At this lab section, we will experiment evaluation trees.

Expression Trees

Asst. Prof. Dr. Feriştah DALKILIÇ
Res. Asst. Fatih DİCLE

PART 1 – Expression Tree

You are given some necessary interfaces and classes to build an expression tree, and some parts are left blank for you to fill them according to the instructions. After completing the blank parts, you should generate an expression tree for the below equation, display the prefix and postfix form of the given expression and evaluate the result.

$$(5 * 2 - 5) * 2 - 4 / 2 + 9 / 3$$

You are expected to fill the given missing parts:

In `class DriverET`

```
public static void main(String[] args)
{

BinaryNode<String> lf1 = new BinaryNode<>("5");
BinaryNode<String> lf2 = new BinaryNode<>("2");
BinaryNode<String> lf3 = new BinaryNode<>("5");
BinaryNode<String> lf4 = new BinaryNode<>("2");
BinaryNode<String> lf5 = new BinaryNode<>("4");
BinaryNode<String> lf6 = new BinaryNode<>("2");
BinaryNode<String> lf7 = new BinaryNode<>("9");
BinaryNode<String> lf8 = new BinaryNode<>("3");

BinaryNode<String> mulnd1 = new BinaryNode<>("*",lf1,lf2);
BinaryNode<String> subnd1 = new BinaryNode<>("-",mulnd1,lf3);
BinaryNode<String> mulnd2 = new BinaryNode<>("*",subnd1,lf4);
BinaryNode<String> divnd1 = new BinaryNode<>("/",lf5,lf6);
BinaryNode<String> divnd2 = new BinaryNode<>("/",lf7,lf8);
BinaryNode<String> sumnd1 = new BinaryNode<>("+",divnd1,divnd2);

BinaryNode<String> rootNode = new BinaryNode<>("-",mulnd2,sumnd1);
ExpressionTree myTree = new ExpressionTree();
myTree.setRootNode(rootNode);
myTree.displayPrefixExpression();
myTree.displayPostfixExpression();
System.out.println(myTree.evaluate());
}
```

In `class ExpressionTree`

```
  private double evaluate(BinaryNode<String> rootNode)
   {
       double result;
      if (rootNode == null)
           result = 0;
        else if (rootNode.isLeaf())
        {
           String variable = rootNode.getData();
           result = Integer.parseInt(variable);
        }
        else
        {
      String operator = rootNode.getData();
BinaryNode<String> left = rootNode.getLeftChild();
BinaryNode<String> right = rootNode.getRightChild();
double leftResult = evaluate(left);
double rightResult = evaluate(right);
result = compute(operator, leftResult, rightResult);
          }

        return result;
    }
```

```
private void postorder(BinaryNode<String> rootNode)
    {
            if (rootNode != null) {
  postorder(rootNode.getLeftChild());
  postorder(rootNode.getRightChild());
  System.out.print(rootNode.getData() + " ");

    }
```

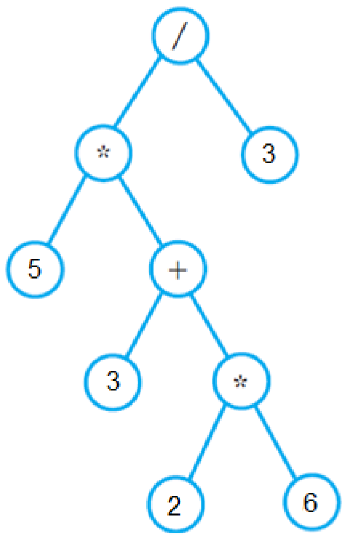
```
   private void preorder(BinaryNode<String> rootNode)
    {
            if (rootNode != null) {
  System.out.print(rootNode.getData() + " ");
  preorder(rootNode.getLeftChild());
  preorder(rootNode.getRightChild());
}

    }
```

Example Scenario:

To illustrate, you are given an equation as: 5 * (3 + 2 * 6) / 3

It can be represented as an expression tree below:



Example Output:

prefix form: / * 5 + 3 * 2 6 3

postfix form: 5 3 2 6 * + * 3 /

```
result: 25
```