

Expression Implementation (Version 4)

```
public abstract class Expression4 {  
    public abstract void accept(ExpressionVisitor visitor);  
}
```

```
public class Constant4 extends Expression4 {  
  
    int value;  
  
    public Constant4(int v) {  
        value = v;  
    }  
  
    public void accept(ExpressionVisitor visitor) {  
        visitor.visitConstant(this);  
    }  
}
```

```
public class Addition4 extends Expression4 {  
  
    Expression4 left, right;  
  
    public Addition4(Expression4 l, Expression4 r)  
    {  
        left = l;  
        right = r;  
    }  
  
    public void accept(ExpressionVisitor visitor) {  
        visitor.visitAddition(this);  
    }  
}
```

Classes for Multiplication and Negation are similar.

```
public interface ExpressionVisitor {  
  
    public void visitConstant(Constant4 c);  
    public void visitAddition(Addition4 a);  
    public void visitMultiplication(Multiplication4 m);  
    public void visitNegation(Negation4 n);  
}
```

```
public class ExpressionEvaluator implements  
ExpressionVisitor {  
  
    /** The calculated value */  
    private int value;  
  
    /** public view of the value */  
    public int getValue() { return value; }  
  
    /** Get the value of a constant expression */  
    public void visitConstant(Constant4 c) {  
        value = c.value;  
    }  
  
    /** Get the value of a sum */  
    public void visitAddition(Addition4 a) {  
        ExpressionEvaluator leftEvaluator = new  
        ExpressionEvaluator();  
        ExpressionEvaluator rightEvaluator = new  
        ExpressionEvaluator();  
        a.left.accept(leftEvaluator);  
        a.right.accept(rightEvaluator);  
        value = leftEvaluator.value +  
        rightEvaluator.value;  
    }  
  
    /** Get the value of a product */  
    public void visitMultiplication(Multiplication4 a) {  
        // similar to Addition...  
        value = leftEvaluator.value * rightEvaluator.value;  
    }  
  
    /** Get the value of a negation */  
    public void visitNegation(Negation4 n) {  
        n.expression.accept(this); value = -value; }  
}
```

```

public class ExpressionFormatter implements
ExpressionVisitor {

    /** The formatted string being built */
    private String string;

    /** Public access to the string */
    public String getString() { return string; }

    /** Initialise with an empty string */
    public ExpressionFormatter() {
        string = "";
    }

    /** Get the string for a constant */
    public void visitConstant(Constant4 c) {
        string += c.value;
    }

    /** Build the string for a sum */
    public void visitAddition(Addition4 a) {
        string += "(";
        a.left.accept(this);
        string += " + ";
        a.right.accept(this);
        string += ")";
    }

    /** Build the string for a product */
    public void visitMultiplication(Multiplication4
a) {
        string += "(";
        a.left.accept(this);
        string += " * ";
        a.right.accept(this);
        string += ")";
    }

    /** Build the string for a negation */
    public void visitNegation(Negation4 n) {
        string += "-(";
        n.expression.accept(this);

```

```

        string += ")";
    }
}

```

Using Expression (Version 4)

```

Expression4 a, b, c, d, e;
ExpressionFormatter f = new ExpressionFormatter();
ExpressionEvaluator v = new ExpressionEvaluator();

a = new Constant4(1);
b = new Constant4(2);
c = new Addition4(a, b);
a = new Constant4(3);
b = new Constant4(4);
d = new Addition4(a, b);
e = new Multiplication4(c, d);

e.accept(f);
e.accept(v);
System.out.println(f.getString() + " = " +
    v.getValue());

```
