Graphical User Interface (GUI)
Interface Gráfica do Usuário

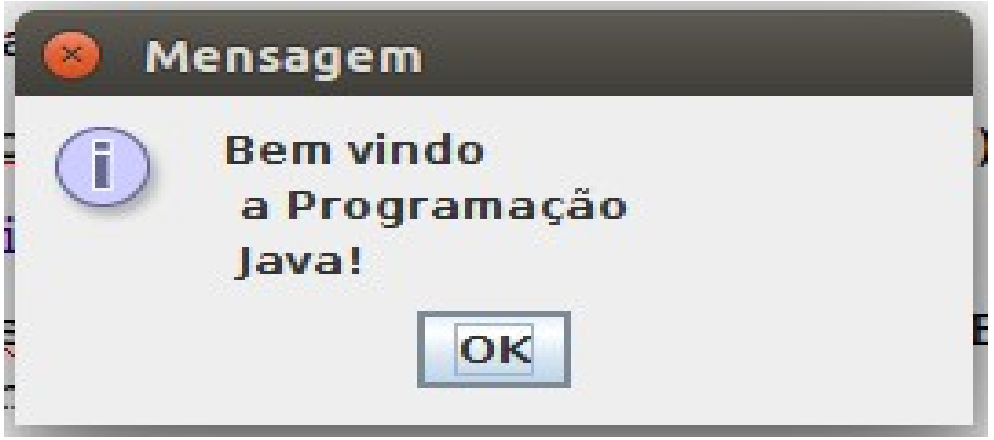
Disciplina de Lógica de Programação
Curso Técnico em Informática para Internet

Exibindo Texto em uma Caixa de Diálogo

- A classe **JOptionPane** fornece caixas de diálogo pré-empacotadas que permitem aos programas exibir janelas que contém mensagens para o usuário, essas janelas são chamadas de **Diálogos de Mensagem**.

Exemplo 1: imprimindo múltiplas linhas na caixa de diálogo

```
import javax.swing.JOptionPane; /* importa a classe
                                   JoptionPane */

public class Dialogo {

   public static void main(String[] args) {

   //exibe um diálogo com a mensagem

   JOptionPane.showMessageDialog(null, "Bem vindo\n a
   Programação\n Java!");

   } //fim de main

} //fim da classe Dialogo
```



Classe JOptionPane

- A classe JOptionPane é importada pela linha:

  javax.swing.JOptionPane;
- JOptionPane é a classe que foi utilizada.
- javax.swing é o pacote onde se encontra a classe JOptionPane.
- O pacote javax.swing contém muitas classes que ajudam os programadores a criarem interfaces gráficas com o usuário.

Método showMessageDialog

- Este método precisa de dois argumentos:
  - O primeiro argumento indica ao aplicativo Java onde posicionar a janela.
  - Quando o argumento é “null”, a caixa de diálogo aparece no centro da tela.
  - O segundo argumento é a String a ser apresentada na caixa de diálogo.
- O método showMessageDialog é um método static. Métodos static são chamado no seguinte formato:

  NomeDaClasse.NomeDoMétodo(argumentos)

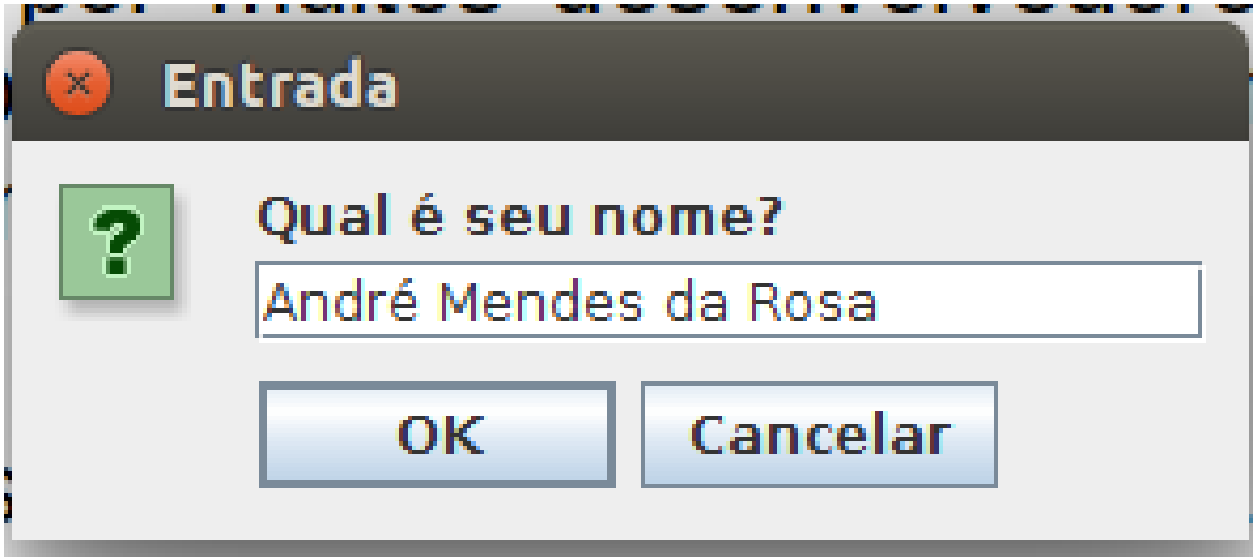
Inserindo texto em uma caixa de diálogo

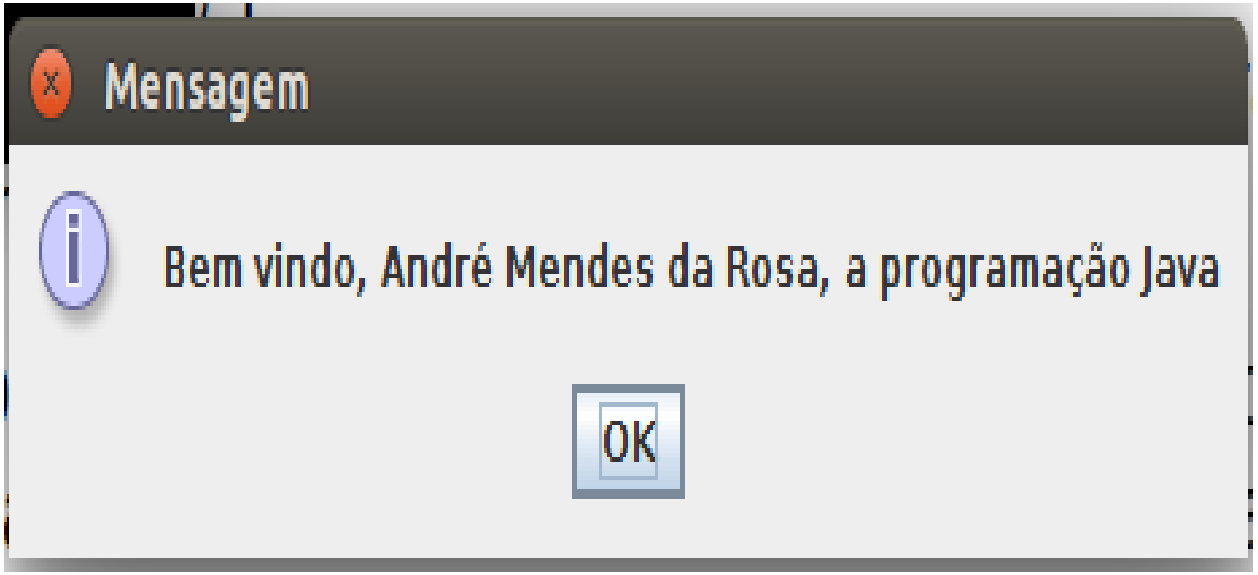
- Outra caixa de diálogo pré-definida da classe JOptionPane é chamada de **diálogo de entrada** que permite ao usuário inserir dados para utilização no programa.
- O exemplo 2 a seguir, solicita o nome do usuário e responde com um cumprimento contendo o nome inserido pelo usuário.

Exemplo 2

```
import javax.swing.JOptionPane;

public class NomeDialogo {

    public static void main(String[] args) {

    //pede ao usuário para inserir seu nome

    String nome = JOptionPane.showInputDialog("Qual é seu nome?");

    //cria a mensagem

    String mensagem = String.format("Bem vindo, %s, a programação Java",
    nome);

    //exibe a mensagem para cumprimentar o usuário pelo nome

    JOptionPane.showMessageDialog(null, mensagem);

    } //fim de main

} //fim da classe NomeDialogo
```

Caixas de Diálogo do Exemplo 2





Método showInputDialog

- Este método da classe JOptionPane exibe um diálogo de entrada simples que contém um prompt e um campo para o usuário inserir texto, campo esse conhecido como **campo de texto.**
- O argumento para o método showInputDialog é o prompt que indica o nome que o usuário deve inserir.
- Clicando em OK ou apertando ENTER, o usuário envia a String para o programa que armazena em uma variável.
- Se o usuário cancelar, o programa receberá **null,** que será mostrado na tela como o nome.

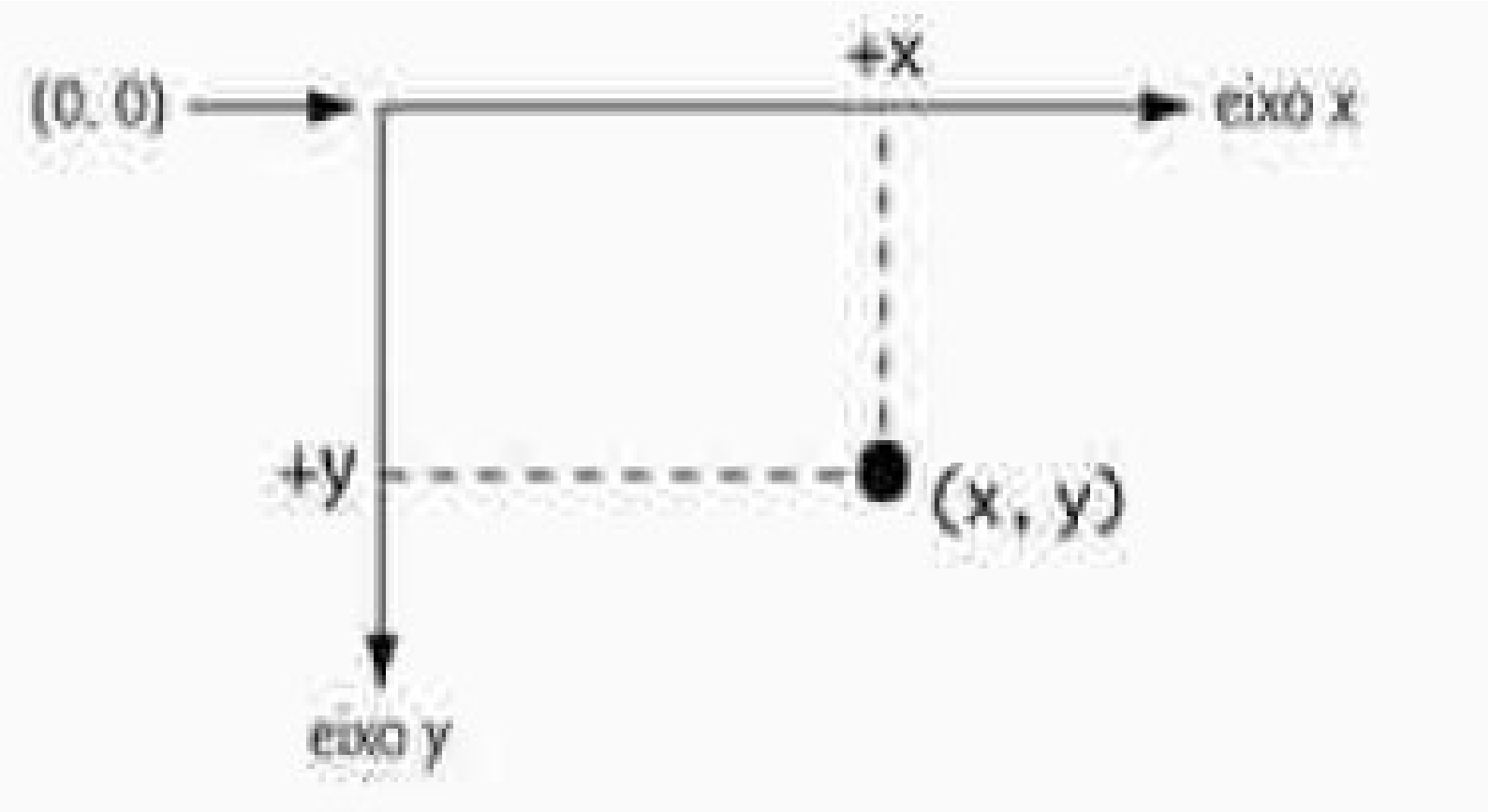
Método static String format

- Retorna uma String que contém a saudação com o nome inserido pelo usuário.
- O método **format** é semelhante ao método **System.out.printf**, exceto pelo fato que format retorna uma String formatada em vez de exibi-la na tela.

Criando desenhos simples

- Primeiramente, é necessário entender o sistema de coordenadas do próximo slide.
- O sistema de coordenadas é esquema para identificar um ponto na tela.
- Por padrão, o canto superior esquerdo de um componente GUI tem as coordenadas (0, 0).
- Um para de coordenadas é composto de uma coordenada x (a coordenada horizontal) e uma coordenada y (a coordenada vertical).
- A coordenada x é a localização horizontal que se estende da esquerda para a direita.
- A coordenada y é a localização vertical que se estende de cima para baixo.
- O eixo x descreve cada coordenada horizontal.
- O eixo y descreve cada coordenada vertical.
- As coordenadas são utilizadas para indicar onde as imagens gráficas devem ser exibidas na tela.

Sistema de coordenadas Java



Sistema de coordenadas Java

- Unidades coordenadas são medidas em pixels.
- Um pixel (picture element – elemento de imagem) é a menor unidade de resolução do monitor.
- No exemplo 3, o aplicativo desenha duas linhas a partir dos cantos.

Exemplo 3

```java
//desenha duas linhas que se cruzam em um painel
import java.awt.Graphics;
import javax.swing.JPanel;

public class DrawPanel extends JPanel {
        //desenha um X a partir dos cantos do painel
        public void paintComponent( Graphics g ) {
                /*chama paintComponent para assegurar
                que o painel é exibido corretamente*/
                super.paintComponent( g );
                int largura = getWidth(); //largura total
                int altura = getHeight(); //altura total
                /*desenha uma linha a partir do canto superior esquerdo
                até o canto inferior direito*/
                g.drawLine(0, 0, largura, altura);
                /*desenha uma linha a partir do canto inferior esquerdo
                até o canto superior direito*/
                g.drawLine(0, altura, largura, 0);
        } //fim do método paintComponent
} //fim da classe DrawPanel
```

Exemplo 3 – Classe DrawPanel

- A classe DrawPanel realiza o desenho real.
- As instruções import permitem utilizar as classes:
  - Graphics (do pacote java.awt) que fornece métodos para desenhar texto e formas na tela.
  - JPanel (do pacote javax.swing) que fornece uma área em que podemos desenhar.
- Na linha 5, a palavra-chave **extends** indica que a classe DrawPanel é um tipo aprimorado de Jpanel.
- A classe drawPanel reutiliza (herda) os dados e métodos da classe JPanel.
- JPanel é a superclasse e DrawPanel é a subclasse.
- O método paintComponent é chamado automaticamente toda vez que é necessário exibir o JPanel.

Exemplo 3 – Classe DrawPanel

- O método paintComponent requer um argumento, um objeto de Graphics, que é oferecido pelo sistema quando ele chama paintComponent.
- A instrução **super.paintComponent(g);** assegura que o painel seja renderizado na tela antes de desenharmos nele.
- Os métodos **getWidth();** e **getHeight();** retornam a largura e a altura do JPanel, respectivamente.
- No método **drawLine**, os primeiros dois argumentos são as coordenadas x e y para umas das extremidades da linha e os outros dois argumentos são as coordenadas da outra extremidade da linha.

Exemplo 3 – Classe DrawPanelTeste

```
1 //aplicativo para exibir um DrawPanel
2 import javax.swing.JFrame;
3 public class DrawPanelTeste {
4         public static void main(String[] args) {
5                 //cria um painel que contém nosso desenho
6                 DrawPanel panel = new DrawPanel();
7                 //cria um novo frame para armazenar o painel
8                 JFrame aplicativo = new JFrame();
9                 /*configura o frame para ser encerrado quando
10                é fechado*/
11                aplicativo.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
12                aplicativo.add( panel ); //adiciona o painel ao frame
13                aplicativo.setSize(250, 250); //configura o tamanho do frame
14                aplicativo.setVisible( true ); //torna o frame visível
15        } //fim de main
16 } //fim da classe DrawPanelTeste
```

Exemplo 3 – Classe DrawPanelTeste

- Na classe **DrawPanelTeste**, é importada a classe **JFrame** (do pacote javax.swing) que cria a janela para exibir o **DrawPanel**.
- A linha 6, cria uma instância da classe DrawPanel que contém nosso desenho.
- A linha 8, cria um novo JFrame que pode armazenar e exibir nosso painel.
- A linha 11, indica que o aplicativo deve terminar quando o usuário fechar a janela.
- A linha 12, o método **add** anexa o DrawPanel.
- A linha 13, o método **setSize** configura o tamanho do Jframe.
-

Desenhando retângulos e círculos

- São utilizados os métodos **drawRect** e **drawOval** da classe **Graphics**.
- O exemplo 4 a seguir, desenha retângulos ou círculos conforme a escolha do usuário.

Exemplo 4

```
1 //demonstra o desenho de diferentes formas
2 import java.awt.Graphics;
3 import javax.swing.JPanel;
4 public class Formas extends JPanel {
5         int escolha; //escolha da forma a desenhar
6         //construtor configura a escolha do usuário
7         public Formas ( int escolhaUsuario ) {
8                 escolha = escolhaUsuario;
9         } //fim do construtor
10        /*desenha uma cascata de formas que
11        iniciam do canto superior esquerdo*/
12        public void paintComponent( Graphics g ) {
13                super.paintComponent( g );
14                for(int i = 0; i < 10; i++) {
15                //seleciona a forma com base na escolha do usuário
16                        switch(escolha) {
17                                case 1: g.drawRect(10 + i * 10, 10 + i * 10,
18                                                50 + i * 10, 50 + i * 10);
19                                        break;
20                                case 2: g.drawOval(10 + i * 10, 10 + i * 10,
21                                                50 + i * 10, 50 + i * 10);
22                                        break;
23                        } //fim do switch
24                } //fim do for
25        } //fim do método paintComponent
26 } //fim da classe Formas
```

Exemplo 4 - explicação

- A classe **Formas** estende a classe **JPanel**.
- Formas tem uma variável de instância, **escolha**, que determina se **paintComponent** deve desenhar retângulos ou círculos.
- Nas linhas 7 a 9, está o **construtor** da classe Formas, o qual inicializa a variável **escolha** com o valor passado no parâmetro **escolhaUsuario**.
- O método paintComponent realiza o desenho real.
- A instrução **super.paintComponent(g);** chama o método **paintComponent** da superclasse.
- A instrução **for** faz o loop 10 vezes para desenhar dez formas.

Exemplo 4 - explicação

- A instrução **switch** escolhe entre desenhar retângulos ou círculos.
- O método **drawRect** (da classe Graphics) recebe quatro argumentos. Os dois primeiros argumentos representam as coordenadas x e y. Os dois próximos argumentos representam a largura e a altura do retângulo.
    - Iniciou-se em uma posição de 10 pixels para baixo e 10 pixels para à direita do canto superior esquerdo, e cada iteração do loop move o canto superior esquerdo outros 10 pixels para baixo e 10 pixels para à direita. A largura e a altura iniciam em 50 pixels e aumentam 10 pixels a cada iteração.
- O método **drawOval** (da classe Graphics) requer os mesmo quatro argumentos de drawRect. Os argumentos especificam a posição e o tamanho do retângulo para o círculo, pois o retângulo é delimitador do círculo a ser criado.

Exemplo 4 - FormasTeste

```
1 //aplicativo de teste que exibi a classe Formas
2 import javax.swing.JFrame;
3 import javax.swing.JOptionPane;
4 public class FormasTeste {
5         public static void main(String[] args) {
6                 //obtém a escolha do usuário
7                 String entrada = JOptionPane.showInputDialog(
8                         "Digite 1 para desenhar retângulos\n"
9                         + "Digite 2 para desenhar círculos");
10                //converte a entrada em inteiro
11                int escolha = Integer.parseInt( entrada );
12                //cria o painel com a entrada do usuário
13                Formas painel = new Formas( escolha );
14                //cria um novo JFrame
15                JFrame aplicativo = new JFrame();
16                aplicativo.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
17                aplicativo.add( painel ); //adiciona o painel ao frame
18                aplicativo.setSize( 300, 300 ); //Configura o tamanho desejado
19                aplicativo.setVisible( true ); //mostra o frame
20        } //fim de main
21 } //fim da classe FormasTeste
```

Exemplo 4 - FormasTeste

- A classe **FormasTeste** é responsável por tratar a entrada do usuário e criar uma janela para exibir o desenho escolhido pelo usuário.
- **JFrame** é importado para tratar a exibição.
- **JOptionPane** é importado para tratar a entrada.
- Na linha 13, uma instância da classe Formas é criada com a escolha do usuário passada para o construtor.

Cores e formas preenchidas

- Adicionar cores traz outra dimensão aos desenhos que o usuário faz na tela do computador.
- Formas preenchidas preenchem regiões inteiras com cores sólidas em vez de apenas exibir os contornos dos desenhos.
- As cores são definidas pelos seus componentes vermelho, verde e azul, denominados valores **RGB** que contém valores inteiros de 0 a 255.
- Quanto mais alto o valor, mais brilhante uma sombra particular ficará na cor final.
- O Java utiliza a classe **Color** do pacote **java.awt** para representar cores utilizando valores RGB.

Cores e formas preenchidas

- O objeto **Color** contém 13 objetos **static Color** predefinidos:
  - Color.BLACK
  - Color.BLUE
  - Color.CYAN
  - Color.DARK_GRAY
  - Color.GRAY
  - Color.GREEN
  - Color.LIGHT_GRAY
  - Color.MAGENTA
  - Color.ORANGE
  - Color.PINK
  - Color.RED
  - Color.WHITE
  - Color.YELLOW

Cores e formas preenchidas

- A classe Color também possui um construtor:

  public Color (int r, int g, int b)

- Assim, é possível criar cores personalizadas especificando os valores de r, g e b individuais de uma cor.
- Retângulos e círculos são preenchidos utilizando os métodos **fillRect** e **fillOval**. Utilizam os mesmos parâmetros de drawRect e drawOval.
- O exemplo 5, desenha uma cara sorridente na tela do computador.

Exemplo 5 – Classe Sorriso

```
 1 //demonstra formas preenchidas
 2 import java.awt.Color;
 3 import java.awt.Graphics;
 4 import javax.swing.JPanel;
 5 public class Sorriso extends JPanel {
 6          public void paintComponent( Graphics g ) {
 7                    super.paintComponent( g );
 8                    //desenha o rosto
 9                    g.setColor( Color.YELLOW );
10                    g.fillOval( 10, 10, 200, 200 );
11                    //desenha os olhos
12                    g.setColor( Color.BLACK );
13                    g.fillOval( 55, 65, 30, 30 );
14                    g.fillOval ( 135, 65, 30, 30 );
15                    //desenha a boca
16                    g.fillOval( 50, 110, 120, 60 );
17                    //"retoca" a boca para criar um sorriso
18                    g.setColor( Color.YELLOW );
19                    g.fillRect( 50, 110, 120, 30 );
20                    g.fillOval( 50, 120, 120, 40 );
21          } //fim do método paintComponent
22 } //fim da classe Sorriso
```

Exemplo 5 – Classe SorrisoTeste

```
1  //aplicativo de teste que exibe um rosto sorridente
2  import javax.swing.JFrame;
3  public class SorrisoTeste {
4          public static void main(String[] args) {
5                  Sorriso painel = new Sorriso();
6                  JFrame aplicativo = new JFrame();
7                  aplicativo.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
8                  aplicativo.add( painel );
9                  aplicativo.setSize( 230, 250 );
10                 aplicativo.setVisible( true );
11         } //fim de main
12 } //fim da classe SorrisoTeste
```