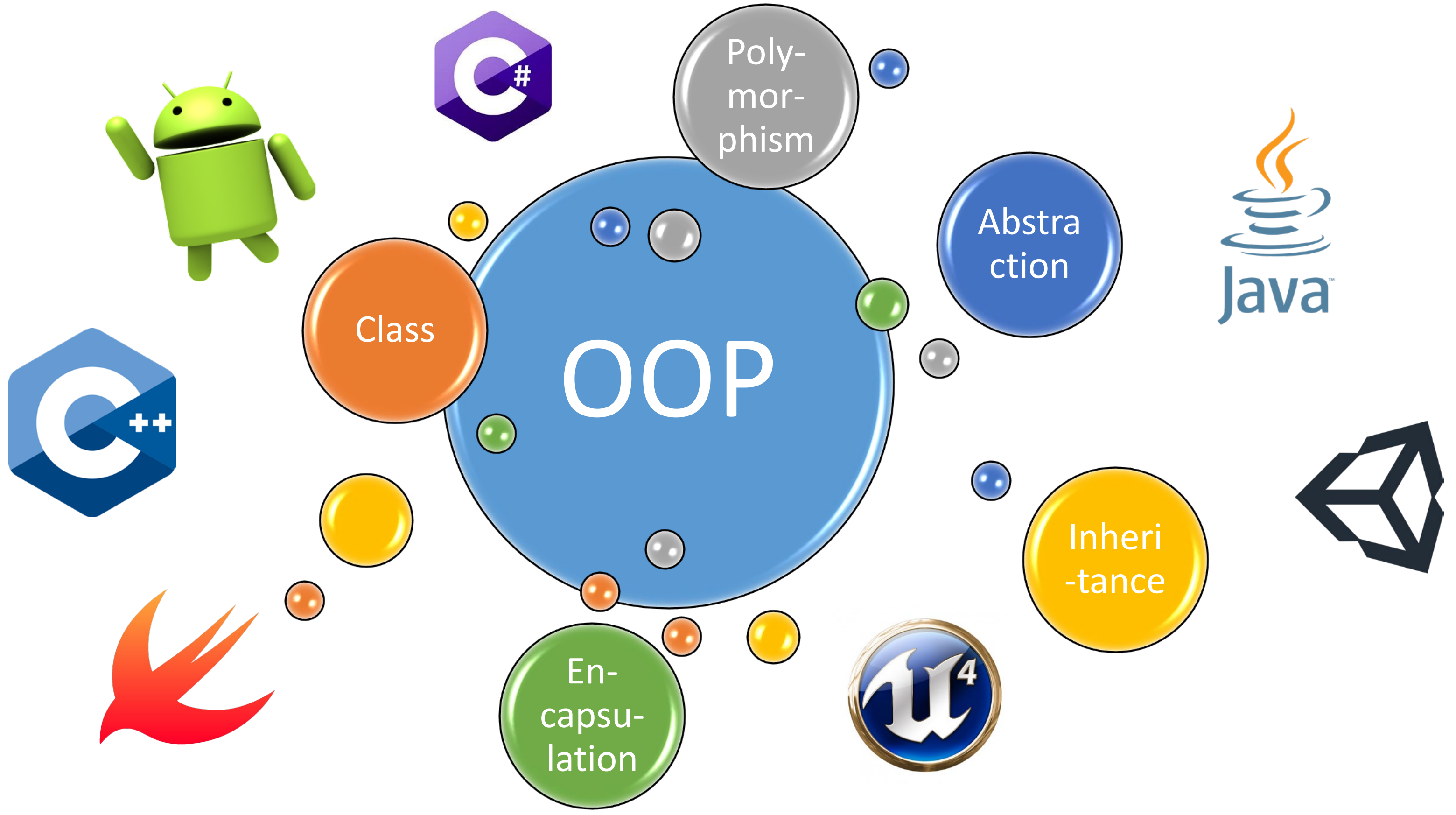
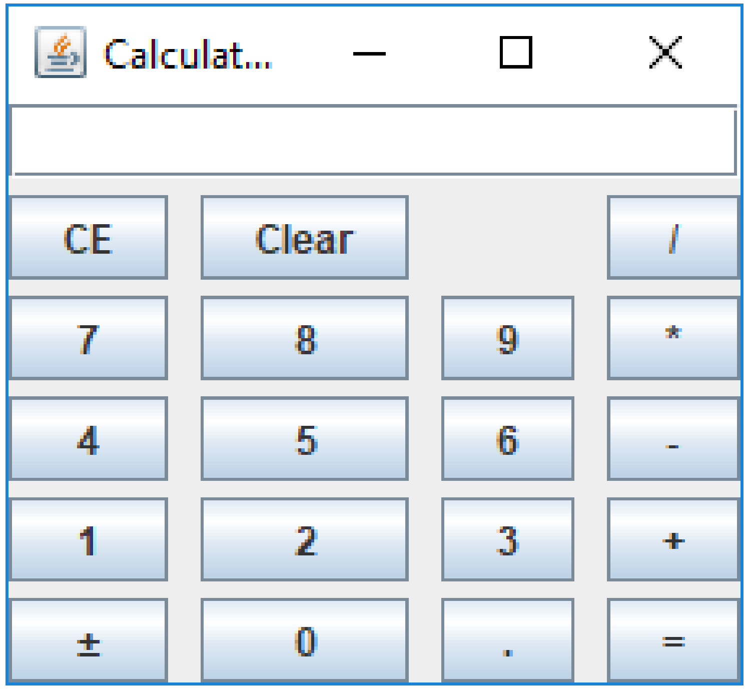


Homework 3 - Basic Calculator

Kuan-Ting Lai
2023/3/25

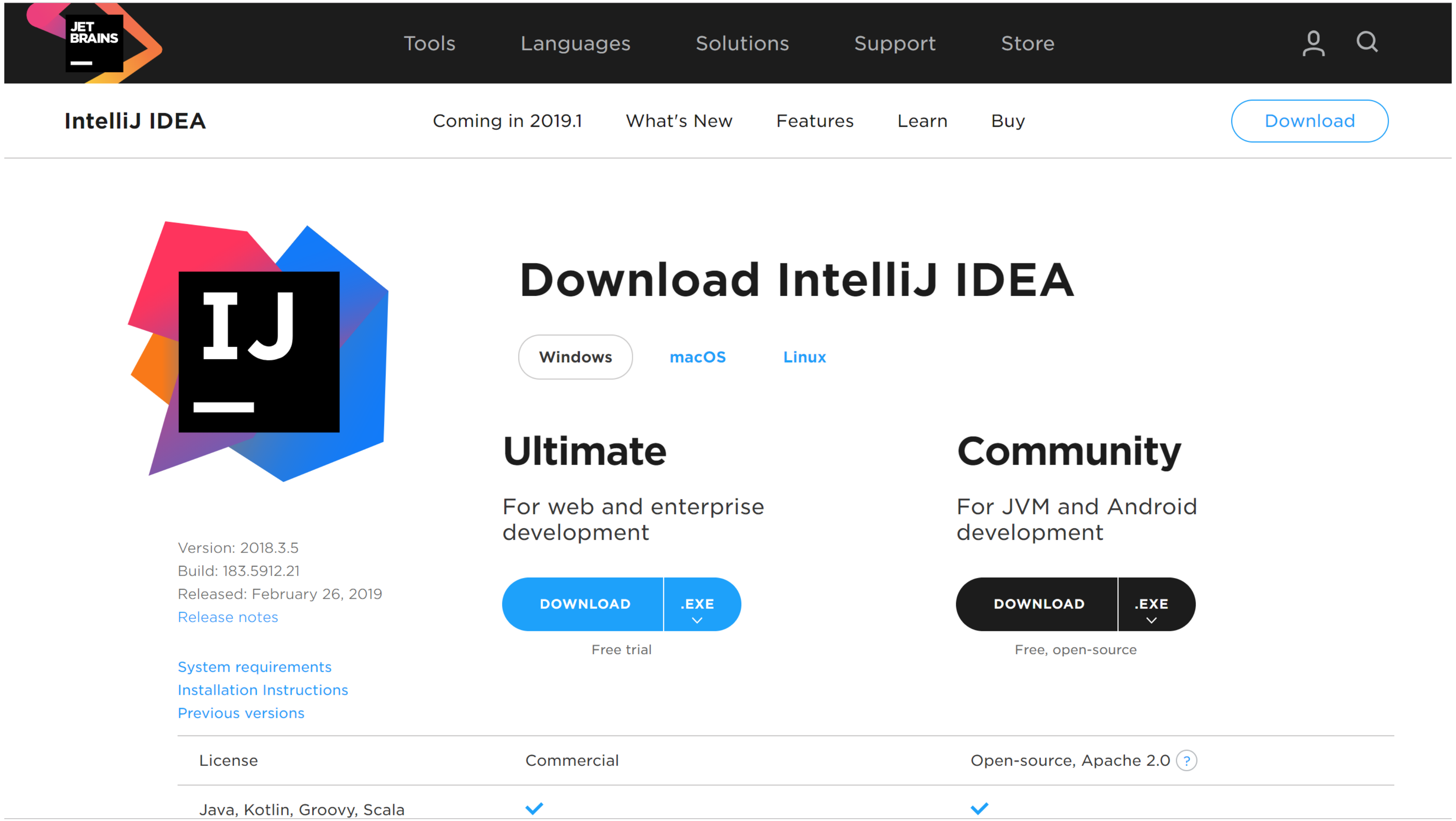
Building a Basic Calculator

- Create a basic calculator using Java Swing
- Implement basic functions:
  - Addition (+)
  - Subtraction (–)
  - Multiplication (*)
  - Division (/)
  - Minus-plus (±)
  - Clear (C)
  - Cancel Entry (CE)



Building Calculator using IntelliJ

- Install IntelliJ Community



Tools Languages Solutions Support Store

IntelliJ IDEA Coming in 2019.1 What's New Features Learn Buy Download

Download IntelliJ IDEA

Windows macOS Linux

Ultimate

For web and enterprise development

DOWNLOAD .EXE

Free trial

Community

For JVM and Android development

DOWNLOAD .EXE

Free, open-source

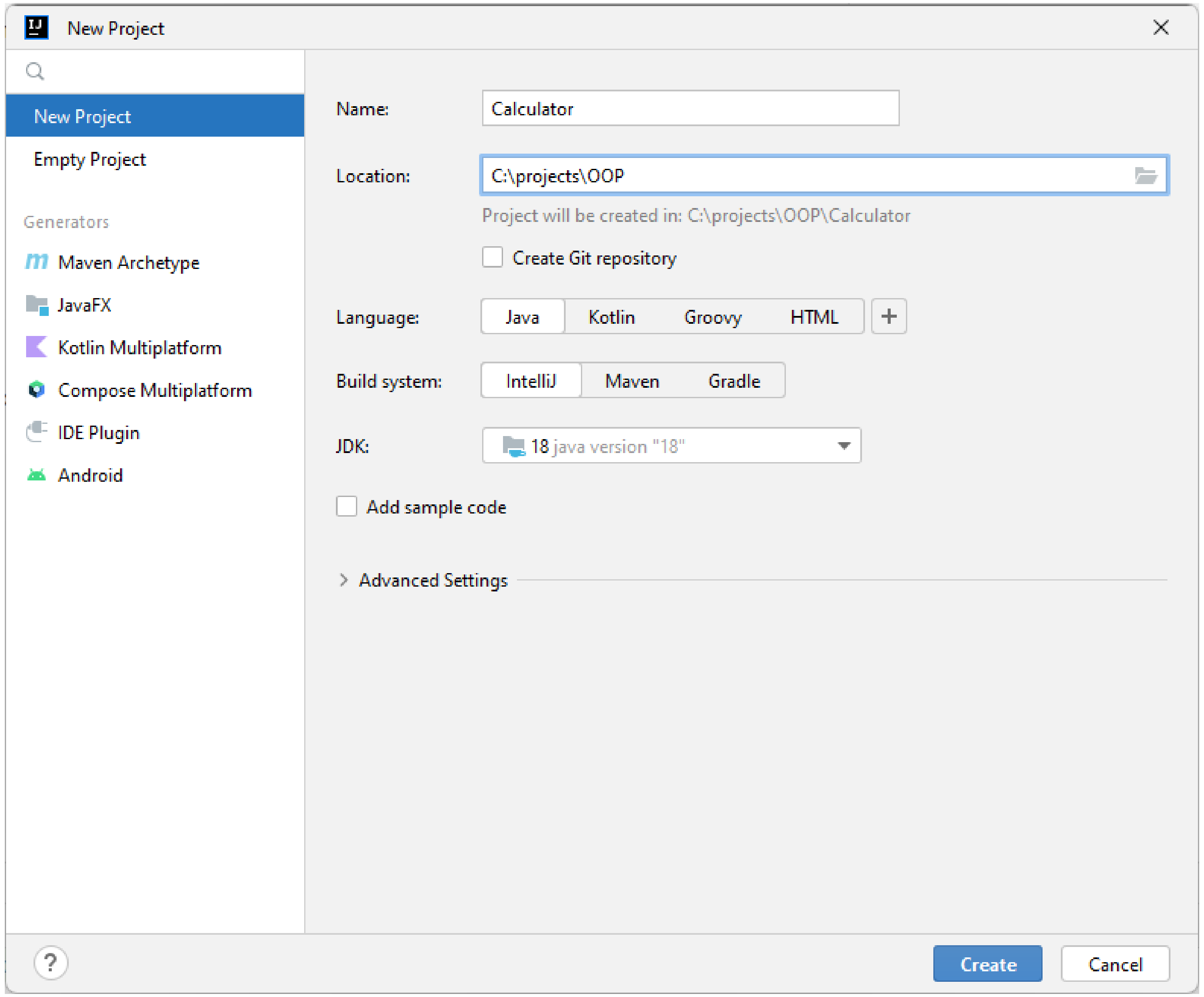
Version: 2018.3.5
Build: 183.5912.21
Released: February 26, 2019
Release notes

System requirements
Installation Instructions
Previous versions

License	Commercial	Open-source, Apache 2.0 ?
Java, Kotlin, Groovy, Scala	✓	✓

Create a Java Project

- Enter project name "Calculator"
- Select location
- Press "Create"



Name: Calculator

Location: C:\projects\OOP

Project will be created in: C:\projects\OOP\Calculator

☐ Create Git repository

Language: Java

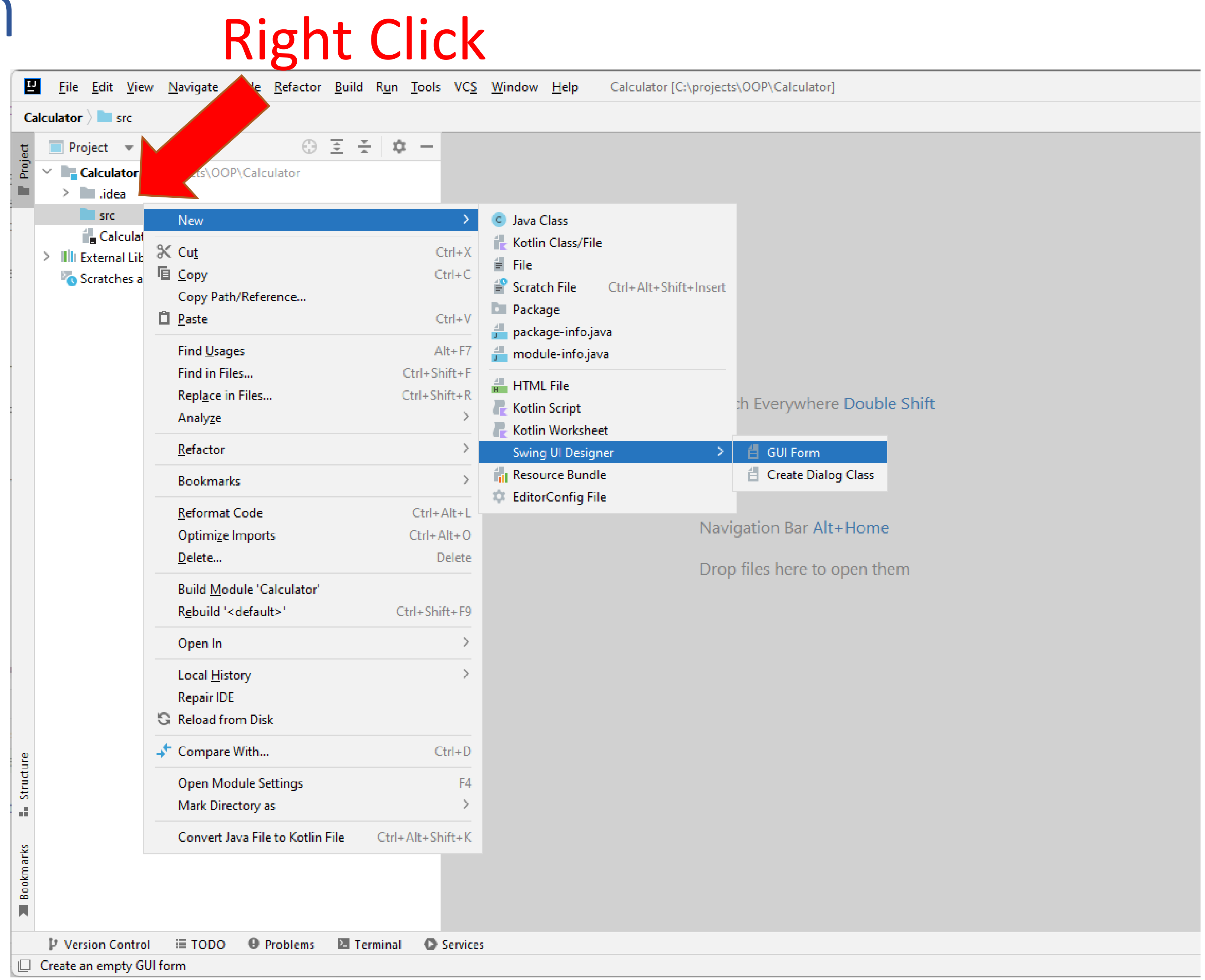
Build system: IntelliJ

JDK: 18 java version "18"

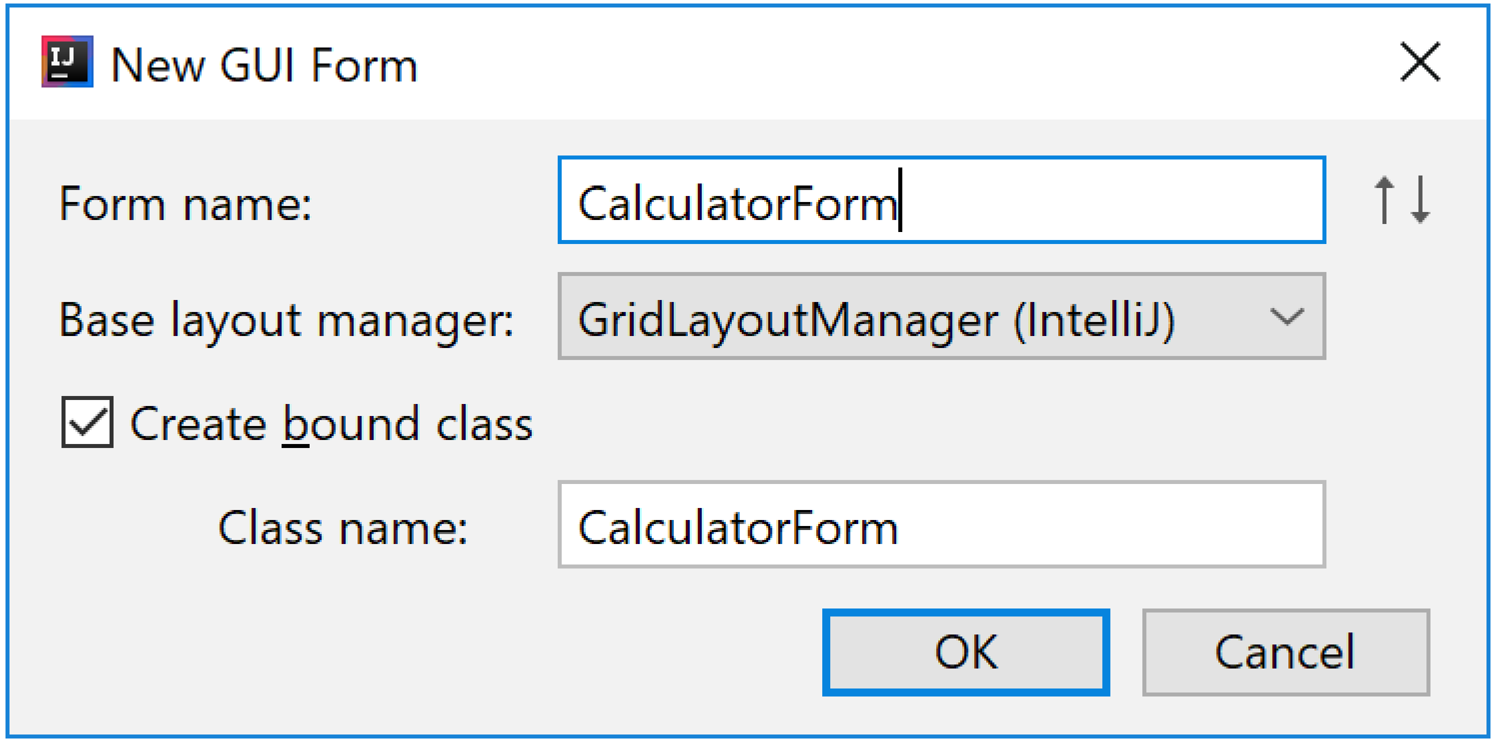
☐ Add sample code

Add GUI Form

- Right click on "src" folder
- On the context menu, select Swing UI Designer -> GUI Form



Name Your Form “CalculatorForm”


New GUI Form

Form name: CalculatorForm

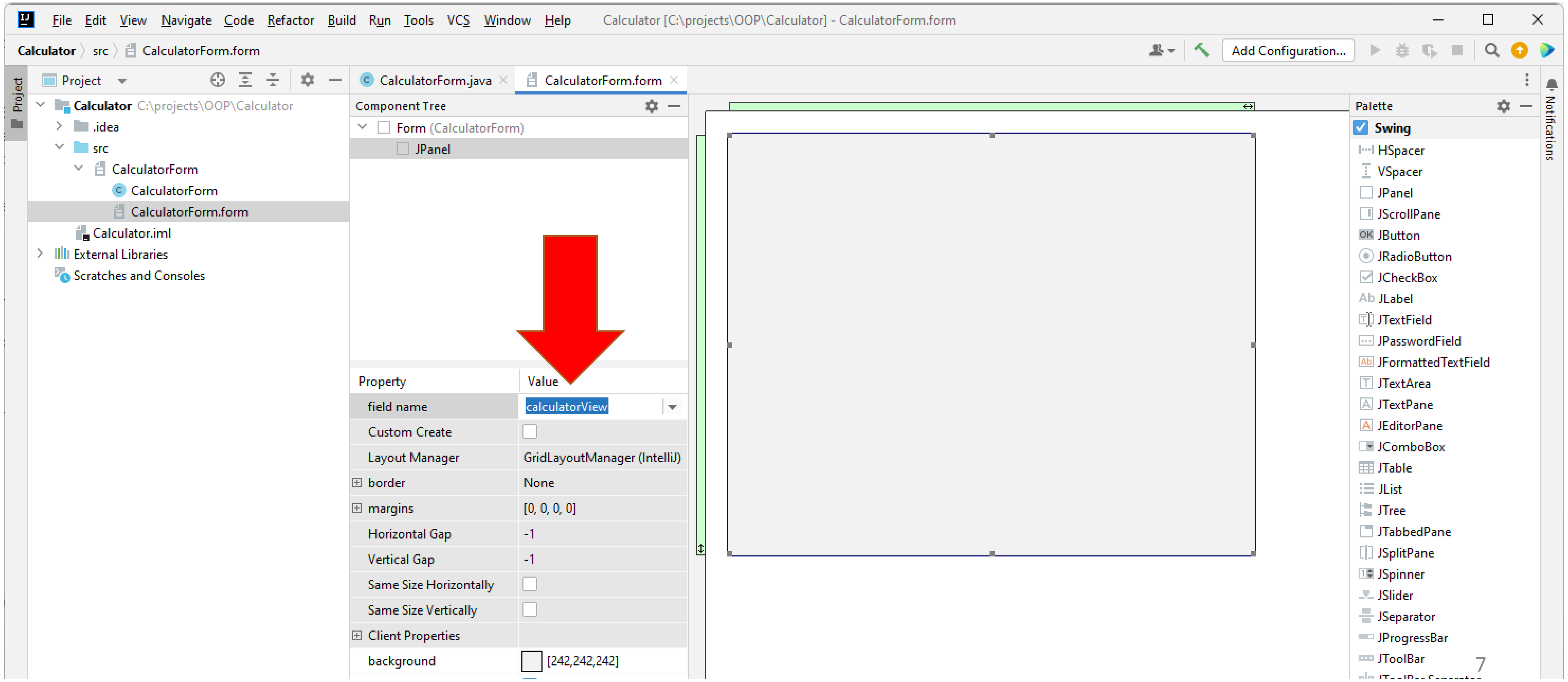
Base layout manager: GridLayoutManager (IntelliJ)

☑ Create bound class

Class name: CalculatorForm

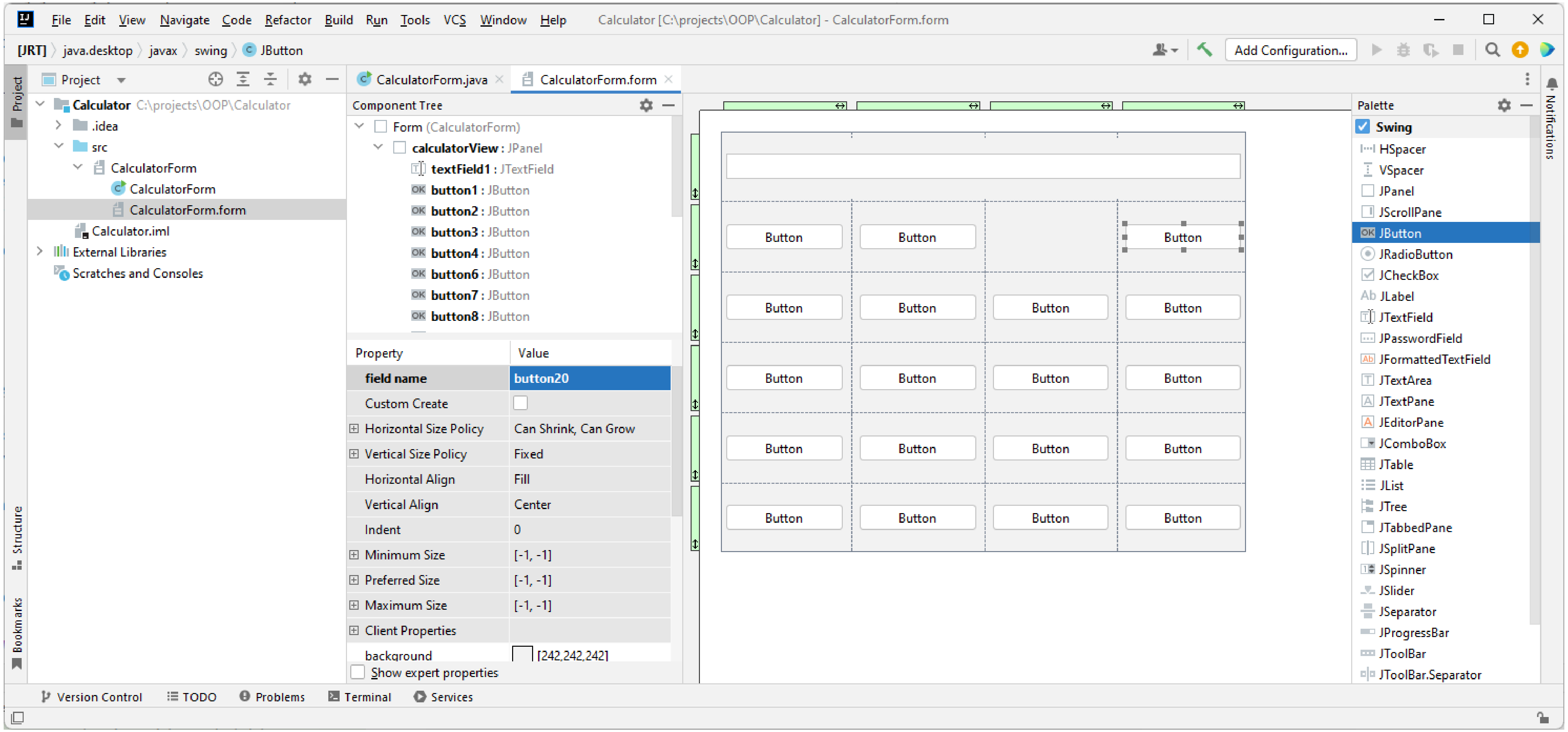
Set JPanel's Name

- Select the JPanel in the Component tree of the form view and update the field name property to calculatorView.



Property	Value
field name	calculatorView
Custom Create	☐
Layout Manager	GridLayoutManager (IntelliJ)
border	None
margins	[0, 0, 0, 0]
Horizontal Gap	-1
Vertical Gap	-1
Same Size Horizontally	☐
Same Size Vertically	☐
Client Properties	
background	[242,242,242]

Put a JTextField and Jbuttons on the Form



Property	Value
field name	button20
Custom Create	☐
Horizontal Size Policy	Can Shrink, Can Grow
Vertical Size Policy	Fixed
Horizontal Align	Fill
Vertical Align	Center
Indent	0
Minimum Size	[-1, -1]
Preferred Size	[-1, -1]
Maximum Size	[-1, -1]
Client Properties	
background	[242,242,242]

☐ Show expert properties

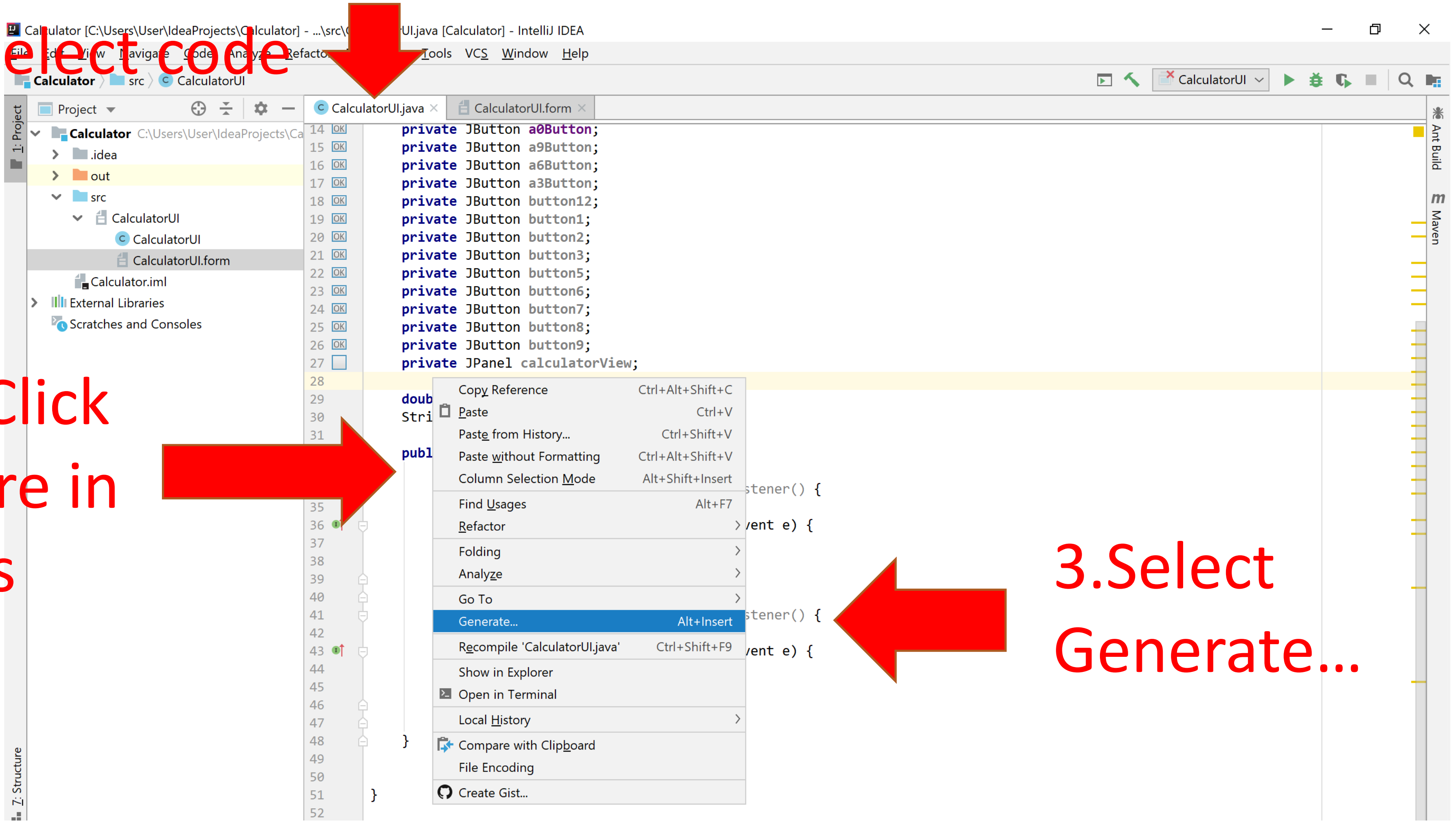
Generate main() Code

- In the code editor of Calculator.java file select -> Generate... -> Form main()

1.Select code

2.Right Click Anywhere in the Class

3.Select Generate...

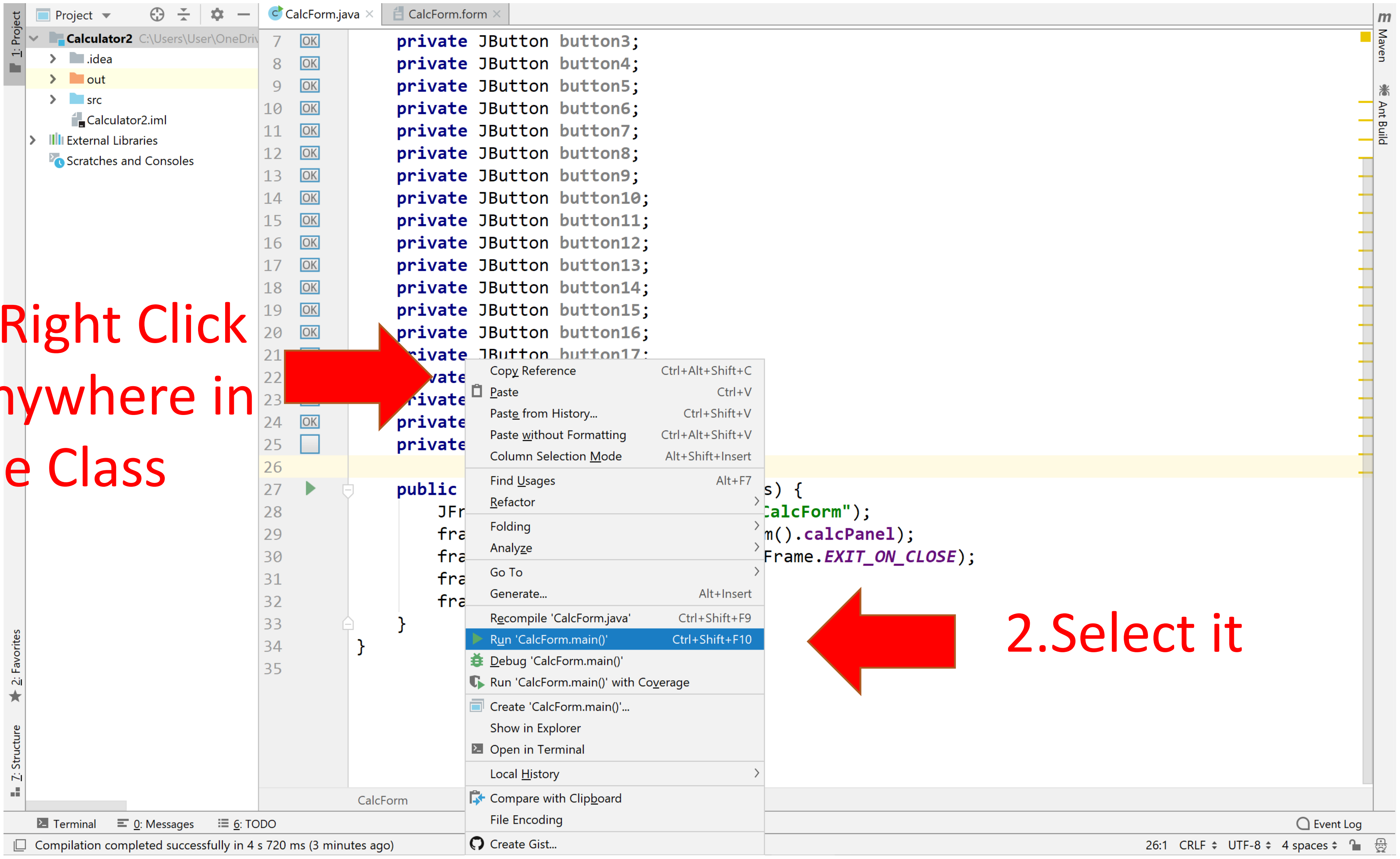

Calculator [C:\Users\User\IdeaProjects\Calculator] - ...\src\CalculatorUI.java [Calculator] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Tools VCS Window Help
Calculator src CalculatorUI
Project
Calculator C:\Users\User\IdeaProjects\Ca
.idea
out
src
CalculatorUI
CalculatorUI
CalculatorUI.form
Calculator.iml
External Libraries
Scratches and Consoles
CalculatorUI.java
CalculatorUI.form
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Paste from History... Ctrl+Shift+V
Paste without Formatting Ctrl+Alt+Shift+V
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Analyze
Go To
Generate... Alt+Insert
Recompile 'CalculatorUI.java' Ctrl+Shift+F9
Show in Explorer
Open in Terminal
Local History
Compare with Clipboard
File Encoding
Create Gist...

```
    private JButton a0Button;
    private JButton a9Button;
    private JButton a6Button;
    private JButton a3Button;
    private JButton button12;
    private JButton button1;
    private JButton button2;
    private JButton button3;
    private JButton button5;
    private JButton button6;
    private JButton button7;
    private JButton button8;
    private JButton button9;
    private JPanel calculatorView;

    doub
    Stri

    publ
                                    stener() {
                                    vent e) {

                                    stener() {
                                    vent e) {

    }

}
```

Run Main()

1.Right Click Anywhere in the Class

2.Select it

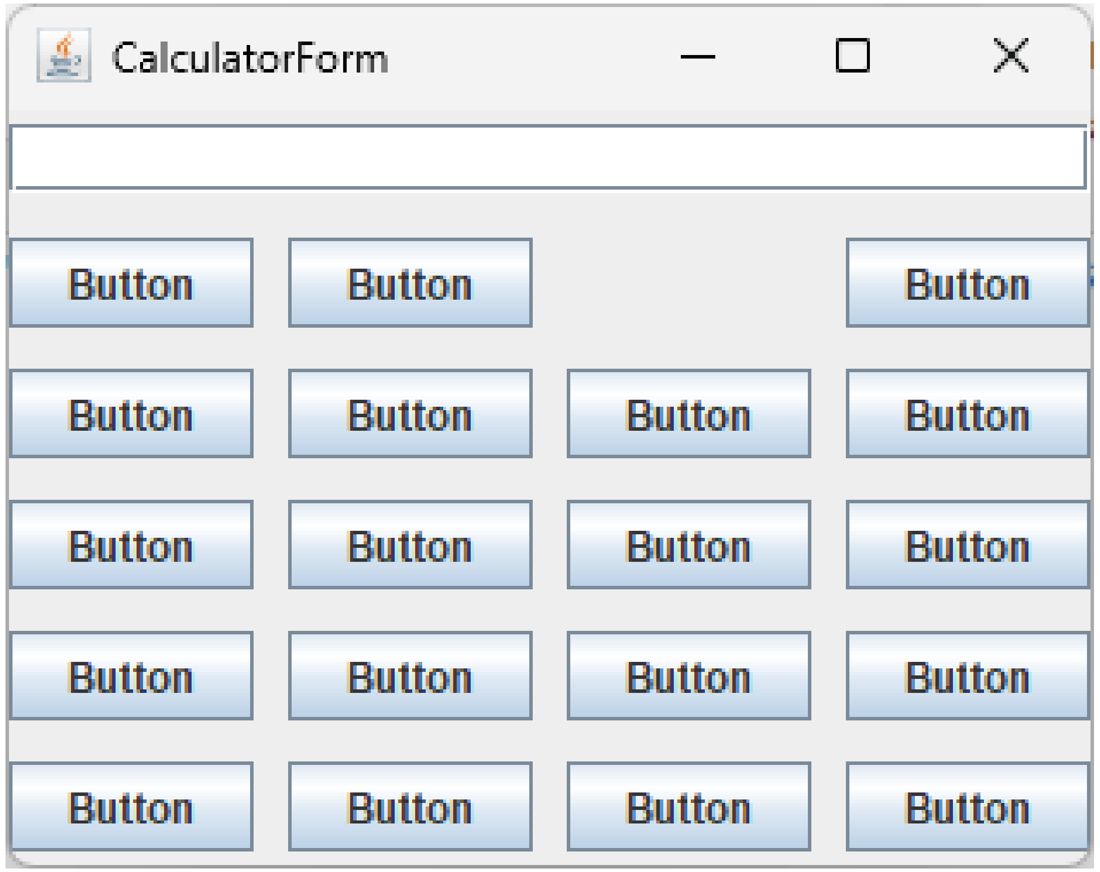


```
    private JButton button3;
    private JButton button4;
    private JButton button5;
    private JButton button6;
    private JButton button7;
    private JButton button8;
    private JButton button9;
    private JButton button10;
    private JButton button11;
    private JButton button12;
    private JButton button13;
    private JButton button14;
    private JButton button15;
    private JButton button16;
    private JButton button17;
    private
    private
    private
    private

    public                                  s) {
        JFr                                 CalcForm");
        fra                                 m().calcPanel);
        fra                                 Frame.EXIT_ON_CLOSE);
        fra
        fra
    }
}
```

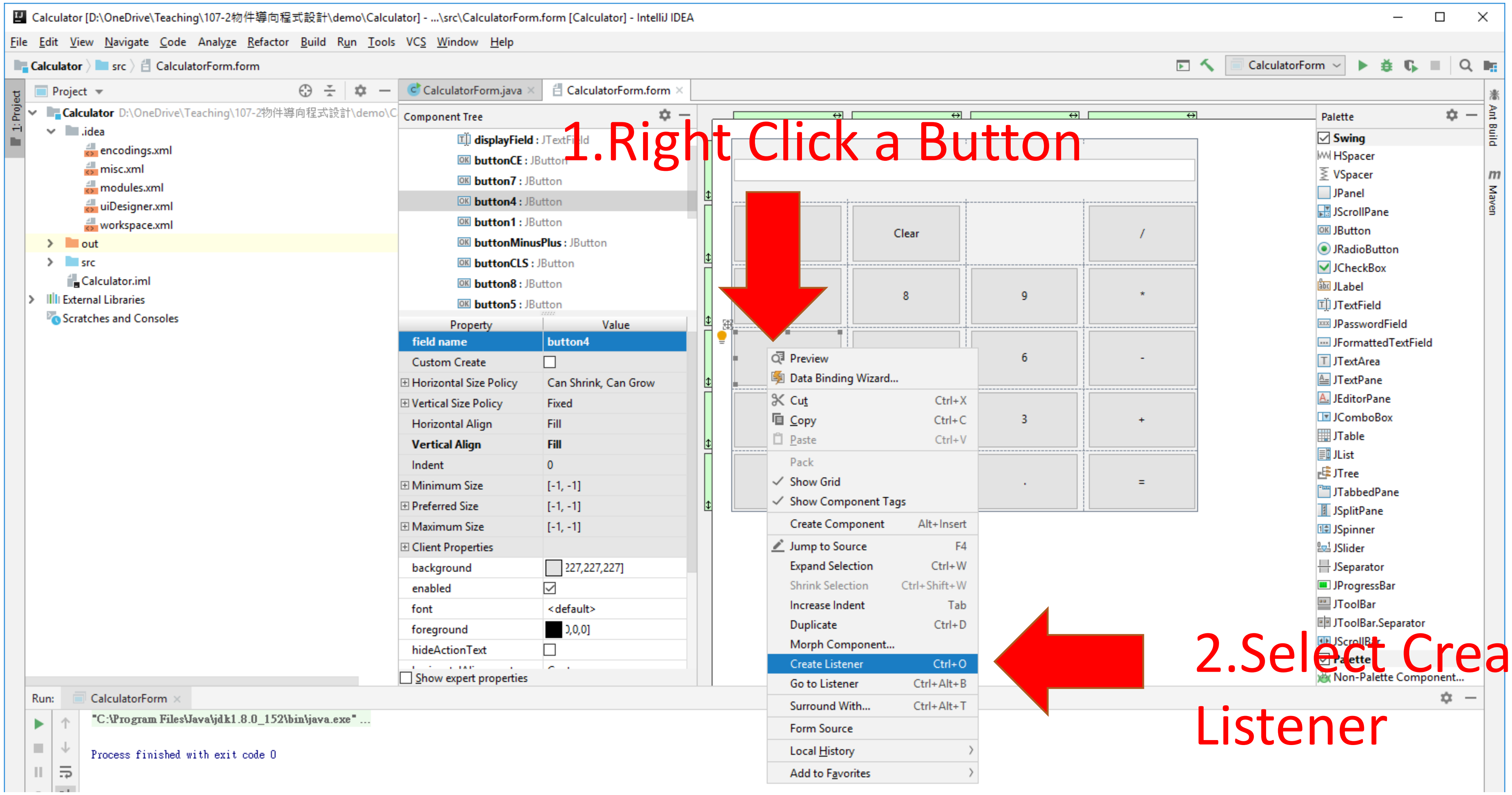
Compilation completed successfully in 4 s 720 ms (3 minutes ago)

Congratulations! Your Calculator prototype

- Next, change the field names of fields and buttons
- Give each button a meaningful name



Create Action Listeners of Buttons



1.Right Click a Button

2.Select Create Listener

Property	Value
field name	button4
Custom Create	☐
Horizontal Size Policy	Can Shrink, Can Grow
Vertical Size Policy	Fixed
Horizontal Align	Fill
Vertical Align	Fill
Indent	0
Minimum Size	[-1, -1]
Preferred Size	[-1, -1]
Maximum Size	[-1, -1]
Client Properties	
background	[227,227,227]
enabled	✓
font	<default>
foreground	[0,0,0]
hideActionText	☐

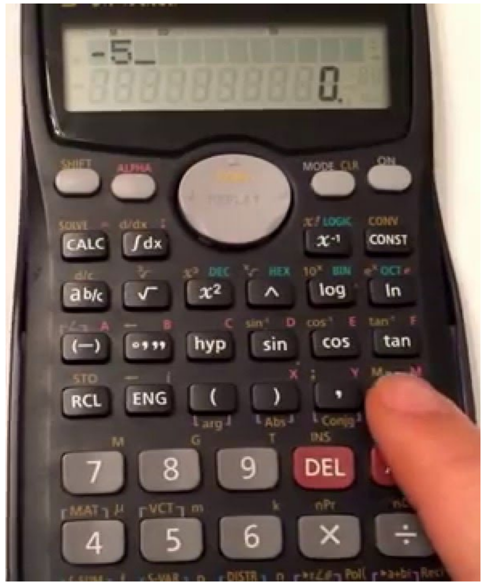
```
"C:\Program Files\Java\jdk1.8.0_152\bin\java.exe" ...

Process finished with exit code 0
```

Enter Your Code in ActionListener

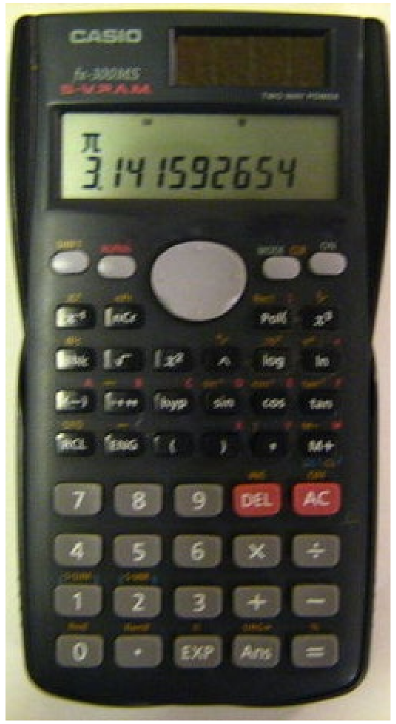
```
public class CalculatorForm {
    private JTextField displayField;
    private JPanel CalcPanel;
    private JButton buttonCE;
    private JButton button0;
    ……
    ……
    public CalculatorForm() {
        button0.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {

            }
        });
     ……
```

Enter Your Own Code Here

Two Modes in Calculator



Entering digits



Show results (temporary or final)

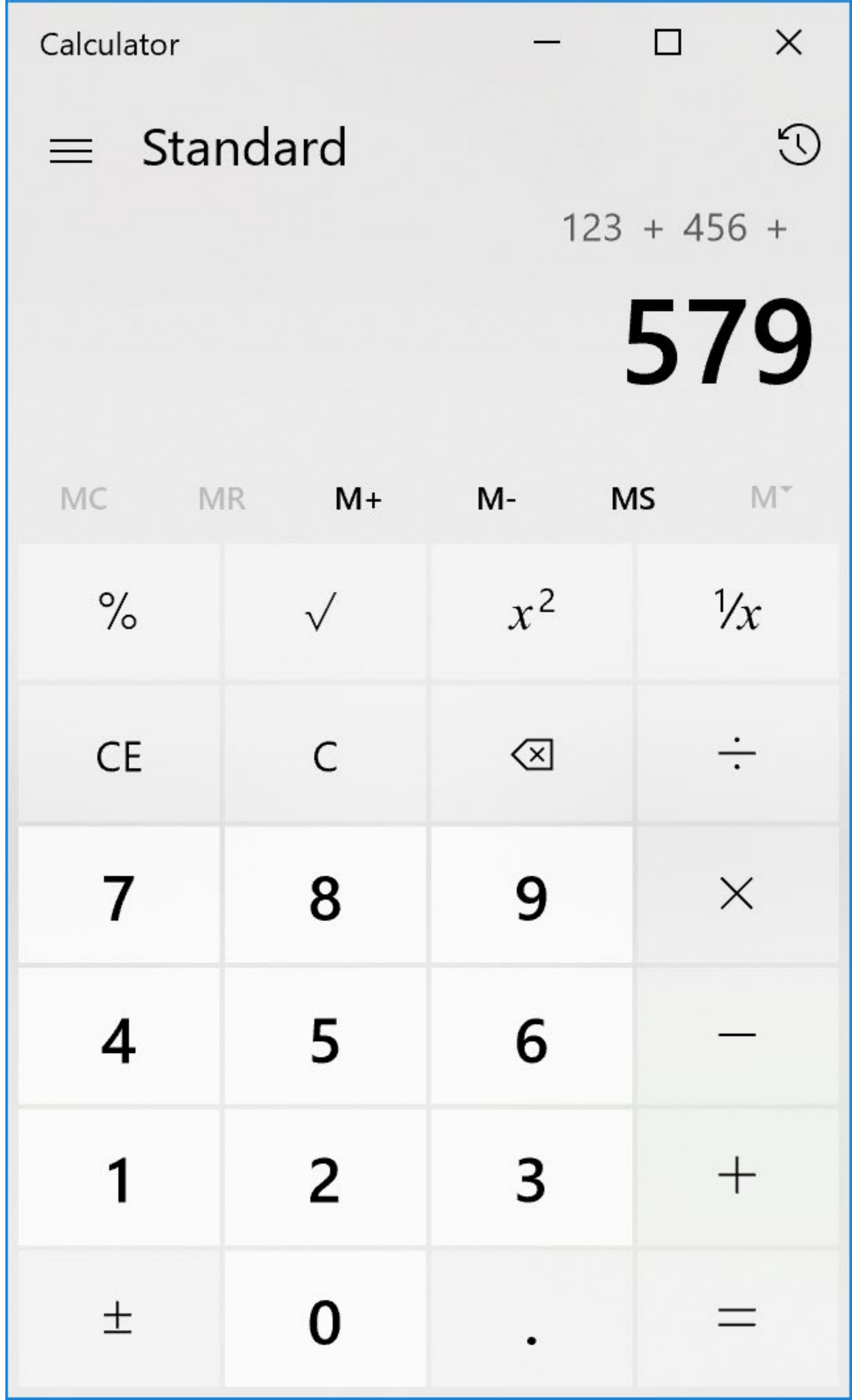
Example

123 + 456 –

Show temp result

579 –

579 – 789…



Define Variables

- Use **enum** to define operations
- Other variables
  - Mode (**isDigitEnterMode**)
  - Current display (**displayString**)
  - Temporary result (**result**)
  - Last operation (**lastOP**)

```
public class CalculatorForm {
    ......
    ......
    enum CalcOP {NONE, ADD, SUB, MULTIPLY, DIVIDE};

    private boolean isDigitEnterMode = false;
    private String displayString = "";
    private double result = 0;
    private CalcOP lastOP = CalcOP.NONE;
    ......
```

Adding Functions to Listeners of Digit Buttons

```
.........
button0.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        enterDigit("0");
    }
});
button1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        enterDigit("1");
    }
});
button2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        enterDigit("2");
    }
});
.........
.........
```

Entering Digits

- Call enterDigit() in each listener of digit buttons

```
private void enterDigit(String digit)
{
    if (!isDigitEnterMode) {
        if (digit == ".")
            displayString = "0.";
        else
            displayString = digit;
        isDigitEnterMode = true;
    }
    else {
        // Only floating-point number
        // can start with 0
        if (displayString == "0" && digit != ".")
            return;
        displayString += digit;
    }
    displayField.setText(displayString);
}
```

Adding Functions to Listeners of OP Buttons

```
.........
buttonMultiply.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        evalLastOP(CalcOP.MULTIPLY);
    }
});
buttonDivide.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        evalLastOP(CalcOP.DIVIDE);
    }
});
buttonEqual.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        evalLastOP(CalcOP.NONE);
    }
});
.........
.........
```

Evaluate Operations

- Evalute operators (+ - * / =)

```
private void evalLastOP(CalcOP currOP)
{
    double value = Double.parseDouble(displayField.getText());
    // Note that we evaluate last Operator, not current
    switch (lastOP) {
            case ADD:
                result += value;
                break;
            case SUB:
                result -= value;
                break;
            case DIVIDE:
                result /= value;
                break;
            case MULTIPLY:
                result *= value;
                break;
            default: // First value
                result = value;
                break;
    }
    displayField.setText(Double.toString(result));
    isDigitEnterMode = false;
    lastOP = currOP;
}
```

Create Test Interfaces

- Need to provide two public test interfaces:

```
1. public void testClick(String button)
   throws Exception
```

```
2. public double getResult() {
       return result;
   }
```

```
3. public void showWindow()
```

```
public void testClick(String button) throws Exception
{
    switch (button)
    {
        case "+": buttonAdd.doClick(); break;
        case "-": buttonSub.doClick(); break;
        case "*": buttonMultiply.doClick(); break;
        case "/": buttonDivide.doClick(); break;
        case ".": buttonDot.doClick(); break;
        case "=": buttonDivide.doClick(); break;
        case "±": buttonMinusPlus.doClick(); break;
        case "CE": buttonCE.doClick(); break;
        case "CLEAR": buttonCLS.doClick(); break;
        case "0": button0.doClick(); break;
        case "1": button1.doClick(); break;
        case "2": button2.doClick(); break;
        case "3": button3.doClick(); break;
        case "4": button4.doClick(); break;
        case "5": button5.doClick(); break;
        case "6": button6.doClick(); break;
        case "7": button7.doClick(); break;
        case "8": button8.doClick(); break;
        case "9": button9.doClick(); break;
        default:
            throw new Exception("Error! No button " + button);
    }
}
```

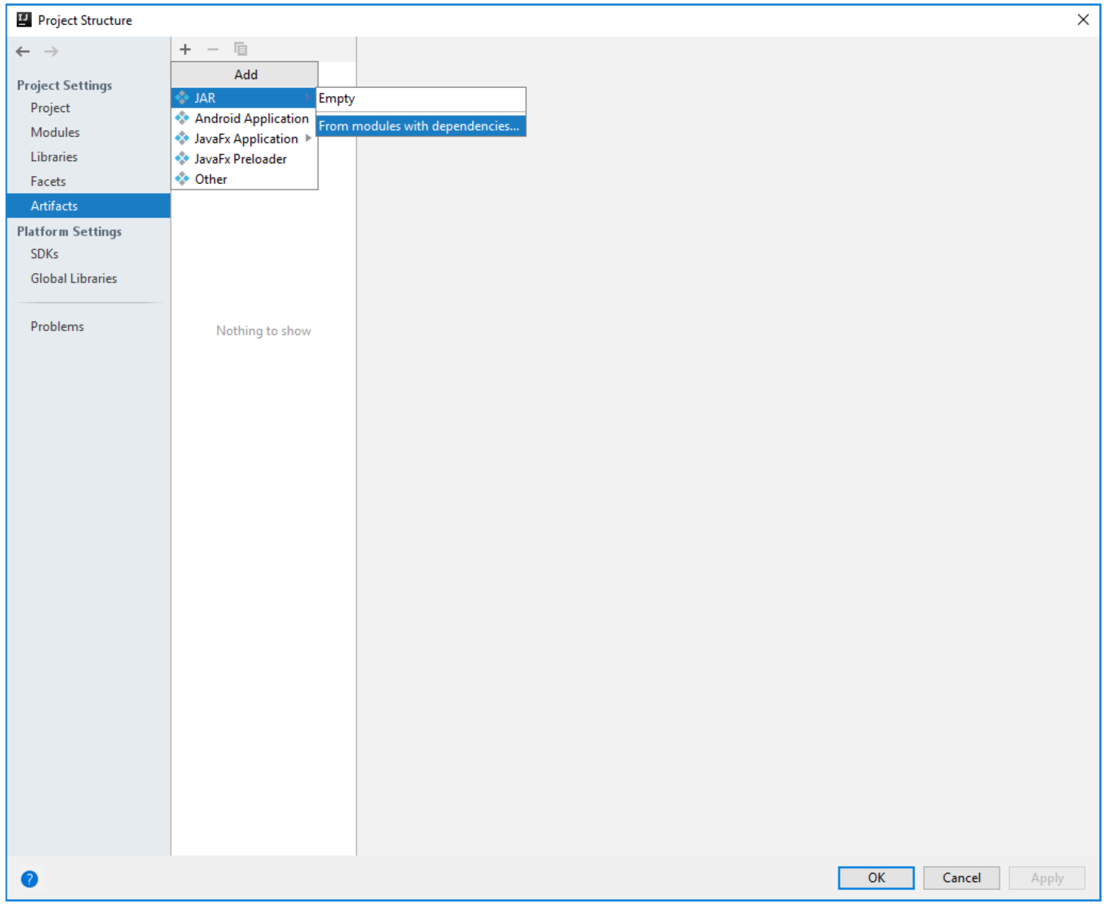
Show the Testing Process

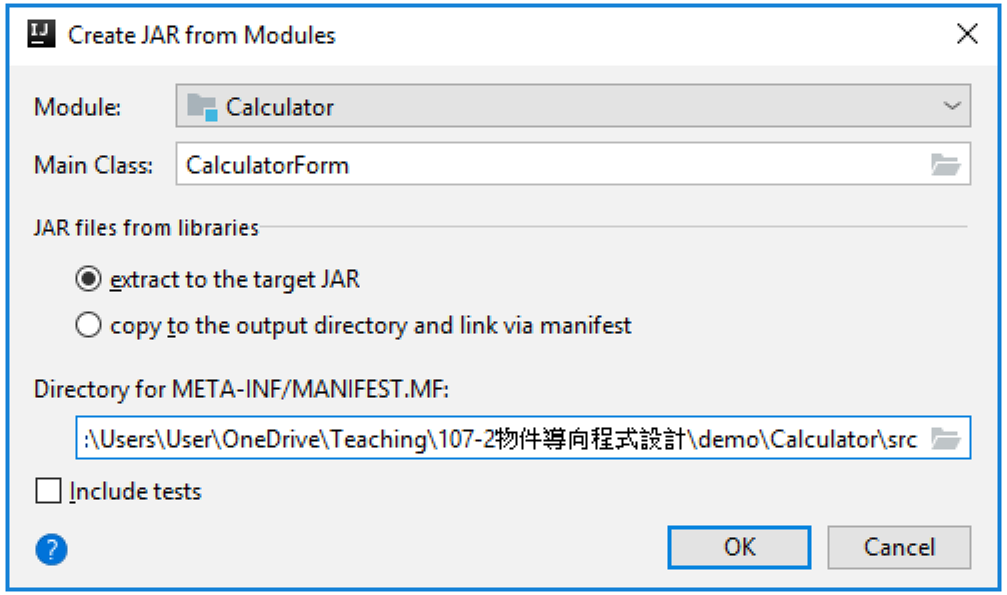
Define a public function `public void showWindow()`

```java
public void showWindow() {
    JFrame frame = new JFrame("Calculator");
    frame.setContentPane(this.CalcPanel);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
}
```

Build a Jar

- File -> Project Structure -> Project Settings -> Artifacts -> Click green plus sign -> Jar -> From modules with dependencies...




Create JAR from Modules

Module: Calculator

Main Class: CalculatorForm

JAR files from libraries

● extract to the target JAR

○ copy to the output directory and link via manifest

Directory for META-INF/MANIFEST.MF:

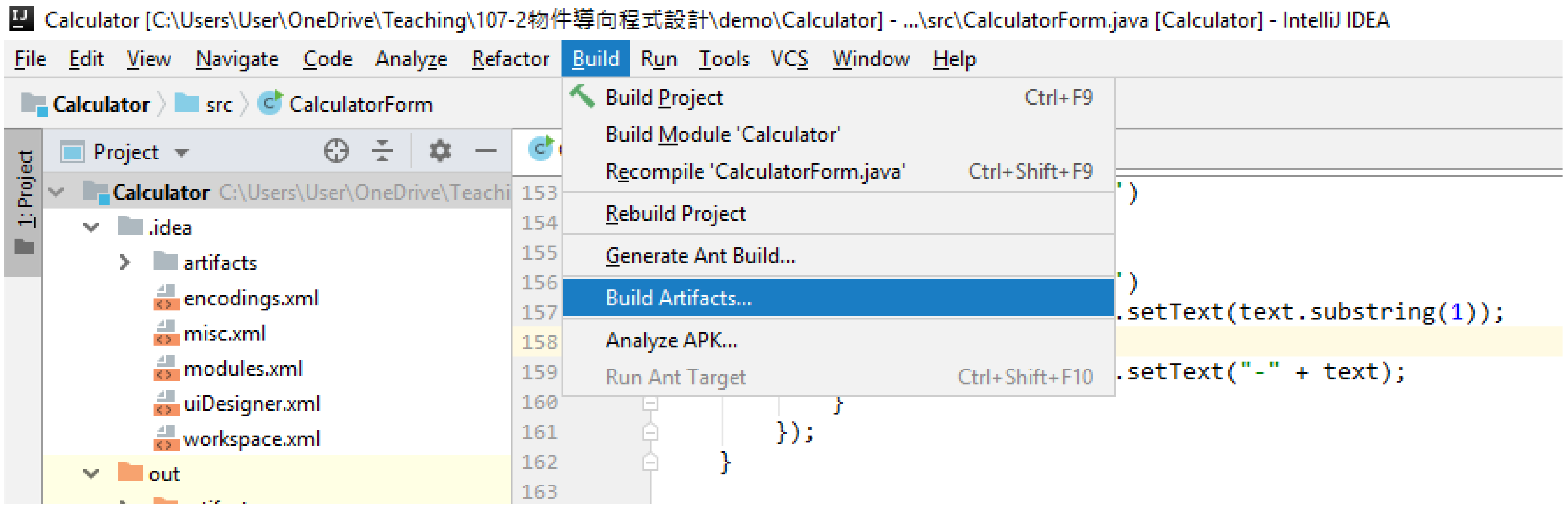
:\Users\User\OneDrive\Teaching\107-2物件導向程式設計\demo\Calculator\src

☐ Include tests

OK Cancel

Build a Jar (Cont'd)

- Build | Build Artifact



Calculator [C:\Users\User\OneDrive\Teaching\107-2物件導向程式設計\demo\Calculator] - ...\src\CalculatorForm.java [Calculator] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Calculator src CalculatorForm
Project
Calculator C:\Users\User\OneDrive\Teachi
.idea
artifacts
encodings.xml
misc.xml
modules.xml
uiDesigner.xml
workspace.xml
out
Build Project Ctrl+F9
Build Module 'Calculator'
Recompile 'CalculatorForm.java' Ctrl+Shift+F9
Rebuild Project
Generate Ant Build...
Build Artifacts...
Analyze APK...
Run Ant Target Ctrl+Shift+F10
1: Project

```
')

')
.setText(text.substring(1));

.setText("-" + text);
        }
    });
}
```

Testing Your Calculator with JUnit

- Download junit-4.12.jar & hamcrest-core-1.3.jar
- Download CalculatorFormTest.java

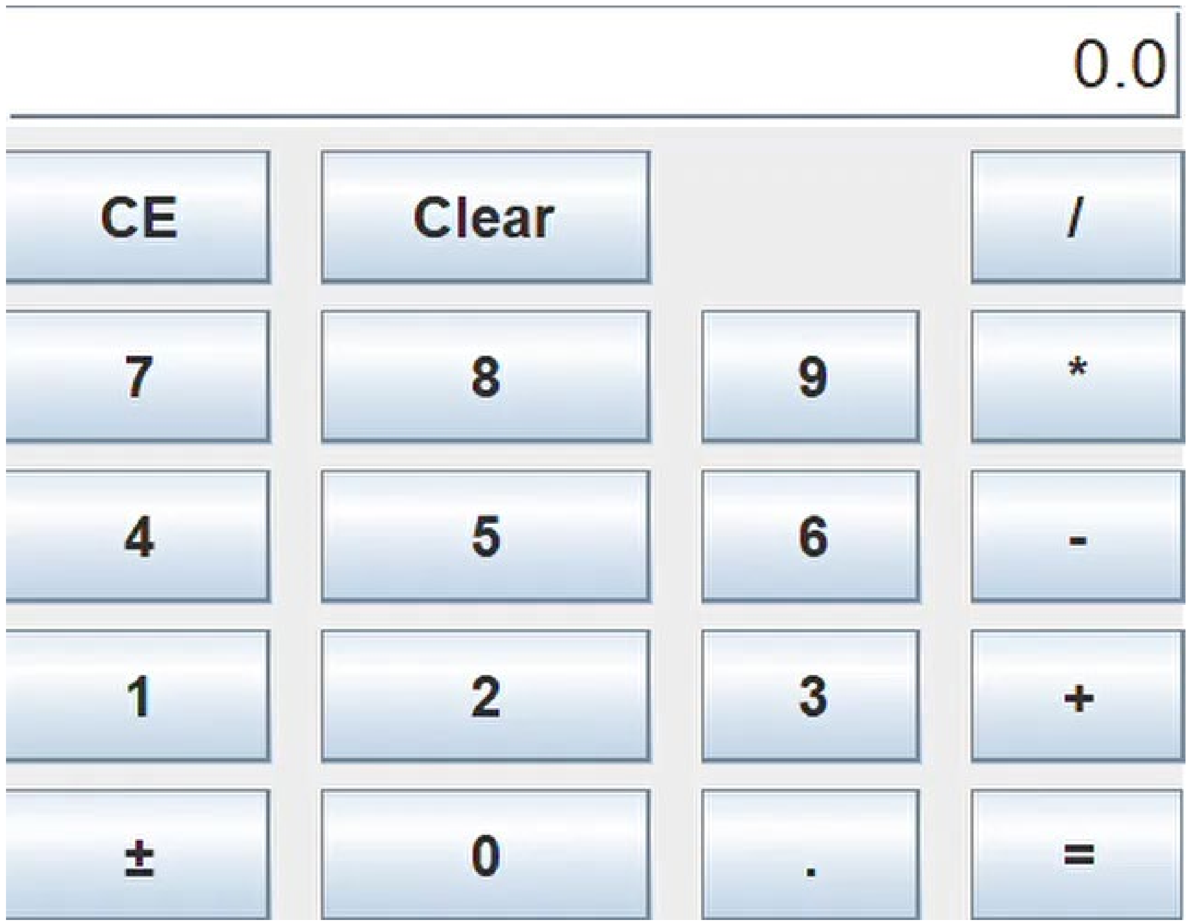
- Compile CalculatorFormTest.java with your jar

```
C:\> javac -cp ".;junit-4.12.jar;Calculator.jar" CalculatorFormTest.java
```

- Run the Test

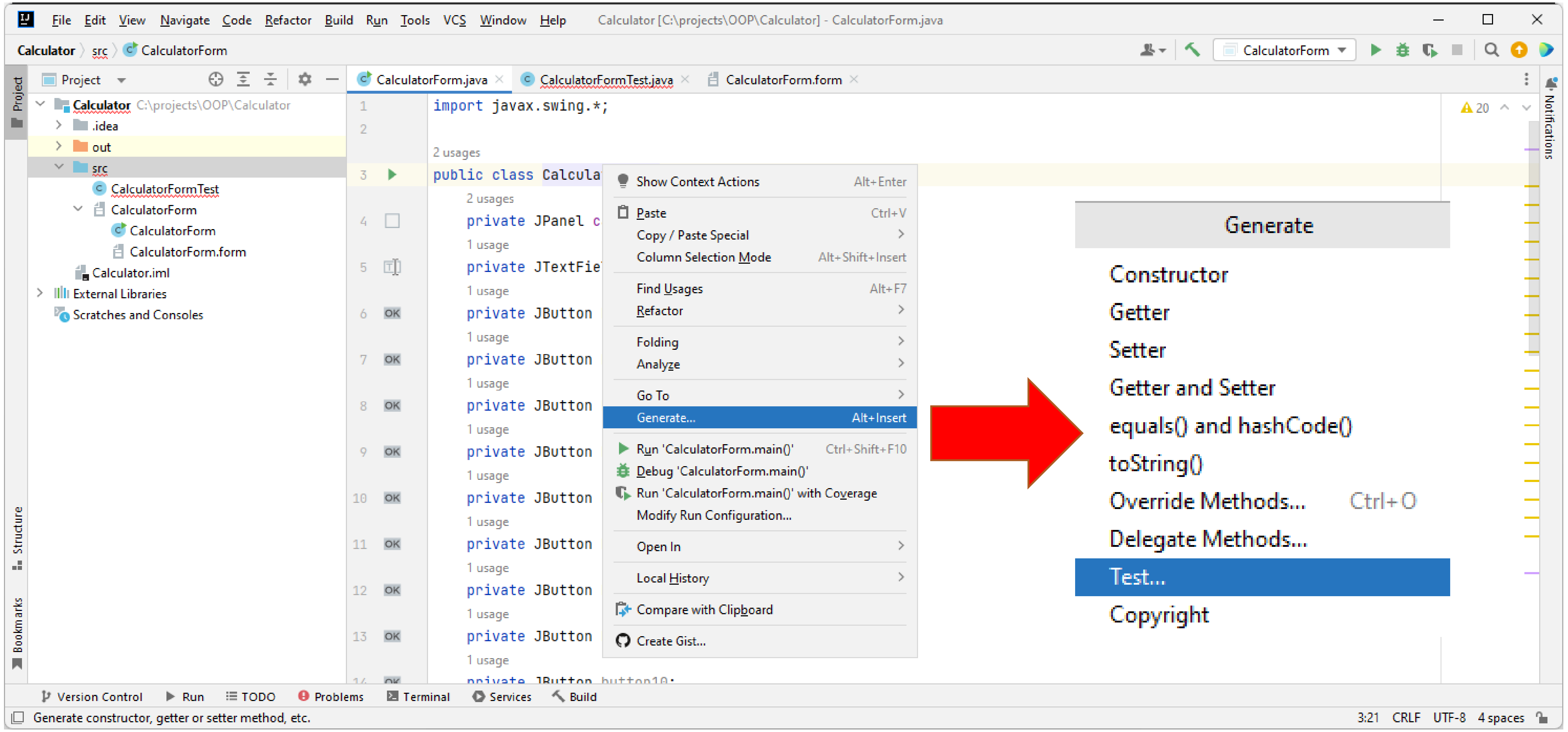
```
C:\> java -cp ".;junit-4.12.jar;Calculator.jar" CalculatorFormTest
```

Testing Process



Create Unit Testing using IntelliJ

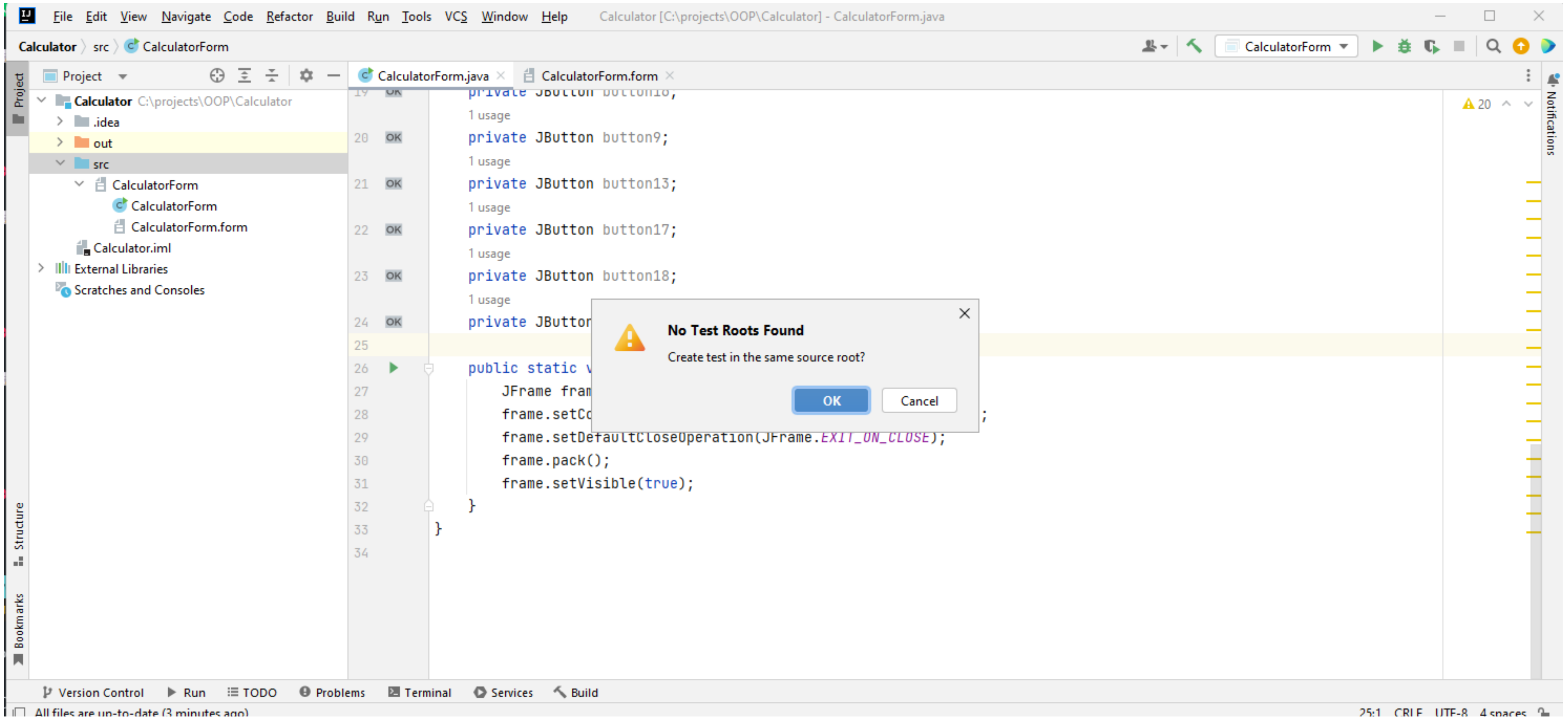
Right click and select “Generate…” -> “Test…”



```
import javax.swing.*;

2 usages
public class Calcula
    2 usages
    private JPanel c
    1 usage
    private JTextFie
    1 usage
    private JButton
    1 usage
    private JButton
    1 usage
    private JButton
    1 usage
    private JButton
    1 usage
    private JButton
    1 usage
    private JButton
    1 usage
    private JButton
    1 usage
    private JButton
    1 usage
    private JButton button10;
```

Click “OK”

- Click “OK” when seeing the message “No Test Roots Found”



```
    private JButton button10;
    1 usage
    private JButton button9;
    1 usage
    private JButton button13;
    1 usage
    private JButton button17;
    1 usage
    private JButton button18;
    1 usage
    private JButtor

    public static v
        JFrame fram
        frame.setCo                                          ;
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```

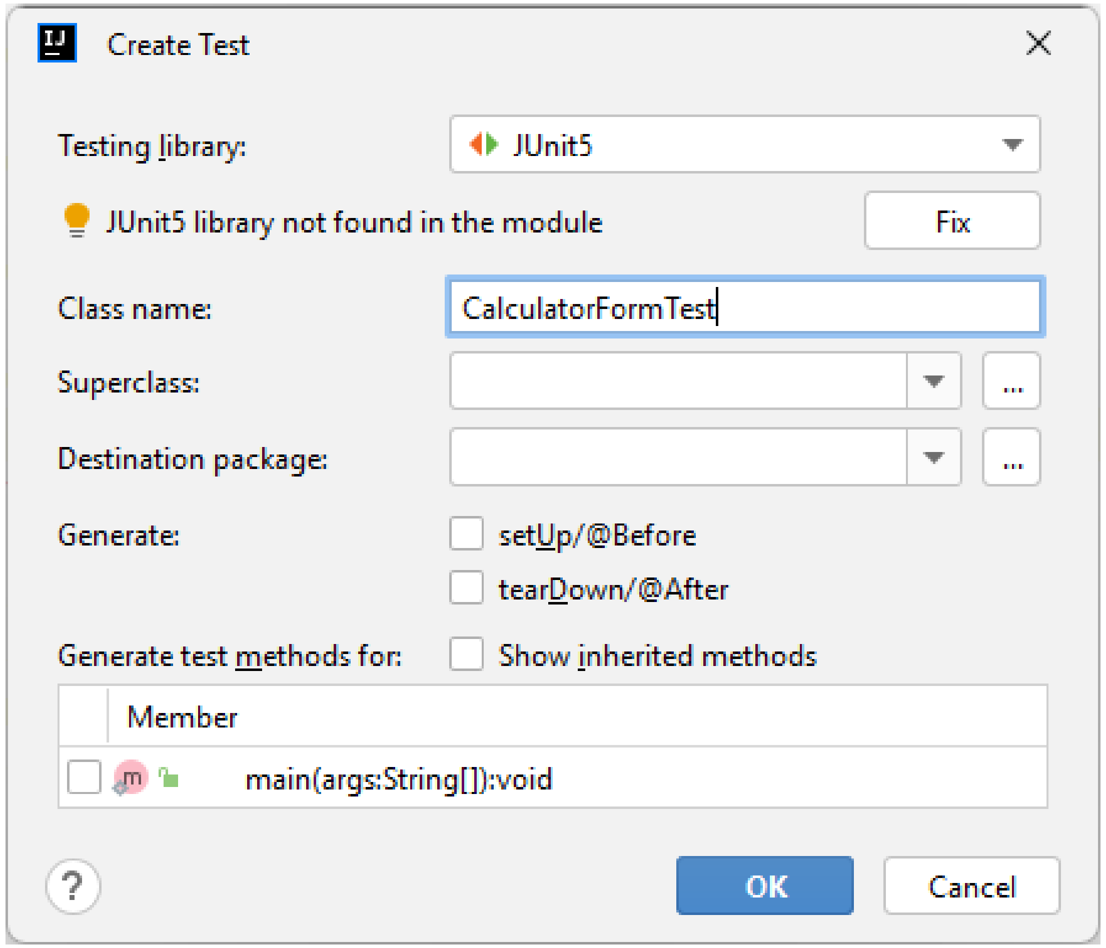
No Test Roots Found

Create test in the same source root?

OK | Cancel

Select Your Test Framework

- Use default JUnit5



Create Test

Testing library: JUnit5

JUnit5 library not found in the module

Class name: CalculatorFormTest

Superclass:

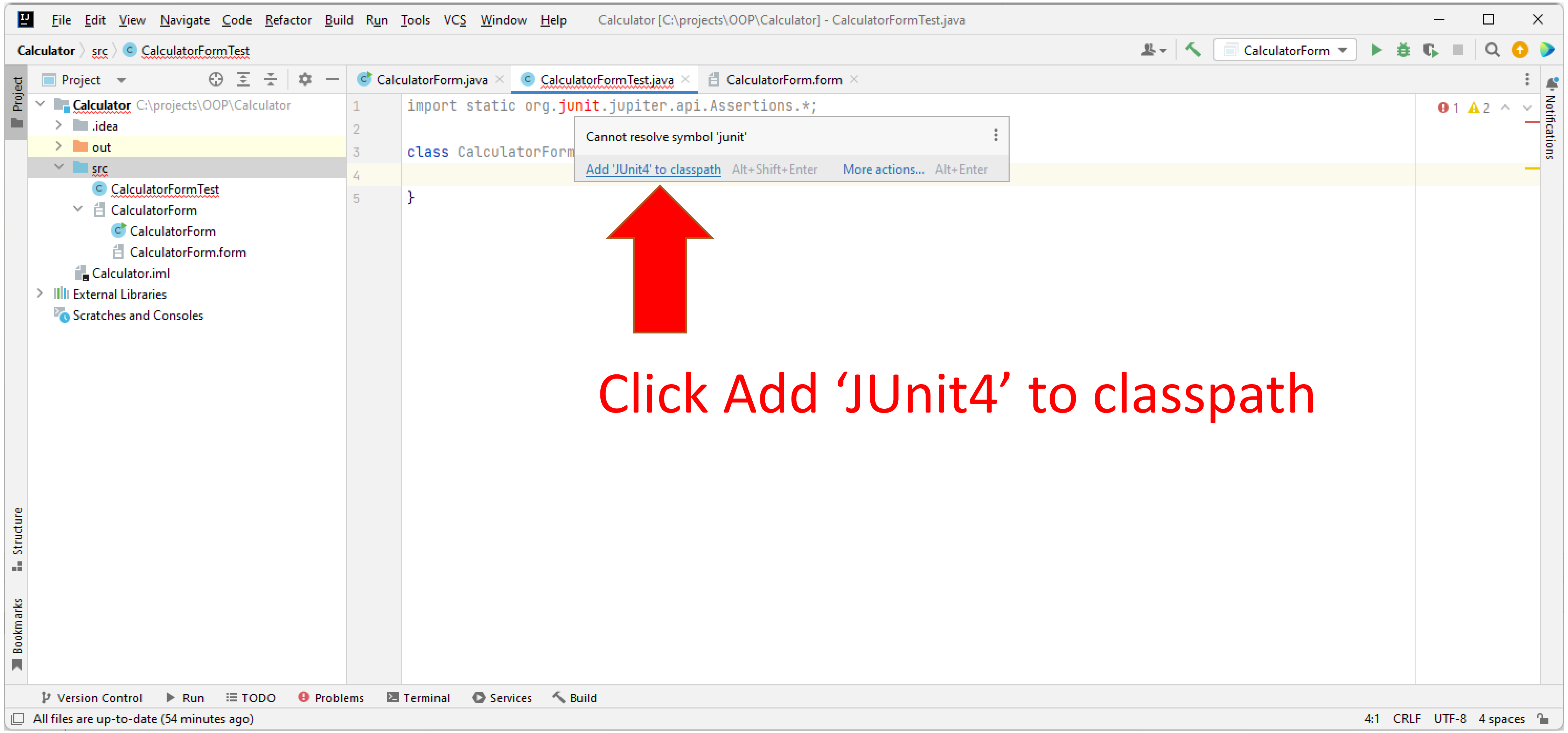
Destination package:

Generate: [ ] setUp/@Before
[ ] tearDown/@After

Generate test methods for: [ ] Show inherited methods

	Member
[ ]	main(args:String[]):void

Click on the error "junit"



```
import static org.junit.jupiter.api.Assertions.*;

class CalculatorForm

}
```

Cannot resolve symbol 'junit'

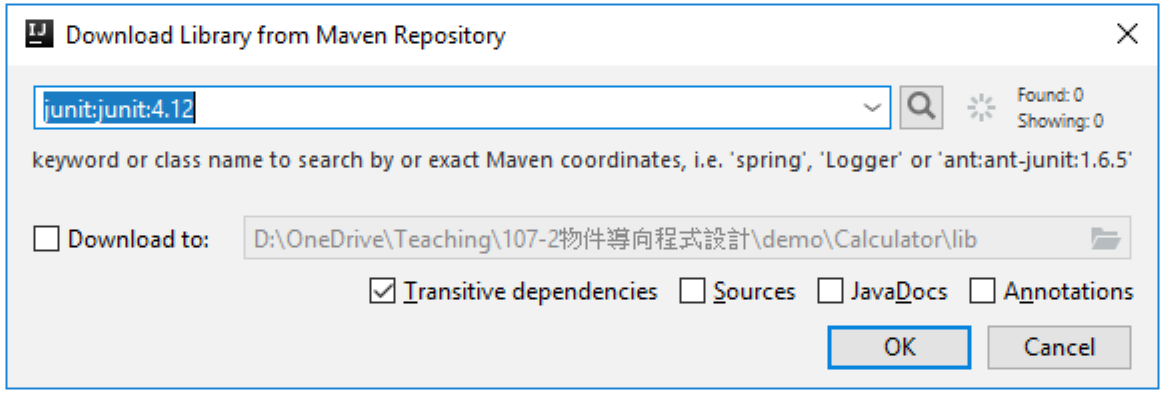
Add 'JUnit4' to classpath Alt+Shift+Enter More actions... Alt+Enter

All files are up-to-date (54 minutes ago)

Click Add 'JUnit4' to classpath

Auto Download Library

- Click OK to download junit library automatically



Download Library from Maven Repository

junit:junit:4.12

Found: 0
Showing: 0

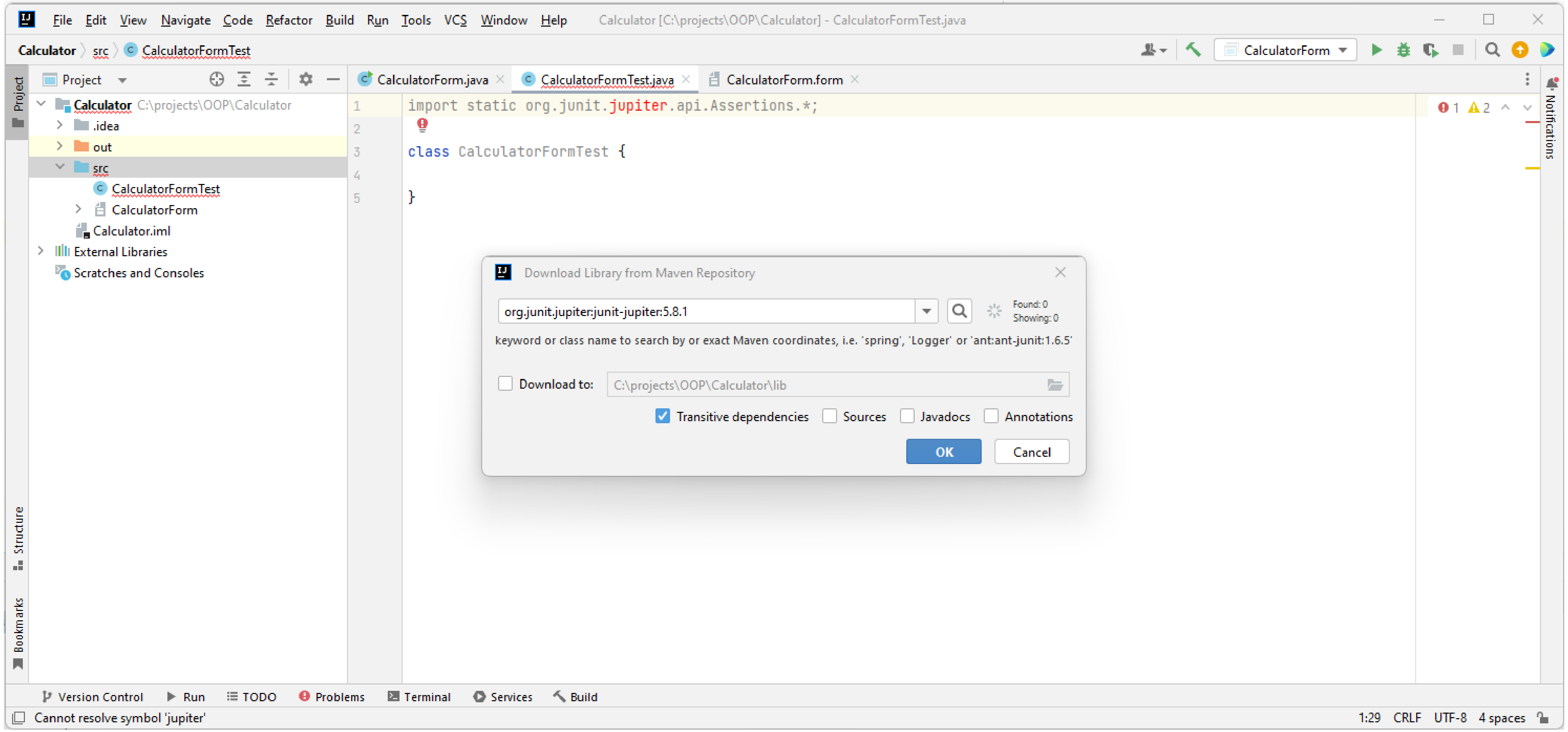
keyword or class name to search by or exact Maven coordinates, i.e. 'spring', 'Logger' or 'ant:ant-junit:1.6.5'

☐ Download to: D:\OneDrive\Teaching\107-2物件導向程式設計\demo\Calculator\lib

☑ Transitive dependencies ☐ Sources ☐ JavaDocs ☐ Annotations

OK Cancel

Click the error “jupiter”



```
import static org.junit.jupiter.api.Assertions.*;

class CalculatorFormTest {

}
```

Download Library from Maven Repository

org.junit.jupiter:junit-jupiter:5.8.1

Found: 0
Showing: 0

keyword or class name to search by or exact Maven coordinates, i.e. 'spring', 'Logger' or 'ant:ant-junit:1.6.5'

☐ Download to: C:\projects\OOP\Calculator\lib

☑ Transitive dependencies ☐ Sources ☐ Javadocs ☐ Annotations

OK Cancel

Cannot resolve symbol 'jupiter'

JUnit 5 Test

New instance

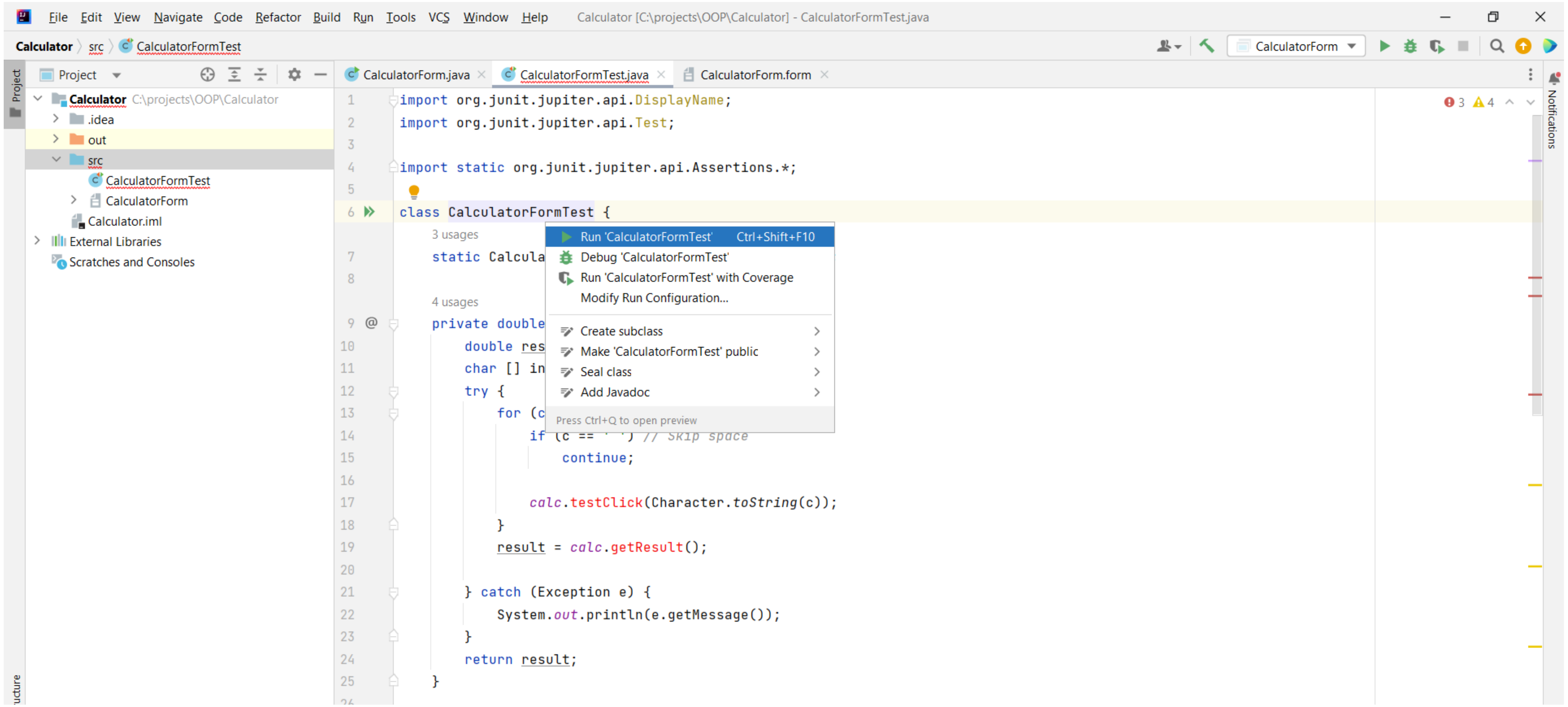
- New an instance of CalculatorForm
- Create test unit using annotation @Test

Test Unit

```
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import java.util.concurrent.ThreadLocalRandom;
import static org.junit.jupiter.api.Assertions.*;

class CalculatorFormTest {
    static CalculatorForm calc = new CalculatorForm();

    private double evalInput(String input) {
        double result = Double.NaN;
        char [] in_array = input.toCharArray();
        try {
            for (char c : in_array) {
                if (c == ' ') // Skip space
                    continue;
                calc.testClick(Character.toString(c));
            }
            result = calc.getResult();

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return result;
    }
    private void clearAll() {
        calc.testClick("CLEAR");
   }

    @Test
    @DisplayName("Floating points")
    public void testFloatingPoint() {
        clearAll();
        double result = evalInput("0.6*0.7*0.8*0.9/2=");
        assertEquals(0.1512, result, 0);
    }
}
```

Run Your Test

- Press “Alt + Enter” on your test class

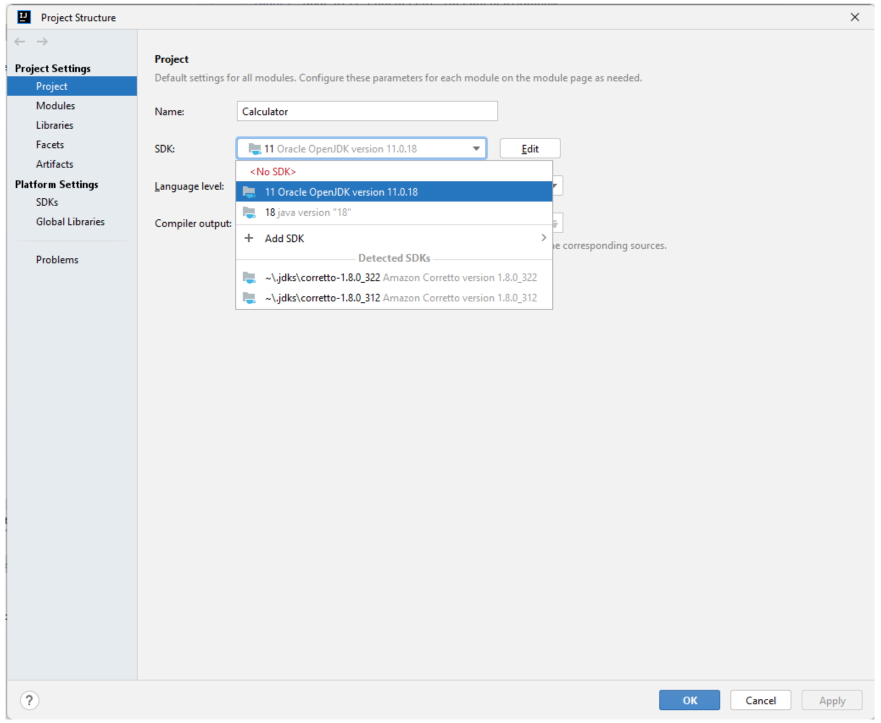

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Calculator [C:\projects\OOP\Calculator] - CalculatorFormTest.java
Calculator › src › CalculatorFormTest
CalculatorForm
Project
Calculator C:\projects\OOP\Calculator
.idea
out
src
CalculatorFormTest
CalculatorForm
Calculator.iml
External Libraries
Scratches and Consoles
CalculatorForm.java
CalculatorFormTest.java
CalculatorForm.form
Run 'CalculatorFormTest' Ctrl+Shift+F10
Debug 'CalculatorFormTest'
Run 'CalculatorFormTest' with Coverage
Modify Run Configuration...
Create subclass
Make 'CalculatorFormTest' public
Seal class
Add Javadoc
Press Ctrl+Q to open preview
Notifications

```
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class CalculatorFormTest {
    3 usages
    static Calcula

    4 usages
    private double
        double res
        char [] in
        try {
            for (c
                if (c == ' ') // skip space
                    continue;

                calc.testClick(Character.toString(c));
            }
            result = calc.getResult();

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return result;
    }
```

Note on Testing in GitHub Classroom (2023/3)

- So far GitHub classroom only support Java 11. Please use it to compile your Calculator.jar!
- Install JDK 11 and set the follows:

File -> Project Structure



Project Structure

Project

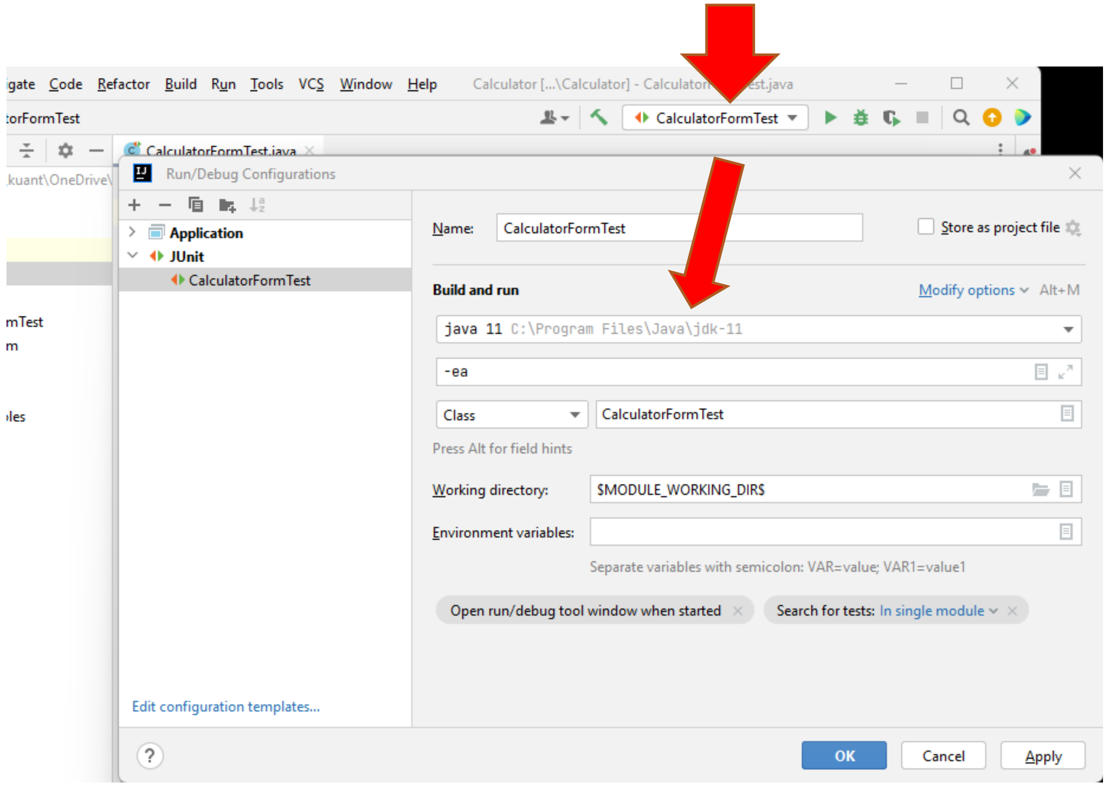
Default settings for all modules. Configure these parameters for each module on the module page as needed.

Name: Calculator

SDK: 11 Oracle OpenJDK version 11.0.18

- <No SDK>
- 11 Oracle OpenJDK version 11.0.18
- 18 java version "18"
- Add SDK
- Detected SDKs
- ~\.jdks\corretto-1.8.0_322 Amazon Corretto version 1.8.0_322
- ~\.jdks\corretto-1.8.0_312 Amazon Corretto version 1.8.0_312

Edit Configuration



Run/Debug Configurations

Name: CalculatorFormTest

Build and run

java 11 C:\Program Files\Java\jdk-11

-ea

Class: CalculatorFormTest

Press Alt for field hints

Working directory: $MODULE_WORKING_DIR$

Environment variables:

Separate variables with semicolon: VAR=value; VAR1=value1

Open run/debug tool window when started

Search for tests: In single module