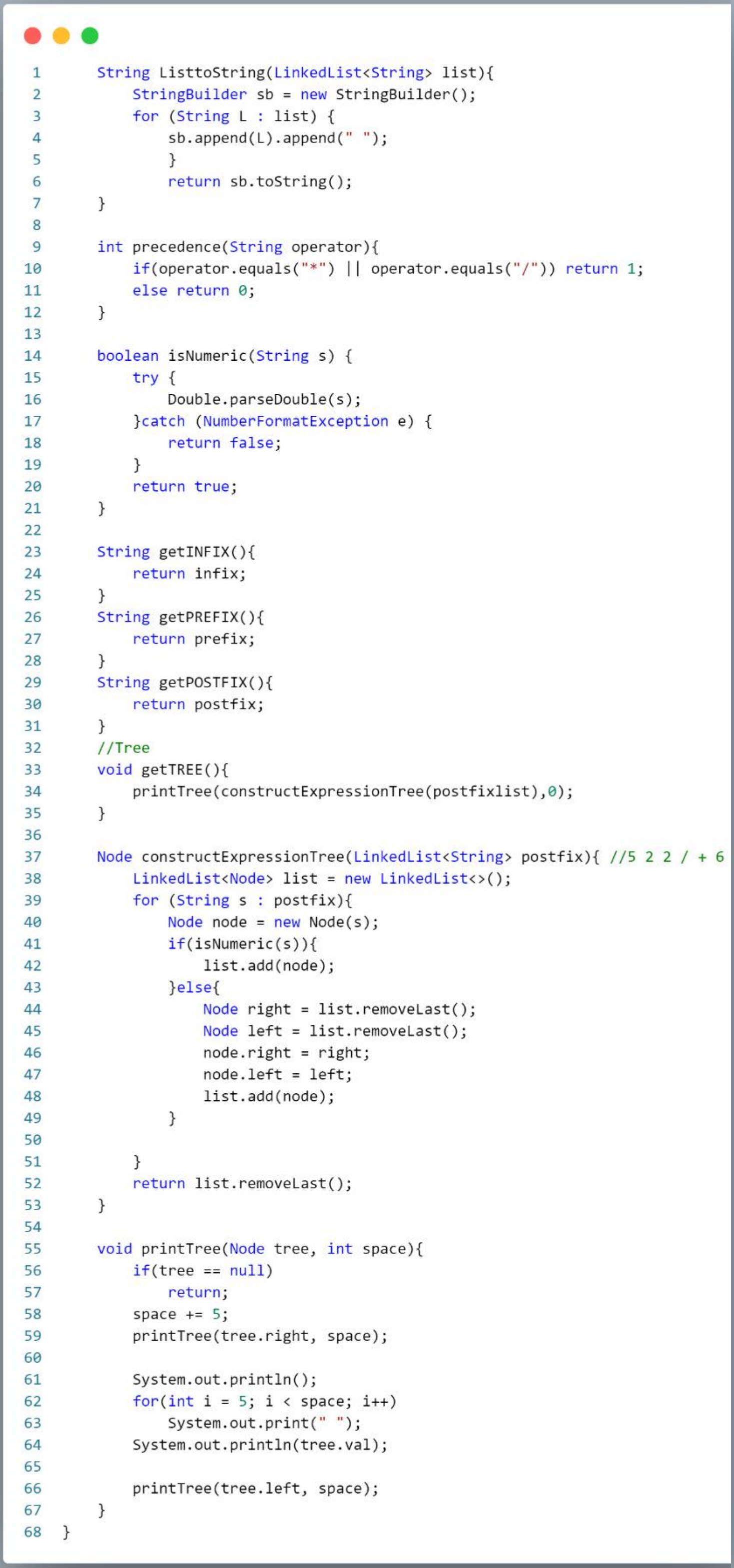
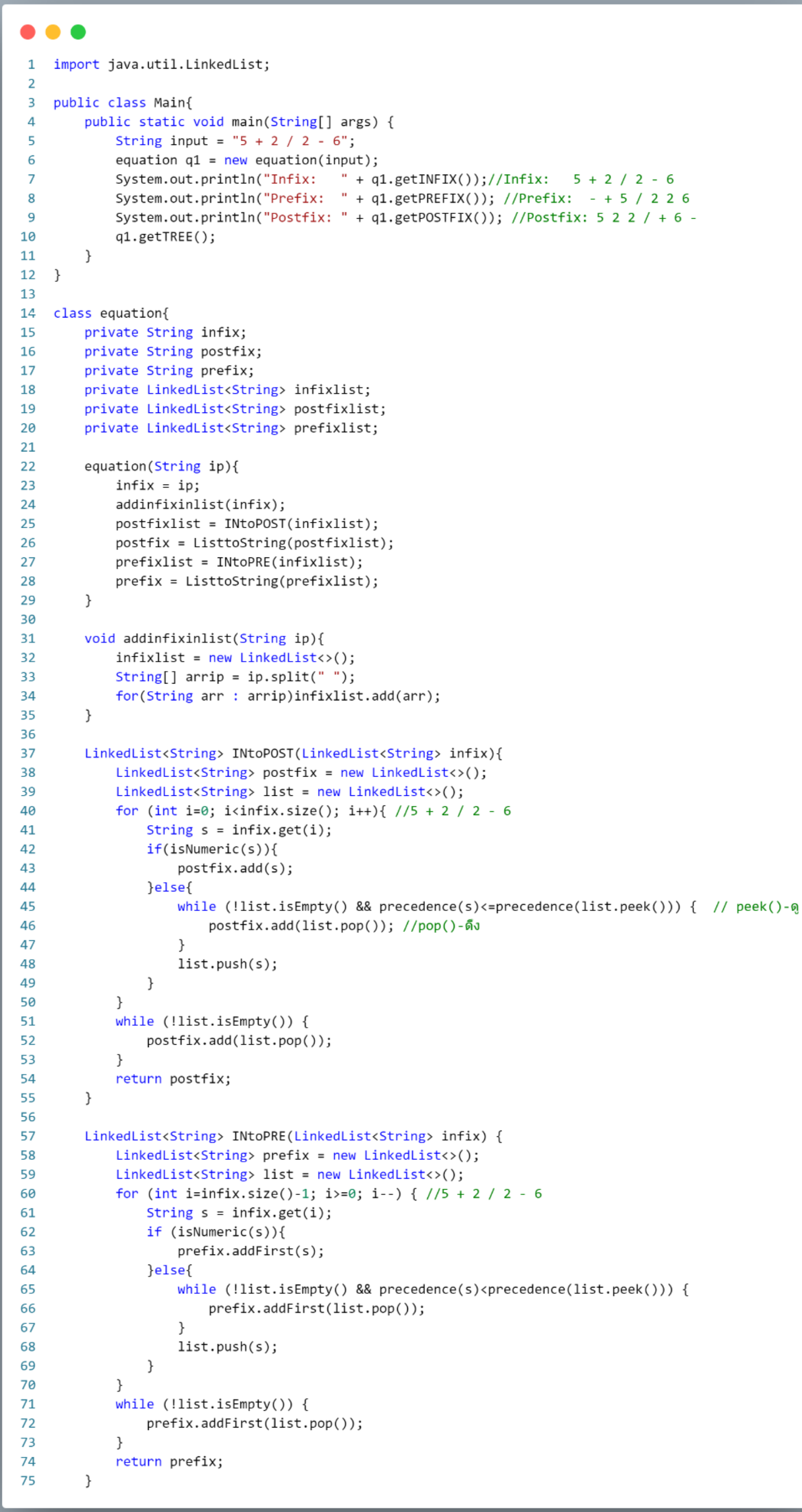
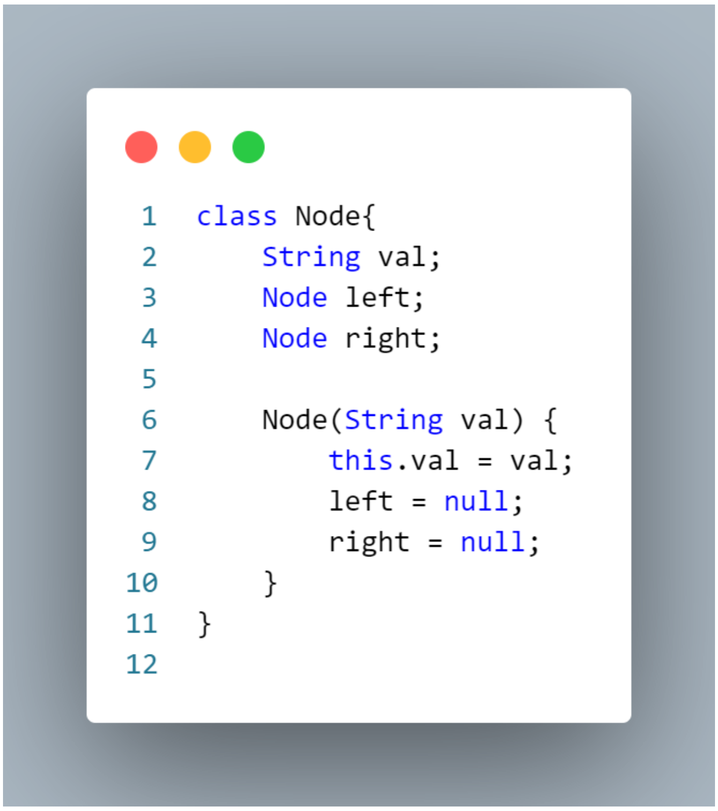


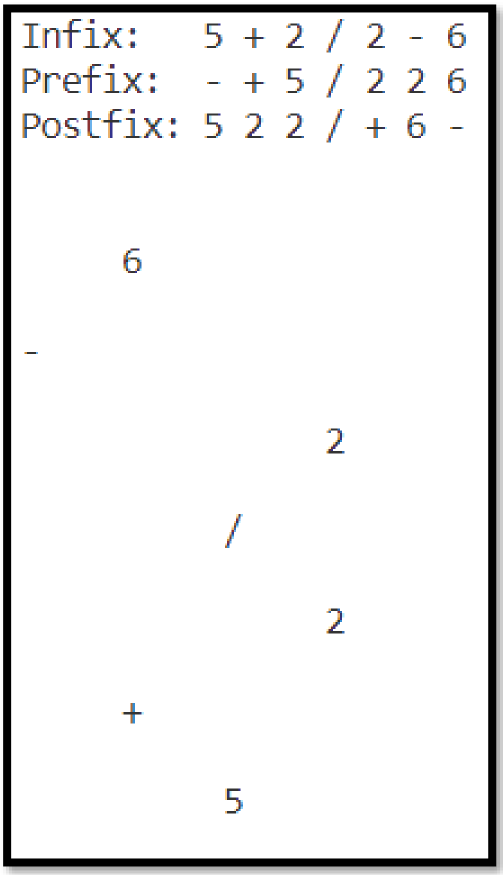
```java
    String ListtoString(LinkedList<String> list){
        StringBuilder sb = new StringBuilder();
        for (String L : list) {
            sb.append(L).append(" ");
            }
            return sb.toString();
    }

    int precedence(String operator){
        if(operator.equals("*") || operator.equals("/")) return 1;
        else return 0;
    }

    boolean isNumeric(String s) {
        try {
            Double.parseDouble(s);
        }catch (NumberFormatException e) {
            return false;
        }
        return true;
    }

    String getINFIX(){
        return infix;
    }
    String getPREFIX(){
        return prefix;
    }
    String getPOSTFIX(){
        return postfix;
    }
    //Tree
    void getTREE(){
        printTree(constructExpressionTree(postfixlist),0);
    }

    Node constructExpressionTree(LinkedList<String> postfix){ //5 2 2 / + 6
        LinkedList<Node> list = new LinkedList<>();
        for (String s : postfix){
            Node node = new Node(s);
            if(isNumeric(s)){
                list.add(node);
            }else{
                Node right = list.removeLast();
                Node left = list.removeLast();
                node.right = right;
                node.left = left;
                list.add(node);
            }

        }
        return list.removeLast();
    }

    void printTree(Node tree, int space){
        if(tree == null)
            return;
        space += 5;
        printTree(tree.right, space);

        System.out.println();
        for(int i = 5; i < space; i++)
            System.out.print(" ");
        System.out.println(tree.val);

        printTree(tree.left, space);
    }
}
```



```java
import java.util.LinkedList;

public class Main{
    public static void main(String[] args) {
        String input = "5 + 2 / 2 - 6";
        equation q1 = new equation(input);
        System.out.println("Infix:   " + q1.getINFIX());//Infix:   5 + 2 / 2 - 6
        System.out.println("Prefix:  " + q1.getPREFIX()); //Prefix:  - + 5 / 2 2 6
        System.out.println("Postfix: " + q1.getPOSTFIX()); //Postfix: 5 2 2 / + 6 -
        q1.getTREE();
    }
}

class equation{
    private String infix;
    private String postfix;
    private String prefix;
    private LinkedList<String> infixlist;
    private LinkedList<String> postfixlist;
    private LinkedList<String> prefixlist;

    equation(String ip){
        infix = ip;
        addinfixinlist(infix);
        postfixlist = INtoPOST(infixlist);
        postfix = ListtoString(postfixlist);
        prefixlist = INtoPRE(infixlist);
        prefix = ListtoString(prefixlist);
    }

    void addinfixinlist(String ip){
        infixlist = new LinkedList<>();
        String[] arrip = ip.split(" ");
        for(String arr : arrip)infixlist.add(arr);
    }

    LinkedList<String> INtoPOST(LinkedList<String> infix){
        LinkedList<String> postfix = new LinkedList<>();
        LinkedList<String> list = new LinkedList<>();
        for (int i=0; i<infix.size(); i++){ //5 + 2 / 2 - 6
            String s = infix.get(i);
            if(isNumeric(s)){
                postfix.add(s);
            }else{
                while (!list.isEmpty() && precedence(s)<=precedence(list.peek())) {  // peek()-ดู
                    postfix.add(list.pop()); //pop()-ดึง
                }
                list.push(s);
            }
        }
        while (!list.isEmpty()) {
            postfix.add(list.pop());
        }
        return postfix;
    }

    LinkedList<String> INtoPRE(LinkedList<String> infix) {
        LinkedList<String> prefix = new LinkedList<>();
        LinkedList<String> list = new LinkedList<>();
        for (int i=infix.size()-1; i>=0; i--) { //5 + 2 / 2 - 6
            String s = infix.get(i);
            if (isNumeric(s)){
                prefix.addFirst(s);
            }else{
                while (!list.isEmpty() && precedence(s)<precedence(list.peek())) {
                    prefix.addFirst(list.pop());
                }
                list.push(s);
            }
        }
        while (!list.isEmpty()) {
            prefix.addFirst(list.pop());
        }
        return prefix;
    }
```



```java
class Node{
    String val;
    Node left;
    Node right;

    Node(String val) {
        this.val = val;
        left = null;
        right = null;
    }
}
```



```
Infix:   5 + 2 / 2 - 6
Prefix:  - + 5 / 2 2 6
Postfix: 5 2 2 / + 6 -


     6

-

               2

          /

               2

     +

          5
```

65050446 นราภรณ์ ดีเย็น