```
import javax.swing.*;
import java.awt.Font;

public class CalcButton extends JButton
{
    private static Font BIG_FONT = new Font("SansSerif", Font.BOLD,24);

    public CalcButton(String s)
    {
        this(s,s);
    }

    public CalcButton(String s, String command)
    {
        super(s);
        setActionCommand(command);
        init();
    }

    protected void init()
    {
        setFont(BIG_FONT);
    }

    public static Font getBigFont()
    {
        return BIG_FONT;
    }

    public void setActionCommand(String s)
    {
        super.setActionCommand(s);
    }
}
```

```
public class CalcMain
{
    public static void main(String[] args)
    {
        CalculatorModel model = new CalculatorModel();
        CalculatorGui gui = new CalculatorGui(model);
    }
}
```

```
import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.event.*;

public class CalculatorGui extends JFrame
{
    private CalculatorPanel myView;

    public CalculatorGui(CalculatorModel model)
    {
        setTitle("APCS Calculator");
        JPanel panel = new JPanel(new BorderLayout());
        myView = new CalculatorPanel(model);

        panel.add(myView, BorderLayout.CENTER);

        setContentPane(panel);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setJMenuBar(makeMenus());
        pack();
        setVisible(true);
    }

    private JMenuBar makeMenus()
    {
        JMenuBar bar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        fileMenu.add(new AbstractAction("Quit")
            {
                public void actionPerformed(ActionEvent e) {
                    System.exit(0);
                }
            });

        JMenu viewMenu = new JMenu("View");
        JRadioButtonMenuItem standard
            = new JRadioButtonMenuItem("standard", true);
        JRadioButtonMenuItem scientific
            = new JRadioButtonMenuItem("scientific", false);
        ButtonGroup group = new ButtonGroup();
        group.add(standard);
        group.add(scientific);

        ActionListener viewAction = new ActionListener()
            {
                public void actionPerformed(ActionEvent e){
                    setVisible(false);
                    myView.changeView();
                    pack();
                    setVisible(true);
                }
            };

        standard.addActionListener(viewAction);
        scientific.addActionListener(viewAction);

        viewMenu.add(standard);
        viewMenu.add(scientific);

        bar.add(fileMenu);
        bar.add(viewMenu);
        return bar;

    }
}
```

```
import java.text.NumberFormat;

public class CalculatorModel
{
    private static final String NO_OP = "";
    private double myStored;
    private double myCurrent;
    private String myOperation;
    private boolean myStarting;
    private NumberFormat myFormatter;
    private ICalculatorView myView;

    public CalculatorModel(ICalculatorView view)
    {
        this();
        myView = view;
    }

    public CalculatorModel()
    {
        myFormatter = NumberFormat.getInstance();
        myFormatter.setMinimumFractionDigits(20);
        clear();
    }
    public void setView(ICalculatorView view)
    {
        myView = view;
    }

    public void clear()
    {
        myStored = myCurrent = 0;
        myStarting = true;
        myOperation = NO_OP;
        update(myCurrent);
    }

    public void clearEntry()
    {
        myCurrent = 0;
        myStarting = true;
        update(myCurrent);
    }

    public void addDigit(int digit)
    {
        if (myStarting){
            myStarting = false;
            myCurrent = 0;
        }
        if (myCurrent >= 0){
            myCurrent = myCurrent * 10 + digit;
        }
        else {
            myCurrent = myCurrent * 10 - digit;
        }

        update(myCurrent);
    }

    public void unaryOp(String op)
    {
        boolean error = false;
        if (op.equals("sin")){
            myCurrent = Math.sin(myCurrent);
        }
        else if (op.equals("cos")){
            myCurrent = Math.cos(myCurrent);
        }
```



```
        else if (op.equals("tan")){
            myCurrent = Math.tan(myCurrent);
        }
        else if (op.equals("e^x")){
            myCurrent = Math.pow(Math.E,myCurrent);
        }
        else if (op.equals("x^2")){
            myCurrent = myCurrent * myCurrent;
        }
        else if (op.equals("1/x")){
            if (myCurrent != 0){
                myCurrent = 1/myCurrent;
            }
            else {
                error = true;
            }
        }
        else if (op.equals("ln")){
            myCurrent = Math.log(myCurrent);
        }
        else if (op.equals("log")){
            if (myCurrent <= 0){
                error = true;
            }
            else {
                myCurrent = Math.log(myCurrent)/Math.log(10);
            }
        }
        else if (op.equals("n!")){
            double dval =  Math.abs(Math.floor(myCurrent));
            if (dval == Double.POSITIVE_INFINITY){
                error = true;
            }
            else {
                int val = (int) dval;
                double prod = 1;
                for(int k=1; k <= val; k++){
                    prod *= (long)k;
                }
                myCurrent = prod;
            }
        }
        else if (op.equals("x^3")){
            myCurrent = myCurrent * myCurrent * myCurrent;
        }
        else if (op.equals("pi")){
            myCurrent = Math.PI;
        }
        else if (op.equals("e")){
            myCurrent = Math.E;
        }
        if (error){
            showError("Error: function evaluation");
            myOperation = NO_OP;
            myStored = myCurrent = 0;
            myStarting = true;
        }
        else {
            update(myCurrent);
        }
    }

    public void binaryOp(String op)
    {
        if (myOperation.equals(NO_OP)){
            myStored = myCurrent;
            update(myStored);
            myOperation = op;
            myStarting = true;
```

```
        }
        else {
            doOp(op);
        }
    }

    public void toggleSign()
    {
        myCurrent = -1*myCurrent;
        update(myCurrent);
    }

    private void doOp(String op)
    {
        double result = 0.0;
        boolean error = false;

        if (myOperation.equals("*")){
            result = myStored * myCurrent;
        }
        else if (myOperation.equals("/")){
            if (myCurrent == 0.0){
                error = true;
            }
            else {
                result = myStored/myCurrent;
            }
        }
        else if (myOperation.equals("+")){
            result = myStored + myCurrent;
        }
        else if (myOperation.equals("-")){
            result = myStored - myCurrent;
        }


        // all done processing, show result

        if (error){
            showError("Error: division by zero");
            myOperation = NO_OP;
            myStored = myCurrent = 0;
            myStarting = true;
        }
        else {
            myStarting = true;
            if (op.equals("=")){
                myOperation = NO_OP;
            }
            else {
                myOperation = op;
            }

            myStored = myCurrent = result;
            update(myStored);
        }
    }

    private void update(double value)
    {
        String s = myFormatter.format(value);
        int dot = s.indexOf(".");
        if (dot != -1){
            int lastNonZero = s.length()-1;
            while (s.charAt(lastNonZero) == '0'){
                lastNonZero--;
            }
            s = s.substring(0,lastNonZero+1);
        }
```



```
        if (myView != null){
            myView.display(s);
        }
    }

    private void showError(String s)
    {
        myView.display(s);
    }
}
```

```
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.*;
import javax.swing.*;

public class CalculatorPanel extends JPanel implements ICalculatorView
{
    private CalculatorModel myModel;
    private JTextField myTextDisplay;

    private JPanel myNumberPanel;
    private JPanel myFunctionPanel;
    private boolean isStandardView;

    /**
     * Create a view with a model
     * @param model is the model for this view
     */
    public CalculatorPanel(CalculatorModel model)
    {
        setLayout(new BorderLayout());

        myModel = model;
        myNumberPanel = new NumberButtonPanel(model);
        myFunctionPanel = new FunctionButtonPanel(model);

        myTextDisplay = new JTextField(20);
        myTextDisplay.setHorizontalAlignment(JTextField.RIGHT);
        myTextDisplay.setFont(CalcButton.getBigFont());
        myTextDisplay.setEditable(false);
        myTextDisplay.setBackground(java.awt.Color.WHITE);


        JPanel sub = new JPanel(new BorderLayout());
        sub.add(myTextDisplay, BorderLayout.NORTH);
        sub.add(getClearButtons(myModel), BorderLayout.SOUTH);
        add(sub,BorderLayout.NORTH);
        add(myNumberPanel, BorderLayout.CENTER);
        isStandardView = true;
        myModel.setView(this);
    }

    public void changeView()
    {
        if (isStandardView){
            isStandardView = false;
            remove(myNumberPanel);
            add(myNumberPanel, BorderLayout.EAST);
            add(myFunctionPanel, BorderLayout.WEST);
        }
        else{
            isStandardView = true;
            remove(myFunctionPanel);
            remove(myNumberPanel);
            add(myNumberPanel, BorderLayout.CENTER);
        }
        validate();
    }

    private JPanel getClearButtons(final CalculatorModel model)
    {
        JPanel panel = new JPanel(new GridLayout(1,4));
        JButton b1 = new JButton();
        JButton b2 = new JButton();
        JButton clearEntry = new CalcButton("CE");
        JButton clear = new CalcButton("C");
        clearEntry.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e)
                {
```



```
                    model.clearEntry();
                }
            });
        clear.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e)
                {
                    model.clear();
                }
            });
        panel.add(b1);
        panel.add(b2);
        panel.add(clearEntry);
        panel.add(clear);
        return panel;
    }

    public void display(String s)
    {
        myTextDisplay.setText(s);
    }
}
```

```
public class CalculatorTester
{
    private CalculatorModel myModel = new CalculatorModel();

    public void testNumber(int value)
    {
        if (value/10 == 0){
            myModel.addDigit(value);
        }
        else {
            testNumber(value/10);
            testNumber(value%10);
        }
    }

    public void testOp(String op)
    {
        myModel.binaryOp(op);
    }

    public static void main(String[] args)
    {
        CalculatorTester test = new CalculatorTester();
        test.testNumber(12);
        test.testOp("*");
        test.testNumber(5);
        test.testOp("=");
        test.testOp("+");
        test.testNumber(34);
        test.testOp("-");
        test.testNumber(80);
        test.testOp("=");
        test.testOp("+");
        test.testOp("+");
        test.testOp("+");
        test.testOp("=");
    }
}
```

```
import java.awt.event.*;
import javax.swing.*;
import java.awt.GridLayout;

class FunctionButtonPanel extends JPanel
{
    private CalculatorModel myModel;

    /**
     * make model
     */

    FunctionButtonPanel(CalculatorModel model)
    {
        // construct superclass and make button array
        super(new GridLayout(4,4));
        myModel = model;
        makeButtons();
    }

    /**
     * Makes the buttons for this GUI/view. The number
     * of buttons is determined my the size of the array
     * myButtons. This helper function takes some of
     * the busy work out of the ButtonPanel constructor.
     */
    protected void makeButtons()
    {
        // make listener to do the move chosen by user
        ActionListener funcProc = new ActionListener(){
                public void actionPerformed(ActionEvent e)
                {
                    String func = e.getActionCommand();
                    myModel.unaryOp(func);
                }
            };

        String[] labels = {
            "e^x", "ln", "log",
            "sin", "cos", "tan",
            "n!", "1/x", "x^2",
            "x^3", "pi", "e"
        };

        for(int k=0; k < labels.length; k++){
            CalcButton button = new CalcButton(labels[k],labels[k]);
            button.setForeground(java.awt.Color.BLUE);
            button.addActionListener(funcProc);
            add(button);
        }
    }
}
```

```
public interface ICalculatorView
{
    public void display(String s);
}
```

```
import java.awt.event.*;
import javax.swing.*;
import java.awt.GridLayout;

class NumberButtonPanel extends JPanel
{
    private CalculatorModel myModel;

    /**
     * make model
     */

    NumberButtonPanel(CalculatorModel model)
    {
        // construct superclass and make button array
        super(new GridLayout(4,4));
        myModel = model;
        makeButtons();
    }

    /**
     * Makes the buttons for this GUI/view. The number
     * of buttons is determined my the size of the array
     * myButtons. This helper function takes some of
     * the busy work out of the ButtonPanel constructor.
     */
    protected void makeButtons()
    {
        // make listener to do the move chosen by user
        ActionListener digitProc = new ActionListener(){
                public void actionPerformed(ActionEvent e)
                {
                    String dig = e.getActionCommand();
                    myModel.addDigit(Integer.parseInt(dig));
                }
            };
        ActionListener opProc = new ActionListener(){
                public void actionPerformed(ActionEvent e)
                {
                    myModel.binaryOp(e.getActionCommand());
                }
            };

        String[] labels = {
            "7", "8", "9", "/",
            "4", "5", "6", "*",
            "1", "2", "3", "-",
            "0", "+/-","=","+"
        };

        for(int k=0; k < labels.length; k++){
            CalcButton button = new CalcButton(labels[k],labels[k]);
            String s = labels[k];
            if (s.equals("+/-")){
                button.addActionListener(
                            new ActionListener(){
                              public void actionPerformed(ActionEvent e)
                              {
                                  myModel.toggleSign();
                              }
                });
            }
            else if (Character.isDigit(s.charAt(0))){
                button.addActionListener(digitProc);
            }
            else {
                button.addActionListener(opProc);
            }
            add(button);
```



```
        }
    }
}
```