MASTER CALCULATOR

Matheus Vinícius Mahl

Marcelo Sehnem

Resumo

Esse projeto tem como finalidade mostrar uma simples calculadora que tem como objetivo, além de realizar cálculos, transformar a expressão infixa, criada pelo usuário, para uma expressão pós-fixa. Utilizando o ambiente Eclipse junto com a linguagem de programação orientada a objetos Java, o projeto utiliza a estrutura de pilhas como meio de transformar a equação para pós-fixa. A interface em que o usuário tem acesso é semelhante a uma calculadora com as operações básicas de soma, subtração, multiplicação e divisão, além do uso de parênteses.

Palavras-chave: Calculadora, Pós-fixa, Transformação.

Funcionamento

Para o fácil uso do programa, foi criada uma interface gráfica simples e semelhante a uma calculadora comum, onde o usuário pode clicar nos botões dos operandos e operadores ou também apertar no teclado a tecla correspondente à da calculadora.

Para fins de verificação e escrita correta da expressão matemática, o usuário não tem permissão de escrever diretamente no campo de criação da equação, portanto ele terá que utilizar os botões para criar a expressão. Para facilitar a escrita da expressão, foi criado um método que verifica o pressionamento de teclas no teclado para realizar um clique no botão correspondente a da tecla física apertada pelo usuário.

O espaço de criação da expressão a ser resolvida também foi limitado a 13 caracteres, apenas para a expressão não ficar muito grande, já que o objetivo do programa não é resolver equações complexas, mas demonstrar a transformação da expressão criada. O usuário pode adicionar números, operadores e parênteses sempre da direita para a esquerda e se precisar apagar um caractere digitado errado poderá utilizar

Graduandos do Curso Bacharelado de Ciência da Computação.
UNOESC - Universidade do Oeste de Santa Catarina http://www.unoesc.edu.br/
Rua Oiapoc, 211 - Bairro Agostini - São Miguel do Oeste - SC - CEP 89900-000.

matheusmahl@outlook.com, marcelosehnem@yahoo.com.br

o botão que possui uma seta para a esquerda, eliminando um a um os caracteres, não podendo, também, eliminar um caractere no meio da expressão sem eliminar os que estão à direita.

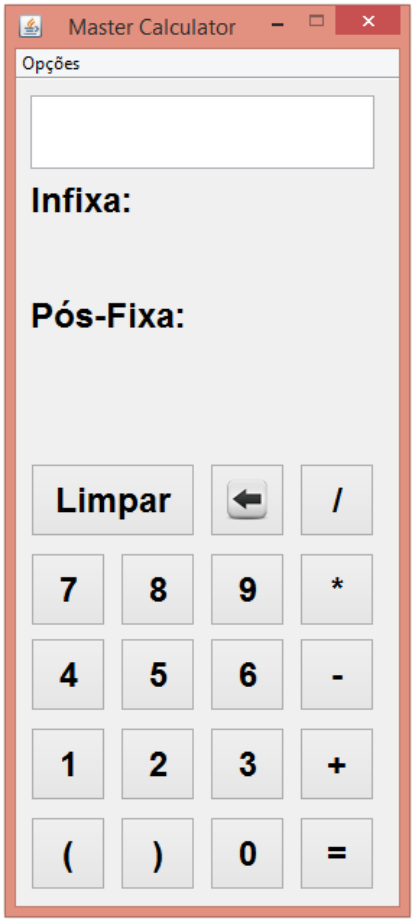

Figura 1: Tela do aplicativo.

Desenvolvimento e Métodos

A transformação da expressão infixa para pós-fixa foi feita utilizando uma pilha de caracteres, e para que pudesse ser feita essa transformação, foi preciso designar algumas regras para que a expressão matemática seja válida e que possa devolver um resultado coerente e correto. Para isso foi criado o método "verificarExpressão()" que retorna verdadeiro se a expressão foi válida, ou falso, caso a expressão não passe pelos critérios descritos a seguir:

- Se a expressão possuir menos que um caractere;
- Primeiro e último caractere da expressão não pode ser um operador;
- Último caractere da expressão também não pode ser parêntese aberto;
- Se houver um parêntese fechado e nenhum parêntese aberto equivalente;
- Se houver dois operadores seguidos ou um operador seguido de um parêntese fechado;
- Se houver um parêntese aberto após um operando;
- Ou se a expressão não possuir nenhum operador.

Caso as regras sejam seguidas, a expressão terá uma forma válida para ser resolvida e transformada para pós-fixa. Para tornar mais fácil a verificação dos caracteres da expressão, eles são gravados em um objeto do tipo "StringBuffer", utilizando o método "adicionarNaLista(char ACaracter)", facilitando a verificação de todos os caracteres, como se fosse um "array" de caracteres, e também facilitando a concatenação de todos os caracteres para formar uma "String", que será utilizada para mostrar a expressão na tela e para que possa ser resolvida.

Ainda como método de validação, temos o "fecharParenteses", que simplesmente conta a quantidade de parênteses abertos e fechados, assim, caso o usuário não tenha fechado todos os parênteses que abriu, esse método adiciona os parênteses fechados faltantes no final da equação. Caso possua mais parênteses fechados que abertos, um dos critérios descritos acima trata de invalidar a expressão.

Para resolver a equação, um objeto da classe "ScriptEngine" foi criado para ser possível utilizar o método "eval()" , com base em JavaScript, que retorna o resultado da equação matemática que foi criada com base nas regras especificadas anteriormente. Esse método foi utilizado para facilitar o trabalho de resolver a equação, já que o foco está na parte de conversão da expressão para pós-fixo.

O método "atualizarTela(boolean ACheck)" possui a função de colocar os caracteres gravados no "stringbuffer" na tela que está visível ao usuário. Ela também chama a função "limparExpressão()" caso o parâmetro passado para ela seja "true", caso que só ocorre quando o botão "=" (igual) for pressionado, para limpar o "stringbuffer" e permitir que seja colocada uma nova expressão.

Ainda temos o método "entradaTeclado()" que por meio de um "KeyEventDispatcher" realiza a comparação das teclas do teclado pressionadas pelo

usuário com o texto dos botões do programa, onde, caso sejam iguais, é realizado o click virtual dos botões da calculadora. Esse método serve apenas para dar uma certa comodidade na criação da expressão matemática, sendo que ele não afeta na resolução da expressão e nem na transformação. Vale lembrar que algumas teclas não são validadas por não possuírem o texto correspondente no teclado, como o botão "Limpar" e o botão "ApagarUm", esse último identificado com uma flecha apontando para a esquerda.

Transformando de Infixo para Pós-Fixo

Para transformar a equação gerada pelo usuário de infixa para pós-fixa foi criada uma classe, chamada "TTransformar" , que contém os métodos necessários tanto para a conversão como para a verificação da equação pelos métodos citados anteriormente.

Os métodos "isOperando(char ACaracter)" e "isOperador(char ACaracter)" tem a simples função de retornar se um caractere é um operando, utilizando o método "isDigit" da classe "Character", ou um operador (Exemplo: '+', '-') , respectivamente. São métodos públicos, pois foram também utilizados pela classe "Calculadora" para fazer a verificação da expressão.

O método privado "getPrioridade(char ACaracter)" é utilizado para retornar a prioridade dos caracteres da lista, necessário para auxiliar o empilhamento e desempilhamento dos caracteres durante a conversão.

Enfim vem o método "ConverterPosFixa(String AExpInfixa)", que irá realizar a conversão da expressão infixa para pós-fixa. O algoritmo de conversão utiliza uma pilha para auxiliar a conversão da expressão. Ela recebe como parâmetro uma String contendo a expressão infixa feita pelo usuário. O método inicia criando uma String "sExpPosFixa" contendo um valor vazio ("") e cria um objeto "oPilha" da classe "Stack", representando a pilha, do tipo "Character".

Assim, um método iterativo (for) vai percorrer cada caractere da expressão Infixa recebida, avaliando o caractere para criar a expressão pós-fixa, um por um. Para essa avaliação, alguns critérios devem ser seguidos:

1. Se o caractere for um operando, ele é adicionado diretamente na expressão de saída.
2. Se for um operador, ele busca a prioridade do operador;

2.1 Enquanto a pilha não estiver vazia e houver um operador, no topo da pilha, com prioridade maior ou igual à prioridade encontrada anteriormente, desempilhe o operador, adicionando na expressão de saída;

2.2 Empilha o operador que foi encontrado.

3. Se encontrar um parêntese de abertura, ele é adicionado na pilha.
4. Se encontrar um parêntese de fechamento, retiram-se os caracteres da pilha para a expressão de saída até que encontre o parêntese de abertura correspondente.

5. No final da avaliação, desempilha-se a pilha para a expressão de saída até que não possua mais nenhum elemento.

Assim, é criada a expressão de saída que já está na forma pós-fixa, a qual é retornada pelo método para o objeto que a chamou.

Conclusão

Esse programa foi criado para transformar uma expressão infixa para pós-fixa utilizando a estrutura de dados do tipo pilha. A forma da calculadora foi feita para prover uma interface gráfica para o usuário interagir e observar a transformação da expressão, além de devolver o resultado dessa expressão. Como o objetivo era mostrar que é possível utilizar a pilha para realizar a transformação, a calculadora foi feita de forma simples, sem suporte a operações mais complexas ou expressões maiores.

Sendo assim, podemos ver que o uso da pilha é simples e útil para realizar trocas de caracteres ou qualquer objeto de posição, além de poder armazená-los para uma posterior utilização, e a utilização da classe “Stack” em Java facilita a criação e uso das pilhas.

Referências

http://www.decom.ufop.br/menotti/aedI092/tps/tp3-sol3.pdf - Acesso em 15/11/2014

http://www.microsoft.com/brasil/msdn/tecnologias/vbnet/visualbasic_stack.mspx - Acesso em 16/11/2014