Faculdade de Computação e Informática
Ciência da Computação
Estrutura de Dados I – 3ª etapa – 2024.2
Professores: André Kishimoto, Charles Boulhosa Rodamilans, Ivan Carlos Alcântara de Oliveira, Leonardo Massayuki Takuno

Atividade Apl1 – Avaliador de expressões matemáticas

Uma aplicação clássica do TAD Pilha é a avaliação de expressões matemáticas.

Podemos encontrar algumas dificuldades na avaliação de uma expressão matemática, tais como:

- A existência de prioridades diferentes para os operadores, que não nos permite efetuá-los na ordem em que os encontramos na expressão;
- A existência de parênteses, que alteram a prioridade das operações.

Uma solução para os itens descritos no parágrafo anterior foi criada pelo polonês Jan Łukasiewicz e consiste em usar uma representação de expressões matemáticas onde não existam prioridades e nem a necessidade do uso de parênteses. Essa solução é conhecida como Notação Polonesa Reversa (*Reverse Polish Notation* ou RPN).

Na notação infixa, a mais "tradicional" na aritmética, o operador aparece entre os operandos. Por exemplo: **A + B**.

Além da notação infixa, temos:

- A notação prefixa: o operador precede os operandos. Por exemplo: **+ A B**.
- A notação posfixa: o operador segue os operandos. Por exemplo: **A B +**.

As formas prefixa e posfixa são denominadas notação polonesa e notação polonesa reversa, respectivamente.

A notação polonesa reversa tem algumas vantagens em relação à notação infixa:

- Cada operador aparece imediatamente após os valores que ele deve operar;
- Não existe a necessidade de usar parênteses.

A tabela a seguir contém alguns exemplos de notação infixa e o equivalente em notação posfixa.

Notação infixa	**Notação posfixa**
A + B * C	A B C * +
A * (B + C)	A B C + *
(A + B) / (C – D)	A B + C D - /
(A + B) / (C – D) * E	A B + C D - / E *

Funcionamento do programa

Desenvolver um programa em Java que implementa um REPL (*Read-Evaluate-Print-Loop*) que aceita entradas via teclado e que avalia e valida as entradas de acordo com os requisitos descritos a seguir.

Expressões matemáticas

- Devem conter somente variáveis e operadores (ver abaixo).
- Podem conter espaços em branco, assim como não ter espaços em branco entre variáveis e operadores.
- Devem estar em notação infixa.
- Devem ser convertidas para a notação posfixa.
- O cálculo da expressão deve ser realizado usando a expressão posfixa, somente se todas as variáveis presentes na expressão possuem valores definidos.
- O programa deve validar as expressões matemáticas, isto é:
  - Aceitar somente as operações indicadas abaixo.
  - Aceitar somente variáveis como operandos, sendo que as variáveis possuem uma única letra, conforme indicado abaixo.
  - Considerar que uma expressão matemática na notação infixa pode conter parênteses que definem a prioridade das operações.
- Caso a expressão inserida seja inválida (por exemplo, a expressão contém algum operador que não seja um dos cinco indicados ou possui uma quantidade incorreta de parênteses, como **`((((A * (B - C))`**, o programa deve exibir uma mensagem informando o erro.

Operações suportadas

- Binárias: **`+ - * / ^`** (adição, subtração, multiplicação, divisão e exponenciação, respectivamente).
- Unária de negação (desafio opcional): **`-`**
- Parênteses: **`( )`**

Variáveis

- **`A-Z`**, *case insensitive*.

REPL: Comandos válidos (case insensitive)

Comando	Descrição
Expressão matemática infixa	Após validação, exibe o resultado em tela.
<VAR> = <VALUE>	Atribui um valor para uma variável, sendo: **<VAR>**: Variável (**`A-Z`**). **<VALUE>**: Número real.
VARS	Lista somente as variáveis com valores definidos e seus respectivos valores.
RESET	Reinicia todas as variáveis.
REC	Começa a gravar (inserir) os comandos digitados em uma fila. A fila deve possuir um limite de 10 comandos. Caso o limite seja atingido, o comando REC é parado automaticamente.

Comando	Descrição
	O programa deve mostrar quantos comandos ainda podem ser gravados na fila. No modo REC, os comandos inseridos não devem ser executados, apenas gravados na fila.
STOP	Para de gravar os comandos.
PLAY	Reproduz os comandos gravados na fila, na sequência correta. A execução do comando **PLAY** não deve perder o conteúdo gravado na fila.
ERASE	Apaga todos os comandos da fila.
EXIT	Encerra o programa.

Quando estiver em modo REC (gravação), somente os quatro primeiros comandos da tabela acima são válidos. Ou seja, comandos relacionados à fila (**REC**, **STOP**, **PLAY**, **ERASE**) e encerramento do programa não são aceitos para gravação (não podem ser inseridos na fila de comandos).

Exemplo de execução do programa

Observação: Conteúdo com fundo cinza representa o que foi digitado pela pessoa usuária do programa.

```
> TESTE
Erro: comando inválido.
> X + Y
Erro: variável X não definida.
Erro: variável Y não definida.
> X = 2
X = 2
> X - TESTE
Erro: expressão inválida.
> (X * X
Erro: expressão inválida.
> Y = 3
Y = 3
> X + Y
5
> X
2
> VARS
X = 2
Y = 3
> RESET
Variáveis reiniciadas.
> VARS
Nenhuma variável definida.
> X + Y
```

```
Erro: variável X não definida.
Erro: variável Y não definida.
> REC
Iniciando gravação... (REC: 0/10)
> A = 10
(REC: 1/10) A = 10
> B = 20
(REC: 2/10) B = 20
> C = 30
(REC: 3/10) C = 30
> (A + B) / C
(REC: 4/10) (A + B) / C
> A % B
(REC: 5/10) A % B
> PLAY
Erro: comando inválido para gravação.
> VARS
(REC: 6/10) VARS
> RESET
(REC: 7/10) RESET
> STOP
Encerrando gravação... (REC: 7/10)
> PLAY
Reproduzindo gravação...
A = 10
B = 20
C = 30
(A + B) / C
1
A % B
Erro: operador inválido.
A = 10
B = 20
C = 30
Variáveis reiniciadas.
> ERASE
Gravação apagada.
> PLAY
Não há gravação para ser reproduzida.
> EXIT
```

Algoritmo para conversão de expressão infixa para posfixa

Um algoritmo para conversão de uma expressão infixa qualquer para posfixa seria:

- Inicie com uma pilha vazia;
- Realize uma varredura na expressão infixa, copiando todos os identificadores encontrados diretamente para a expressão de saída.

  a) Ao encontrar um operador:
  1. Enquanto a pilha não estiver vazia e houver no seu topo um operador com prioridade maior ou igual ao encontrado, desempilhe o operador e copie-o na saída;
  2. Empilhe o operador encontrado;

  b) Ao encontrar um parêntese de abertura, empilhe-o;

  c) Ao encontrar um parêntese de fechamento, remova um símbolo da pilha e copie-o na saída, até que seja desempilhado o parêntese de abertura correspondente.

- Ao final da varredura, esvazie a pilha, movendo os símbolos desempilhados para a saída.

Um exemplo de execução do algoritmo de conversão de infixa para posfixa, supondo a expressão **A*(B+C)/D**, é apresentado abaixo:

Símbolo	Ação	Pilha	Saída
A	copia para a saída	P:[]	**A**
*****	pilha vazia, empilha	P:[*]	**A**
(	sempre deve ser empilhado	P:[(, *]	**A**
B	copia para a saída	P:[(,*]	**AB**
+	prioridade maior, empilha	P:[+, (, *]	**AB**
C	copia para a saída	P:[+, (, *]	**ABC**
)	desempilha até achar ‘)’	P:[*]	**ABC+**
/	prioridade igual, desempilha	P:[/]	**ABC+***
D	copia para a saída	P:[/]	**ABC+*D**
	final, esvazia pilha	P:[]	**ABC+*D/**

Algoritmo de avaliação de uma expressão na forma posfixa

- Primeiramente, atribui-se valores numéricos às variáveis da expressão a ser avaliada;
- Inicia-se com uma pilha vazia;
- Varre-se a expressão e, para cada elemento encontrado:
  a) Se for operando, então empilhar seu valor;
  b) Se for operador, então desempilhar os dois últimos valores, efetuar a operação com eles e empilhar de volta o resultado obtido;
- No final do processo, o resultado da avaliação estará no topo da pilha.

Um exemplo de execução do algoritmo de avaliação de uma expressão na forma posfixa é apresentado abaixo, assumindo a expressão infixa **(A+B)/(C-D)*E** que foi convertida para a expressão posfixa **AB+CD-/E*** e que **A=7**, **B=3**, **C=6**, **D=4** e **E=9**.

Expressão	Elemento	Ação	Pilha
AB+CD-/E*			P:[ ]
B+CD-/E*	A	empilha valor de A = 7	P:[7]
+CD-/E*	B	empilha valor de B = 3	P:[3, 7]
CD-/E*	+	desempilha 3	P:[7]
		desempilha 7	P:[]
		empilha 3 + 7	P:[10]
D-/E*	C	empilha valor de C = 6	P:[6, 10]
-/E*	D	empilha valor de D = 4	P:[4, 6, 10]
/E*	-	desempilha 4	P:[6, 10]
		desempilha 6	P:[10]
		empilha 6 - 4	P:[2, 10]
E*	/	desempilha 2	P:[10]
		desempilha 10	P:[ ]
		empilha 10/2	P:[5]
*	E	empilha valor de E = 9	P:[9, 5]
		desempilha 9	P:[5]
		desempilha 5	P:[ ]
	*	empilha 5 * 9	P:[45]

Desenvolvimento

Grupo:
A atividade deve ser realizada em **grupo de, no máximo, 3 pessoas**.

- A solução deve ser implementada em linguagem Java e deve usar uma versão adaptada da implementação do TAD Pilha estática/sequencial realizada durante as aulas.
- Fica proibido o uso de estruturas de dados fornecidas pela linguagem Java (restrição inclui uso de Stack, Map, Hashtable, Vector, etc). Projetos que usarem tais estruturas serão desconsiderados – zero.
- Inclua a identificação do grupo (nome completo e RA de cada integrante) no início de cada arquivo de código, como comentário.
- Inclua todas as referências (livros, artigos, sites, vídeos, entre outros) consultadas para solucionar a atividade como comentário no arquivo .java que contém a main().

Entrega

Compacte o código-fonte (somente arquivos *.java) no **formato zip**.

Atenção: O arquivo zip não deve conter arquivos intermediários e/ou pastas geradas pelo compilador/IDE (ex. arquivos *.class, etc.).

Prazo de entrega: via link do Moodle até 01/10/2024 23:59.

Critérios de avaliação

A nota da atividade é calculada de acordo com os critérios da tabela a seguir.

Item avaliado	Pontuação máxima
Implementação do **REPL** e comando **EXIT**.	até 0,5 ponto
Comando **<VAR> = <VALUE>**.	até 0,5 ponto
Comandos **VARS** e **RESET**.	até 0,5 ponto
Comandos **REC** e **PLAY**.	até 1,5 ponto
Comandos **STOP** e **ERASE**.	até 0,5 ponto
Conversão da expressão em notação infixa para notação posfixa.	até 1,5 ponto
Avaliação da expressão em notação posfixa (cálculo e apresentação do resultado da expressão).	até 1,0 ponto
Validação das entradas da pessoa usuária do programa.	até 1,0 ponto
Explicação sobre a solução do problema (apresentação e/ou questionário em horário de aula – a definir).	até 3,0 pontos
Opcional: suporte ao operador unário de negação.	até 0,5 ponto

Tabela 1 - Critérios de avaliação.

A tabela a seguir contém critérios de avaliação que podem **reduzir** a nota final da atividade.

Item indesejável	Redução de nota
O projeto é cópia de outro projeto.	Projeto é zerado
O projeto usa a classe Stack e/ou outras estruturas de dados fornecidas pela linguagem Java.	Projeto é zerado
Há erros de compilação e/ou o programa trava durante a execução.[1]	-1,0 ponto
Não há identificação do grupo no código-fonte. Não há indicação de referências no código-fonte. Arquivos enviados em formatos incorretos. Arquivos e/ou pastas intermediárias que são criadas no processo de compilação foram enviadas junto com o código-fonte.	-1,0 ponto

Tabela 2 - Critérios de avaliação (redução de nota).

[1] Sobre erros de compilação: considere apenas erros. Não há problema se o projeto tiver *warnings* (embora *warnings* podem avisar sobre possíveis travamentos em tempo de execução, como loop infinito, divisão por zero, etc.).

O código-fonte será compilado com o compilador **javac** (21.0.2) na plataforma Windows da seguinte forma:

```
> javac *.java -encoding utf8
```

O código compilado será executado com **java** (21.0.2) na plataforma Windows da seguinte forma:

```
> java <Classe>
```

Sendo que **<Classe>** deve ser substituído pelo nome da classe que contém o método **public static void main(String[] args)**.