

Programação Orientada a Objetos

ITE - 003

profº Mauricio Conceição Mario

Aplicações Gráficas

```
// programa Desenho – Aplicações Gráficas
```

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

```
public class Desenho extends JFrame{
```

```
    public Desenho() {
```

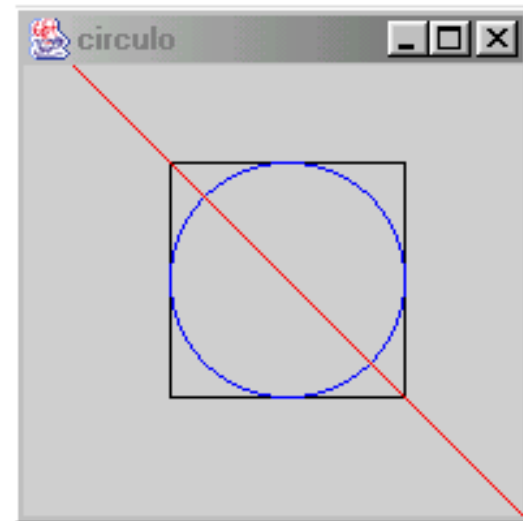
```
        super ("circulo");  
        setSize(200,200);  
        setVisible(true);  
    }
```

```
    public void paint (Graphics p)
```

```
    {  
        super.paint(p);  
        p.setColor(Color.black);  
        p.drawRect(60,60,90,90);  
        p.setColor(Color.blue);  
        p.drawArc(60,60,90,90,0,360);  
        p.setColor(Color.red);  
        p.drawLine(0,0,200,200);  
    }
```

```
    public static void main (String args[])
```

```
    {  
        Desenho implementa = new Desenho();  
        implementa.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    }  
}
```



Aplicações Gráficas

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

public class Figuras1 extends JFrame {

    public Figuras1()
    {
        super("Figuras");
        setSize(300, 300);
        setVisible(true);
    }

    public void paint (Graphics g)
    {
        super.paint(g);

        g.setColor(Color.white);
        g.drawArc(100, 100, 60, 60, 0, 180);

        g.setColor(Color.white);
        g.drawArc(160, 100, 60, 60, 180, 180);

        g.setColor(Color.green);
        int xpontos[] = {100, 220 };
        int ypontos[] = {130, 130 };
        g.drawPolyline(xpontos, ypontos, 2);
    }
}
```

Aplicações Gráficas

```
g.setFont(new Font("Monospaced", Font.ITALIC, 14));  
g.drawString("Aplicações 2D", 40, 240);
```

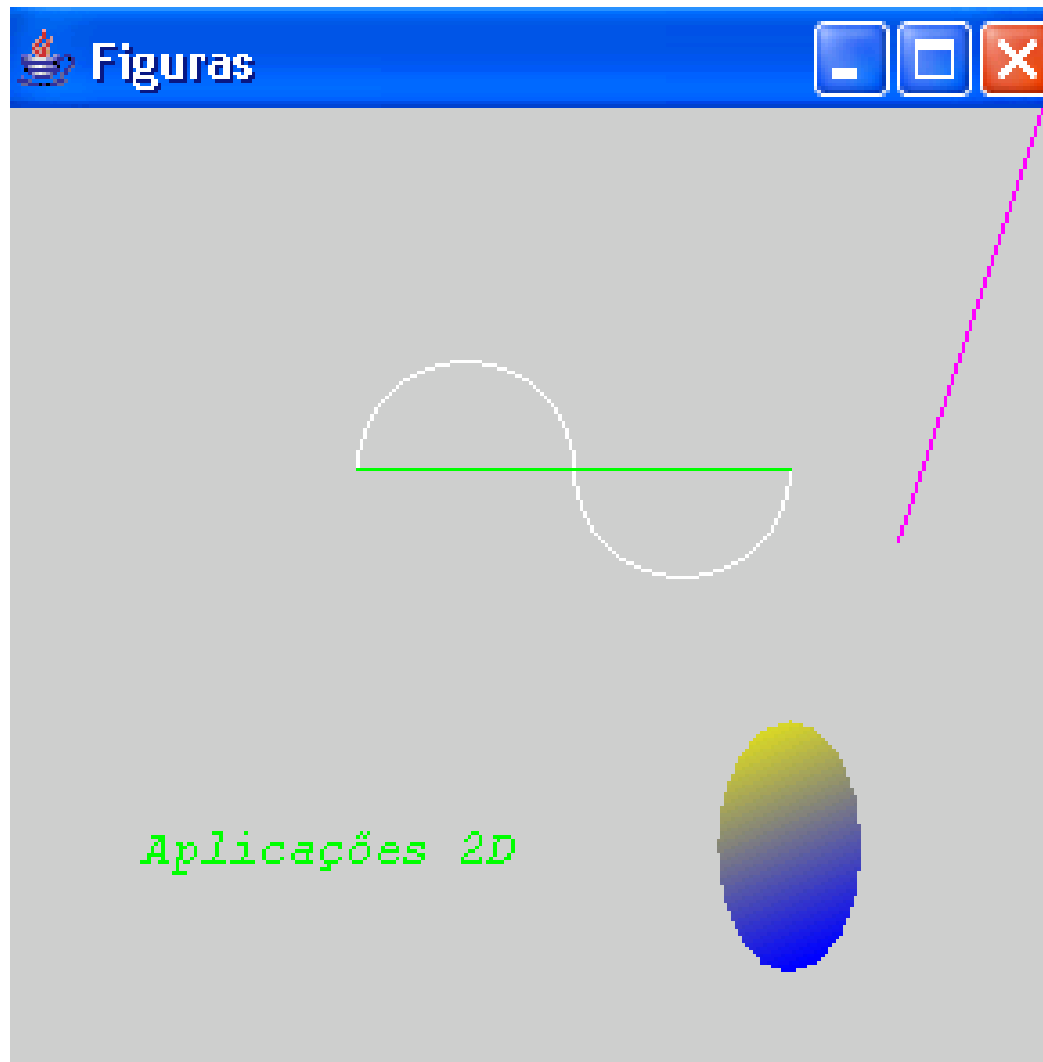
```
//conversão de g para 2D  
Graphics2D p = (Graphics2D) g;
```

```
p.setPaint(new GradientPaint (5, 30, Color.blue, 35, 100, Color.yellow, true));  
p.fill (new Ellipse2D.Double(200, 200, 40, 70));
```

```
p.setPaint(Color.magenta);  
p.draw(new Line2D.Double(300, 0, 250, 150));    }
```

```
public static void main( String args[] )  
{    Figuras1 aplicacao = new Figuras1();  
    aplicacao.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}  
}
```

Aplicações Gráficas



Aplicações Gráficas – painel de cores

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

public class Painelcores extends JFrame{

    private JButton changeColorButton;
    private Color cor = Color.lightGray;
    private Container janela;

    //configuração da interface com usuário GUI
    public Painelcores() {

        super ("usando JColorChooser");
        janela = getContentPane();
        janela.setLayout (new FlowLayout());
        changeColorButton = new JButton ("muda cor");

        //configura changeColorButton e registra o tratador de eventos
        changeColorButton.addActionListener (

            //classe interna anônima
            new ActionListener() {
```

Aplicações Gráficas – painel de cores

//exibe JColorChooser quando clica botão

```
public void actionPerformed(ActionEvent evento)
```

```
{
```

```
    cor = JColorChooser.showDialog
```

```
    (Painelcores.this,"escolha a cor",cor);
```

```
    if (cor == null)
```

```
        cor = Color.lightGray;
```

//muda a cor de fundo do painel

```
janela.setBackground(cor);
```

```
}
```

```
} //fim da classe interna anônima
```

);// fim da chamada para addActionListener

```
janela.add(changeColorButton);
```

```
setSize(400,130);
```

```
setVisible(true);
```

```
}
```

```
public static void main (String args[])
```

```
{
```

```
    Painelcores implementa = new Painelcores();
```

```
    implementa.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
}
```

```
}
```

Importação de imagem

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*; // carregamento de um ícone/imagem

public class Imagem1 extends JFrame {
    JLabel L1;
    ImageIcon figura = new ImageIcon ("C:/des1.gif");//declaração de objeto da classe ImageIcon,
    com endereço da imagem.gif

    Imagem1()
    {
        setTitle(" ");
        setSize(300,300);
        setLocation(50,50);
        L1 = new JLabel(figura);
        getContentPane().setBackground(new Color(255,255,255));
        getContentPane().setLayout(new GridLayout());
        getContentPane().add(L1);
    }

    public static void main( String args[] )
    {

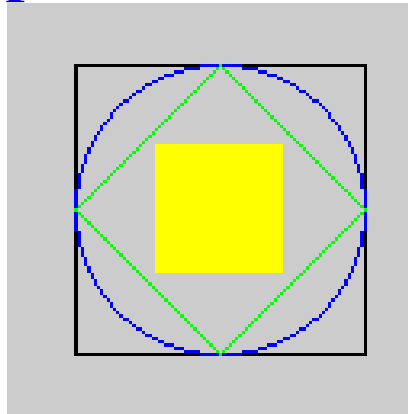
        JFrame Janela = new Imagem1();
        Janela.show();
        WindowListener y = new WindowAdapter()
        {

            public void windowClosing (WindowEvent e)
            {
                System.exit(0);
            }
        };
        Janela.addWindowListener(y);
    }
}
```


Exercícios:

27. Compilar e executar as classes Desenho e Figuras1. Relacionar os objetos gráficos com os respectivos métodos nas classes.

28. Criar aplicação para construir a figura abaixo:



29. Compilar e executar a classe Painelcores. Descrever as funções do método void main(), do método construtor e da classe interna anônima.

30. Compilar e executar a classe Imagem1. Adaptá-la para ler uma imagem diferente da especificada no código.

Uso de applets

- `import java.awt.Graphics;` instrução `import` diz ao compilador para carregar a classe `Graphics` do pacote `java.awt`. A classe `Graphics` permite a um applet Java desenhar gráficos como linhas, retângulos, elipses e strings de caracteres e desenhos.
- `import javax.swing.JApplet;` instrução `import` diz ao compilador para carregar a classe `JApplet` do pacote `javax.swing`. Quando se cria um applet em java, normalmente se importa a classe `JApplet`.
- Cada applet Java criado possui pelo menos uma definição de classe; ao se criar uma definição de classe, normalmente se utilizam fragmentos de uma definição de classe já existente: *herança*.
`public class Applet1 extends JApplet {`
inicia uma definição de `class` para a classe `Applet1`; `class` introduz a definição de classe. `Applet1` é o nome da classe. A palavra-chave `extends` indica que a classe `Applet1` herda pedaços existentes de uma outra classe, no caso `JApplet`, que aparece à direita. Nesse relacionamento de herança, `JApplet` é chamada de *superclasse* ou *classe básica* e `Applet1` é chamada de *subclasse* ou *classe derivada*. Utilizar a herança resulta em uma definição de classe `Applet1` que tem *atributos* (dados) e *comportamentos* (métodos) da classe `JApplet` (capacidade de imprimir a frase no applet).

Criação de applets

1. Criar arquivo e salvar como .html:

Applet1.html

```
<html>  
<applet code = "Applet1.class" width = 300 height = "45">  
</applet>  
</html>
```

Criação de applets

2. Criar arquivo e salvar como Applet1.java

Applet1.java

```
//programa utilizando applet

import java.awt.Graphics; //importa a classe Graphics

//pacotes de extensão de Java

import javax.swing.JApplet; //importa a classe JApplet

public class Applet1 extends JApplet {

    //desenha texto sobre o fundo do applet

    public void paint ( Graphics g )
    {
        //chama versão herdada do método paint
        super.paint( g );

        //desenha string nas coordenadas x=25 e y=25
        g.drawString( "aplicativo Applet!", 25, 25 );

    }

} //fim da classe Applet1
```

Criação de applets

3. Comandos para execução e resultados

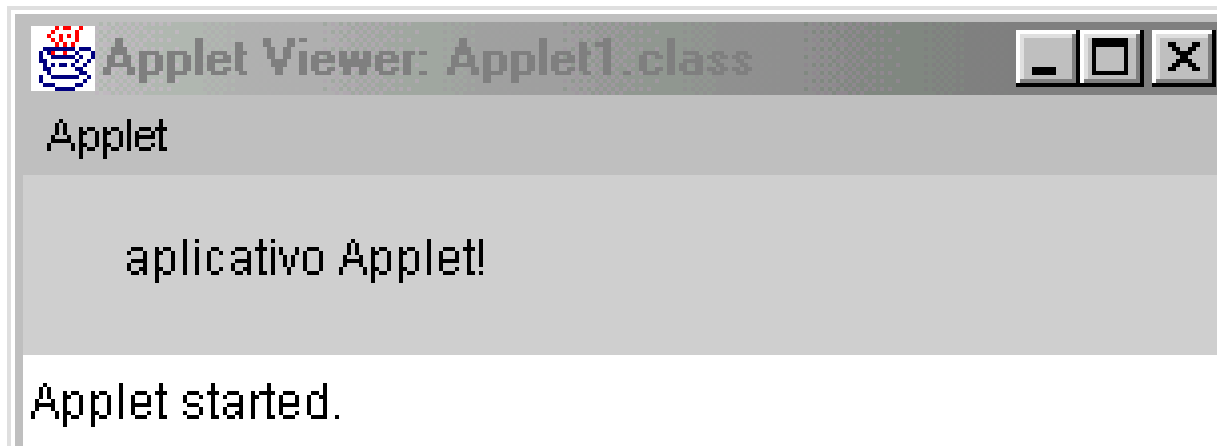
Comandos:

```
C:\Meus documentos>cd\jdk1.3.1\bin
```

```
C:\jdk1.3.1\bin>javac Applet1.java
```

```
C:\jdk1.3.1\bin>appletviewer Applet1.html
```

Resultado:



Utilizando Métodos e Applets

Criação de applets

//programa utilizando applet

```
import java.awt.Container;
```

//pacotes de extensão de Java

```
import javax.swing.*;
```

```
public class Applet2 extends JApplet { //contêiner de applets chama método init() do applet
```

```
    public void init ( )
```

```
    {
```

```
        JTextArea saidadados = new JTextArea(); //saidadados declarada como objeto tipo JTextArea
```

```
        Container janela = getContentPane(); //área de exibição do applet conterá painel (objeto da classe Container do pacote java.awt)
```

```
        janela.add( saidadados ); //referência a janela, da classe Container, e atribuição a ela o resultado da chamada ao método getContentPane, herança da classe Japplet.
```

```
        int resultado;
```

```
        String saida = "";
```

```
        for (int x = 1; x <= 10; x++)
```

```
        {
```

```
            resultado = quadrado (x);
```

```
            saida += "o quadrado de " + x + " é " + resultado + "\n";
```

```
        }
```

```
        saidadados.setText(saida);
```

```
    }
```

```
    public int quadrado (int y) //método quadrado calcula x2
```

```
    {
```

```
        return y*y;
```

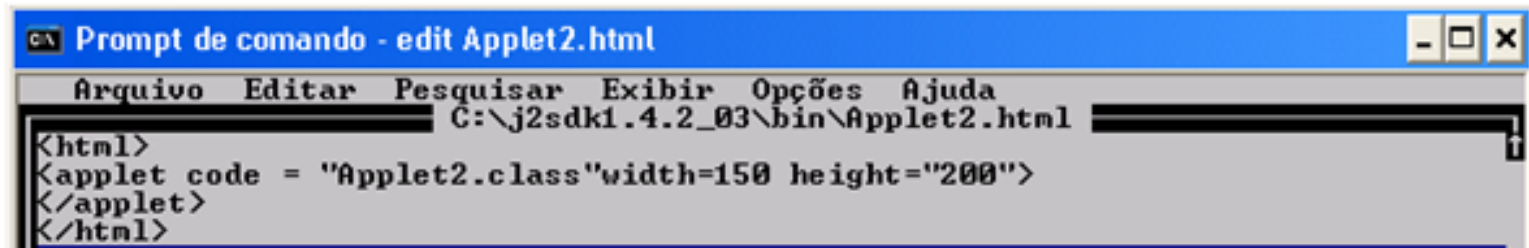
```
    }
```

```
}
```

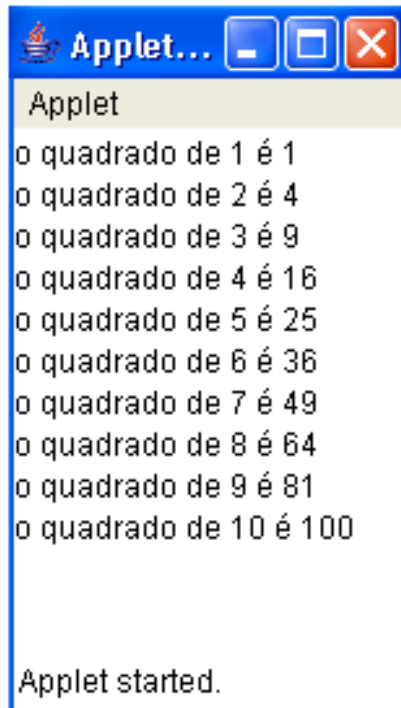
4. Arquivo html e resultados

Criação de applets

Construção do Applet



```
Prompt de comando - edit Applet2.html
Arquivo  Editar  Pesquisar  Exibir  Opções  Ajuda
C:\j2sdk1.4.2_03\bin\Applet2.html
<html>
<applet code = "Applet2.class"width=150 height="200">
</applet>
</html>
```



Exercícios:

31. Compilar e executar as classes Applet1 e Applet2.
32. Criar um applet que possua a funcionalidade de verificar a senha de um usuário.

Uso de Lista

```
package Lista;
```

```
import java.util.ArrayList;  
import java.util.List;
```

```
public class Roda_lista {
```

```
    public static void main(String args []){
```

```
        List dados = new ArrayList();
```

```
        dados.add("O Futuro Está no Passado ");  
        dados.add(37.00);
```

```
        for (int b = 0; b < dados.size(); b++)
```

```
            System.out.println("dados na lista após inserção \n" + dados.get(b));
```

```
        dados.remove(0);
```

```
        dados.remove(0);
```

```
        System.out.println("lista vazia após remoção \n" + dados.isEmpty());
```

```
    }
```

```
}
```

Problems @ Javadoc Declaration Console

<terminated> Roda_lista [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (26/05/2015 10:28:27)

dados na lista após inserção

O Futuro Está no Passado

dados na lista após inserção

37.0

lista vazia após remoção

true

Referências Bibliográficas

- Java 7 - Ensino Didático
Sérgio Furgeri - Editora Érica