**Java program that converts infix expression into postfix form**

|  |
| --- |
| **import** java.io.\*;  **class** stack  {  **char** stack1[]=**new** **char**[20];  **int** top;  **void** push(**char** ch)  {  top++;  stack1[top]=ch;  }  **char** pop()  {  **char** ch;  ch=stack1[top];  top--;  **return** ch;  }  **int** pre(**char** ch)  {  **switch**(ch)  {  **case** '-':**return** 1;  **case** '+':**return** 1;  **case** '\*':**return** 2;  **case** '/':**return** 2;  }  **return** 0;  }  **boolean** operator(**char** ch)  {  **if**(ch=='/'||ch=='\*'||ch=='+'||ch=='-')  **return** **true**;  **else**  **return** **false**;  }  **boolean** isAlpha(**char** ch)  {  **if**(ch>='a'&&ch<='z'||ch>='0'&&ch=='9')  **return** **true**;  **else**  **return** **false**;  }  **void** postfix(String str)  {  **char** output[]=**new** **char**[str.length()];  **char** ch;  **int** p=0,i;  **for**(i=0;i<str.length();i++)  {  ch=str.charAt(i);  **if**(ch=='(')  {  push(ch);  }  **else** **if**(isAlpha(ch))  {  output[p++]=ch;  }  **else** **if**(operator(ch))  {  **if**(stack1[top]==0||(pre(ch)>pre(stack1[top]))||stack1[top]=='(')  {  push(ch);  }  }  **else** **if**(pre(ch)<=pre(stack1[top]))  {  output[p++]=pop();  push(ch);  }  **else** **if**(ch=='(')  {  **while**((ch=pop())!='(')  {  output[p++]=ch;  }  }  }  **while**(top!=0)  {  output[p++]=pop();  }  **for**(**int** j=0;j<str.length();j++)  {  System.out.print(output[j]);  }  }  }  **class** intopost  {  **public** **static** **void** main(String[] args)**throws** Exception  {  String s;  BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.in));  stack b=**new** stack();  System.out.println("Enter input string");  s=br.readLine();  System.out.println("Input String:"+s);  System.out.println("Output String:");  b.postfix(s);  }  }  **Output:**  Enter input string a+b\*c Input String:a+b\*c Output String: abc\*+  Enter input string a+(b\*c)/d Input String:a+(b\*c)/d Output String: abc\*d/)(+  **Postfix Evaluation**  // Java proram to evaluate value of a postfix expression    import java.util.Stack;    public class Test  {      // Method to evaluate value of a postfix expression      static int evaluatePostfix(String exp)      {          //create a stack          Stack<Integer> stack=new Stack<>();            // Scan all characters one by one          for(int i=0;i<exp.length();i++)          {              char c=exp.charAt(i);                // If the scanned character is an operand (number here),              // push it to the stack.              if(Character.isDigit(c))              stack.push(c - '0');                //  If the scanned character is an operator, pop two              // elements from stack apply the operator              else              {                  int val1 = stack.pop();                  int val2 = stack.pop();                    switch(c)                  {                      case '+':                      stack.push(val2+val1);                      break;                        case '-':                      stack.push(val2- val1);                      break;                        case '/':                      stack.push(val2/val1);                      break;                        case '\*':                      stack.push(val2\*val1);                      break;                }              }          }          return stack.pop();      }        // Driver program to test above functions      public static void main(String[] args)      {          String exp="231\*+9-";          System.out.println(evaluatePostfix(exp));      }  }  Output:  Value of 231\*+9- is -4 |

**Another Example**

// This program evaluates "postfix" expressions (also called "Reverse Polish

// Notation"), which are mathematical expressions but with the operators placed

// after operands instead of between.

// For example: 1 + 2 \* 3 + 4 is written as 1 2 3 \* + 4 +

import java.util.\*;

public class Postfix {

public static void main(String[] args) {

System.out.println(postfixEvaluate("1 2 +")); // 3

System.out.println(postfixEvaluate("1 2 3 \* + 4 +")); // 11

System.out.println(postfixEvaluate("5 2 4 \* + 7 -")); // 6

System.out.println(postfixEvaluate("2 3 + 4 5 \* +")); // 25

System.out.println(postfixEvaluate("8 5 \* 7 4 2 + \* +")); // 82

System.out.println(postfixEvaluate("6 8 2 / 1 - \*")); // 18

}

// Evaluates the given postfix expression string and returns the result.

// Precondition: expression consists only of integers, +-\*/, and spaces in

// proper postfix format such as "2 3 - 4 \*"

public static int postfixEvaluate(String exp) {

Stack<Integer> s = new Stack<Integer> ();

Scanner tokens = new Scanner(exp);

while (tokens.hasNext()) {

if (tokens.hasNextInt()) {

s.push(tokens.nextInt());

} else {

int num2 = s.pop();

int num1 = s.pop();

String op = tokens.next();

if (op.equals("+")) {

s.push(num1 + num2);

} else if (op.equals("-")) {

s.push(num1 - num2);

} else if (op.equals("\*")) {

s.push(num1 \* num2);

} else {

s.push(num1 / num2);

}

// "+", "-", "\*", "/"

}

}

return s.pop();

}

}