Educational Offering

Design Pattern Primer

Exercise: Observer

Overview
In this exercise, you will apply the Observer pattern to the calculator implementation.

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

Introduction
In order to help reinforce understanding of this pattern, this exercise will only provide sparse instructions requiring the student to fill in the gaps. Instructions are provided below.

Design Requirement
The Calculator implementation needs to be refactored so that any number of objects can be notified when operations, results are put into and recalled from memory. When these actions occur, they will be reported to the console in XML and text form.

Two types of observer classes are required; one that will output calculator changes to the console in the XML format, and the other will output in text format. Format examples are shown below.

XML tag format…
```
<operation>
   +
</operation>
<memory>
  100
</memory>
<recall>
  200
</recall>
```

Text format…
```
operation = +
memory = 100
recall =200
```

Restrictions

- You must use the Observer pattern.
- Outputting the XML and text structures must be defined in separate classes that do not appear in the same hierarchy.

Hints

Calculator class will need to be modified to hold a collection of observable instances. In addition, the ability to add observer instances to the calculator is required.

- The Observer class is an interface that defines methods for each type of notification (memory, recall, operation). The argument for these interface methods will contain operation specific data that is formatted for display

- Two classes for format type will be created and will implement the Observer interface.

- The Tester class main method will be modified to accept the Observer instances. An example is shown below.

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

    // basic calculator

    System.out.println("* * Basic Calculator * *");

    Calculator calc =
ConcreteCalculatorFactory.getInstance().basic();

    // add Observers

    calc.addObserver( new XMLObserver() );
    calc.addObserver( new TextObserver() );

    // Create Memory caretaker

    CalculatorMemory memory = new CalculatorMemory();

    calc.execute("+", 10.0);
    // remember result
    memory.remember(calc);
    calc.execute("/", 2.0);
    // recall from memory
    memory.recall(calc);
    calc.execute("*",5.0);
    calc.print();


}
```