Educational Offering

Design Pattern Primer

Exercise: Composite

Overview
In this exercise, you will apply the Composite 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.

The composite pattern will allow additional functionality to be added without having to change the Calculator instance. Instead an existing hierarchy will utilize the recursive composition attribute of the pattern to provide an parent/child operation structure.

Design Requirement
The Calculator implementation in this exercise has been modified to execute instances of the Operation class. This means that a BinaryOperation can be executed using the expression below:

```
BinaryOperation o = new BinaryOperation();
aCalculator.exec(o);
```

The intent of the Composite pattern is to allow uniform treatment of part-whole object structures. The Calculator can utilize this pattern to introduce a Formula operation that is treated and executed in the same way that an individual operation is executed. Example usage expression is shown below.

```
FormulaOperation formula = new FormulaOperation();
formula.add(new AddOperation(),100.00);
formula.add(new SubtractOperation(),200.00);

aCalculator.exececute(formula); // performs formula operations
```

The FormulaOperation class has not yet been implemented; this is task of this lab. The example shows single Operation instances can be added to instances of FormulaOperations along with a decimal value. Obviously, this implementation will be implemented by the FormulaOperation class.

Restrictions

- You cannot change the implementation of the Calculator class. Calculator instances utilize the abstract Operation execute behavior.

Hints

1. The FormulaOperation class is a subclass in a current hierarchy.
2. FormulaOperation instances will reference multiple Operation instances, along with a decimal value to perform the operations against. You will have to utilize some structure to hold both of these values, and of preserver the order in which they have been added.
3. When FormulaOperations are sent the execute() method they will have to perform all of their referenced operations. Remember you will have reference to a Calculator instance. Therefore, you can call the appropriate execute() method defined by a Calculator instance.