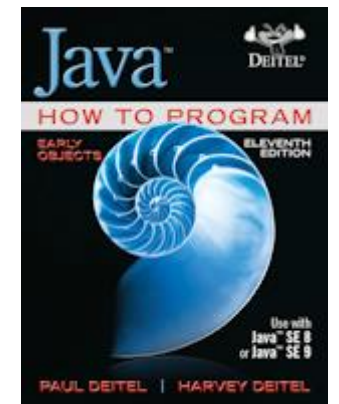


Guia de Laboratório

Lab 04

Prof. André Bernardi

andrebernardi@unifei.edu.br





4º Laboratório ECOP15

27 e 28 de setembro de 2021

- **Utilizar como referência os exemplos:**
 - InitArray.java: Inicializar arrays;
 - SumArray.java: Soma os elementos de um vetor;
 - Exemplos Cap. 7. Java How to Program 10ed;

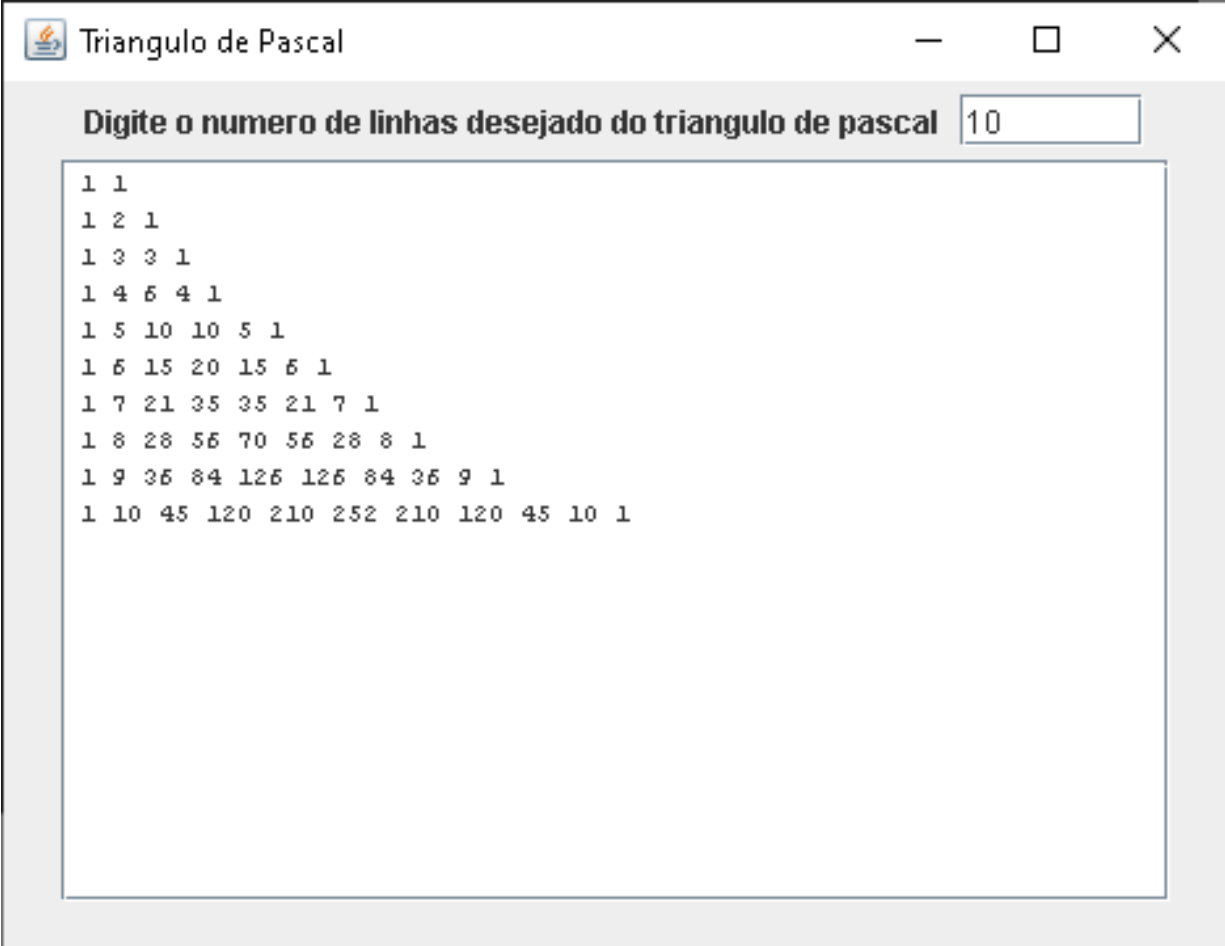


1ª Tarefa:

- Escreva um programa em Java que use um array irregular para representar um Triângulo de Pascal. O triângulo de Pascal é uma série de séries de valores onde cada valor é obtido somando-se o valor acima (linha anterior) e o valor à esquerda do valor acima. Parte do triângulo de Pascal é mostrado abaixo.

1ª Tarefa:

```
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
```



Triangulo de Pascal

Digite o numero de linhas desejado do triangulo de pascal

```
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
```

```

public class PascalFrame extends JFrame
{
    private JTextField texto;
    private JTextArea display;
    public PascalFrame() {
        super("Triangulo de Pascal");
        setLayout ( new FlowLayout() ); // altera o layout da janela
        add(new JLabel("Digite o numero de linhas desejado do triangulo de pascal "));
        texto = new JTextField(6); // aloca caixa de texto para mostrar 6 colunas
        // registrar o evento de ação para texto
        texto.addActionListener( new ActionListener() {
            public void actionPerformed(ActionEvent e){
                montaTrianguloPascal();
            }
        });
        add(texto); //coloca na interface grafica
        display = new JTextArea(20,70);
        display.setFont(new Font("Monospaced", Font.PLAIN, 10));
        add(new JScrollPane(display));
        setSize(500,400);
        setVisible(true);
    }
}

```

```

private void montaTrianguloPascal()
{
    int linha;
    linha = Integer.parseInt(texto.getText());
    display.setText(" "); //Alocar a Matriz triangular
    int mat[][];
    mat = new int [linha][];
    for(int i = 0 ; i< mat.length; i++)
        mat[i] = new int [i+2];
    mat[0][0] = 1;
    mat[0][1] = 1;
    for(int i = 1; i< mat.length; i++){
        mat[i][0] = 1;          mat[i][mat[i].length-1] = 1;
        for(int j = 1; j< mat[i].length-1; j++)
            mat[i][j] = mat[i-1][j] + mat[i-1][j - 1];
    }
    String triangulo=""; // montar a interface de saida.
    for(int i = 0; i< mat.length; i++){
        for(int j = 0; j< mat[i].length; j++)
            triangulo += " " + mat[i][j];
        triangulo += "\n";
    }
    display.setText(triangulo);
}
} // end class

```



Exemplo - PascalTest

```
// Aplicativo de teste que exhibe triangulo de pascal.  
public class PascalTest  
{  
    public static void main(String[] args)  
    {  
        PascalFrame janela = new PascalFrame();  
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    }  
} // fim da classe PascalTest
```



2ª Tarefa:

- Para encontrar uma raiz real de um polinômio $p(x)=a_0+a_1x+\dots+a_nx^n$, ($n > 2$), pode-se aplicar o método de Newton, que consiste em refinar uma aproximação inicial x_0 dessa raiz através da expressão:

$$x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n)}$$

onde:

$n = 0, 1, 2, \dots,$

$p'(x)$ é a primeira derivada de $p(x)$.

- Usualmente, repete-se esse refinamento até que $|x_{n+1} - x_n| < \varepsilon$, $\varepsilon \geq 0$, ou até que m iterações tenham sido executadas.
- Dados um polinômio $p(x) = a_0 + a_1x + \dots + a_nx^n$, uma aproximação inicial x_0 da raiz de $p(x)$, $\varepsilon \geq 0$ e o número máximo de iterações que devem ser executadas, determine uma aproximação da raiz de $p(x)$ pelo método de Newton.
- Utilize uma função para ler um polinômio.
- Utilize uma função que, dado um polinômio $p(x)$, calcula a derivada $p'(x)$
- Para calcular $p(x_n)$ e $p'(x_n)$ em cada iteração, uma função que calcula o valor de um polinômio em um ponto.

Obs:

- represente o polinômio através de um vetor;
- crie um programa em Java que utilize as funções descritas acima.



Exemplo numérico

- Considere o polinômio $p(x) = x^2 - 5x + 6$, que possui duas raízes reais em 2 e 3. Sua derivada é $p'(x) = 2x - 5$.

Para testar o método de Newton podemos supor uma aproximação inicial $x_0 = 500$ e observar sua convergência.

$$x_1 = 500 - p(500)/p'(500) = 500 - 247506/995 = 251.250251256$$

$$x_2 = x_1 - p(x_1) / p'(x_1) = 251.250251256 - 61876.4374999/497.5005025 = 126.87$$

$$x_3 = x_2 - p(x_2) / p'(x_2) = 126.87562814 - 16097.42/248.75 = 62.1626885688$$

...

$$x_n = 3$$

```
import static java.lang.Math.pow;

public class Polinomio {

    public double[] polinome;

    public Polinomio( int n )
    {
        polinome = new double[ n ];
    }

    public Polinomio( double[] n )
    {
        polinome = new double[n.length];
        for(int i = 0; i < polinome.length; i++)
            polinome[i] = n[n.length - 1 - i];
    }

    public Polinomio Derivate()
    {
        Polinomio d = new Polinomio(this.polinome.length - 1);
        for(int i = 0; i < d.polinome.length; i++)
            d.polinome[i] = polinome[i+1] * (i+1);
        return d;
    }
}
```

```

public double valueAt( double x )
{
    double total = 0;

    for(int i = 0; i < polinome.length; i++)
        total += polinome[i] * pow(x,i);

    return total;
}

public double root( double x )
{
    double raiz = 0;
    double pertoDeZero = .0000000000000001;
    for(int i = 0; i < 50; i++)
    {
        raiz = x - this.valueAt(x)/this.Derivate().valueAt(x);
        if(Math.abs(raiz - x) < pertoDeZero)
            break;
        x = raiz;
    }
    return raiz;
}
} // end class

```

```

import java.util.Scanner;
import java.util.StringTokenizer;

public class PolinomioTest {
    public static void main(String[] args) {
        String polinomio;
        Scanner input = new Scanner(System.in);
        System.out.print("Entre com os coeficientes do polinomio separados por "
            + "espaços em ordem decrescente de grau: ");
        polinomio = input.nextLine();
        StringTokenizer strT;
        strT = new StringTokenizer( polinomio );

        double[] coef = new double[ strT.countTokens() ];
        int i = 0;
        while(strT.hasMoreTokens()) {
            coef[i] = Double.valueOf(strT.nextToken());
            i++;
        }
        Polinomio p = new Polinomio( coef );
        System.out.print("\nUma das raízes é : " + p.root(1000));
    }
} // end class

```

```

D:\2021\ECOP15\Lab4\ex2>java PolinomioTest
Entre com os coeficientes do polinomio separados por espaços em ordem decrescente de grau: 1 -5 6

Uma das raízes é : 2.9999999999999996
D:\2021\ECOP15\Lab4\ex2>

```



3ª Tarefa:

Crie um programa em Java que declare e inicialize dois arrays unidimensionais do mesmo tamanho, e calcule o produto escalar deles. Por exemplo, se os dois arrays forem { 9; 2; 6; 7; 0 } e { 1; 4; 5; 9; 2 } o produto escalar será:

$$9 \times 1 + 2 \times 4 + 6 \times 5 + 7 \times 9 + 0 \times 2 = 110.$$

Baseado na implementação do produto escalar descrito nesta questão, crie um método que verifica se dois vetores são ortogonais.

Monte uma interface gráfica (utilizando dois JTextField) com eventos de ação para pegar os valores dos vetores usando o método split da classe String. Representar o produto escalar em um JLabel.

```
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;
```

```
public class ProdutoEscalarFrame extends JFrame {
```

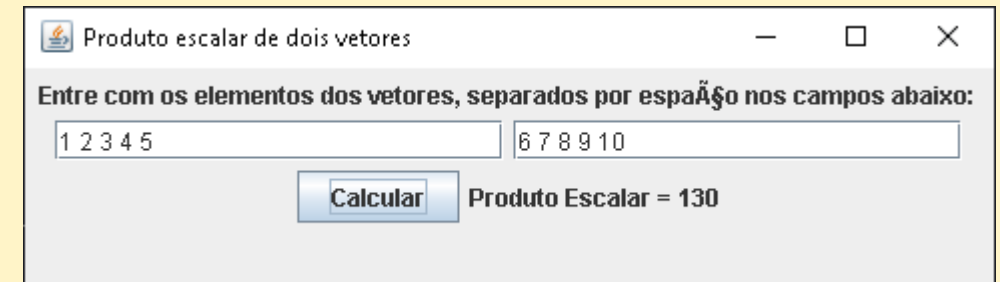
```
    private JTextField textoV1;  
    private JTextField textoV2;  
    private JButton btnCalcular;  
    private JLabel display;
```

```
    public ProdutoEscalarFrame() {  
        super("Produto escalar de dois vetores");
```

```
        setLayout ( new FlowLayout() ); // altera o layout da janela
```

```
        add(new JLabel("Entre com os elementos dos vetores, " +  
            "separados por espaço nos campos abaixo: "));
```

```
        textoV1 = new JTextField(20); // aloca caixa de texto para mostrar 20 colunas  
        add(textoV1);                // coloca na interface grafica  
        textoV2 = new JTextField(20); // aloca caixa de texto para mostrar 20 colunas  
        add(textoV2);                // coloca na interface grafica
```



```

btnCalcular = new JButton("Calcular");
btnCalcular.addActionListener( new ActionListener() {
    public void actionPerformed(ActionEvent e){
        display.setText("Produto Escalar = " +
            produtoEscalar(textoV1.getText(), textoV2.getText()));
    }
});
add (btnCalcular);
display = new JLabel("          ");
add(display);
setSize(500,300);
setVisible(true);
}

```

```

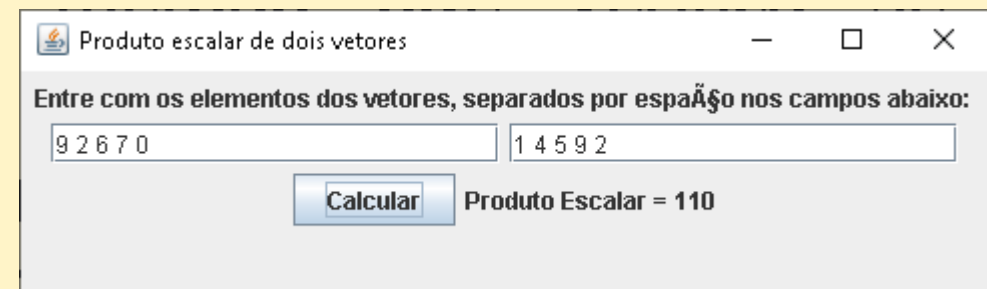
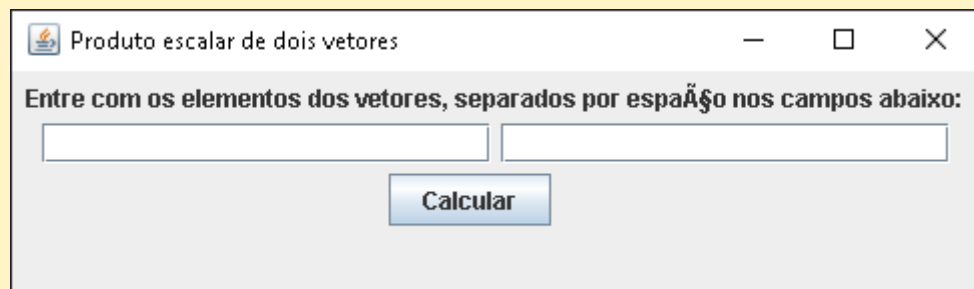
private int produtoEscalar(String v1, String v2) {
    int total = 0;
    String [] vetor1, vetor2;
    vetor1 = v1.split(" ");
    vetor2 = v2.split(" ");
    int tam = Math.min(vetor1.length, vetor2.length);
    for(int i = 0; i < tam; i++)
        total += Integer.parseInt(vetor1[i]) * Integer.parseInt(vetor2[i]);
    return total;
}

```



```
import javax.swing.*;

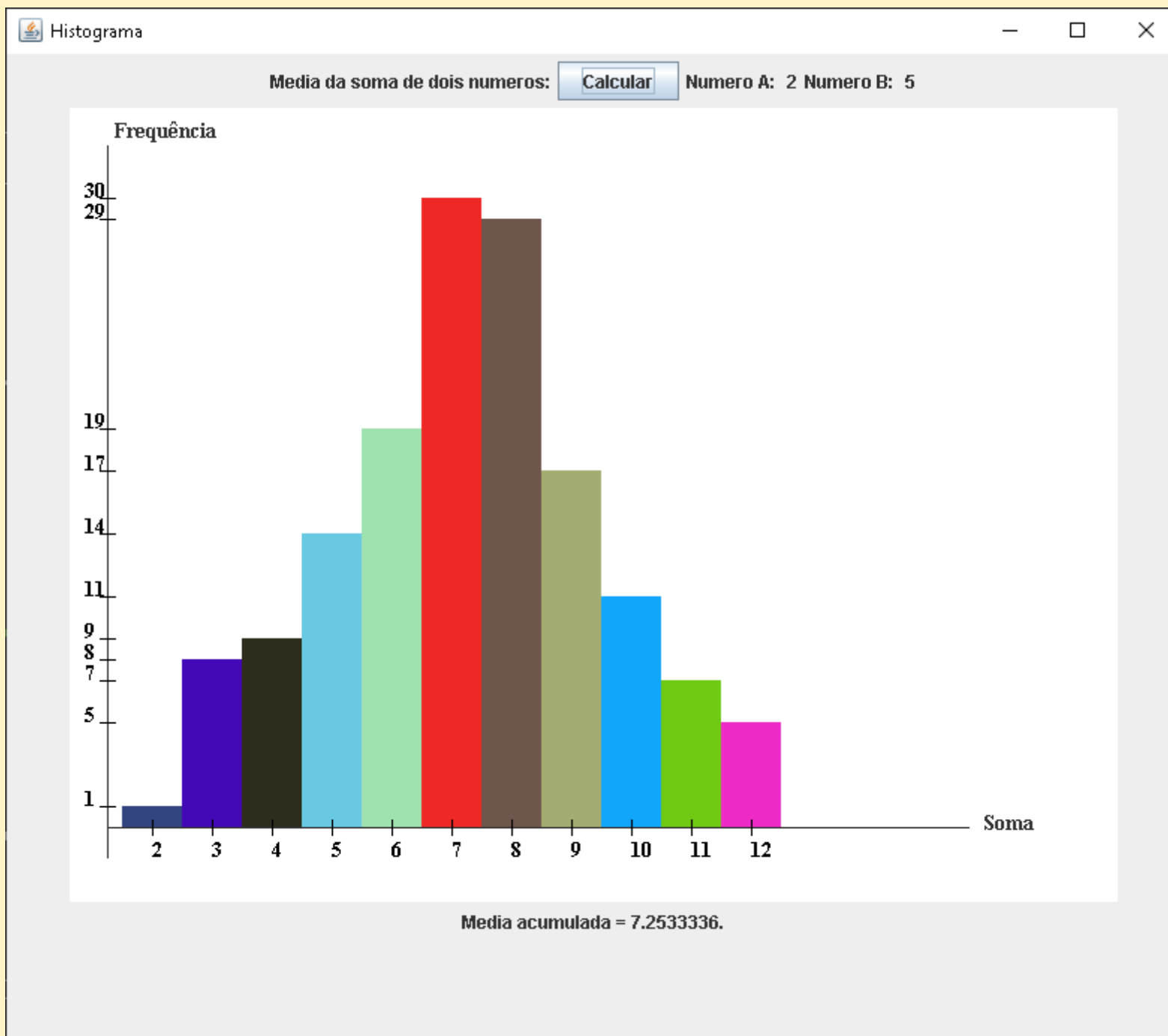
// Aplicativo de teste.
public class ProdutoEscalarTest
{
    public static void main(String[] args)
    {
        ProdutoEscalarFrame janela = new ProdutoEscalarFrame();
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
} // fim da classe ProdutoEscalarTest
```





4ª Tarefa:

Crie um aplicativo Java com interface gráfica que tenha dois campos de texto e um botão. O botão deverá disparar uma função que gere dois números aleatórios entre 1 e 6, colocando cada número em um dos campos. Armazene os valores de cada vez que o usuário apertar o botão em um *ArrayList* e mostre em um label a média dos valores obtidos bem como o desenho de um histograma com a frequência de aparição da soma dos dados.



```

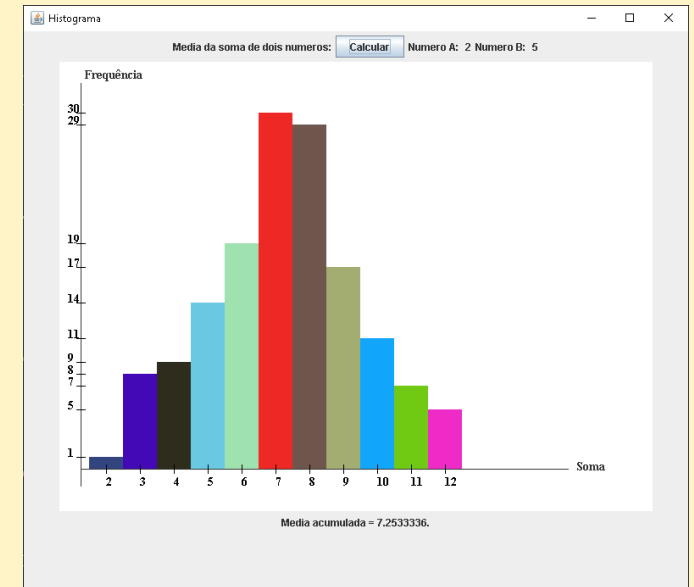
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;

public class Histograma extends JFrame{
    private final JButton btnCalcular;
    private JLabel vetor1;
    private JLabel vetor2;
    private JLabel display;
    private drawHistograma histograma;

    public Histograma() {
        super("Histograma");
        setLayout ( new FlowLayout() );
        add(new JLabel("Media da soma de dois números:\n"));

        Random random = new Random();
        ArrayList<Integer> numbers = new ArrayList<Integer>();
        histograma = new drawHistograma();
        btnCalcular = new JButton("Calcular");
    }
}

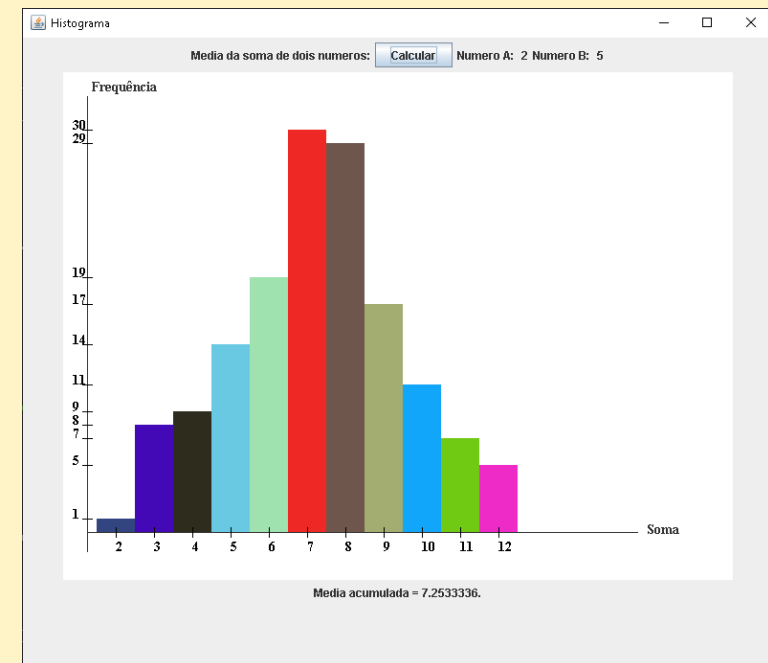
```



```

btnCalcular.addActionListener( new ActionListener() {
    public void actionPerformed(ActionEvent e){
        vetor1.setText(""+(random.nextInt(6) + (1)));
        vetor2.setText(""+(random.nextInt(6) + (1)));
        numbers.add(Integer.parseInt(vetor1.getText()) +
            Integer.parseInt(vetor2.getText()));
        histograma.setDados(numbers);
        display.setText("Media acumulada = " + media(numbers) + ".");
    }
});
add(btnCalcular);
add(new JLabel("Numero A: "));
vetor1 = new JLabel(" ");
add(vetor1);
add(new JLabel("Numero B: "));
vetor2 = new JLabel(" ");
add(vetor2);
add(histograma);
display = new JLabel(" ");
add(display);
setSize(800,700);
setVisible(true);
}

```



```

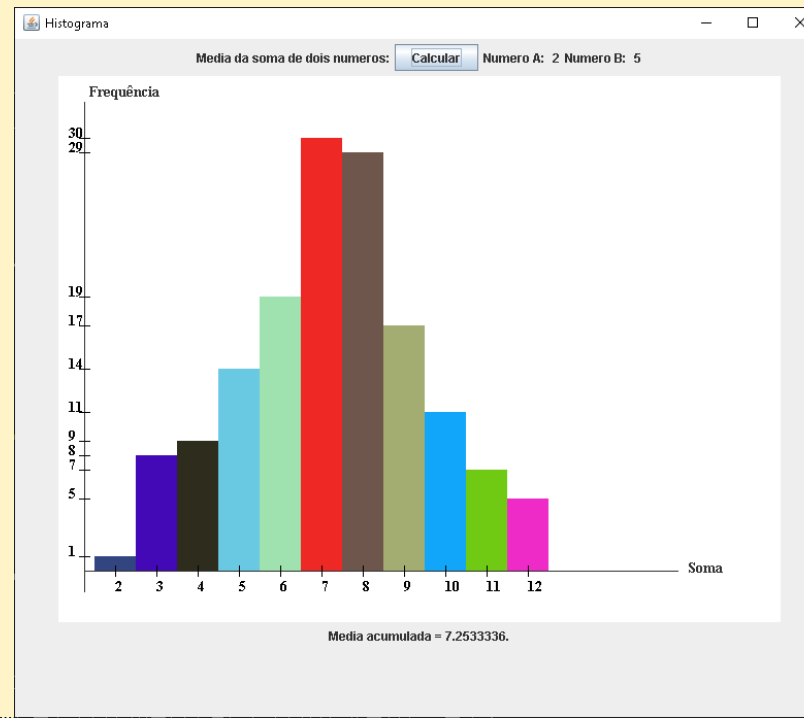
public float media(ArrayList<Integer> a) {
    int m = 0;
    for(int i = 0; i <a.size(); i++)
        m += a.get(i);
    return (float)m/a.size();
}

```

```

public static void main ( String arg[]) {
    Histograma histograma = new Histograma();
    histograma.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

```



```

import java.awt.*;
import java.util.*;
import javax.swing.*;

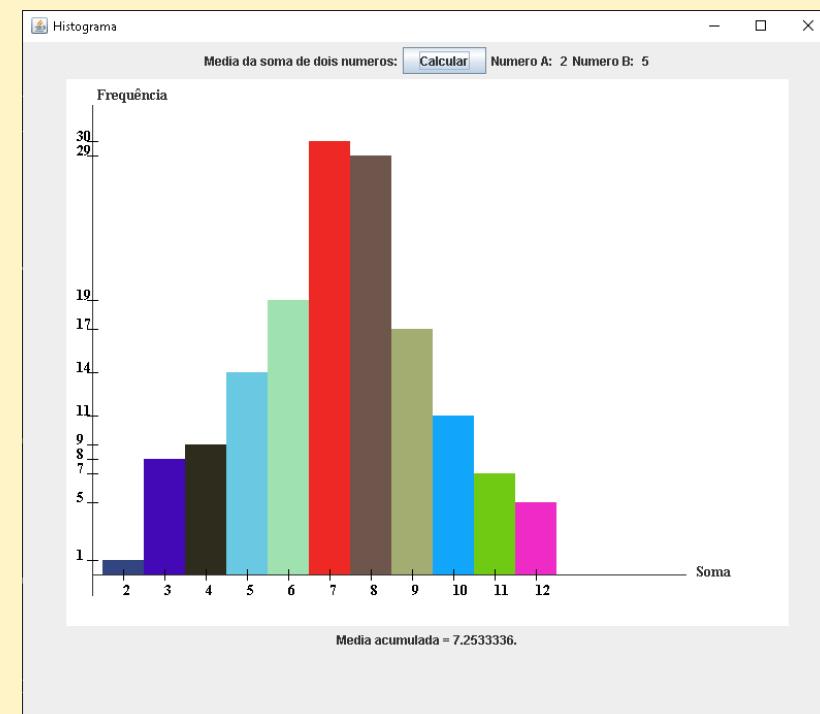
public class drawHistograma extends JPanel {

    private final ArrayList<Integer> dados = new ArrayList<Integer>();
    private final ArrayList<Integer> freq = new ArrayList<Integer>();

    public Dimension getPreferredSize() {
        return new Dimension(700,530);
    }

    public void setDados(ArrayList<Integer> a) {
        //limpar vetores internos
        dados.clear();
        freq.clear();
        //criar vet temporario para manipulações
        int[] copia = new int[a.size()];
        for(int i = 0; i < a.size(); i++){
            copia[i] = a.get(i);
        }
    }

```



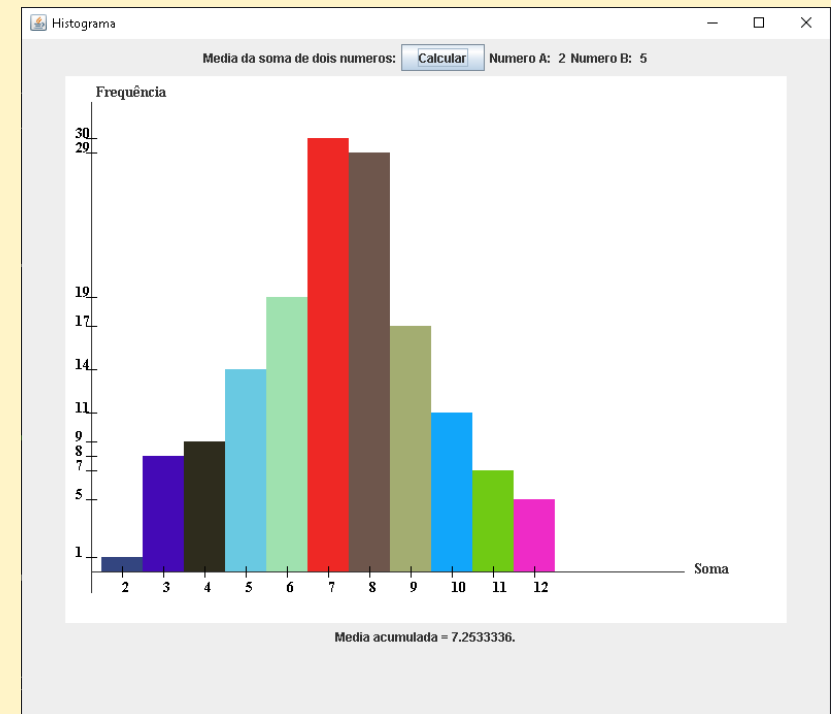
```

//ordenar dados
Arrays.sort(copia);
//encontrar frequencia e preencher dados
int cont;
for(int i = 0; i < a.size(); i++){
    cont = 1;
    for(int j = a.size() - 1; j >= 0; j--){
        if(copia[i] == copia[j] && i != j)
            cont++;
        if(!dados.contains(copia[i])){
            dados.add(copia[i]);
            freq.add(cont);
        }
    }
    repaint();
}

public void paintComponent(Graphics g){
    super.paintComponent(g);
    this.setBackground(Color.WHITE);

    g.setFont(new Font("Serif", Font.BOLD, 14));
    Random random = new Random();
    //Desenha eixos
    g.drawLine(25, 25, 25, 500);

```




```

g.drawString("Frequência", 30, 20);
g.drawLine(25, 480, 600, 480);
g.drawString("Soma", 610, 482);

//encontrar maior valor para fazer a escala.
int aux = 0;
for(int i = 0; i < freq.size(); i++)
    if(freq.get(i) > aux)
        aux = freq.get(i);
aux++;

for(int i = 0; i < dados.size(); i++){
    g.setColor(new Color(random.nextInt(256), random.nextInt(256),
        random.nextInt(256)));
    g.fillRect(35 + 40*i, 480 - (455/aux)*freq.get(i),
        40, (455/aux)*freq.get(i));
    g.setColor(Color.BLACK);
    g.drawLine(20, (480 - 455/aux*freq.get(i)), 30,
        (480 - 455/aux*freq.get(i)));
    g.drawLine(55 + 40*i, 475, 55 + 40*i, 485);
    g.drawString("" + dados.get(i), 55+40*i, 500);
    g.drawString("" + freq.get(i), 10, (480 - 455/aux*freq.get(i)));
}
}
}

```

