ISC Project

Computer Science

Submitted to:

[redacted]

Submitted by:

[redacted]

11th Non-medical

Acknowledgement:

We would like to express our special thanks of gratitude to our Computer Science teacher [redacted] as well as our principal [redacted] who gave us the golden opportunity to do this wonderful project on making a BODMAS calculator.

We really appreciate the hard work of the teachers to guide and motivate us, because of which, we have been able to complete this project.

It is because of the guidance of our teacher that we were able to complete this project. We were able to learn a lot of new things, and we really enjoyed making this project!

We would also like to thank our classmates who helped us to finalize this project within the limited time frame.

[redacted]
XI Non-medical (2)

Manbir Singh
XI Non-medical (4)

Topic:

BODMAS CALCULATOR

using java

Key features:

Calculates complex BODMAS equations.

No limit on operators

Infinite operands

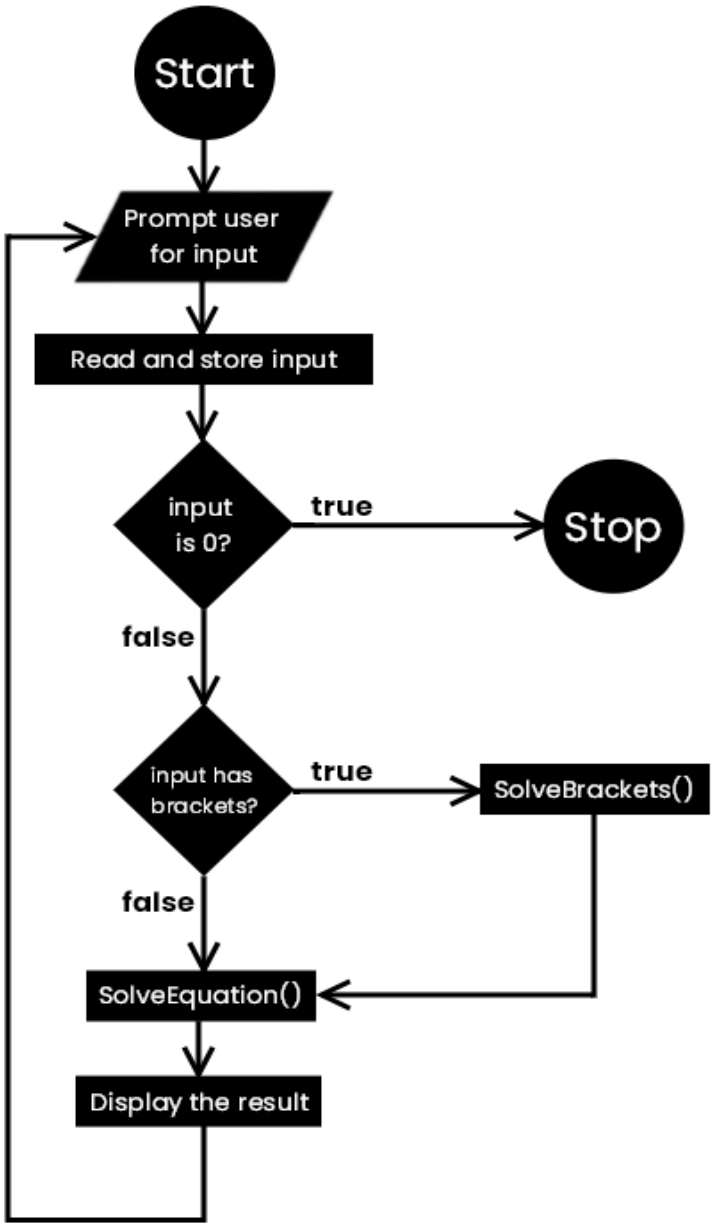
Description:

BODMAS is an abbreviation that stands for **B**rackets, **O**f, **D**ivision, **M**ultiplication, **A**ddition, **S**ubtraction. It is used to explain the order of operation of a mathematical expression.

Concepts used:

- String manipulation
- Conditional Statements
- Loops
- Arrays
- Classes:
    - Math
    - Scanner
    - Wrapper Classes

Algorithm:


Start
Prompt user for input
Read and store input
input is 0?
true
Stop
false
input has brackets?
true
SolveBrackets()
false
SolveEquation()
Display the result

Structure:

```
import java.util.Scanner;

public class calc
{
	[variables]
	main()		⇐ handles the input, and solves parenthesis
	callback()	⇐ checks if the input is incomplete
	calc()		⇐ calculates the base equation
}
```

Global Variables:

```
static String Output = "";
static String NewString = "";
String Final = "";
String printer;
String tab=
"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";
```

Methods:

calc(String inp)

Description:

This method accepts a string as an input, which contains operators and operands, without brackets.
It first checks if the format of the equation is valid.
Then it uses an algorithm to solve the equation.

A ‘for’ loop runs 4 times, each for a different operator **(/, *, +, -)** and replaces the operator and operands with the result of the same.

Eg:
15+5*16/2 ⇒ 15+5*8 ⇒ 15+40 ⇒ **55**
40+0.5*55-10 ⇒ 40+27.5-10 ⇒ 67.5-10 ⇒ **57.5**

Code:

```
static void calc(String inp)
   {
      boolean terminate=false, print=true;
      Output = inp;



      //input only contains negative number
      if(!inp.contains("/") && !inp.contains("*") && !inp.contains("+") &&
inp.charAt(0)=='-'){
         inp = "0"+inp;
      }

      //input starts with an operator
      char c = inp.charAt(0);
      if((c=='/' || c=='*' || c=='+') && !terminate){
         if(c=='+'){
           inp = "0"+inp;
         }
         else{

            terminate = true;
            print = false;
         }
      }

      //input contains no operations
      if(!inp.contains("/") && !inp.contains("*") && !inp.contains("+") &&
!inp.contains("-") && !terminate){
```

```
            terminate = true;
            print = false;
        }

        //input doesnt contain a number after a symbol
        if(!Character.isDigit(inp.charAt(inp.length()-1)) && !terminate){
            printer = "Invalid Format!";
            terminate = true;
            print = false;
        }
        String sampler = "";
        //input contains 2 consecutive symbols
        for(int i=0; i<inp.length()-1; i++){
            if(!Character.isDigit(inp.charAt(i)) && !Character.isDigit(inp.charAt(i+1))
&& !terminate){
                if((inp.charAt(i)=='+' && inp.charAt(i+1)=='-')||(inp.charAt(i)=='-' &&
inp.charAt(i+1)=='+')){
                int j = 0;
                while(j<inp.length()){
                if(j==i){
                 sampler += "-";
                 j=j+2; continue;
                }
                sampler += inp.charAt(j);
                j++;
               }
              inp = sampler;
             }
             else if(inp.charAt(i)=='-' && inp.charAt(i+1)=='-'){
                int j = 0;
                while(j<inp.length()){
```

```
        if(j==i){
         sampler += "+";
         j=j+2; continue;
        }
        sampler += inp.charAt(j);
        j++;
       }
       inp = sampler;
     }
     else{
         printer = "Invalid Format";
         terminate = true;
         print = false;
         break;
     }
   }
}

for(int i=0; i<4; i++)
{
    char[] list = {'/', '*', '+', '-'};

    while(inp.indexOf(list[i]) != -1 && !terminate)
    {
        int indexI = inp.indexOf(list[i]);
        int i1=indexI-1, i2=indexI+1;
        String num1="", num2="";

        //if negative sign is at start
        if(i==3 && indexI==0){
            //skip the negative and set indexI to the index of next negative
```

```
    for(int j=1; j<inp.length(); j++)
    {
        if(inp.charAt(j) == '-')
        {
            indexI = j;
            i1=indexI-1; i2=indexI+1;
            break;
        }
    }
}

//store the number to the left of operator in 'num1'
while(i1 != -1)
{
    char ch = inp.charAt(i1);
    if(ch=='/' || ch== '*' || ch== '+' || ch== '-')
        break;
    num1 += ch;
    i1--;
}
//store the number to the right of operator in 'num2'
while(i2 < inp.length())
{
    char ch = inp.charAt(i2);
    if(ch=='/' || ch== '*' || ch== '+' || ch== '-')
        break;
    num2 += ch;
    i2++;
}

//convert string to int, reverse num1
```

```
String rev1="", rev2="";
float n1, n2, n=0;
for(int j=num1.length()-1; j>=0; j--)
   rev1+=num1.charAt(j);

n1 = Float.parseFloat(rev1);
n2 = Float.parseFloat(num2);

switch(list[i])
{
   case '/':
   n=n1/n2;
   break;
   case '*':
   n=n1*n2;
   break;
   case '+':
   boolean negative = false;
   int k=indexI-1;
   if(k==0)
      n=n1+n2;

   if(inp.substring(0,k).indexOf('-')!=-1)
   {
      while(k>0)
      {
         k--;
         char ch = inp.charAt(k);
         if(ch=='*' || ch=='/' || ch=='+')
         {
            n = n1+n2;
```

```
            break;
        }
        else if(ch=='-')
        {
            //found negative
            negative = true;
            break;
        }
    }
    if(negative)
    {
        negative = false;
        if(n2>n1)
        {
            n = n2-n1;
            i1--;
        }
        else
        {
            n = n1-n2;
            if(k==0 && inp.indexOf('-')==inp.lastIndexOf('-'))
            {
                n = (float)Math.round(n * 1000f) / 1000f;
                inp = "-"+Float.toString(n);
                terminate = true;
                continue;
            }
        }
    }
}else
    n=n1+n2;
```

```
break;

case '-':
if(inp.indexOf('-')==inp.lastIndexOf('-'))
{
   //only one - remains
   n = n1 - n2;
   n = (float)Math.round(n * 1000f) / 1000f;
   inp = Float.toString(n);
   terminate = true;
   continue;
}
else
{
   //more than one -
   int count = inp.length() - inp.replace("-","").length();
   if(inp.charAt(0)=='-' && count==2)
   {
      //only one - at start
      n = n1 + n2;
      n = (float)Math.round(n * 1000f) / 1000f;
      inp = "-"+Float.toString(n);
      terminate = true;
      continue;
   }else if(inp.charAt(0)=='-'){
      //more than one -, and one at start
      n = Math.abs(n1)+Math.abs(n2);
   }else{
      //more than one -, none at start
      n = n1 - n2;
   }
```

```
            }
        }

        //round off the result
        n = (float)Math.round(n * 1000f) / 1000f;

        //store the new equation
        inp = inp.substring(0,i1+1) + n + inp.substring(i2);
      }
    }

    //removing extra 0s
    if(inp.charAt(inp.length()-1)=='0' && inp.charAt(inp.length()-2)=='.')
      inp = inp.substring(0,inp.length()-2);

    if(print)
      Output = inp;
    printer = inp;
}
```

callback(String s)

Description:

This method checks if the provided input has incomplete parenthesis.

Code:

```
static void calcbrack(String s){
    //to check if the input has incomplete parentheses
    String inserter = "";
    int c = 0;
    int count_open=0,count_close=0;
    for(c=0;c<s.length();c++){
       if(s.charAt(c)=='('
       count_open++;
       else if(s.charAt(c)==')')
       count_close++;
    }
    if(count_open != count_close){

for(c=0;c<Math.max(count_open,count_close)-Math.min(count_open,count
_close);c++){
        if(count_open > count_close){
        inserter +=")";
        }
        else if(count_open < count_close){
        inserter += "(";
       }
    }
```

```
        if(count_open > count_close){
            s=s+inserter;
        }
        else if(count_open < count_close){
            s=inserter+s;
        }
    }
      inserter = "";
      for(int ab = 0;ab<s.length();ab++){
       if(s.charAt(ab)=='(' && (int)s.charAt(ab-1)>=48 &&
(int)s.charAt(ab-1)<=57){
          inserter += "*";
       }else if(s.charAt(ab)==')'){
          if(ab<s.length()-1 && (int)s.charAt(ab+1)>=48 &&
(int)s.charAt(ab+1)<=57){
          inserter = "*";
          }
       }
       inserter += s.charAt(ab);
       }
       s=inserter;

       int start =0 ,end = s.length()-1 ;
       int i = 0,count=0;
       NewString = "";
       while(i<s.length()){
          if(s.charAt(i)=='('){
             count++;
          }
          i++;
       }
```

```
    for(int n = 0;n<count;n++){
        int x = 0;start = 0;end = s.length()-1;
        while(x<s.length()){
            if(s.charAt(x)=='('){
                start = x+1;
            }
            else if(s.charAt(x)==')'){
                end = x;
                break;
            }
            x++;
        }
            String brackets = s.substring(start,end);
            calc(brackets);

            for(int m = 0;m<s.length();m++){
                if(m==start-1 && m<end){
                    NewString += Output;
                }
                else if(m<start-1 ||m>end){
                    NewString += s.charAt(m);
                }
            }
            s = NewString;
            NewString = "";
            Final = s;

    }
}
```

main()

Description:

The main method of the program is responsible for input handling, and manipulation of the equation in the form readable by **calc()** method.

It uses an infinite loop to get inputs from the user.

It terminates the program if the input is 0, otherwise, solves the problem and provides the desired result.

Code:

```
public static void main()
  {
    Scanner s = new Scanner(System.in);
        System.out.print("\u000C");
      System.out.print( tab+"   _____\n"+
      "  | _____ |\n"+
      "  ||  enter value  ||\n"+
      "  ||----------|||\n"+
      "  | ___ ___ ___  ___ |\n"+
      "  || 7 | 8 | 9 || + ||\n"+
      "  ||___|___|___||___||\n"+
      "  || 4 | 5 | 6 || - ||\n"+
      "  ||___|___|___||___||\n"+
      "  || 1 | 2 | 3 || x ||\n"+
      "  ||___|___|___||___||\n"+
      "  || . | 0 | = || / ||\n"+
      "  ||___|___|___||___||\n"+
      "  |_____|\n");
```

```
   System.out.println("\n (in order to exit please enter 0)\n");
while(true)
{
   System.out.print("\tInput - ");
   String inp = s.nextLine();
   String firstinp = inp;

   if(inp.equals("0")){
              System.out.print( tab+"    _____\n"+
   "   |  _____  |\n"+
   "   ||    Thank You    ||\n"+
   "   ||-----------------||\n"+
   "   |  ___ ___ ___   ___  |\n"+
   "   || 7 | 8 | 9 || + ||\n"+
   "   ||___|___|___||___||\n"+
   "   || 4 | 5 | 6 || - ||\n"+
   "   ||___|___|___||___||\n"+
   "   || 1 | 2 | 3 || x ||\n"+
   "   ||___|___|___||___||\n"+
   "   || . | 0 | = || / ||\n"+
   "   ||___|___|___||___||\n"+
   "   |_____|\n\n\n");
   System.out.println("  credits - Manbir Singh (+1 Non-Medical)");
   System.out.println("  credits -  Dhruv Puri  (+1 Non-Medical)");
      break;}
   else if(inp.contains("(")||inp.contains(")")){
      calcbrack(inp);
      inp = Final;
      if(!inp.contains("(")||!inp.contains(")"))
      calc(inp);
   }
```

```
        else
        calc(inp);
        /*
       for(int i=0; i<printer.length()-1; i++){
          if(!Character.isDigit(printer.charAt(i)) &&
!Character.isDigit(printer.charAt(i+1))){
              printer = "Nan";  break;
           }
        }*/
       if(inp.contains("*+")||inp.contains("(+")||inp.contains("/+")){
        String medium = "";
        for(int d = 0;d<inp.length()-1;d++){
         if((inp.charAt(d)=='*'||inp.charAt(d)=='(')&&(inp.charAt(d+1)=='+')){
           medium += "*";
           d++;continue;
           }
         if((inp.charAt(d)=='/')&&(inp.charAt(d+1)=='+')){
           medium += "/";
           d++;continue;
           }
         medium += inp.charAt(d);
         }
        inp = medium;
        calc(inp);
        }
       else if(inp.contains("*-")||inp.contains("(-")||inp.contains("/-")){
          int index = 0;
          String med = "";
          for(int r = 0;r<inp.length()-1;r++){
           if(!Character.isDigit(inp.charAt(r)) &&
Character.isDigit(inp.charAt(r+1))){
```

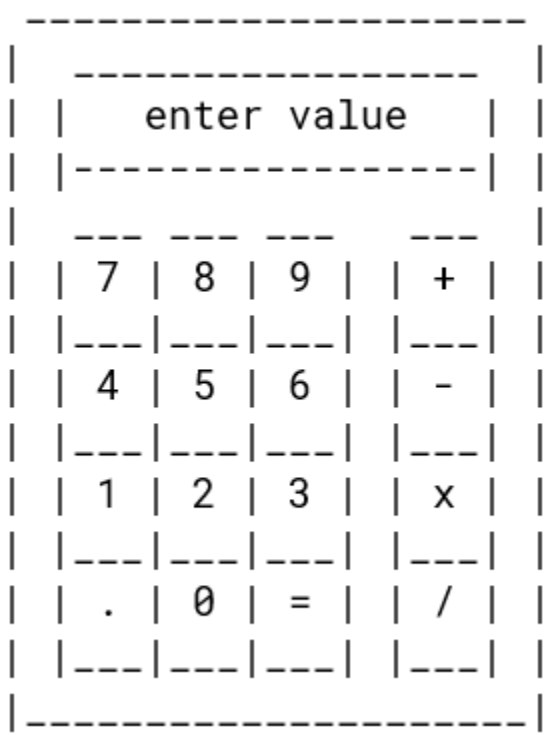
```
            index = r+1;
           }
           if((inp.charAt(r)=='*'||inp.charAt(r)=='(' ||inp.charAt(r)=='/'
)&&(inp.charAt(r+1)=='-')){
              for(int j = 0;j<inp.length();j++){
              if(j==index){
               med +="-(";
               }
               if((inp.charAt(j)=='*'||inp.charAt(j)=='(' )&&(inp.charAt(j+1)=='-')){
                   med += "*";
                   j++;continue;
                 }
               if((inp.charAt(j)=='/')&&(inp.charAt(j+1)=='-')){
                   med += "/";
                   j++;continue;
                 }
                 med += inp.charAt(j);
              }
            }
           }
          inp = med;

          if(inp.contains("(")||inp.contains(")")){
           calcbrack(inp);
           inp = Final;
           if(!inp.contains("(")||!inp.contains(")"))
           calc(inp);
        }
        else
        calc(inp);
```

```
        }
       if(inp.contains("/0"))
        printer = "cannot divide with 0";
        System.out.print( tab+"    _____\n"+
        "   |  _____  |\n"+
        "   | |");
        int p=0;
        for( p= 0;p<printer.length();p++){
        if(p==16)
        break;
        System.out.print(printer.charAt(p));
        }
        int space_left = 16-p;
        for( p= 0;p<=space_left;p++){
         System.out.print(" ");
        }
        System.out.print("| |\n"+
        "   | |_____|_|\n"+
        "   |  ___ ___ ___   ___  |\n"+
        "   | | 7 | 8 | 9 | | + | |\n"+
        "   | |___|___|___| |___| |\n"+
        "   | | 4 | 5 | 6 | | - | |\n"+
        "   | |___|___|___| |___| |\n"+
        "   | | 1 | 2 | 3 | | x | |\n"+
        "   | |___|___|___| |___| |\n"+
        "   | | . | 0 | = | | / | |\n"+
        "   | |___|___|___| |___| |\n"+
        "   |_____|\n");
        System.out.println("\n\n (in order to exit please enter 0)");
    }
}
```

Example Outputs:

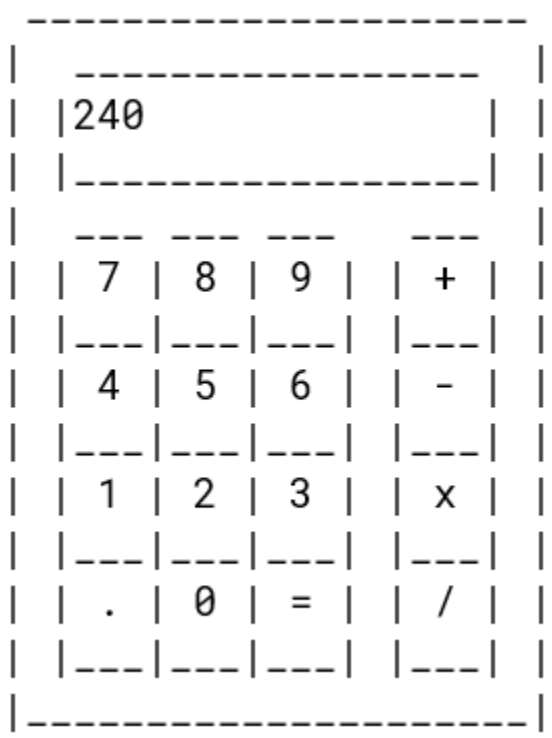
Interface:



Input/Output interface:

Input: 16/4*3(2+18)

Output:



Sample outputs:

Input	**Output**
7+(8-3*2)	9
0.5*0.25+12/6-0.1	2.025
4585/5+150*35-452(19/2)	1873
4(10+15/5*4-2*2)	72
5(5)+10/22	25.455
1400-458(2)+485/5-56(7*8)	-2555
1+25/78+25-0.156	26.165
484/22+45*2-78-45(2.5)	-78.5
718745+1875*(16/2*3)+78(50)	767645

Bibliography:

For finalizing this project, we used:

- ISC Computer Science textbook
- BlueJ
- MS Word

Signature