

Boanerges Junior /Matheus Amancio

Trabalho de Paradigma de Linguagens de Programação

Brasil

3 de dezembro de 2019

Boanerges Junior /Matheus Amancio

Trabalho de Paradigma de Linguagens de Programação

Relatório de estudos realizados envolvendo
equações matemáticas e físicas

Universidade Federal de Lavras

Ciência da Computação

Brasil

3 de dezembro de 2019

Sumário

1	INTRODUÇÃO	3
2	SOBRE O CÓDIGO	4
3	DIFICULDADES E OPERAÇÕES	6
4	ESCOLHA E DESISTÊNCIA DE LINGUAGEM	7

1 Introdução

O sistema deverá apresentar ao usuário a listagem das equações disponível para solução. Dependendo da equação escolhida, os valores das variáveis, exceto uma, devem ser solicitados e, após o cálculo, o valor da variável, cujo valor não foi informado, deve ser exibido. Esse loop de interação deve ser realizado continuamente, até que o usuário escolha sair (por alguma opção do sistema).

Por exemplo (veja a figura 1).

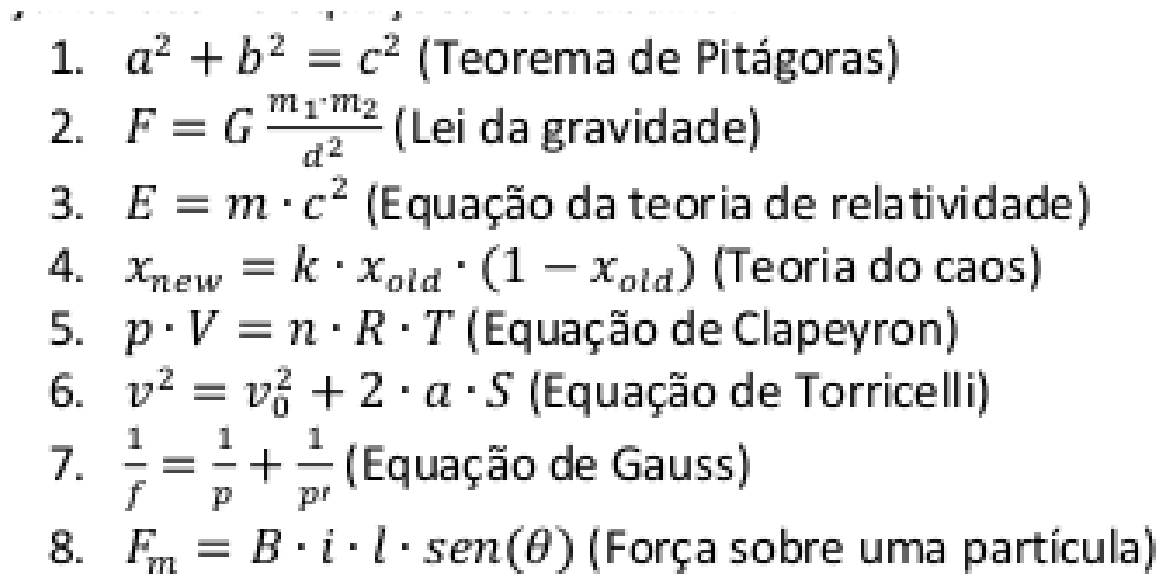
- 
1. $a^2 + b^2 = c^2$ (Teorema de Pitágoras)
 2. $F = G \frac{m_1 \cdot m_2}{d^2}$ (Lei da gravidade)
 3. $E = m \cdot c^2$ (Equação da teoria de relatividade)
 4. $x_{new} = k \cdot x_{old} \cdot (1 - x_{old})$ (Teoria do caos)
 5. $p \cdot V = n \cdot R \cdot T$ (Equação de Clapeyron)
 6. $v^2 = v_0^2 + 2 \cdot a \cdot S$ (Equação de Torricelli)
 7. $\frac{1}{f} = \frac{1}{p} + \frac{1}{p'}$ (Equação de Gauss)
 8. $F_m = B \cdot i \cdot l \cdot \text{sen}(\theta)$ (Força sobre uma partícula)

Figura 1 – Equações

2 Sobre o código

O código descrito foi feito na linguagem JAVA orientada a objetos e utilizando a IDE NetBeans para facilitar a implementação de uma interface gráfica orietada a eventos.

A praticidade do NetBeans deve-se ao fato que para montar e estruturar a interface nós podemos 'codar' os campos de texto, botões, etc ou arrastar essas aplicações sobre um JFrame, deste modo toda aplicação que é arrastada automaticamente é criado um trecho de código referenciando-a.

Por exemplo:

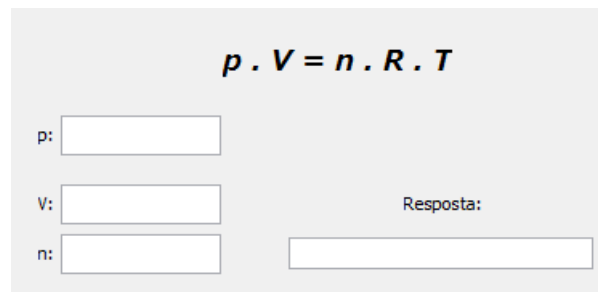


Figura 2 – JFrame

```
private void pKeyReleased(java.awt.event.KeyEvent evt) {  
    p.setText(p.getText().replaceAll("[^0-9 | ^.]", ""));  
}
```

Figura 3 – Trecho

Agora para atender aos requisitos pedidos, retornar um valor de variável cujo valor não foi informado, executamos novamente em JAVA. Para indicar qual variável estamos utilizando foi criado um JLabel com o nome respectivo, logo em frente posicionamos um JTextField para inserir o valor int,float ou double. Após inserir os valores existe o botão 'Enviar', que realiza o cálculo e seta o valor em um JTextField 'Resposta'.

```
private void send5ActionPerformed(java.awt.event.ActionEvent evt) {  
    if(p.getText().isEmpty()){  
        double _V, _n, _R, _T, res;  
        String result;  
        _V = Double.parseDouble(V.getText());  
        _n = Double.parseDouble(n.getText());  
        _R = Double.parseDouble(R.getText());  
        _T = Double.parseDouble(T.getText());  
        res = (_n * _R * _T) / _V;  
        result = Double.toString(res);  
        resposta.setText(result);  
    }  
}
```

Figura 4 – Send

'variable.get()' - método 'getter' para recuperar o valor da variável.

'variable.set()' - método 'setter' para setar/envia um valor para a variável.

'Type.parseType(variable.get())' - conversões de tipos.

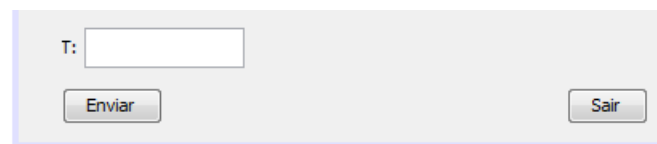
A small Java Swing window with a light gray background. It has a title bar with standard window controls (close, maximize, minimize). The main area contains a label 'T:' followed by a text input field. Below the input field are two buttons: 'Enviar' on the left and 'Sair' on the right.

Figura 5 – Buttom

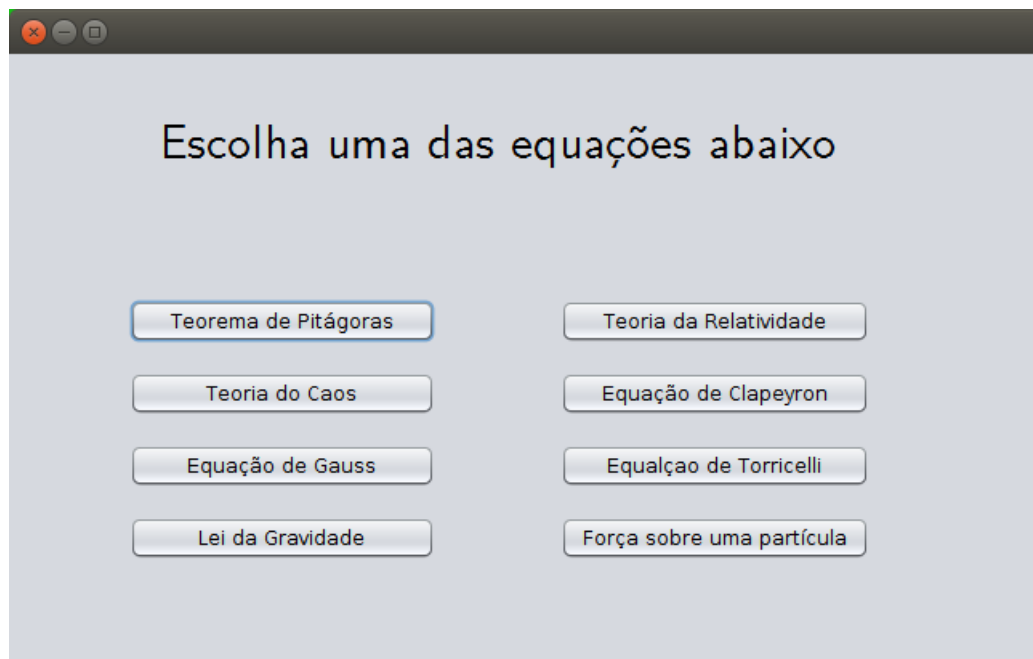


Figura 6 – mainView

3 Dificuldades e operações

A princípio iríamos utilizar Haskell para prova dos teoremas propostos, porém após alguns códigos feitos encontramos dificuldades devido ao retorno da variável que ficaria vazia, ou seja, a variável que procuramos encontrar a resposta, portanto resolvemos utilizar JAVA para o projeto total, assim acarretando no problema de nossos campos tratarem apenas de string, porém a solução foi simples, já que a linguagem fornece métodos práticos para conversões de tipos primitivos.

Como nossas variáveis criadas são privadas, nosso método de operação foi:

```
private void send4ActionPerformed(java.awt.event.ActionEvent evt)
    //chama funcao para calcular (getXOld, getXNew, getk)
    if (xNew.getText().isEmpty())
        float xold, k, res;
        String result;
        xold = Float.parseFloat(xOld.getText());
        k = Float.parseFloat(k.getText());
        res = k * xold * (1 - xold);
        result = Float.toString(res);
        resposta.setText(result);
```

Implementamos no método *send4ActionPerformed* a ação do botão 'Enviar'.

Em seguida verificamos qual `TextField` está vazio.

Criamos variáveis do tipo `float` para receber dados e uma `string` para enviar dados.

Como os `TextField` trabalham com `string`'s temos que pegar o valor do field e transformar para `int/float/double` para realizar as operações.

Em seguida voltamos o valor para o tipo `string` para que o field 'Resposta' possa receber a informação.

Finalmente *resposta.setText(result)* é responsável por setar a informação e deixá-la visível ao usuário.

4 Escolha e desistência de linguagem

Para realizar nosso trabalho optamos por escolher a linguagem JAVA como a principal, contudo a IDE NetBeans open source multiplataforma.

Com esta IDE a criação de uma interface visual (GUI: Graphical User Interface) é facilitada pelo fato do 'arraste' de blocos. Torna-se perfeitamente possível um iniciante na linguagem JAVA operar, construir e configurar uma interface rápida e eficiente para uso.

Com a interface definida, para os cálculos e provas de teoremas iríamos utilizar HASKELL, linguagem matemática criada para prova de teoremas, deste modo teríamos a resolução de forma direta e sem perigo e erros pelo fato da linguagem possuir transparência referencial, onde os dados de saída depende somente dos dados de entrada.

Entretanto encontramos dificuldades em lidar com a linguagem, tanto na implementação quanto na pesquisa em livros e sites relacionados. Os problemas foram surgindo no momento em que, para obtermos o resultado de uma das variáveis disponíveis na equação escolhida era necessário passar os valores das restantes, realizar o cálculo e atribuí-lo a variável escolhida. Deste modo, se a equação tiver 4 variáveis é necessário operar deste modo em cada uma delas separadamente. Além da falta de conhecimento da linguagem a falta de informações foi um obstáculo que acarretou na mudança de linguagem para a resolução das equações.

Deste modo optamos por resolvê-las no próprio código da interface, gerando um evento em um botão criado posteriormente (JButton send). Com isto tivemos de adaptar o código para os conformes JAVA, mas como a parte criada para equações é similar a C, estudado anteriormente com mais profundidade, facilitou a construção do código.