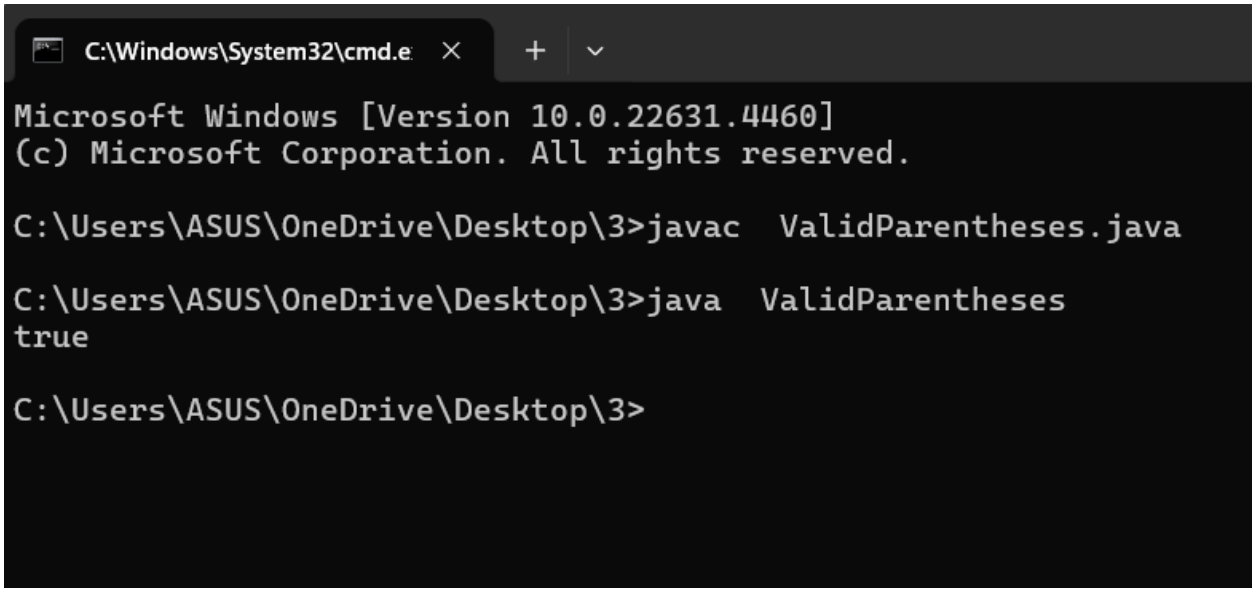
1)valid parentheses

```
import java.util.Stack;

public class ValidParentheses {
    public static boolean isValid(String s) {
        Stack<Character> stack = new Stack<>();
        for (char c : s.toCharArray()) {
            if (c == '(' || c == '[' || c == '{') {
                stack.push(c);
            } else {
                if (stack.isEmpty()) return false;
                char top = stack.pop();
                if ((c == ')' && top != '(') || (c == ']' && top != '[') || (c == '}' && top !=
'{')) {
                    return false;
                }
            }
        }
        return stack.isEmpty();
    }

    public static void main(String[] args) {
        String s = "()[]{}";
        System.out.println(isValid(s));
    }
}
```

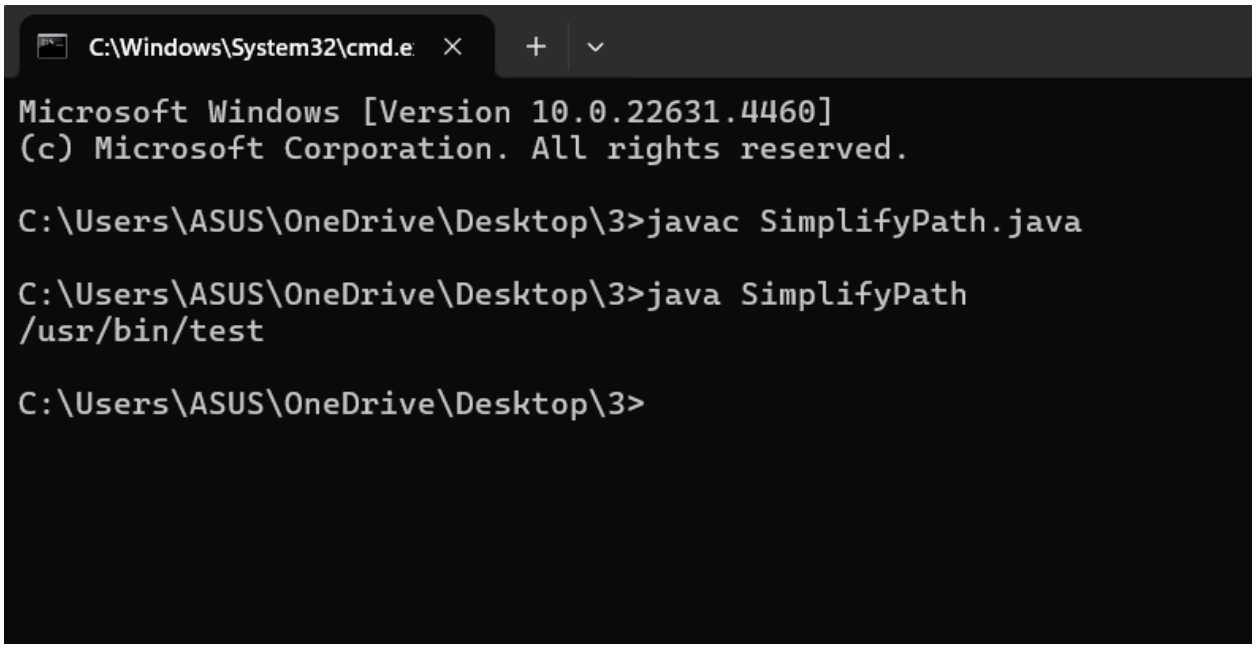


Time:O(n)

2)Simplify path

```
import java.util.Stack;

public class SimplifyPath {
    public static String simplifyPath(String path) {
        Stack<String> stack = new Stack<>();
        String[] dirs = path.split("/");

        for (String dir : dirs) {
            if (dir.equals("..")) {
                if (!stack.isEmpty()) stack.pop();
            } else if (!dir.equals("") && !dir.equals(".")) {
                stack.push(dir);
            }
        }

        StringBuilder result = new StringBuilder();
        for (String dir : stack) {
            result.append("/").append(dir);
```

```
        }

        return result.length() == 0 ? "/" : result.toString();
    }

    public static void main(String[] args) {
        String path = "/home/../usr//bin/./test";
        System.out.println(simplifyPath(path));
    }
}
```



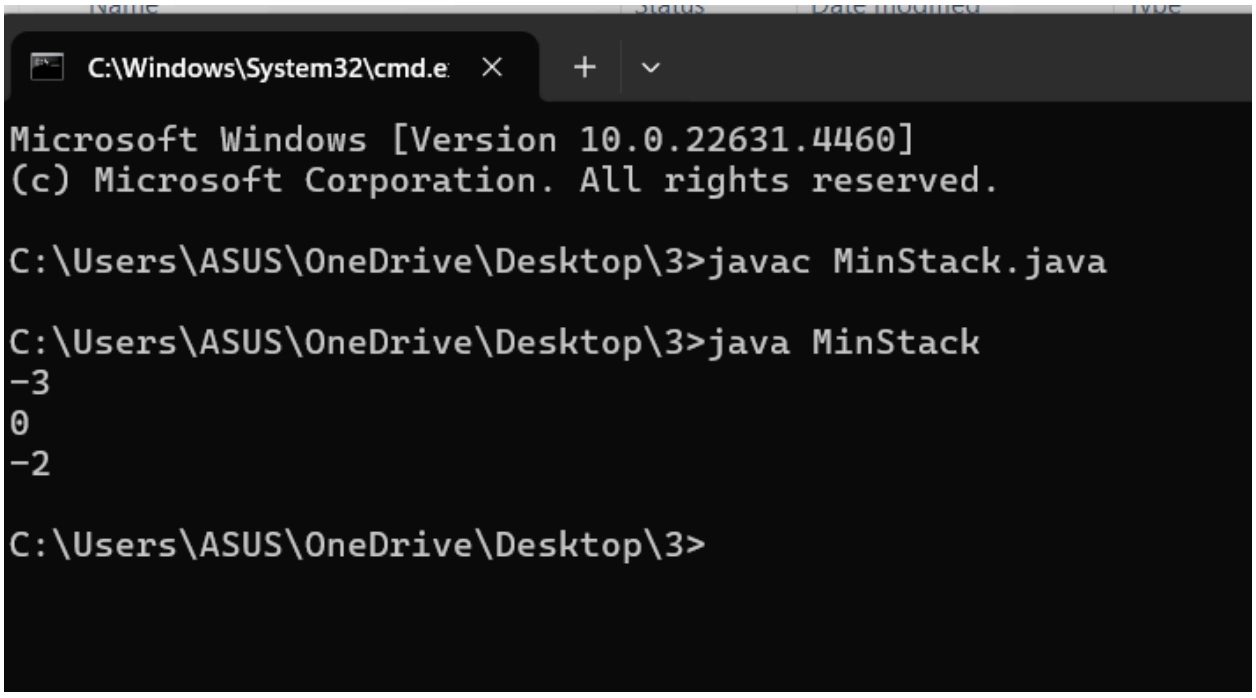
Time:O(n)

3)Min stack

```
import java.util.Stack;

public class MinStack {
    private Stack<Integer> stack;
    private Stack<Integer> minStack;
```

```
public MinStack() {
    stack = new Stack<>();
    minStack = new Stack<>();
}

public void push(int val) {
    stack.push(val);
    if (minStack.isEmpty() || val <= minStack.peek()) {
        minStack.push(val);
    }
}

public void pop() {
    if (stack.peek().equals(minStack.peek())) {
        minStack.pop();
    }
    stack.pop();
}

public int top() {
    return stack.peek();
}

public int getMin() {
    return minStack.peek();
}

public static void main(String[] args) {
    MinStack minStack = new MinStack();
    minStack.push(-2);
    minStack.push(0);
    minStack.push(-3);
    System.out.println(minStack.getMin()); // -3
    minStack.pop();
    System.out.println(minStack.top());    // 0
```

```
        System.out.println(minStack.getMin()); // -2
    }
}
```

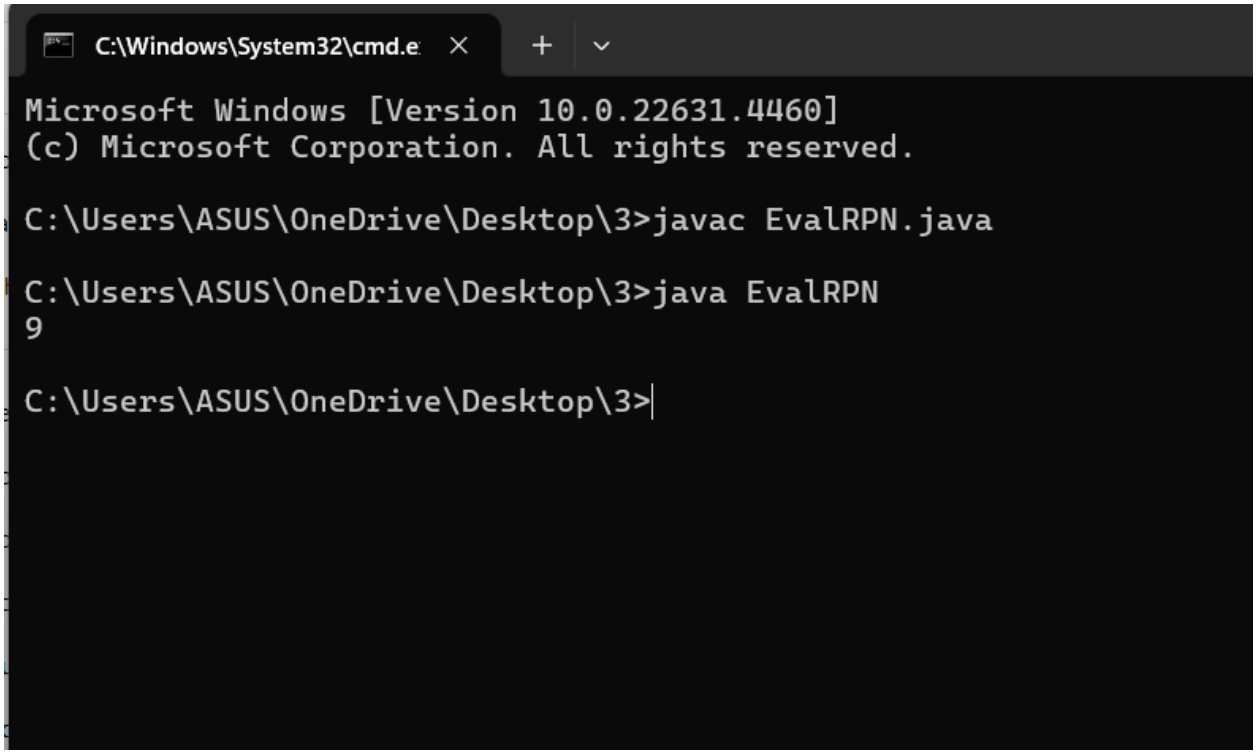


Time:O(1)

4)Evaluate Reverse polish notation

```
import java.util.Stack;

public class EvalRPN {
    public static int evalRPN(String[] tokens) {
        Stack<Integer> stack = new Stack<>();

        for (String token : tokens) {
            if (token.equals("+")) {
                int b = stack.pop();
                int a = stack.pop();
                stack.push(a + b);
            } else if (token.equals("-")) {
                int b = stack.pop();
```

```
                int a = stack.pop();
                stack.push(a - b);
            } else if (token.equals("*")) {
                int b = stack.pop();
                int a = stack.pop();
                stack.push(a * b);
            } else if (token.equals("/")) {
                int b = stack.pop();
                int a = stack.pop();
                stack.push(a / b);
            } else {
                stack.push(Integer.parseInt(token));
            }
        }
        return stack.pop();
    }

    public static void main(String[] args) {
        String[] tokens = {"2", "1", "+", "3", "*"};
        System.out.println(evalRPN(tokens));      }
}
```

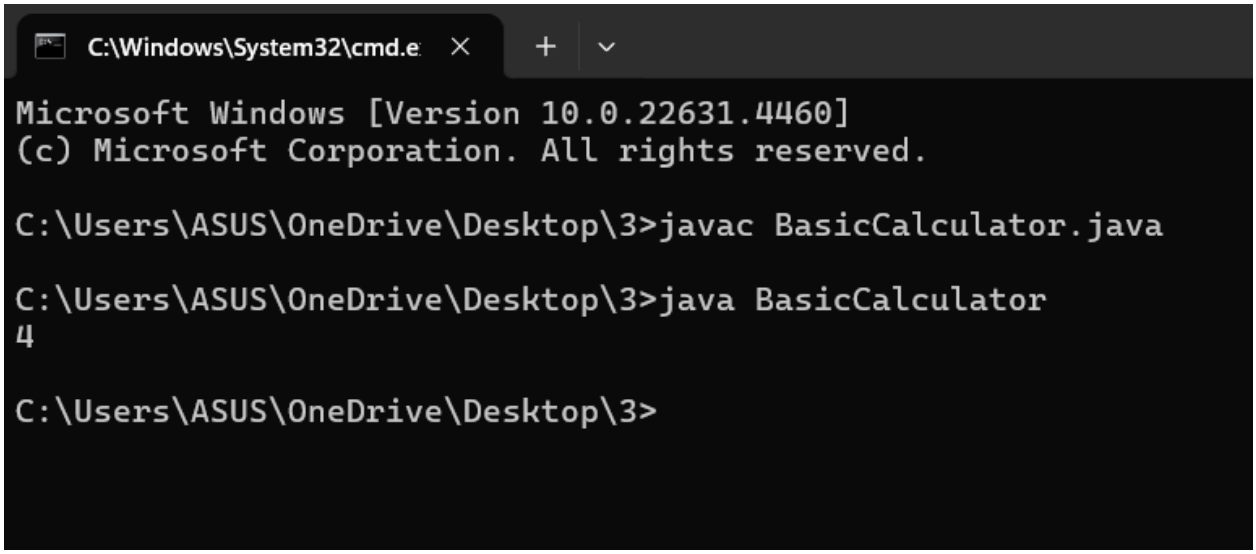


Time:O(n)

5)Baisc calculator

```
import java.util.Stack;

public class BasicCalculator {
    public static int calculate(String s) {
        Stack<Integer> stack = new Stack<>();
        int num = 0;
        int sign = 1;
        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);

            if (Character.isDigit(c)) {
                num = num * 10 + (c - '0');
            }

            if (c == '+') {
                stack.push(sign * num);
```

```
                sign = 1;
                num = 0;
            } else if (c == '-') {
                stack.push(sign * num);
                sign = -1;
                num = 0;
            } else if (c == '(') {
                stack.push(Integer.MAX_VALUE);
            } else if (c == ')') {
                int tempSum = 0;
                while (stack.peek() != Integer.MAX_VALUE) {
                    tempSum += stack.pop();
                }
                stack.pop();
                stack.push(tempSum * sign);
            }
        }

        stack.push(sign * num);
        int result = 0;
        while (!stack.isEmpty()) {
            result += stack.pop();
        }
        return result;
    }

    public static void main(String[] args) {
        String s = "1 + (2 - (3 + 4))";
        System.out.println(calculate(s));
    }
}
```


C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.
C:\Users\ASUS\OneDrive\Desktop\3>javac BasicCalculator.java
C:\Users\ASUS\OneDrive\Desktop\3>java BasicCalculator
4
C:\Users\ASUS\OneDrive\Desktop\3>

Time:O(n)