ASSIGNMENT 5.2 -62070501064 ONWIPA KUJAROENPAISAN-

SOURCE CODE

```java
package ass5;

import java.util.Scanner;
import java.util.Stack;
import java.util.ArrayList;

public class ass5 {

    public static String Transform(String x) { //function to split the operator
        x = x.replace("+"," + ");
        x = x.replace("-"," - ");
        x = x.replace("*"," * ");
        x = x.replace("/"," / ");
        x = x.replace("^"," ^ ");
        x = x.replace("("," ( ");
        x = x.replace(")"," ) ");
        x = x.replaceAll("\\s+"," "); //change the space
        x = x.trim(); //cut the space before and behind
        return x;
    }

    public static boolean isNum(String str) { //function to check the number
        boolean check = true;
        try {
            Double.parseDouble(str);
        } catch(Exception e) {
            check = false;
        }
        return check;
    }

    public static int check_Function(String str) { //function to check string in form math function
        String[] myfunc = {"","sin","cos","tan","cosec","sec","cot","asin","acos","atan","sqrt","log","exp","abs"};
        int check = 0;
        for(int i = 1; i < myfunc.length; i++) {
            if(myfunc[i].equalsIgnoreCase(str))
                check = i;
        }

        return check;
    }

    public static boolean isConstant(String str) { //function to check math constant
        String[] myconst = {"pi","e","ans"};
        boolean check = false;
        for(int i = 0; i < myconst.length; i++) { //and count the number
            if(myconst[i].equalsIgnoreCase(str))
                check = true;
        }
        return check;
    }

    public static double ans;

    public static double ValueOf(String str) { //function to declare value of math constant
        if(str.equalsIgnoreCase("pi"))
            return Math.PI;
        else if(str.equalsIgnoreCase("E"))
            return Math.E;
        else if(str.equalsIgnoreCase("ans")) //if user want to use last answer as new input
            return ans;
        else
            return Double.parseDouble(str);
    }

    public static int check_group(String str) { //separate operator in group and return by number
        if(isNum(str))
            return 1;
        else if(isConstant(str))
            return 1;
        else if(str.matches("[-+]"))
            return 2;
        else if(str.matches("[/*]"))
            return 3;
        else if(str.equals("^"))
            return 4;
        else if(str.equals("!"))
```

```
            return 5;
        else if(str.equals("("))
            return 7;
        else if(str.equals(")"))
            return 8;
        else if(check_Function(str) > 0)
            return 6;
        else
            return 0; //if its in unknown group then return 0
    }

    public static int check_state(String[] token, int count) {
        int state = 0, next = 0, b = 0, i;
        for(i = 0; i < count && state >= 0; i++) {
            state = next; //declare state to be new value
            next = check_group(token[i]);
            if(next >= 2 && next <= 4) next = 2; //make state of +-*/^ be state number 2
            if(next == 0) state = -1; //unknown state is error
            if(next == 7) b++; //check (
            else if(next == 8) b--; //check )
            if(b < 0)
                state = -1; //if its ) more than or before ( then show error
            else if(state == 0 && (next == 8 || next == 2))
                state = -1; //error if its begin with ) or -+
            else if(state == 1 && (next == 1 || next == 6 || next == 7))
                state = -1; //error if its variant fn or ( after number
            else if(state == 2 && (next == 2 || next == 8))
                state = -1;//error if its operator or ) after operator
            else if(state == 5 && (next == 2 || next == 8))
                state = -1; //error if its operator or ) after sign
            else if(state == 6 && (next != 7))
                state = -1;//error if its fn but not after by (
            else if(state == 7 && (next == 8 || next == 2))
                state = -1;//error if its be like () or (-+
        }//loop until end
            if(b != 0 || state == -1) //if theres not couple of () or state is error
                return -1;
            else

                return next; //return last state

    }

    public static void change_neg_sign(String token[]) { //function to change the negative statement
        int i;
        if(token[0].equals("-"))
            token[0] = "!";
        for(i = 0; i < token.length -1; i++)
            if(token[i+1].equals("-") && (token[i].matches("[-+/(*\\^]")))
                token[i+1] = "!";
    }

    static ArrayList<String> postfix = new ArrayList<String>();
    static Stack<String> oprstack = new Stack<String>();

    //function to change infix form to postfix form
    public static void infixToPostfix(String[] token) {
        int group, cur, prior = 0, i;
        String buff;

        //clear the stack
        postfix.clear();
        oprstack.clear();

        for(i=0; i < token.length; i++) { //loop word for word
            group = check_group(token[i]);
            if(group == 7) {//if it's (
                oprstack.push(token[i]);
            }else if(group == 1){ //if it's number
                postfix.add(token[i]);
            }else if(group >= 2 && group <=6) {//if it's operator or function
                if(oprstack.empty()) //check if stack is empty
                    oprstack.push(token[i]);
                else {
                    do {
                        cur = check_group(token[i]);
                        buff = oprstack.peek();
```

```
153                         prior = check_group(buff);
154                         if(prior >= cur && prior <= 6) {
155                             buff = oprstack.pop();
156                             postfix.add(buff);
157                         }
158                     }while(prior >= cur && prior <= 6 && !oprstack.empty());
159                         oprstack.push(token[i]);
160                 }
161             }else if(group == 8) { //if it's )
162                 do {
163                     buff = oprstack.pop();
164                     if(!buff.equals("("))
165                         postfix.add(buff);
166                 }while(!buff.equals("(") && !oprstack.empty());
167             }
168         }
169         while(!oprstack.empty()) //loop until stack is empty
170             postfix.add(oprstack.pop());
171     }
172
173     static Stack<Double> numstack = new Stack<Double>();
174
175     //function to calculate the postfix
176     public static double Calculate() {
177         double num, num1, num2;
178         int i, group;
179         String token = new String("");
180         for(i = 0; i < postfix.size(); i++) {
181             token = postfix.get(i);
182             group = check_group(token);
183             if(group == 1) { //if it's number
184                 num1 = ValueOf(token);
185                 numstack.push(num1);
186             }else if(group == 6) { //if it's function
187                 if(token.equalsIgnoreCase("sin")) {
188                     num = Math.sin(numstack.pop() * Math.PI / 180);
189                     numstack.push(num);
190                 }else if(token.equalsIgnoreCase("cos")) {
191                     num = Math.cos(numstack.pop() * Math.PI / 180);
192                     numstack.push(num);
193                 }else if(token.equalsIgnoreCase("tan")) {
194                     num = Math.tan(numstack.pop() * Math.PI / 180);
195                     numstack.push(num);
196                 }else if(token.equalsIgnoreCase("asin")) {
197                     num = Math.asin(numstack.pop()) * 180 / Math.PI;
198                     numstack.push(num);
199                 }else if(token.equalsIgnoreCase("acos")) {
200                     num = Math.acos(numstack.pop()) * 180 / Math.PI;
201                     numstack.push(num);
202                 }else if(token.equalsIgnoreCase("atan")) {
203                     num = Math.atan(numstack.pop()) * 180 / Math.PI;
204                     numstack.push(num);
205                 }else if(token.equalsIgnoreCase("sqrt")) {
206                     num = Math.sqrt(numstack.pop());
207                     numstack.push(num);
208                 }else if(token.equalsIgnoreCase("log")) {
209                     num = Math.log(numstack.pop());
210                     numstack.push(num);
211                 }else if(token.equalsIgnoreCase("exp")) {
212                     num = Math.exp(numstack.pop());
213                     numstack.push(num);
214                 }else if(token.equalsIgnoreCase("abs")) {
215                     num = Math.abs(numstack.pop());
216                     numstack.push(num);
217                 }
218             }else if(group >= 2 && group <= 4) { //if it's operator
219                 num1 = numstack.pop();
220                 num2 = numstack.pop();
221                 if((group == 2) && token.equals("+"))
222                     numstack.push(num2 + num1);
223                 else if((group == 2) && token.equals("-"))
224                     numstack.push(num2 - num1);
225                 else if((group == 3) && token.equals("*"))
226                     numstack.push(num2 * num1);
227                 else if((group == 3) && token.equals("/")) {
228                     numstack.push(num2 / num1);
```

```
            }else if(group == 4)
                numstack.push(Math.pow(num2, num1));
            }else if(group == 5) {//if it's negative sign
                num1 = numstack.pop();
                numstack.push(-num1);
            }
        }
        ans = numstack.pop(); //output answer
        return ans;
    }

    public static void main(String[] args) {
        String str;
        Scanner in = new Scanner(System.in);
        String[] token;
        int count = 0 ;
        //double ans;
        do {
        System.out.printf("expression> ");
        str = in.nextLine(); //read string 1 line
        str = str.toLowerCase();//change input to lower character
        str = Transform(str); //send input to split the operator
        token = str.trim().split("\\s+"); //keep input that split by apce in token
        count = token.length; //count the token
        change_neg_sign(token);
        int i = check_state(token,count); //send to check state

        if((i == -1 || i == 2 || i == 4 || i == 5 || i == 7) && !token[0].equalsIgnoreCase("end") && !token[0].equalsIgnoreCase("help")) {
            System.out.println("answer> error");
        }
        else if(!token[0].equalsIgnoreCase("end") && !token[0].equalsIgnoreCase("help")) {
            //System.out.println("answer> OK");
            infixToPostfix(token); //send token to change into postfix form
            //System.out.println(postfix);
            ans = Calculate(); //calculate and return as ans
            System.out.printf("answer> " + ans + "\n"); //print answer to calculating
```

```
265         }
266         else if(token[0].equalsIgnoreCase("help")) {
267             System.out.println("answer> token = sin, cos, tan, asin, acos, atan, sqrt, log, exp, abs, +, -, *, /, ^, (, ), pi, ans");
268         }
269         }while(!token[0].equalsIgnoreCase("end")); //loop until user input 'end'
270         System.out.printf("End Program\n");
271         System.out.printf("Program written by ONWIPA KUJAROENPAISAN 62070501064");
272         in.close();
273     }
274
275 }
```

คำอธิบายโปรแกรม

เมื่อรันโปรแกรม จะแสดงข้อความให้ผู้ใช้กรอกสมการทางคณิตศาสตร์ สแกนสิ่งที่รับเข้ามา ส่งไปเปลี่ยนเป็นตัวพิมพ์เล็ก จากนั้นก็ทำการเว้นช่องว่างจะหว่างเครื่องหมายทางคณิตศาสตร์ทุกตัวที่เจอ และแบ่งเป็นคำนำไปเก็บเป็น token ส่ง token ไปแปลงค่าที่เป็นนิเสธหรือตัวที่ติดลบ จากนั้นส่ง token ไป check_state เพื่อเช็คความถูกต้องโดย return กลับมาในค่า i

ในการ check_state นั้น เมื่อเข้าไปแล้ว จะส่ง token ทีละตัวไปฟังก์ชัน check_group เพื่อแบ่งประเภทของทุกตัวอักษรใน token นั้นๆ และนำมาเทียบกับตัวก่อนหน้า ว่าผิดรูปแบบหรือไม่ เมื่อวนลูปครบทุก token แล้ว ก็จะส่งค่ากลับไปที่ main

เมื่อ return กลับมาแล้วไม่ error จะนำค่า token เดิมไปเปลี่ยนจากฟอร์ม infix ให้เป็น postfix และนำรูป postfix นั้น ไปคำนวณทางคณิตศาสตร์ด้วยฟังก์ชัน Calculate โดยจะส่งไป check_group อีกครั้งเพื่อแบ่งกลุ่มการคำนวณ และ return กลับมาเป็นตัวแปร group เมื่อ return กลับมาแล้ว

- ถ้าพบว่าเป็นตัวเลข จะ push ใส่ใน stack ของเลขจำนวนจริง
- ถ้าเป็นฟังก์ชันทางคณิตศาสตร์ จะ pop ตัวเลขมา ตัว จาก stack และนำไปคำนวณตามคำสั่ง แล้ว push คำตอบที่ได้กลับไปใน stack
- ถ้าเป็นเครื่องหมายทางคณิตศาสตร์ (+-*/^) จะ pop ตัวเลขจาก stack มา 2 ตัว มาคำนวณ และ push คำตอบกลับไปใน stack
- ถ้าเป็นนิเสธ จะ pop ตัวเลขมา 1 ตัว นำมาใส่เครื่องหมายลบ และ push กลับเข้าไปใน stack

จากนั้นทำซ้ำจนกว่าข้อมูลจะหมด เมื่อวนจนครบแล้ว ออกจากลูป for และ pop ตัวเลขจาก stack (จะเหลือตัวเดียว) ออกมาเป็นคำตอบ

และจะแสดงข้อความให้กรอกใหม่ซ้ำๆจนกว่าผู้ใช้จะกรอก end จึงจบโปรแกรม

TEST CASE

```
expression> 1.23 4.56 +
answer> error
expression> sin 90
answer> error
expression> 0/1
answer> 0.0
expression> 1/sin(0)
answer> Infinity
expression> -2^-2-2
answer> -1.75
expression> -1+2^3/(4-5*6)+pi
answer> 1.8339003458974854
expression> sqrt(log(10^2)+exp(3))
answer> 4.968974452457545
expression> sqrt(3^2+4^2)
answer> 5.0
expression> ans^-2
answer> 0.04
expression> asin(1-cos(0)+sin(90))
answer> 90.0
expression> sin(30)^2+cos(30)^2
answer> 1.0
expression> end
End Program
Program written by ONWIPA KUJAROENPAISAN 62070501064
```

ปัญหาที่พบในการทำ ASSIGNMENT

- ตอนที่ stack ไม่มีข้อมูล จะ peek ไม่ได้ จึงต้องเช็คก่อนว่า stack ว่างหรือไม่
- เขียนสูตรคำนวณ asin สลับ เพราะต้องคำนวณ asin ก่อนแล้วค่อย *180/pi
- เขียน infix to postfix ผิด
- ยังสับสนการเขียนจาวากับภาษาซีอยู่บ้าง

Self-Assessment : 3 เข้าใจแต่มีปัญหาบางช่วงยังต้องขอความช่วยเหลือ