

# 70094 17

## Interactive Applications

### Submitters

Mark: 19/20

Very nice implementation! You've implemented more calculator functions than expected. However, you may need to pay a little attention to code duplication in tests. Well done!

---

**jh5723**

**sf23**

**James Hartley**

**Shihan Fu**

# Emarking

**Final Tests****TestSummary.txt: 1/1****James Hartley - jh5723:v5**

```
1: Final Tests: Summary for jh5723 of v5
2: -----
3:
4:   Public Tests:
5:     Compiles:      1 / 1           Great ✓
6:     Tests Pass:    1 / 1
7:     Style Checks:  1 / 1
8:
9: Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_spring/SE_Design_Ex7_jh5723.git
10: Commit ID: 61d0c
```

```

1: package ic.doc;
2:
3: import org.junit.Test;
4: import org.jmock.lib.concurrent.Synchroniser;
5: import org.jmock.Expectations;
6: import org.jmock.Mockery;
7: import static org.junit.Assert.fail;
8:
9: public class RpnCalculatorModelTest {
10:
11:     // Need synchroniser for mockery to ensure thread safety
12:     private Mockery context = new Mockery() {{
13:         setThreadingPolicy(new Synchroniser());
14:     }};
15:     private final Observer mockObserver = context.mock(Observer.class);
16:
17:     @Test
18:     public void pushValueNotifiesObserver() {
19:         RpnCalculatorModel calculator = new RpnCalculatorModel();
20:         calculator.addObserver(mockObserver);
21:
22:         double value = 5.0;
23:
24:         context.checking(new Expectations() {{
25:             oneOf(mockObserver).update(value);
26:         }});
27:
28:         calculator.pushValue(value);
29:
30:         context.assertIsSatisfied();
31:     }
32:
33:     @Test
34:     public void performOperationNotifiesObserverWithResult() {
35:         RpnCalculatorModel calculator = new RpnCalculatorModel();
36:         calculator.addObserver(mockObserver);
37:
38:         final double value1 = 5.0;
39:         final double value2 = 10.0;
40:         final double expectedResult = 15.0;
41:
42:         context.checking(new Expectations() {{
43:             exactly(1).of(mockObserver).update(5.0);
44:             exactly(1).of(mockObserver).update(10.0);
45:             exactly(1).of(mockObserver).update(expectedResult);
46:         }});
47:
48:         calculator.pushValue(value1);
49:         calculator.pushValue(value2);
50:         calculator.performOperation("+");
51:
52:         context.assertIsSatisfied();
53:     }
54:
55:     @Test
56:     public void canClearStack() {
57:         RpnCalculatorModel calculator = new RpnCalculatorModel();
58:         calculator.addObserver(mockObserver);
59:
60:         context.checking(new Expectations() {{
61:             exactly(1).of(mockObserver).update(5.0);
62:             exactly(1).of(mockObserver).update(10.0);
63:         }});
64:
65:         calculator.pushValue(5.0);
66:         calculator.pushValue(10.0);

```

Good to use mock object to test in isolation

You could move the setup outside the methods to avoid code duplication

```

67: calculator.clearStack();
68: try {
69:     calculator.performOperation("+");
70:     fail("Expected IllegalStateException to be thrown");
71: } catch (IllegalStateException e) {
72:     System.out.println("Expected IllegalStateException was caught: "
73:         + e.getMessage());
74: }
75:
76: context.assertIsSatisfied();
77: }
78:
79: @Test
80: public void addsTogetherTopTwoNumbersOnStack() {
81:     RpnCalculatorModel calculator = new RpnCalculatorModel();
82:     calculator.addObserver(mockObserver);
83:
84:     context.checking(new Expectations() {{
85:         exactly(1).of(mockObserver).update(1.0);
86:         exactly(1).of(mockObserver).update(2.0);
87:         exactly(1).of(mockObserver).update(3.0);
88:     }});
89:
90:     calculator.pushValue(1.0);
91:     calculator.pushValue(2.0);
92:     calculator.performOperation("+");
93:
94:     context.assertIsSatisfied();
95: }
96:
97: @Test
98: public void multipliesTogetherTopTwoNumbersOnStack() {
99:     RpnCalculatorModel calculator = new RpnCalculatorModel();
100:    calculator.addObserver(mockObserver);
101:
102:    context.checking(new Expectations() {{
103:        exactly(1).of(mockObserver).update(4.0);
104:        exactly(1).of(mockObserver).update(2.0);
105:        exactly(1).of(mockObserver).update(8.0);
106:    }});
107:
108:    calculator.pushValue(4.0);
109:    calculator.pushValue(2.0);
110:    calculator.performOperation("*");
111:
112:    context.assertIsSatisfied();
113: }
114:
115: @Test
116: public void subtractsTopNumberOnStackFromOneBeneath() {
117:     RpnCalculatorModel calculator = new RpnCalculatorModel();
118:     calculator.addObserver(mockObserver);
119:
120:     context.checking(new Expectations() {{
121:         exactly(1).of(mockObserver).update(5.0);
122:         exactly(1).of(mockObserver).update(2.0);
123:         exactly(1).of(mockObserver).update(3.0);
124:     }});
125:
126:     calculator.pushValue(5.0);
127:     calculator.pushValue(2.0);
128:     calculator.performOperation("-");
129:
130:     context.assertIsSatisfied();
131: }
132:

```

Final Tests RpnCalculatorModelTest.java: 3/3 James Hartley - jh5723:v5

```
133:  @Test
134:  public void dividesSecondNumberOnStackByFirst() {
135:      RpnCalculatorModel calculator = new RpnCalculatorModel();
136:      calculator.addObserver(mockObserver);
137:
138:      context.checking(new Expectations() {{
139:          exactly(1).of(mockObserver).update(6.0);
140:          exactly(1).of(mockObserver).update(2.0);
141:          exactly(1).of(mockObserver).update(3.0);
142:      }});
143:
144:      calculator.pushValue(6.0);
145:      calculator.pushValue(2.0);
146:      calculator.performOperation("/");
147:
148:      context.assertIsSatisfied();
149:  }
150: }
```

Final Tests RpnCalculatorGuiTest.java: 1/1 James Hartley - jh5723:v5

```
1:  // NOTE TO EXAMINER -- We have commented out this test as, even with
2:  // java.awt.headless set to true, LabTS throws an exception
3:
4:  //package ic.doc;
5:  //
6:  //import org.junit.Test;
7:  //import static org.junit.Assert.assertEquals;
8:  //import org.junit.BeforeClass;
9:  //
10: //public class RpnCalculatorGuiTest {
11: //
12: //    @BeforeClass
13: //    public static void setUp() {
14: //        System.setProperty("java.awt.headless", "true");
15: //    }
16: //
17: //    @Test
18: //    public void updateUpdatesDisplay() {
19: //        RpnCalculatorModel model = new RpnCalculatorModel();
20: //        RpnCalculatorGui gui = new RpnCalculatorGui(model) {
21: //
22: //            // Directly verify update
23: //            @Override
24: //            public void update(double result) {
25: //                super.update(result);
26: //                assertEquals("Expected display text", String.valueOf(result),
27: //                    this.display.getText());
28: //            }
29: //        };
30: //        gui.update(5.0);
31: //    }
32: //}
33:
```

## Final Tests

RpnCalculatorModel.java: 1/2

James Hartley - jh5723:v5

```

1: /**
2:  * Represents the model component of a Reverse Polish Notation (RPN) calculator.
3:  * This class manages the arithmetic operations using a stack to store numerical
4:  * values. It implements the Observable interface to notify observers about
5:  * changes in the calculation result. Observers are notified whenever a new
6:  * value is pushed onto the stack or after the top of the stack is changed.
7:  */
8:
9: package ic.doc;
10:
11: import java.util.ArrayList;
12: import java.util.List;
13: import java.util.Stack;
14:
15: public class RpnCalculatorModel implements Observable {
16:     private Stack<Double> stack = new Stack<>();
17:     private List<Observer> observers = new ArrayList<>(); Good to set a list of observers instead of one single observer
18:
19:     // Function to push displayed number to the stack
20:     public void pushValue(double value) {
21:         stack.push(value);
22:         notifyObservers();
23:     }
24:
25:     // Function defining behaviour of operation buttons
26:     public void performOperation(String operation) {
27:
28:         if (stack.size() == 0) {
29:             throw new IllegalStateException("Error: Insufficient operands");
30:         }
31:         if (stack.size() < 2) {
32:             return; // Allows operation on
33:         }
34:         double b = stack.pop();
35:         double a = stack.pop();
36:         double result = 0;
37:         switch (operation) {
38:             case "+":
39:                 result = a + b;
40:                 break;
41:             case "-":
42:                 result = a - b;
43:                 break;
44:             case "*":
45:                 result = a * b;
46:                 break;
47:             case "/":
48:                 result = a / b;
49:                 break;
50:             default:
51:                 throw new IllegalArgumentException("Unsupported operation: "
52:                     + operation);
53:         };
54:         stack.push(result); Great!
55:         notifyObservers();
56:     }
57:
58:     public void clearStack() {
59:         stack.clear();
60:     }
61:
62:     @Override
63:     public void addObserver(Observer observer) {
64:         observers.add(observer);
65:     }
66:

```

## Final Tests

RpnCalculatorModel.java: 2/2

James Hartley - jh5723:v5

```

67:     @Override
68:     public void removeObserver(Observer observer) {
69:         observers.remove(observer);
70:     }
71:
72:     @Override
73:     public void notifyObservers() {
74:         double result = stack.isEmpty() ? 0 : stack.peek();
75:         for (Observer observer : observers) {
76:             observer.update(result);
77:         }
78:     }
79: }

```

## Final Tests

RpnCalculatorGui.java: 1/2

James Hartley - jh5723:v5

```

1: /**
2:  * Represents the Graphical User Interface (GUI) for a Reverse Polish Notation
3:  * calculator, with a visual interface for users to input numbers and
4:  * operations. It subscribes to the RpnCalculatorModel as an observer,
5:  * sending values and operations to the model and updating the display in
6:  * response to changes in the model's state.
7:  */
8:
9: package ic.doc;
10:
11: // Swing classes implemented
12: import javax.swing.JFrame;
13: import javax.swing.JButton;
14: import javax.swing.JTextArea;
15: import javax.swing.JPanel;
16:
17: // awt classes implemented
18: import java.awt.BorderLayout;
19: import java.awt.GridLayout;
20: import java.awt.event.ActionEvent;
21: import java.awt.event.ActionListener;
22:
23:
24: public class RpnCalculatorGui implements Observer {
25:     JTextArea display; It'd be better to write the access modifier explicitly
26:     private RpnCalculatorModel rpnCalculator;
27:     private JFrame frame;
28:
29:     public RpnCalculatorGui(RpnCalculatorModel model) {
30:         this.rpnCalculator = model;
31:         this.rpnCalculator.addObserver(this); // Observer to the RpnCalculatorModel
32:         frame = new JFrame("Reverse Polish Notation Calculator");
33:         frame.setSize(300, 200);
34:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35:         initializeUi();
36:     }
37:
38:     // Function to create the buttons and display and add ActionListeners
39:     private void initializeUi() {
40:         frame.setLayout(new BorderLayout());
41:         display = new JTextArea();
42:         frame.add(display, BorderLayout.NORTH);
43:
44:         JPanel buttonPanel = new JPanel();
45:         buttonPanel.setLayout(new GridLayout(4, 4));
46:         frame.add(buttonPanel, BorderLayout.CENTER);
47:
48:         String[] buttons = {"1", "2", "3", "+",
49:                             "4", "5", "6", "-",
50:                             "7", "8", "9", "*",
51:                             "0", "Enter", "C", "/"};
52:         for (String buttonText : buttons) {
53:             JButton button = new JButton(buttonText);
54:             button.addActionListener(new ActionListener() {
55:                 @Override
56:                 public void actionPerformed(ActionEvent e) {
57:                     buttonPressed(e);
58:                 }
59:             });
60:             buttonPanel.add(button);
61:         }
62:     }
63:
64:     // Function to communicate keystrokes to the model
65:     private void buttonPressed(ActionEvent e) {
66:         String command = e.getActionCommand();

```

## Final Tests

RpnCalculatorGui.java: 2/2

James Hartley - jh5723:v5

```

67:     if ("Enter".equals(command)) {
68:         try {
69:             double value = Double.parseDouble(display.getText());
70:             rpnCalculator.pushValue(value);
71:             display.setText(""); Great!
72:         } catch (NumberFormatException ex) {
73:             display.setText("Error");
74:         }
75:     } else if ("C".equals(command)) {
76:         // "C" clears the stack and display
77:         rpnCalculator.clearStack();
78:         display.setText("");
79:     } else if ("+" .equals(command) || "-" .equals(command)
80:               || "*" .equals(command) || "/" .equals(command)) {
81:         // Attempts to apply operation to most recent pair of values in the stack
82:         rpnCalculator.performOperation(command);
83:     } else {
84:         // Append digits to the display
85:         display.append(command);
86:     }
87: }
88:
89: @Override
90: public void update(double result) {
91:     display.setText(String.valueOf(result));
92: }
93:
94: public void show() {
95:     frame.setVisible(true);
96: }
97: }
98:

```

## Final Tests

Observer.java: 1/1

James Hartley - jh5723:v5

```
1: /**
2:  * Observer interface for receiving updates about changes in a subject's state.
3:  * Implementations of this interface should define how to handle the updated
4:  * result. The update method is called with the latest result value whenever
5:  * the subject's state changes.
6:  */
7:
8: package ic.doc;
9:
10: public interface Observer {
11:     void update(double result);
12: }
```

Great!

## Final Tests

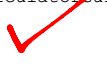
Observable.java: 1/1

James Hartley - jh5723:v5

```
1: /**
2:  * Observable interface for objects that can be observed by Observers.
3:  * This interface declares methods for adding and removing Observers,
4:  * as well as notifying them of any state changes. Implementing classes
5:  * should maintain a list of Observers and call their update method
6:  * when there are relevant changes to notify them about.
7:  */
8:
9: package ic.doc;
10:
11: public interface Observable {
12:     void addObserver(Observer observer);
13:
14:     void removeObserver(Observer observer);
15:
16:     void notifyObservers();
17: }
```

Good design

```
1: package ic.doc;
2:
3: import javax.swing.SwingUtilities;
4:
5: public class Main {
6:     public static void main(String[] args) {
7:         // Place GUI initialization code inside a Runnable to ensure thread safety
8:         SwingUtilities.invokeLater(() -> {
9:             RpnCalculatorModel model = new RpnCalculatorModel();
10:            RpnCalculatorGui calculatorGui = new RpnCalculatorGui(model);
11:            calculatorGui.show();
12:        });
13:    }
14: }
```





## Final Tests

testResults.txt: 1/1

James Hartley - jh5723:v5

```
1: ----- Test Output -----
2: Running LabTS build... (Thu 29 Feb 14:51:08 UTC 2024)
3:
4: Submission summary...
5: You made 6 commits
6:   - 274b316 Feat: adds combinations of 1 and 2 in Reverse Polish Notation [8 files changed, 277 insertions, 4 deletions]
7:   - 0bcbc12 Feat: can clear stack [3 files changed, 50 insertions, 15 deletions]
8:   - 50e9111 Feat: multiplies, divides, subtracts and adds integers inputted through integer keys 0-9 [3 files changed, 100 insertions, 27 deletions]
9:   - efbf0fc Fix: sets aws property to headless in Gui tests to pass on LabTS [1 file changed, 6 insertions, 1 deletion]
10:  - 7e51217 Fix: Deleted GUI updateDisplay test as cannot be passed on LabTS, even with headless set to true [1 file changed, 29 deletions]
11:  - 61d0c53 Refactor: created display helper functions and extracted GUI main function to Main.java [4 files changed, 50 insertions, 9 deletions]
12:
13: Preparing...
14:
15: BUILD SUCCESSFUL in 649ms
16:
17: Compiling...
18: BUILD SUCCESSFUL in 4s
19:
20: Running tests...
21:
22: ic.doc.RpnCalculatorModelTest > multipliesTogetherTopTwoNumbersOnStack PASSED
23:
24: ic.doc.RpnCalculatorModelTest > dividesSecondNumberOnStackByFirst PASSED
25:
26: ic.doc.RpnCalculatorModelTest > subtractsTopNumberOnStackFromOneBeneath PASSED
27:
28: ic.doc.RpnCalculatorModelTest > addsTogetherTopTwoNumbersOnStack PASSED
29:
30: ic.doc.RpnCalculatorModelTest > performOperationNotifiesObserverWithResult PASSED
31:
32: ic.doc.RpnCalculatorModelTest > canClearStack PASSED
33:
34: ic.doc.RpnCalculatorModelTest > pushValueNotifiesObserver PASSED
35:
36: BUILD SUCCESSFUL in 1s
37:
38: Checking code style...
39: BUILD SUCCESSFUL in 5s
40: Finished auto test. (Thu 29 Feb 14:51:30 UTC 2024)
41:
42: ----- Test Errors -----
43:
```

Clear commit messages