ASSIGNMENT NO: B3

1. TITLE

A mobile application needs to be designed for using a Calculator (+, - ,*, /, Sin, Cos, sqroot) with Memory Save/Recall using Extended precision floating point number format. Give the Required modelling, Design and Positive-Negative test cases.

2. PREREQUISITES

- 64-bit Fedora or equivalent OS with 64-bit Intel-i5/i7
- Java 1.7.0
- Android Studio

3. OBJECTIVE

- To learn the Android Studio.
- To study the design and implementation of mobile application for calculator.

4. MATHEMATICAL MODEL

Let, S be the System Such that,
A={ S, E, I,O, F, DD, NDD, success, failure }
Where,
S= Start state,
E= End State,
I= Set of Input
O= Set of Out put
F =Set of Function
DD=Deterministic Data
NDD=Non Deterministic Data
Success Case: It is the case when all the inputs are given by system are entered correctly.
Failure Case: It is the case when the input does not match the validation Criteria.

5. THEORY

Android Studio Overview
Android Studio is the official IDE for Android application development, based on IntelliJ IDEA. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle-based build system
- Build variants and multiple apk file generation
- Code templates to help you build common app features
- Rich layout editor with support for drag and drop theme editing
- lint tools to catch performance, usability, version compatibility, and other problems
- ProGuard and app-signing capabilities

- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine
- And much more

Android Project Structure

By default, Android Studio displays your project files in the *Android* project view. This view shows a flattened version of your project's structure that provides quick access to the key source files of Android projects and helps you work with the Gradle-based build system. The *Android* project view:

- Shows the most important source directories at the top level of the module hierarchy.
- Groups the build files for all modules in a common folder.
- Groups all the manifest files for each module in a common folder.
- Shows resource files from all Gradle source sets.
- Groups resource files for different locales, orientations, and screen types in a single group per resource type

  java/ - Source files for the module.
  manifests/ - Manifest files for the module.
  res/ - Resource files for the module.
  Gradle Scripts/ - Gradle build and property files.

Positive Testing:

Test Case ID	Expected Result	Actual Result	Status
1	Check if all the numbers are working (0 to 9)	All the numbers are working (0 to 9)	
2	Check if the arithmetic keys (+, -, *, %, /) are working	The arithmetic keys (+, -, *, %, /) are working	
3	Check if the brackets keys are working	The bracket keys are working	
4	Check if the square and square root key is working	The square and square root key is working	
5	Check if the sin, cos, tan, cot keys are working	The sin, cos, tan, cot keys are working	
6	Check if it is showing the correct values for sin, cos, tan and cot	It is showing the correct values for sin, cos, tan and cot	
7	Check the addition of two sin and cos values	The addition of two sin and cos values	
8	Check the addition of two tan and cot values	The addition of two tan and cot values	
9	Check that it is returning the float values or integer values	It is returning the float values or integer values	

10	Check if the functionality using BODMAS/BIDMAS works as expected	Working Properly	

Negative Testing:

Test Case ID	Expected Result	Actual Result	Status
1	Check if it is allowing letters instead of numbers	It is taking only numbers as input	
2	Check if it is returning float values instead of integer	It is returning integer values only	
3	Check if it is returning integer values instead of float	It is returning float values only	
4	Check if the functionality using BODMAS/BIDMAS works as expected	Functioning Properly	

6. APPLICATION FLOW

1 You can use Android studio IDE/android-adt-bundle to create an Android application under a package com.example.calci.myapplication;. While creating this project, make sure you Target SDK and Compile With at the latest version of Android SDK to use higher levels of APIs.

2 Modify src/MainActivity.java file to add Calculator code.

3 Modify the res/layout/activity_main to add respective XML components

4 Create a new folder under Calculator

5 Run the application and choose a running android device and install the application on it and verify the results

7. CONCLUSION

A mobile application is designed for a Calculator (+, - ,*, /, Sin, Cos, sq-root) with Memory Save/Recall using Extended precision floating point number format.

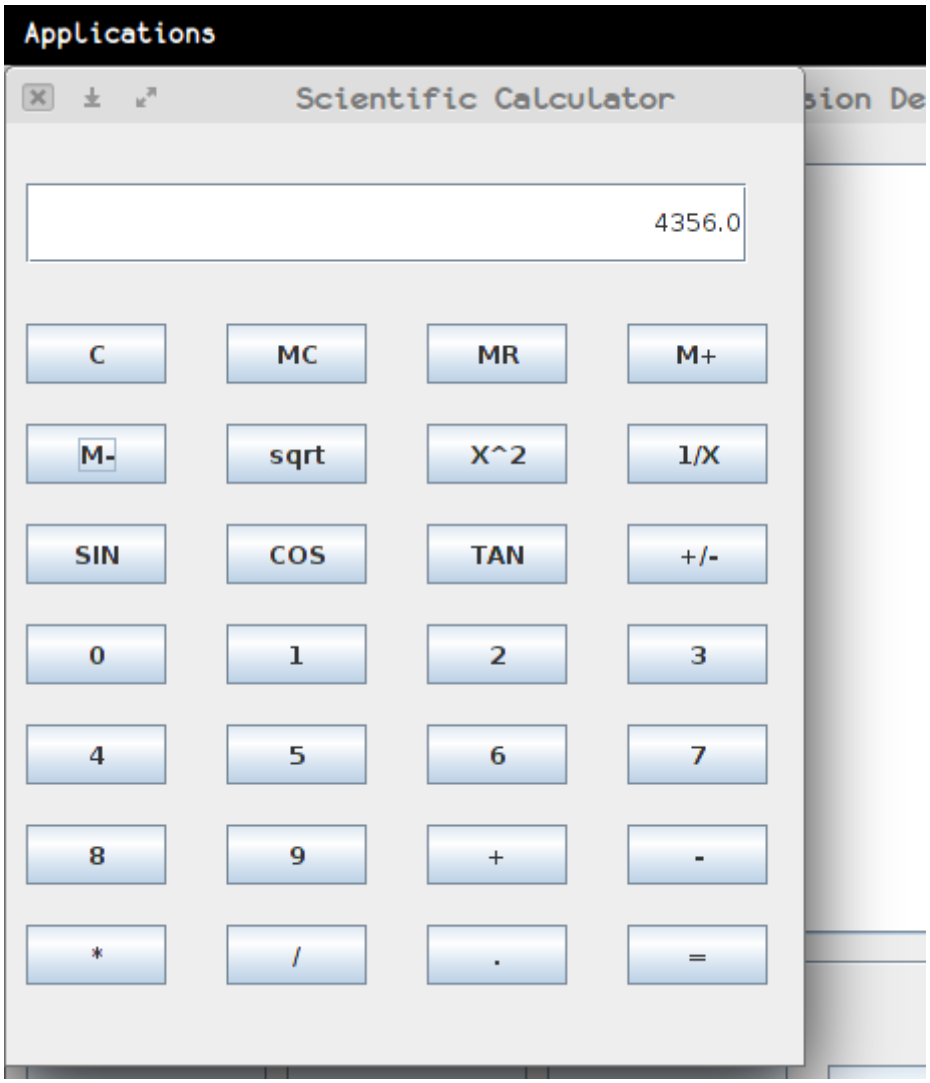
```
import java.awt.*;
```

```
import java.text.DecimalFormat;
import java.awt.event.*;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;

import javax.swing.*;
public class ScientificCalc implements ActionListener{
JFrame frame;
JPanel panel;
JTextField ansfield;
JButton buttons[];
String buttonsText[]={"C","MC","MR","M+","M-","sqrt","X^2","1/X","SIN","COS","TAN","+/-
","0","1","2","3","4","5","6","7","8","9","+","-","*","/",".","="};
int maxx=400,maxy=500;

static final String DIGITS = "0123456789.";
 Boolean userIsInTheMiddleOfTypingANumber = false;
 CalculatorBrain mCalculatorBrain=new CalculatorBrain();



public ScientificCalc() {
        frame = new JFrame("Scientific Calculator");
        frame.setSize(maxx, maxy);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        panel = new JPanel();
        panel.setLayout(null);

        ansfield = new JTextField();
        ansfield.setBounds(10,30,maxx-40,40);
        ansfield.setHorizontalAlignment(JTextField.RIGHT);


        buttons=new JButton[buttonsText.length];

        int currentx=0,currenty=0;

        for(int i=0;i<buttonsText.length;i++)
        {
                buttons[i]=new JButton(buttonsText[i]);
                buttons[i].addActionListener(this);
                if(currentx==0&&currenty==0)
                {
                        buttons[i].setBounds(10,100,70,30);
                        currentx=10;
                        currenty=100;
                }
                else
                {
                        if(currentx<maxx-100)
                        {
                        currentx+=100;
                        buttons[i].setBounds(currentx,currenty,70,30);
                        }
                        else
                        {
                                currentx=10;
                                currenty+=50;
                                buttons[i].setBounds(currentx,currenty,70,30);
                        }

                }
        panel.add(buttons[i]);
        }

        panel.add(ansfield);
        frame.add(panel);
        frame.setVisible(true);
}
public void actionPerformed(ActionEvent ae) {
```

```
        String buttonObj="";

        for(int i=0;i<buttonsText.length;i++)
        {
                if(ae.getSource()==buttons[i])
                {
                buttonObj=buttons[i].getText().toString();

                break;
                }
        }
        calc(buttonObj);

}
void calc(String buttonObj)
{
        if (DIGITS.contains(buttonObj)) {


            if (userIsInTheMiddleOfTypingANumber) {

                if (buttonObj.equals(".") && ansfield.getText().toString().contains(".")) {

                } else {
                    ansfield.setText(ansfield.getText()+buttonObj);
                }

            } else {

                if (buttonObj.equals(".")) {


                    ansfield.setText(0 + buttonObj);
                } else {
                    ansfield.setText(buttonObj);
                }

                userIsInTheMiddleOfTypingANumber = true;
            }

        } else {

            if (userIsInTheMiddleOfTypingANumber) {

                mCalculatorBrain.setOperand(Double.parseDouble(ansfield.getText().toString()));
                userIsInTheMiddleOfTypingANumber = false;
            }
            try
            {
            mCalculatorBrain.performOperation(buttonObj);
            }catch(Exception e){}
            ansfield.setText(""+mCalculatorBrain.getResult());

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

        new ScientificCalc();
}

public class CalculatorBrain {


    private double mOperand;
    private double mWaitingOperand;
    private String mWaitingOperator;
    private double mCalculatorMemory;


    public static final String ADD = "+";
    public static final String SUBTRACT = "-";
    public static final String MULTIPLY = "*";
```

```
public static final String DIVIDE = "/";

public static final String CLEAR = "C" ;
public static final String CLEARMEMORY = "MC";
public static final String ADDTOMEMORY = "M+";
public static final String SUBTRACTFROMMEMORY = "M-";
public static final String RECALLMEMORY = "MR";
public static final String SQUAREROOT = "sqrt";
public static final String SQUARED = "X^2";
public static final String INVERT = "1/X";
public static final String TOGGLESIGN = "+/-";
public static final String SINE = "SIN";
public static final String COSINE = "COS";
public static final String TANGENT = "TAN";

PrintWriter writer;
public CalculatorBrain() {
    mOperand = 0;
    mWaitingOperand = 0;
    mWaitingOperator = "";
    mCalculatorMemory = 0;
}

public void setOperand(double operand) {
    mOperand = operand;
}

public double getResult() {
    return mOperand;
}
public void setMemory(double calculatorMemory) {
    mCalculatorMemory = calculatorMemory;
}
public double getMemory() {
    return mCalculatorMemory;
}

public String toString() {
    return Double.toString(mOperand);
}

protected double performOperation(String operator) throws Exception {

    if (operator.equals(CLEAR)) {
        mOperand = 0;
        mWaitingOperator = "";
        mWaitingOperand = 0;
        // mCalculatorMemory = 0;
    } else if (operator.equals(CLEARMEMORY)) {
        mCalculatorMemory = 0;

        writer = new PrintWriter("memoryFile.txt", "UTF-8");
        writer.println(""+mCalculatorMemory);

        writer.close();
    } else if (operator.equals(ADDTOMEMORY)) {
        mCalculatorMemory = mCalculatorMemory + mOperand;

        writer = new PrintWriter("memoryFile.txt", "UTF-8");
        writer.println(""+mCalculatorMemory);

        writer.close();

    } else if (operator.equals(SUBTRACTFROMMEMORY)) {
        mCalculatorMemory = mCalculatorMemory - mOperand;

        writer = new PrintWriter("memoryFile.txt", "UTF-8");
        writer.println(""+mCalculatorMemory);

        writer.close();
    } else if (operator.equals(RECALLMEMORY)) {
        mOperand = mCalculatorMemory;
    } else if (operator.equals(SQUAREROOT)) {
```

```
            mOperand = Math.sqrt(mOperand);

        } else if (operator.equals(SQUARED)) {
            mOperand = mOperand * mOperand;

        } else if (operator.equals(INVERT)) {
            if (mOperand != 0) {
                mOperand = 1 / mOperand;
            }
        } else if (operator.equals(TOGGLESIGN)) {
            mOperand = -mOperand;
        } else if (operator.equals(SINE)) {
            mOperand = Math.sin(Math.toRadians(mOperand));

        } else if (operator.equals(COSINE)) {
            mOperand = Math.cos(Math.toRadians(mOperand));
        } else if (operator.equals(TANGENT)) {
            mOperand = Math.tan(Math.toRadians(mOperand));
        } else {
            performWaitingOperation();
            mWaitingOperator = operator;
            mWaitingOperand = mOperand;
        }

        return mOperand;
    }

    protected void performWaitingOperation() {

        if (mWaitingOperator.equals(ADD)) {
            mOperand = mWaitingOperand + mOperand;
        } else if (mWaitingOperator.equals(SUBTRACT)) {
            mOperand = mWaitingOperand - mOperand;
        } else if (mWaitingOperator.equals(MULTIPLY)) {
            mOperand = mWaitingOperand * mOperand;
        } else if (mWaitingOperator.equals(DIVIDE)) {
            if (mOperand != 0) {
                mOperand = mWaitingOperand / mOperand;
            }
        }

    }
}
}
```


Applications
Scientific Calculator
4356.0
C
MC
MR
M+
M-
sqrt
X^2
1/X
SIN
COS
TAN
+/-
0
1
2
3
4
5
6
7
8
9
+
-
*
/
.
=