Educational Offering

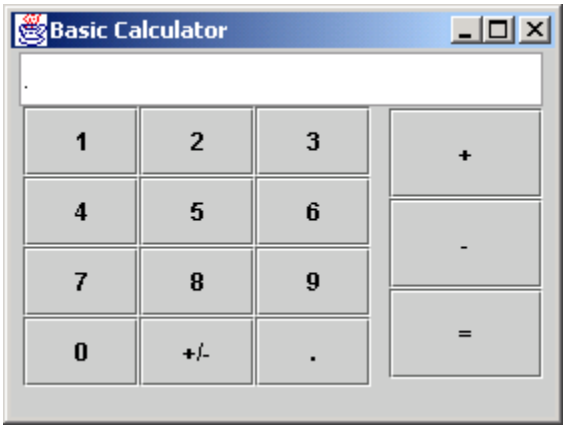
Design Pattern Primer
Exercise: Calculator Interface

Overview
In this exercise, you will explore the advantage of a composition-based design by applying a graphical user interface on the Calculator design.

User Requirement
You need to utilize Java compiler and byte code interpreter to create and compile classes, methods, and fields.

Introduction
Object composition allows for implementations that are more configurable. This exercise will reinforce this concept by presenting a Calculator user interface that utilizes composition to automatically react to arbitrary Operation instances, resulting in a configurable user interface.

A screen shot is shown below:

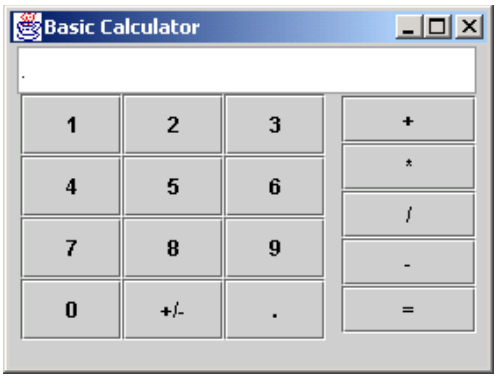


Source for this exercise is defined in the dp.lab.gui package.

Exercise Instructions
1. Execute the Calculator Interface
In the dp.lab.gui package, execute the BasicCalculator classes main method. Notice the calculator only supports add and subtract operations. The implementation is designed to

automatically utilize operations installed for a given Calculator instance. This occurs in the BasicCalculator main method. Modify this method so that division and multiplication operations are installed.

When installed, the BasicCalculator should look like this:



The main method of the BasicCalculator creates and assigns a Calculator reference to the CalculaterPanel instance that is added to a JFrame. Study the createOperationsPanel() method defined in the CalculatorPanel.

2. Create a Scientific Calculator
Using the same design, a ScientificCalculator can be created by implementing and executing the class definition shown below. Or copy the BasicCalculator and install the appropriate operations.

```
import dp.lab.strategy.*;


public class ScientificCalculator {

	/**
	 * Constructor for ScientificCalculator.
	 */
	public ScientificCalculator() {
		super();
	}

	public static void main(String[] args) {

		try {
			javax.swing.JFrame frame = new javax.swing.JFrame();
			frame.setTitle("Scientific Calculator");
			CalculatorPanel aCalculatorPanel;
			Calculator calc = new Calculator();
			calc.install(new AddOperation());
			calc.install(new SubtractOperation());
			calc.install(new MultiplyOperation());
			calc.install(new DivideOperation());
			calc.install(new LogOperation());
```

```
                calc.install(new TanOperation());
                calc.install(new SinOperation());

                aCalculatorPanel = new CalculatorPanel(calc);

                frame.setContentPane(aCalculatorPanel);
                frame.setSize(aCalculatorPanel.getSize());
                frame.addWindowListener(new
java.awt.event.WindowAdapter() {
                    public                                          void
windowClosing(java.awt.event.WindowEvent e) {
                        System.exit(0);
                    };
                });
                frame.show();
                java.awt.Insets insets = frame.getInsets();
                frame.setSize(
                    frame.getWidth() + insets.left + insets.right,
                    frame.getHeight() + insets.top + insets.bottom);
                frame.setVisible(true);
            } catch (Throwable exception) {
                System.err.println(
                    "Exception    occurred    in    main()    of
javax.swing.JPanel");
                exception.printStackTrace(System.out);
            }

        }
}
```