Module Title: Informatics 1 — OOP (Resit)
Exam Diet (Dec/April/Aug): August 2010

Brief notes on answers

Given the limited nature of the tasks in these questions, it is difficult for students to demonstrate excellence over and above giving correct answers. Therefore, in general a set of answers that produces the correct results will receive full marks. Question (1.c) is intentionally more challenging than the preceding questions.

Answers which fail to produce correct results will still be given partial credit, according to the following guidelines:

- If there appears to be a good understanding of the logic of the task, but there are minor syntactic errors, then 25% will be deducted from the marks available.
- If there is a significant error in the logic of the task, then 50% will be deducted from the marks available.

Obviously, these penalties will be adjusted according to the severity of the problem.

Question 1 Solutions

1.
```
public class OneA {

    /**
     * @param a A list of integers
     * @return the sum of the elements in a
     */
    public int product( int [] a ) {

        int product = 1;
        for ( int i = 0; i < a.length; i++ ) {
            product *= a[i];
        }
        return product;
    }
}
```

2.
```
public class OneB {

        /**
         * @param a A list of integers
         * @param b A list of integers
         * @return A list of integers. If a[i] and b[i] differ then the element returned
         *         is the sum of the two, otherwise it is just a[i]
         */
        public int [] sumDiffs( int [] a, int [] b ) {

                int [] sum = new int [a.length];
                for ( int i = 0; i < a.length; i++ ) {
                        if ( a[i] == b[i] ) {
                                sum[i] = a[i];
                        } else {
                                sum[i] = a[i] + b[i];
                        }
                }
                return sum;
        }
}
```

3.
```
public class OneC {

        public int [] stretch(int[] a) {

                int [] stretch = new int [a.length * 2];
                for ( int i = 0; i < a.length; i++ ) {
                        stretch[ i * 2 ] = a[i]/2 + a[i]%2;
                        stretch[ i * 2 + 1] = a[i]/2;
                }
                return stretch;

        }
}
```

4. The main rationale for this task is to encourage students to budget some time for testing. They have seen many examples of "launcher" code which runs some methods inside `main()`. If they have two reasonable-looking calls for each of their three methods, they will receive full marks.

Question 2 Solutions

Files Supplied

`Operation.java`

```
public interface Operation {

    int apply(Expression left, Expression right);

    String toString();

}
```

`Literal.java`

```
public class Literal implements Expression {

    private int value;

    public Literal(int value) {
        this.value = value;
    }

    public int getValue() {
        return value;
    }

    public String toString() {
        return Integer.toString(value);
    }
}
```

`Addition.java`

```
public class Addition implements Operation {

    public int apply(Expression left, Expression right) {
        return left.getValue() + right.getValue();
    }

    public String toString() {
        return "+";
    }
}
```

Subtraction.java

```
public class Subtraction implements Operation {

    public int apply(Expression left, Expression right) {
        return left.getValue() - right.getValue();
    }

    public String toString() {
        return "-";
    }

}
```

Files to be Submitted

Expression.java

```
public interface Expression {

    public String toString();

    public int getValue();

}
```

Division.java

```
public class Division implements Operation {

    public int apply(Expression left, Expression right) {
        return left.getValue() / right.getValue();
    }

    public String toString() {
        return "/";
    }
}
```

Multiplication.java

```
public class Multiplication implements Operation {

    public int apply(Expression left, Expression right) {
        return left.getValue() * right.getValue();
    }

    public String toString() {
        return "*";
    }
}
```

CompoundExpression.java

```
public class CompoundExpression implements Expression {

    Expression left;
    Expression right;
    Operation op;

    public CompoundExpression(Expression left, Operation op, Expression right) {
        this.left = left;
        this.right = right;
        this.op = op;
      }

    public int getValue() {
        return op.apply(left, right);
    }

    public String toString() {
        return String.format("(%s %s %s)", left, op, right);
    }

}
```

CalculatorLauncher.java

```
public class CalculatorLauncher {

   public static void main(String[] args) {
        Expression e1 = new Literal(1);
        Expression e2 = new Literal(2);
        Expression e3 = new Literal(3);
        Expression e4 = new Literal(4);
        Expression e5 = new Literal(5);

        Operator a = new Add();
        Operator s = new Subtract();
        Operator m = new Mult();
        Operator d = new Divide();

        Expression ce1 = new CompoundExpression(e1, s, e5); // 1 - 2
        Expression ce2 = new CompoundExpression(e4, d, e2); // 4 / 5
        Expression ce3 = new CompoundExpression(ce1, a, e3); // (1 - 2) + 3
        Expression ce4 = new CompoundExpression(ce2, m, ce3); // (4 / 5) * ((1 - 2) + 3)

        // alternative presentation
        Expression ce01 = new CompoundExpression(new Literal(3), a,
                new Literal(10)); // 3 +10

        Expression ce02 = (new CompoundExpression(new CompoundExpression(
                new Literal(3), a, new Literal(10)), s, new Literal(17))); // (3 + 10) - 17


        Expression exps[] = { ce1, ce2, ce3, ce4, ce01, ce02 };

        for (Expression e : exps) {
            System.out.println(e + " = " + e.getValue());
        }
    }
}
```