Домашнее задание № 2

Двоичное дерево представлено структурой, в которой узлы бывают двух видов: листья и промежуточные узлы. Каждый из типов узлов наследует общему предку `Node`, в котором определена функция обработки разного вида узлов. Обработка листьев осуществляется функцией типа `Function<V,R>`, получающей на вход значение типа `V`, хранящееся в узле, и выдающей результат некоторого определенного типа `R`. Обработка промежуточных узлов осуществляется функцией специального типа

```
@FunctionalInterface
public interface TreeFunction<T,R> {
     R apply(T arg1, R agr2, R arg3);
}
```

которая получает на вход значение, хранящееся в узле, и результаты обработки левого и правого узлов, и также выдает значение типа `R`. Таким образом, тип данных абстрактного узла будет выглядеть так:

```
public interface Node<V, T> {
     <R> R process(Function<V,R> leafProcessor,
                   TreeFunction<T,R> biNodeProcessor);
}
```

Тип листьев определен следующим образом:

```
public class Leaf<V, T> implements Node<V, T> {
     private V leafInfo;
     public Leaf(V leaf) { leafInfo = leaf; }
     public <R> R process(Function<V,R> leafProcessor,
                          TreeFunction<T,R> biNodeProcessor) {
          return leafProcessor.apply(leafInfo);
     }
}
```

а тип промежуточных узлов так:

```
public class BiNode<V, T> implements Node<V, T> {
     private T binInfo;
     private Node<V, T> left, right;

     public BiNode(T info, Node<V, T> left, Node<V, T> right) {
          binInfo = info;
          this.left = left;
          this.right = right;
     }

     public <R> R process(Function<V,R> leafProcessor,
```

```
                        TreeFunction<T,R> biNodeProcessor) {
        R res1 = left.process(leafProcessor, biNodeProcessor);
        R res2 = right.process(leafProcessor, biNodeProcessor);
        return biNodeProcessor.apply(binInfo, res1, res2);
    }
}
```

Требуется построить дерево некоторого выражения, листьями которого являются целые числа, а в промежуточных узлах находятся знаки операций `'+'`, `'-'` или `'*'`, и с помощью метода `process` написать следующие способы обработки дерева:

- вычисление значения выражения;
- преобразование дерева в символьную строку;
- построение нового дерева, в котором структура остается той же самой, но все значения в листьях инвертированы относительно нуля.

Программа должна содержать комментарии по существу работы алгоритма **на английском языке**!