



Relatório da Aplicação 1

Atividade Apl1 – Avaliador de expressões matemáticas

Nome do Integrante (ordem alfabética)	TIA
Erik Samuel Viana Hsu	32265921
Mateus Kenzo Iochimoto	32216289
Rodrigo Machado de Assis Oliveira de Lima	32234678
Thiago Shihan Cardoso Toma	32210744

Conteúdo do Relatório

Printscreen de testes de execução das opções do Menu:

Exemplos de entradas com erros:

“” – (input vazio)

```
Menu de Opções
1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:1
Leitura infixa

Digite a expressão matemática em notação infixa:
Nenhuma variável detectada, expressão inválida.

Menu de Opções
1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:
```

“a++”



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados I



```
Digite a expressão matemática em notação infixa: a++  
Dois operadores seguidos, expressão inválida.
```

```
Menu de Opções
```

- 1 - Leitura infixa
- 2 - Entrada Variáveis
- 3 - Converte pósfixa
- 4 - Calcula Resultado
- 5 - Encerra

```
Opção:
```

“(a”

```
Menu de Opções
```

- 1 - Leitura infixa
- 2 - Entrada Variáveis
- 3 - Converte pósfixa
- 4 - Calcula Resultado
- 5 - Encerra

```
Opção:1
```

```
Leitura infixa
```

```
Digite a expressão matemática em notação infixa: (a  
Parênteses incorretos, expressão inválida.
```

```
Menu de Opções
```

- 1 - Leitura infixa
- 2 - Entrada Variáveis
- 3 - Converte pósfixa
- 4 - Calcula Resultado
- 5 - Encerra

```
Opção:
```

“((a+b)”



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados I



```
Menu de Opções

1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:1
Leitura infixa

Digite a expressão matemática em notação infixa: ((a+b)
Parênteses incorretos, expressão inválida.

Menu de Opções

1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:|
```

“aa”

```
Menu de Opções

1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:1
Leitura infixa

Digite a expressão matemática em notação infixa: aa
Apenas variáveis de 1 são letra aceitas, expressão inválida.

Menu de Opções

1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:
```

Exemplo de expressão válida:

$(A+B)/(C-D)*E$

{A=7, B=3, C=6, D=4, E=9}

Expressão pósfixa esperada: AB+CD-/E*

resultado = 45



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados I



Menu de Opções

- 1 - Leitura infixa
- 2 - Entrada Variáveis
- 3 - Converte pósfixa
- 4 - Calcula Resultado
- 5 - Encerra

Opção:1

Leitura infixa

Digite a expressão matemática em notação infixa: (A+B)/(C-D)*E
Expressão válida.

Menu de Opções

- 1 - Leitura infixa
- 2 - Entrada Variáveis
- 3 - Converte pósfixa
- 4 - Calcula Resultado
- 5 - Encerra

Opção:2

Opção:2

Entrada de Variáveis

Qual o valor de A? 7

Qual o valor de B? 3

Qual o valor de C? 6

Qual o valor de D? 4

Qual o valor de E? 9

Menu de Opções

- 1 - Leitura infixa
- 2 - Entrada Variáveis
- 3 - Converte pósfixa
- 4 - Calcula Resultado
- 5 - Encerra

Opção:3

Converte Pósfixa

posfixa = AB+CD-/E*

Menu de Opções



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados I



```
Menu de Opções

1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:4
Calcula Resultado

resultado = 45
posfixa = AB+CD-/E*
variáveis = {A=7, B=3, C=6, D=4, E=9}

Menu de Opções

1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:5
```

```
Opção:5
Encerra o programa
```

Outro exemplo de expressão válida:

$(A-B) \cdot (A+B)^C / D \cdot (E+E)$

$\{A = 3, B = 2, C = 2, D = 5, E = 5\}$

Expressão pósfixa esperada: $AB-AB+C^*D/EE+*$

resultado = 50



UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados I



Menu de Opções

- 1 - Leitura infixa
- 2 - Entrada Variáveis
- 3 - Converte pósfixa
- 4 - Calcula Resultado
- 5 - Encerra

Opção:1

Leitura infixa

Digite a expressão matemática em notação infixa: $(A-B)*(A+B)^C/D*(E+E)$
Expressão válida.

Menu de Opções

- 1 - Leitura infixa
- 2 - Entrada Variáveis
- 3 - Converte pósfixa
- 4 - Calcula Resultado
- 5 - Encerra

Opção:2

Opção:2

Entrada de Variáveis

Qual o valor de A? 3

Qual o valor de B? 2

Qual o valor de C? 2

Qual o valor de D? 5

Qual o valor de E? 5

Menu de Opções

- 1 - Leitura infixa
- 2 - Entrada Variáveis
- 3 - Converte pósfixa
- 4 - Calcula Resultado
- 5 - Encerra

Opção:3



```
Opção:3
Converte Pósfixa
posfixa = AB-AB+C^*D/EE+*

Menu de Opções

1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:4
Calcula Resultado

resultado = 50
posfixa = AB-AB+C^*D/EE+*
variáveis = {A=3, B=2, C=2, D=5, E=5}

Menu de Opções

1 - Leitura infixa
2 - Entrada Variáveis
3 - Converte pósfixa
4 - Calcula Resultado
5 - Encerra

Opção:5

Opção:5
Encerra o programa
```

Código fonte desenvolvido:

- **Código fonte Pilha.java:**



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados I



```
1  public class Pilha<T>{
2      private static int TAM_DEFAULT = 100;
3      private int topoPilha;
4      private T[] e;
5
6      //CONSTRUTOR COM PARÂMETRO
7      public Pilha(int tamanho){
8          this.e = (T[]) new Object[tamanho];
9          this.topoPilha = -1;
10     }
11     //CONSTRUTOR VAZIO
12     public Pilha(){
13         this(TAM_DEFAULT);
14     }
15
16     //VERIFICA SE A PILHA ESTÁ VAZIA
17     public boolean isEmpty(){
18         return this.topoPilha == -1;
19     }
20
21     //VERIFICA SE A PILHA ESTÁ CHEIA
22     public boolean isFull(){
23         return this.topoPilha == this.e.length - 1;
24     }
25
26     //INSERE UM ELEMENTO NO TOPO DA PILHA
27     public void push(T elem){
28         if (!this.isFull())
29             this.e[++this.topoPilha] = elem;
30         else
31             System.out.println("Stack Overflow");
32     }
33 }
```




```
33
34 //REMOVE O ELEMENTO DO TOPO DA PILHA
35 public T pop(){
36     if (!this.isEmpty())
37         return this.e[this.topoPilha--];
38     else{
39         System.out.println("Stack Underflow");
40         return null;
41     }
42 }
43
44 //OBTÉM O ELEMENTO DO TOPO DA PILHA
45 public T top(){
46     if (! this.isEmpty())
47         return this.e[this.topoPilha];
48     else{
49         System.out.println("Stack Underflow");
50         return null;
51     }
52 }
53
54 //OBTÉM A QUANTIDADE DE ELEMENTOS NA PILHA
55 public int sizeElements(){
56     return topoPilha+1;
57 }
58 }
```

- Código fonte Main.java:



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados I



```
/
8  import java.lang.Math;
9  import java.util.*;
10
11 class Main {
12     public static boolean validaExp(String exp) {
13         //Vetor com operadores validos
14         char[] opValidos = new char[5];
15         opValidos[0] = '+';
16         opValidos[1] = '-';
17         opValidos[2] = '*';
18         opValidos[3] = '/';
19         opValidos[4] = '^';
20
21         int contAbre = 0;
22         int contFecha = 0;
23         int contVariavel = 0;
24
25         //loop que varre expressão
26         for(int i=0; i < exp.length();i++) {
27             if (exp.charAt(i) == '(')
28                 contAbre++;
29             if (exp.charAt(i) == ')')
30                 contFecha++;
31
32             //Se parênteses for fechado sem que tenha sido aberto retorna false
33             if (contFecha > contAbre) {
34                 System.out.print("Parênteses incorretos, expressão inválida.");
35                 return false;
36             }
37             if(Character.isLetter(exp.charAt(i)))
38                 contVariavel++;
39             //Se duas letras vierem em sequência retorna false
40             if (i!= exp.length()-1 && Character.isLetter(exp.charAt(i)) &&
41                 Character.isLetter(exp.charAt(i+1))) {
42                 System.out.print("Apenas variáveis de 1 são letra aceitas, expressão inválida.");
43                 return false;
44             }
45             //Se caractere não for letra nem parênteses compara com vetor de operadores válidos
46             if(Character.isLetter(exp.charAt(i)) == false && exp.charAt(i) != '(' && exp.charAt(i) !=
47                 ')') {
48                 int valido = 0;
49                 for (int j=0; j < 5;j++) {
50                     if (exp.charAt(i) == opValidos[j])
51                         valido = 1;
52                 }
53                 //Caso caractere não estiver no vetor retorna false
54                 if (valido == 0) {
55                     System.out.print("Operando ou operador não reconhecido, expressão inválida.");
56                 }
57             }
58         }
59     }
60 }
```



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Estruturas de Dados I



```
52     System.out.print("Operando ou operador não reconhecido, expressão inválida.");
53     return false;
54 }
55 //Se o caractere do próximo índice também for operador retorna false
56 else {
57     if(valido == 1 && i < exp.length()-1) {
58         for(int k = 0; k < 5; k++) {
59             if((exp.charAt(i+1)) == opValidos[k]) {
60                 System.out.print("Dois operadores seguidos, expressão inválida.");
61                 return false;
62             }
63         }
64     }
65 }
66 }
67 //Se operador for último caractere da expressão retorna false
68 if (i == exp.length()-1) {
69     System.out.print("Operador não pode ser ultimo caractere da expressão, expressão
70 inválida.");
71     return false;
72 }
73 }
74 //Se não houver nenhuma variável na expressão retorna false
75 if (contVariavel == 0) {
76     System.out.print("Nenhuma variável detectada, expressão inválida.");
77     return false;
78 }
79 //Se parênteses aberto não for fechado retorna false
80 if (contAbre != contFecha) {
81     System.out.print("Parênteses incorretos, expressão inválida.");
82     return false;
83 }
84 System.out.print("Expressão válida.");
85 return true;
86 }
87 //Etapa de parentização e conversão da expressão para pósfixa
88 public static String ConvertePosfixa(String exp) {
89     Pilha<Character> pilha1 = new Pilha<Character>();
90     String expConvertida = "";
91
92     for (int i = 0; i < exp.length(); i++) {
93         char c = exp.charAt(i);
94         //Se for '(', sempre deve ser empilhado, pois indica o início de uma operação matemática
95         if (exp.charAt(i) == '(') {
96             pilha1.push(exp.charAt(i));
97         }
98         //Se for operando (letra), copiá-lo diretamente para a expressão pósfixa
99         else if (Character.isLetter(exp.charAt(i)) == true) {
```



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados I



```
100     expConvertida += exp.charAt(i);
101     }
102     //Se for operador, fazê-lo aguardar
103     else if(exp.charAt(i) == '+' || exp.charAt(i) == '-' || exp.charAt(i) == '*' || exp.charAt(i)
104 == '/' || exp.charAt(i) == '^') {
105         while(pilha1.isEmpty() == false && pilha1.top() != '(' && precedencia(pilha1.top()) >=
106 precedencia(c)) {
107             //Enquanto for verdadeiro, o topo da pilha é removido e adicionado à string
108             expConvertida += pilha1.pop();
109         }
110         //Quando a pilha estiver vazia, o operador atual é empilhado na pilha.
111         pilha1.push(exp.charAt(i));
112     }
113     //Se for ')', adicionar o último operador (topo da pilha) na expressão
114     //Essa etapa garante que a expressão dentro dos parênteses seja calculada antes do
115     restante da expressão
116     else if(exp.charAt(i) == ')') {
117         //Loop para remover os operadores da pilha até que encontre '('
118         while(pilha1.isEmpty() == false && pilha1.top() != '(') {
119             expConvertida += pilha1.pop();
120         }
121         //Se for encontrado, ele é removido da pilha e o loop é encerrado
122         if (pilha1.isEmpty() == false && pilha1.top() == '(') {
123             pilha1.pop();
124         }
125     }
126     //Se não, a expressão é inválida e retorna null
127     else {
128         System.out.println("Expressão inválida");
129         return null;
130     }
131 }
132 //Desempilhar operadores e operandos que restaram na pilha
133 //Garantindo que todos os caracteres sejam processados e que a expressão seja válida
134 while(pilha1.isEmpty() == false) {
135     //Se for '(', a expressão é inválida e retorna null
136     if (pilha1.top() == '(') {
137         System.out.println("Expressão inválida");
138         return null;
139     }
140     //Caso contrário, o operador ou operando é adicionado à expConvertida
141     expConvertida += pilha1.pop();
142 }
143 //Retorna a expressão pósfixa
144 return expConvertida;
145 // Final da função ParentizaExp
146 }
```

```
145 //Função que determina precedência dos operadores matemáticos
146 //Quanto maior for o número retornado, maior a precedência do caracter
```



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Estruturas de Dados I



```
147 //Operadores com a mesma precedência serão avaliados da esquerda para a direita
148 ~ public static int precedencia(char c) {
149 ~     if (c == '^') { //Operador de potencialização é o com maior precedência
150         return 4;
151     }
152 ~     else if(c == '*' || c == '/') { //Em seguida os operadores de multiplicação e divisão
153         return 3;
154     }
155 ~     else if(c == '+' || c == '-') { //Em seguida os operadores de soma e subtração
156         return 2;
157     }
158 ~     else {
159         return 1;
160     }
161 }
162
163 ~ public static void main(String[] args) {
164     Map<Character, Integer> valorVariaveis = new HashMap<>();
165     Pilha<Character> op = new Pilha<Character>(); // pilha para operações
166     Pilha<Integer> variaveis = new Pilha<Integer>();
167     Scanner s = new Scanner(System.in);
168     String expressao = "";
169     String posfixa = "";
170     int aux;
171     String operacoes = "+-*/^()":
172     ~ boolean flag = false;
173
174     String opcoes = "\n\nMenu de Opções\n\n1 - Leitura infixa\n2 - Entrada Variáveis\n3 - Converte
175     pósfixa\n4 - Calcula Resultado\n5 - Encerra\n\n Opção:";
176
177     Scanner ent = new Scanner(System.in);
178     int opcao = 0;
179     do{
180         System.out.print(opcoes);
181         opcao = ent.nextInt();
182         switch(opcao){
183             case 1:
184                 System.out.println("Leitura infixa");
185                 System.out.print("\nDigite a expressão matemática em notação infixa: ");
186                 expressao = s.nextLine();
187                 flag = validaExp(expressao); // flag para dizer se a expressão é válida ou não
188                 valorVariaveis.clear(); //limpa o dicionário
189                 //op.clear(); //limpa a pilha
190                 //variaveis.clear(); //limpa a pilha
191
192                 break;
193             case 2:
194                 ~ if (flag != false){ // não faz a opção caso a flag seja falsa
195                     System.out.println("Entrada de Variáveis");
```



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira

Estruturas de Dados I



```
196     for (int i = 0; i < expressao.length(); i++){ // for que percorre cada char da
    expressao
197     if (operacoes.indexOf(expressao.charAt(i)) == -1){ // "se o char não estiver em
    operacoes"
198         if (valorVariaveis.containsKey(expressao.charAt(i))) // "se o char estiver no
    dicionário" pule a iteração
199             continue;
200         System.out.printf("\nQual o valor de %c? ", expressao.charAt(i));
201         int valor = s.nextInt();
202         valorVariaveis.put(expressao.charAt(i), valor); // adiciona ao dicionário
203     }
204 }
205 break;
```

```
206
207 case 3:
208     if (flag != false){ // não faz a opção caso a flag seja falsa
209         System.out.println("Converte Pósfixa");
210         posfixa = ConvertePosfixa(expressao);
211         System.out.printf("posfixa = %s", posfixa);
212     }
213     break;
214
215 case 4:
216     System.out.println("Calcula Resultado");
```

```
217
218
219     for (int i=0; i<posfixa.length(); i++){ // for que percorre cada char de posfixa
220         if (operacoes.indexOf(posfixa.charAt(i)) == -1) // se o char não for uma operação
221             variaveis.push(valorVariaveis.get(posfixa.charAt(i)));
222
223         else{
224             if (posfixa.charAt(i) == '+')
225                 variaveis.push(variaveis.pop()+variaveis.pop());
226             if (posfixa.charAt(i) == '-'){
227
228                 aux = variaveis.pop();
229                 variaveis.push(variaveis.pop()-aux);
230             }
231             if (posfixa.charAt(i) == '/'){
232                 aux = variaveis.pop();
233                 variaveis.push(variaveis.pop()/aux);
234             }
235             if (posfixa.charAt(i) == '*')
236                 variaveis.push(variaveis.pop()*variaveis.pop());
237
238             if (posfixa.charAt(i) == '^'){
239                 aux = variaveis.pop();
240                 variaveis.push((int) Math.pow(variaveis.pop(), aux));
```



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados I



```
241     }
242     }
243 }
244 System.out.printf("\nresultado = %d", variaveis.pop()); // print do resultado da operação
245 System.out.printf("\nposfixa = %s", posfixa);
246 System.out.println("\nvariáveis = "+valorVariaveis);
247 break;
248
249 case 5:
250     System.out.println("Encerra o programa");
251     break;
252 }
253 }while (opcao != 5);
254
255
256 }}
257
```