MSc Java Lab - Calculator - Day 1

Robert Chatley and William Lee

1 Aims

The aim of this lab is to allow you to gain experience using the Java language. You will write a program comprising a number of related classes, compile and run them. In this way you will become familiar with using the tools that come with the Java Development Kit. We will develop a very simple calculator. We suggest you work in pairs.

2 Getting started

Use a Linux machine (Java is cross platform and available on the Windows machines too, but for simplicity and consistency we will specify the lab exercise in the Linux environment).
Create a new directory for the exercise.
Create a file called Calculator.java using your favourite text editor (vi, emacs, nedit etc.)
Create a public class called **Calculator** with a **main** method in this file
In the body of this method, print out a message.
Save the file.
cd into your directory for the exercise.
Compile your file with the command **javac Calcluator.java**
Examine the files now in the directory.
Run your program with **java Calculator**

3 Creating a Set of Classes

Create a new directory called **operators** inside your working directory. This will correspond to a **package** that will contain all the mathematical operators for our calculator. Create an **abstract** class **Operator**, in its own file, with an abstract method **apply** that takes two integers as parameters and returns a resultant integer.
Now create four subclasses of Operator called Plus, Minus, Multiply and Divide (in their own files) in the **package** operators. Implement appropriate apply() methods in each subclass.

Back in your Calculator class, **import** all the operators in from your operators package. In your main() method, instantiate one of your operators. Use the apply() method to do a calculation. Print the result.

4 Expressions

We will now develop the program so that we can make and evaluate more complex expressions using our operators. Create another package, called **expressions** at the same level as the operators package. Inside this package create an abstract class **Expression** with an abstract method evaluate(), returning an int. Create two subclasses of Expression called BinaryExpression and Value. Add a **private field** and a **constructor** to Value, so that a Value can be created that represents an integer. Value's evaluate() method should return this integer.

The BinaryExpression class should have one private field holding an Operator and two others representing sub Expressions. The evaluate() method should return the result of applying the Operator to the two sub-expressions. Add a suitable constructor too.

You will need to update all of your Operators so that their apply methods take Expression parameters rather than integers.

Again back in Calculator, import your expressions package, create some expressions and evaluate them. Can you represent 2 + 3 * 4 + 5 and (2 + 3) * (4 + 5)?

5 Interfaces

We will now add a capability to expressions so that they can be displayed. Create an **interface** Printable (in its own .java file) in a new package called printing, with one method, show(). This method should print a representation of the expression on the screen. Make all of your expression and operator classes **implement** the Printable interface. This will work in a similar way to evaluate().

You should now be able to print both an expression and the result of evaluating it, e.g. 2 + 3 + 4 = 9