

# **Programação de Sistemas Distribuídos - Java para Web**

## **5 - Applets**

### **Citeforma**

Jose Aser Lorenzo, Pedro Nunes, Paulo Jorge Martins

[jose.l.aser@sapo.pt](mailto:jose.l.aser@sapo.pt), [pedro.g.nunes@gmail.com](mailto:pedro.g.nunes@gmail.com),  
[paulojsm@gmail.com](mailto:paulojsm@gmail.com)

Abril de 2010

V1.3

## Sumário

<b>5- Applets.....</b>	<b>5-3</b>
<b>5.1- Objectivos.....</b>	<b>5-4</b>
<b>5.2- Ambiente de trabalho.....</b>	<b>5-5</b>
5.2.1- Criar projecto .....	5-5
5.2.2- Alterar parâmetros de deploy.....	5-8
5.2.3- Criar Applet .....	5-10
<b>5.3- Applet .....</b>	<b>5-11</b>
5.3.1- A applet e a segurança .....	5-12
5.3.2- A applet e a página HTML .....	5-13
5.3.2.1- A tag <applet>.....	5-13
5.3.2.2- Alinhar a applet com o texto HTML.....	5-14
5.3.3- Passar parâmetros do HTML para a applet.....	5-16
5.3.4- Ciclo de vida de uma applet.....	5-18
<b>5.4- Desenhar.....</b>	<b>5-21</b>
5.4.1- Desenhar linhas e rectângulos.....	5-22
5.4.1.1- O JPanel e o contexto gráfico.....	5-22
5.4.1.2- Desenhar linhas .....	5-23
5.4.1.3- Desenhar rectângulos .....	5-24
5.4.2- Desenhar curvas.....	5-26
5.4.2.1- Desenhar circunferências e ovais .....	5-26
5.4.2.2- Desenhar curvas e arcos .....	5-27
<b>5.5- Texto .....</b>	<b>5-29</b>
5.5.1- Escrever .....	5-30
5.5.2- Alterar o tipo de letra .....	5-31
<b>5.6- Cor .....</b>	<b>5-34</b>
5.6.1- Alterar a cor .....	5-35
5.6.2- Aplicar no contexto gráfico .....	5-36
5.6.3- Aplicar no background.....	5-36
5.6.4- Exemplo.....	5-36
<b>5.7- Imagem .....</b>	<b>5-39</b>
5.7.1- Desenhar uma imagem.....	5-40
5.7.1.1- Ler a imagem.....	5-40
5.7.1.2- Desenhar.....	5-41
5.7.2- Exemplo .....	5-41
<b>5.8- Som .....</b>	<b>5-43</b>
5.8.1- Tocar Som.....	5-44
5.8.1.1- Carregar.....	5-44
5.8.1.2- Tocar .....	5-44
5.8.2- Exemplo .....	5-45
<b>5.9- Exercício .....</b>	<b>5-47</b>
5.9.1- Exercício: Gráfico de barras .....	5-48



# Programação de Sistemas Distribuídos - Java para Web

---

## Capítulo 5 – *Applets*

José Aser Lorenzo  
Pedro Nunes  
Paulo Jorge Martins



Java Web  
© Citeforma

## 5- Applets

## Objectivos

- Integrar a *applet* na página HTML;
- Importar dados vindos do HTML;
- Desenhar figuras geométricas planas;
- Escrever texto;
- Mudar a cor de desenho e de escrita;
- Importar imagens;
- Importar sons.



### 5.1- Objectivos

Uma applet é um programa Java que se integra numa página HTML. Quando a página é carregada por um browser, a applet também o é, sendo depois executada pela máquina virtual Java contida no browser.

No fim deste capítulo o leitor estará apto a escrever applets. Estes programas podem interagir com o utilizador via interface gráfico, trabalhar com ficheiros, aceder a bases de dados ou produzir animação com imagem e som.

As applets permitem a produção de aplicações para a Internet, onde se inclui o “e-business”, graças à segurança e conectividade oferecidas pela linguagem Java. Além disso permitem a produção de aplicações em ambiente cliente servidor tradicional que não necessitam da instalação de software no cliente, usufruindo de todas as propriedades do Swing. A partir do momento em que o cliente tem um JRE (Java Runtime Environment) e um browser, ele é capaz de carregar toda a aplicação do servidor, não sendo necessário instalar software adicional no cliente.

## Sumário

- Ambiente de trabalho;
- *Applet*;
- Desenhar;
- Texto;
- Cor;
- Imagem;
- Som;
- Exercício.



### 5.2- Ambiente de trabalho

Esta capítulo requer um ambiente de trabalho com configuração especial. Nos próximos passos vamos preparar o NetBeans nesse sentido.

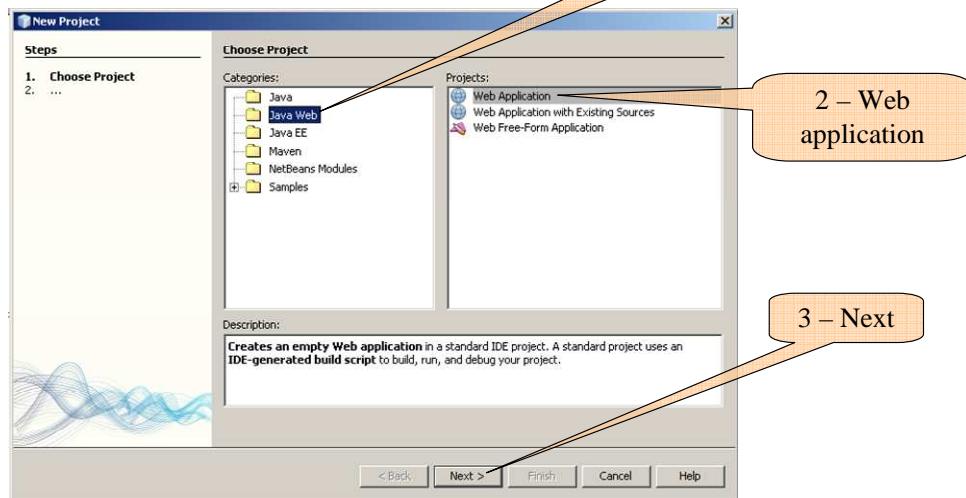
#### 5.2.1- Criar projecto

Vamos começar por criar um projecto do tipo WEB (com Tomcat).

## NetBeans - Criar um projecto

#1/4

○ File → New Project



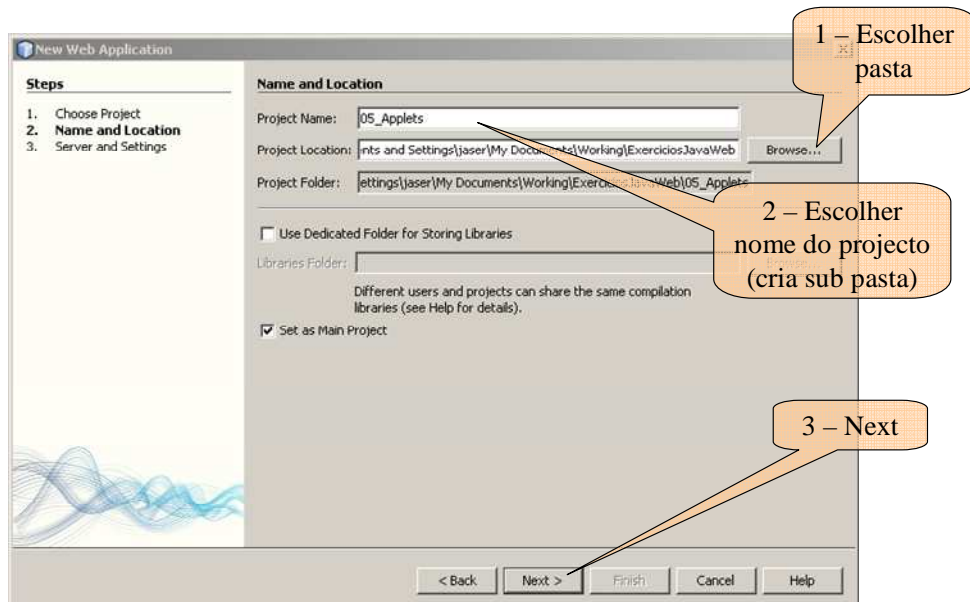
Java Web  
© Citeforma

Capítulo 5 - Applets

3

## NetBeans - Criar um projecto

#2/4



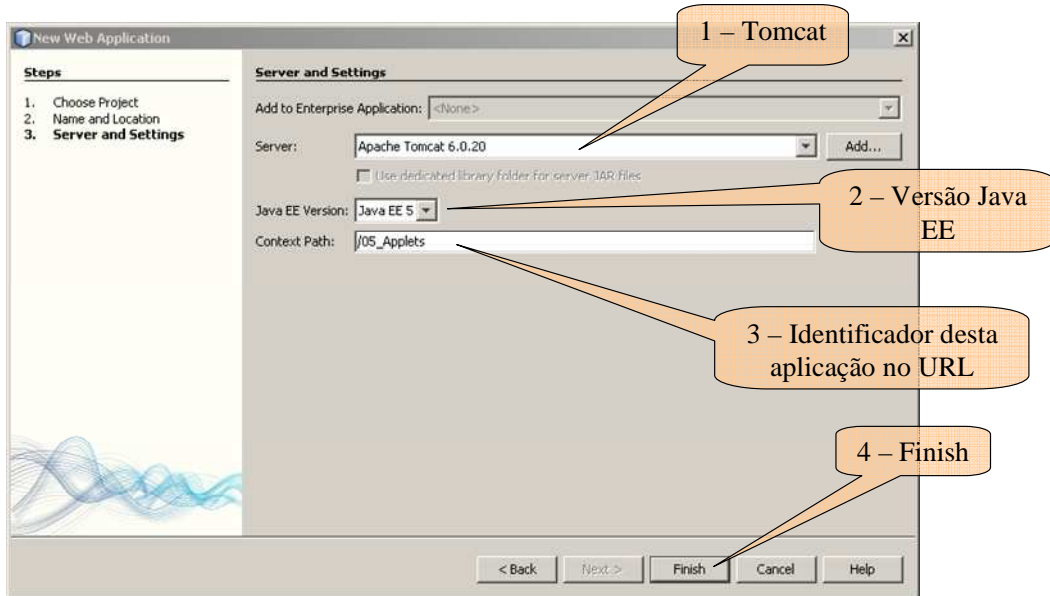
Java Web  
© Citeforma

Capítulo 5 - Applets

4

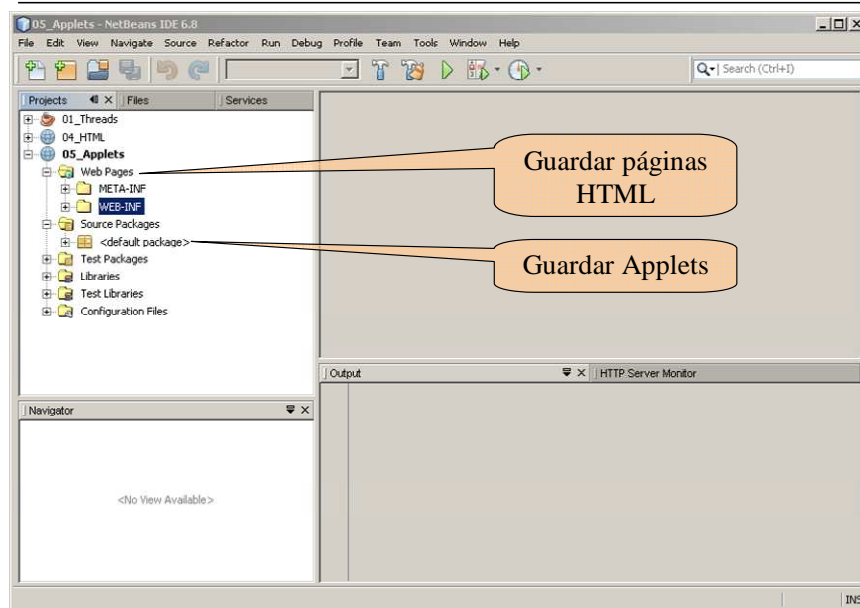
## NetBeans - Criar um projecto

#3/4



## NetBeans - Criar um projecto

#4/4



## NetBeans – Alterar parâmetros de deploy #1/2

- O NetBeans guarda os **source files** numa pasta diferente dos ficheiros HTML;
- Quando o NetBeans faz **deploy** da aplicação os class files mantêm a mesma estrutura de pastas;
- Neste projecto queremos os class files na mesma directoria que o HTML. Para isso:
  - No **desenvolvimento** do projecto mantemos a separação entre source files e HTML;
  - **Configuramos o deploy** para guardar os class files na directoria do HTML;



### 5.2.2- Alterar parâmetros de deploy

Como o slide indica a configuração por omissão do NetBeans não serve os nossos interesses. É necessário alterar os parâmetros que configuram o deploy da aplicação para que os ficheiros fiquem guardados no sítio certo.

A palavra deploy representa o conjunto de operações que têm que ser feitas para levar a aplicação WEB do ambiente de desenvolvimento (ou qualidade) para o ambiente de produção. O NetBeans invoca o Tomcat de forma automática e coloca os ficheiros da aplicação no sítio esperado pelo Tomcat.

Nesta aplicação em concreto queremos que os class files (ficheiros resultantes da compilação dos source files) sejam guardados na mesma directoria que os ficheiros HTML. Não vamos forçar essa arrumação durante o desenvolvimento, mas vamos configurar o deploy para que no Tomcat os ficheiros fiquem arrumados dessa forma.



## NetBeans – Alterar parâmetros de deploy #2/2

1- Separador Files

2- nbproject → project.properties

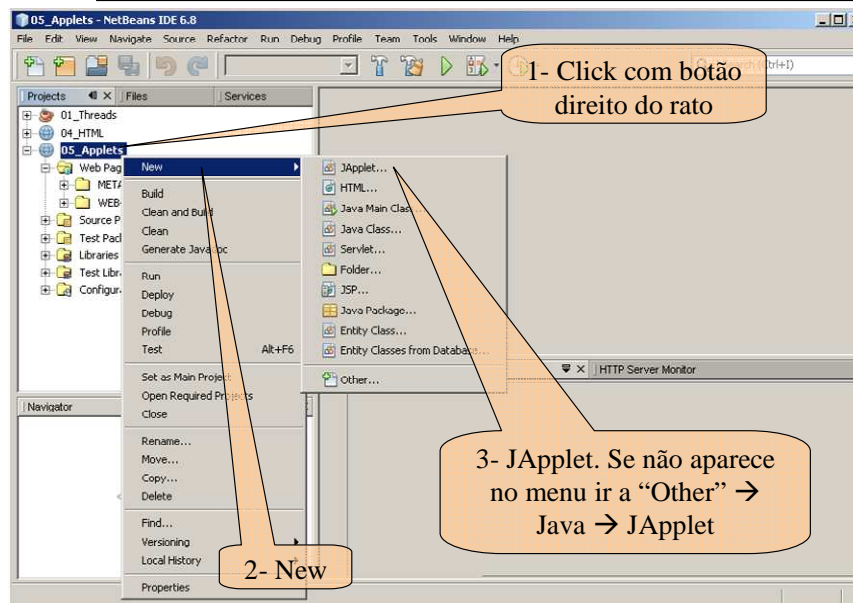
3- Alterar a directoria onde são deployed os class files. Colocar a mesma que HTML

```
1 build.classes.dir=${build.web.dir}
2 build.classes.excludes=**/*.java,**/*.form
3 build.dir=build
4 build.generated.dir=${build.dir}/generated
5 build.generated.sources.dir=${build.dir}/generated-sources
6 build.test.classes.dir=${build.dir}/test/classes
7 build.test.results.dir=${build.dir}/test/results
8 build.web.dir=${build.dir}/web
9 build.web.excludes=${build.classes.excludes}
10 client.urlPart=
11 compile.jsps=false
12 conf.dir=${source.root}/conf
13 debug.classpath=${build.classes.dir}:${javac.classpath}
```

Esta configuração requer uma alteração no ficheiro **project.properties**, que configura todo o projecto no NetBeans. Este ficheiro não é visível no separador **Projects**, pelo que temos que invocar o separador **Files**. Aqui abrimos a pasta **nbproject**.

Este ficheiro possui uma entrada que define a directoria onde vão ser guardados os class files (**build.classes.dir**). Por omissão o seu valor depende da variável **build.web.dir** acrescido de **WEB-INF/classes**. Se removermos estas entradas os class files serão armazenados na mesma directoria que os ficheiros HTML (**build.web.dir**) que por omissão recebe o valor **web**.

## NetBeans – Criar applet



### 5.2.3- Criar Applet

Tendo o projecto criado e os parâmetros de deploy configurados podemos começar a escrever os exemplos deste capítulo. As páginas HTML serão escritas como vimos no capítulo sobre HTML. As applets Java serão classes Java que deverão ser criadas como descreve o slide.

## Sumário

---

- Ambiente de trabalho;
- *Applet*;
- Desenhar;
- Texto;
- Cor;
- Imagem;
- Som;
- Exercício.



### 5.3- Applet

Vamos ver o que é uma applet e como construí-la.

## A *applet* e a segurança

- A *applet* é descarregada do *Web Server* e executada pela *JVM* que está integrada no *browser* do cliente.
- A *JVM* não deixa a *applet*:
  - Ler/escrever do/no disco do computador local. Pode fazê-lo do/no servidor de onde foi importada;
  - Executar programas armazenados no computador local;
  - Comunicar com outras máquinas para além daquela de onde veio.



### 5.3.1- A *applet* e a segurança

Como foi visto anteriormente, existe uma diferença entre um programa normal Java e uma *applet*. A *applet* é executada pela máquina virtual Java incluída no *browser* do cliente, possuindo algumas restrições de segurança no que toca à utilização de recursos da máquina onde o *browser* corre. Poderá a *applet* apagar toda a informação contida no disco? Poderá enviar informação do disco para o exterior? Em termos de segurança, as especificações da SUN Microsystems impõem:

- As *applets* estão proibidas de ler/escrever do/no disco do PC local. Podem fazê-lo do/no servidor de onde foram importadas.
- Não pode haver comunicação entre a *applet* e outros servidores WWW para além daquele de onde foi importada.
- As *applets* não podem executar programas armazenados no PC local, nem carregar bibliotecas.

A implementação destas restrições está a cargo da máquina virtual Java que está a ser utilizada. Alguns fabricantes de máquinas virtuais Java para ambiente Windows permitem que as *applets* acedam a API's do Windows, o que viola as especificações originais da máquina virtual, dando ao programa a possibilidade de violar as normas de segurança. Para o evitar a SUN desenvolveu um JRE (Java Runtime Environment) para Windows que respeita todas as normas de segurança e disponibiliza-o sem encargos, podendo ser obtido no site da SUN.

As *applets* são subclasses de **java.applet.Applet** onde é definido todo o seu comportamento. Neste capítulo usaremos **java.swing.JApplet**.

## A *applet* e a página HTML

- A TAG *applet*:
  - `<applet code="Alinhamento" width="150" height="50" align="top"> </applet>`
  - **WIDTH** - comprimento da *applet* em pixel;
  - **HEIGHT** - altura da *applet* em pixel;
  - **HSPACE** - espaço horizontal entre a *applet* e o texto em pixel;
  - **VSPACE** - espaço na vertical entre a *applet* e o texto em pixel;
  - **CODE** - nome da classe Java (*applet*) que vai ser executada;
  - **CODEBASE** - URL de localização da classe Java;
  - **ALIGN** - alinhamento da *applet* em relação ao texto HTML (BOTTON, ABSMIDDLE, TOP, LEFT, RIGHT);
  - **ARCHIVE** - indica um ficheiro "\*.zip" ou "\*.jar" que arquiva classes Java, imagens e som a serem importados pela *applet*;
- **Exercício: HTML + classe Java. Ver *applet* a correr no browser e variar posicionamento da *applet*.**



### 5.3.2- A *applet* e a página HTML

Neste ponto vamos ver como uma *applet* interage com o browser e o código HTML que a manda executar.

#### 5.3.2.1- A tag `<applet>`

A *applet* começa a ser executada quando a página HTML que lhe faz referência é chamada. Para referenciar a *applet* no código HTML usa-se a TAG `<applet>` como exemplificado a seguir:




```
<applet code="NomeFicheiroClass" configuração>
</applet>
```

A TAG `<applet>` aceita os seguintes atributos:-

- **WIDTH** - define o comprimento da *applet* em pixel;
- **HEIGHT** - define a altura da *applet* em pixel;
- **HSPACE** - define o espaço horizontal entre a *applet* e o texto em pixel;
- **VSPACE** - define o espaço na vertical entre a *applet* e o texto em pixel;
- **CODE** - nome da classe java (*applet*) que vai ser executada;
- **CODEBASE** - localização da classe java que vai ser executada, usando o formato URL (Universal Resource Locator). Informa o browser onde procurar a *applet*;

- ALIGN - alinhamento da área reservada à applet em relação ao resto do conteúdo da página HTML;
- ARCHIVE - indica um ficheiro "\*.zip" ou "\*.jar" que arquiva várias classes Java, ficheiros de imagens e/ou ficheiros de som a serem importados para a applet. Reduz as comunicações já que usa algoritmos de compactação.

Exemplo:




```
<applet code="Alinhamento" width=150 height=50 align=top>
</applet>
```

### 5.3.2.2- Alinhar a applet com o texto HTML

Para alinhar a applet com o texto da página HTML usa-se o atributo **align**. Este pode receber os seguintes valores:

- BOTTOM → o rectângulo da applet é alinhado com a base da linha de texto que o rodeia;
- ABSMIDDLE → o meio do rectângulo da applet é colocado a meio da linha de texto que o rodeia;
- TOP → o topo do rectângulo da applet é colocado no topo da linha de texto que o rodeia;
- LEFT → o rectângulo da applet é encostado à esquerda na página;
- RIGHT → o rectângulo da applet é encostado à direita da página.

Para exemplificar vamos escrever a seguinte applet:




```
import java.awt.Color;
import java.awt.Graphics;

public class Alinhamento extends javax.swing.JApplet {

    public void paint(Graphics g) {
        this.setBackground(Color.LIGHT_GRAY);
        g.drawString("Applet running", 10, 25);
    }
}
```

Escrever o código HTML abaixo:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>alinhamento</title>
    <!-- alinhamento.html -->
</head>
<body>
    <p>align<applet code="Alinhamento" width=100 height=40
align="bottom"></applet>bottom</p>
    <br>
    <hr>
    <p>align<applet code="Alinhamento" width=100 height=40
align="middle"></applet>middle</p>
    <br>
    <hr>
    <p>align<applet code="Alinhamento" width=100 height=40
align="top"></applet>top</p>
    <br>
    <hr>
```

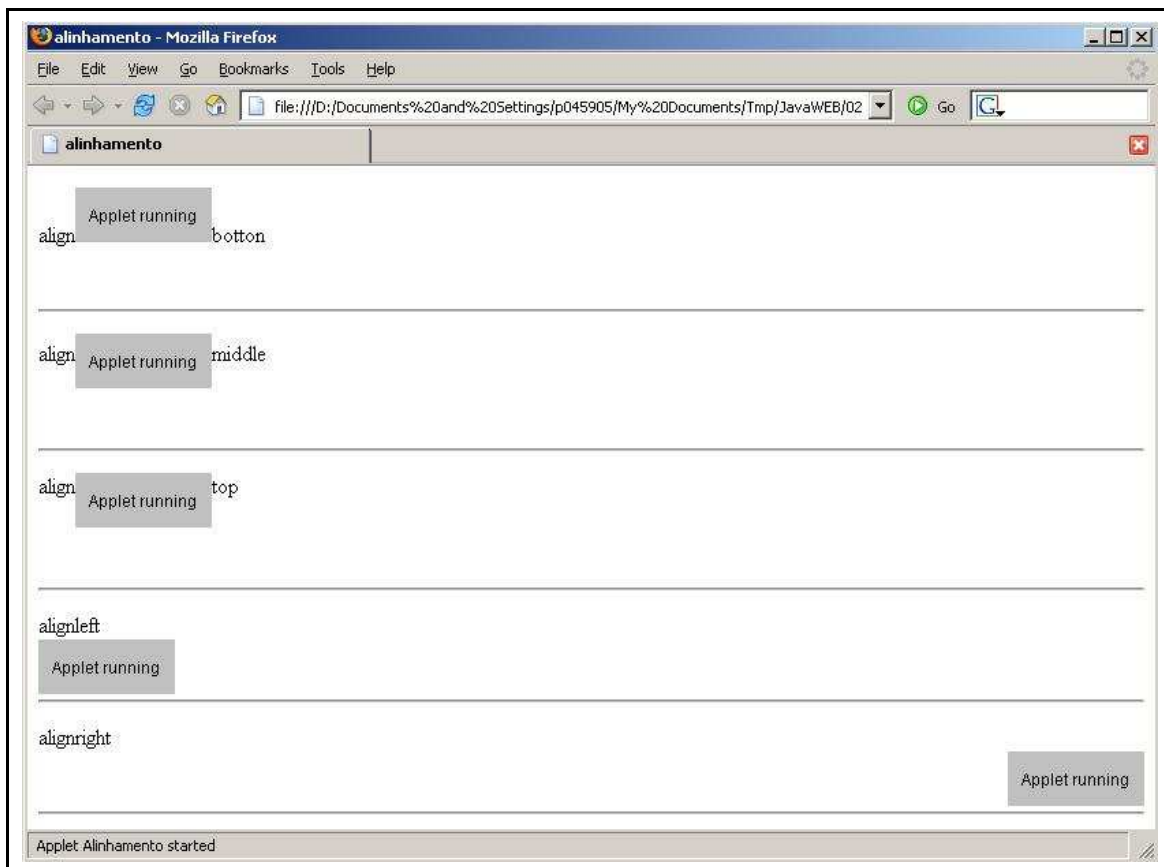
```

<p>align<applet code="Alinhamento" width=100 height=40
align="left"></applet>left</p>
<br>
<hr>
<p>align<applet code="Alinhamento" width=100 height=40
align="right"></applet>right</p>
<br>
<hr>
</body>
</html>

```

Para executar a applet é necessário executar a página HTML, o que é feito seleccionando a página e usando o botão do lado direito do rato, onde aparece a opção RUN.

A página fica com o aspecto abaixo:



A página anterior executa a mesma applet 5 vezes em simultâneo. Cada uma tem o seu espaço de visualização, enquadrado no texto da página.

## Passar parâmetros do HTML para a *applet*

```
<applet code="Parametros" width=250 height=100>
  <param name="var1" value="Teste com raiz quadrada de um
    número:">
  <param name="var2" value="6.25f">
</applet>
```

```
public void init() {
    var1 = getParameter("var1");
    var2 = getParameter("var2");
}
```

- Exemplo: Página HTML e *applet* Parametros.



### 5.3.3- Passar parâmetros do HTML para a *applet*

Podem ser passados parâmetros de uma página HTML para o código da *applet* que está em execução. Dentro da TAG `<applet>` podemos usar o atributo `<param>` que define o nome e valor do parâmetro a passar para o programa Java. A sintaxe a utilizar é a seguinte:



```
<applet code="nomeFicheiro" configuração>
  <param name="var1" value="valor1">
  <param name="var2" value="valor2">
  <param name="var3" value="valor3">
  Etc...
</applet>
```

O valor passado pode ser recuperado no interior da *applet* utilizando o método `getParameter()`, que aceita como argumento o nome da variável a passar como string (ex. `var1`) e devolve o respectivo valor como string (ex. `valor1`). Se o utilizador pretender manipular outros tipos de dados deverá efectuar a conversão de string para o tipo pretendido. Veja-se o exemplo a seguir:

O programa fonte é o seguinte:



```
import java.awt.Graphics;


public class Parametros extends javax.swing.JApplet {
    String var1, var2;

    /* obter o parametro que vem da pagina html */
```



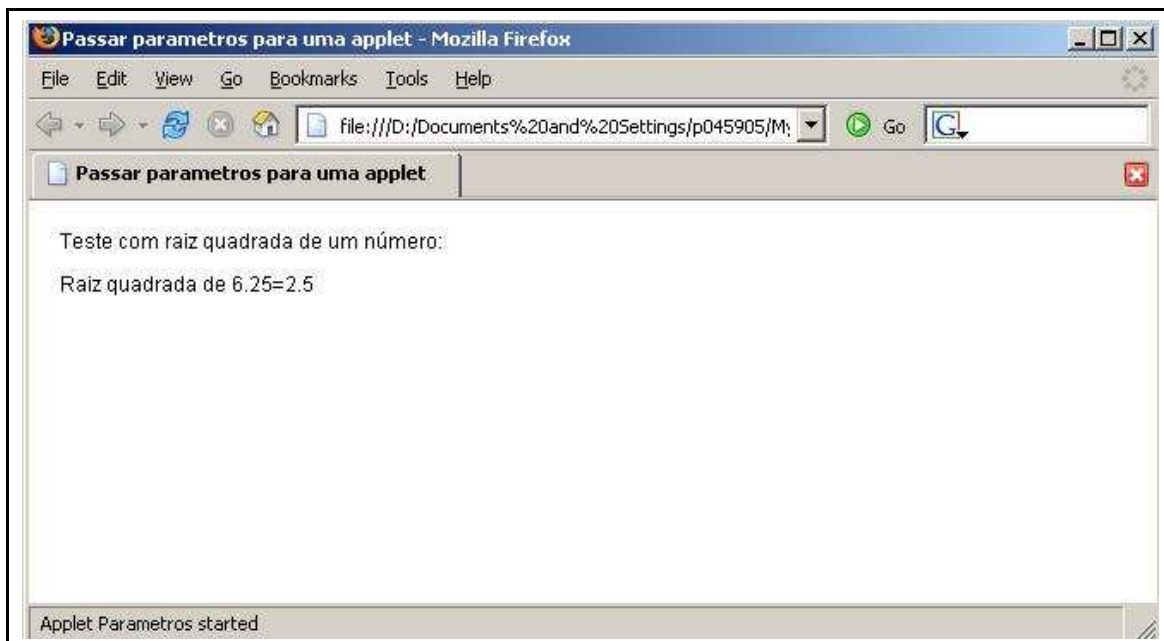
```
public void init() {  
    var1 = getParameter("var1");  
    var2 = getParameter("var2");  
}  
  
/* "pinta" o que esta na var1 e var2... */  
  
public void paint(Graphics g) {  
    float a = Float.valueOf(var2).floatValue();  
    g.drawString(var1, 10, 20);  
    g.drawString("Raiz quadrada de " + a + "=" + (Math.sqrt(a)), 10, 45);  
}  
}
```

O código HTML que chama a applet:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html>  
<head>  
    <title>Passar parametros para uma applet</title>  
    <!-- parametros.html -->  
</head>  
<body>  
    <applet code="Parametros" width=250 height=100>  
        <param name="var1" value="Teste com raiz quadrada de um número:">  
        <param name="var2" value="6.25f">  
    </applet>  
</body>  
</html>
```

O output no browser:



## Ciclo de vida de uma *applet*



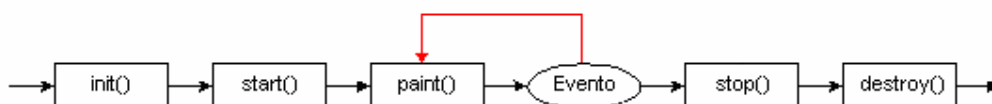
○ Nem sempre foi assim...

○ **Exercício:** Página HTML + *Applet* Ciclo de Vida.



### 5.3.4- Ciclo de vida de uma applet

Um programa tradicional Java é executado de acordo com o método `main()`, que sendo o primeiro, cria os objectos principais e chama os outros métodos. Pelo contrário, uma applet é controlada por um conjunto de métodos, executados numa sequência preestabelecida e em resposta aos eventos provocados pelo utilizador sobre a applet e sobre o browser. O diagrama apresentado a seguir mostra os estados que a applet atravessa durante o seu ciclo de vida. A cada estado corresponde um método.



Os estados / métodos são os seguintes:-

- **init()** → método executado quando a applet é carregada pela primeira vez, sendo aproveitado para criar objectos ou inicializar variáveis;
- **start()** → executado depois do `init()`;
- **stop()** → executado depois de um evento que provoque a paragem da applet;
- **destroy()** → executado quando a applet termina;


- **paint()** → define o que a applet desenha. É utilizado em animação para desenhar as diferentes frames separadas por um intervalo de tempo, ou para refrescar a applet quando é tapada por outra janela ou quando o browser é “restored” após ter sido minimizado;
- **evento** → representa um evento sobre a applet ou página HTML. Por exemplo quando a applet é tapada por outra janela, quando o browser é minimizado ou restored. Representa o período de tempo durante o qual a applet aguarda que algo aconteça;

Estes métodos estão definidos na classe Applet, mas com um comportamento nulo. Para que a nossa applet faça o que se pretende é necessário reescreve-los, tendo o cuidado de os definir como public.

Em seguida vamos construir uma applet que mostra os diferentes estados pelos quais passa durante a sua execução. O código da applet inclui um contador de execução para cada método. Após incrementar o contador é adicionada uma linha a uma string, que será depois escrita na área da applet que aparece no browser. O código Java escreve também mensagens no standard output, cuja visualização depende do tipo de execução:

- Se a applet for invocada por uma página HTML e tiver sido instalada uma versão do JRE da SUN igual ou posterior a 1.3, então o browser deve utilizar esse JRE. Este possui uma consola para onde é escrito o standard output, que por omissão não aparece visível. Para a mostrar ir ao CONTROL PANEL, activar a vista clássica e escolher JAVA (parametrização do JRE). Agora temos que escolher o parâmetro que mostra a consola em permanência e cuja localização varia com a versão.
- Se a applet for executada pelo AppletViewer o standard output será a janela que o desencadeou. Este programa permite simular a máquina virtual Java de um browser, recebendo como input uma página HTML e mostrando cada uma das suas applets numa janela separada.

O código da applet é o seguinte:



```
import java.awt.Graphics;

public class CicloVida extends javax.swing.JApplet {
    String estado = "";
    int init = 0, start = 0, paint = 0, stop = 0, destroy = 0;


    public void init() {
        init += 1;
        estado = estado + "init(" + init + ")";
        System.out.println("init(" + init + ")");
    }
    public void start() {
        start += 1;
        estado = estado + "start(" + start + ")";
        System.out.println("start(" + start + ")");
    }
    public void stop() {
        stop += 1;
        estado = estado + "stop(" + stop + ")";
        System.out.println("stop(" + stop + ")");
    }
    public void destroy() {
        destroy += 1;
        estado = estado + "destroy(" + destroy + ")";
        System.out.println("destroy(" + destroy + ")");
    }
    public void paint(Graphics g) {
        paint += 1;
        estado = estado + "paint(" + paint + ")";
    }
}
```

```

        g.drawString(estado, 10, 20);
        System.out.println("paint(" + paint + ")");
    }
}

```

O código HTML para a testar é o seguinte:



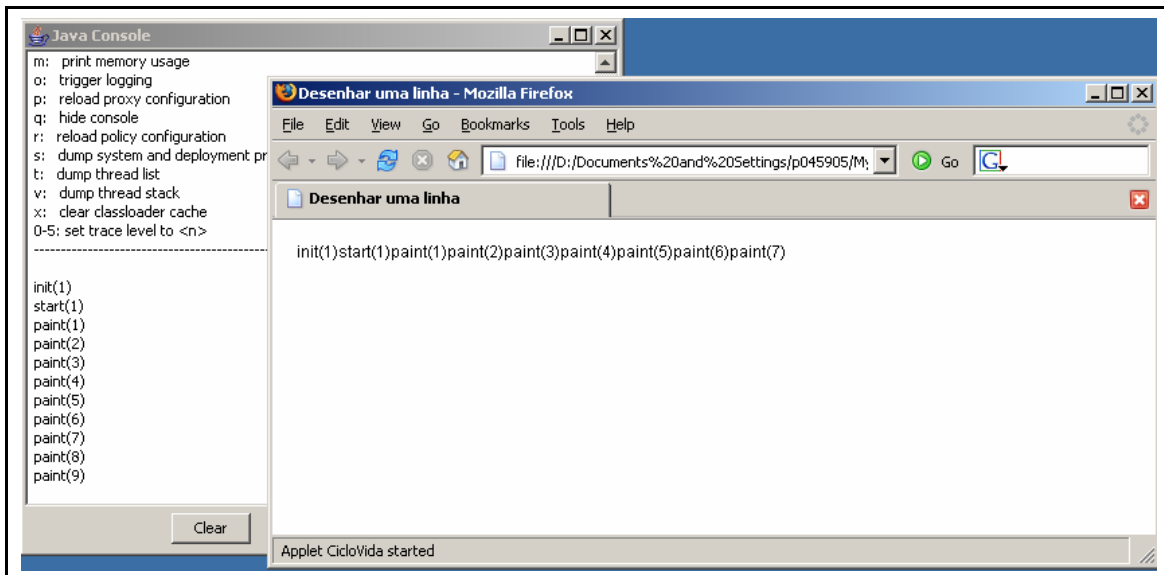
```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>CicloVida</title>
    <!-- ciclovida.html -->
</head>
<body bgcolor="Blue">

    <applet code="CicloVida" width=400 height=50>
    </applet>
</body>
</html>

```

No browser o output fica assim:



Repare que:

- Ao **minimizar** e **maximizar** a janela do browser o paint() é executado automaticamente;
- Ao fazer **PgDown** e **PgUp** é provocado um refresco do ecrã, o que lança o paint() outra vez;
- Ao **redimensionar** a janela do browser o paint() é executado;
- Ao **sobrepor** uma janela de outra aplicação sobre o browser, quando retorna ao browser o paint() é chamado outra vez;
- Ao utilizar o botão **"reload"** (ou refresh no IE) são executados os métodos stop(), destroy() e depois start(), init() e paint(). O mesmo se passa fazendo "back" (ir para a página anterior) e depois "forward" para regressar a esta página.

## Sumário

---

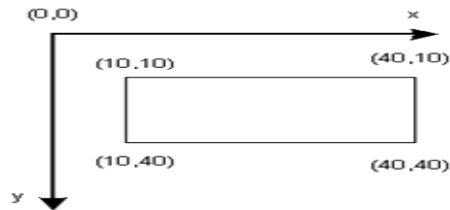
- Ambiente de trabalho;
- *Applet*;
- Desenhar;
- Texto;
- Cor;
- Imagem;
- Som;
- Exercício.



### 5.4- Desenhar

Neste ponto vamos ver como se trabalha em Java com elementos. Os exemplos estão orientados para applets, mas podem ser também aplicados em programas Swing.

## Desenhar linhas e rectângulos



```
public void paint(Graphics g) {  
    g.drawLine(0, 0, 240, 140);  
    drawRect(10,10,40,40); //fillRect(...)  
}  
drawRoundRect(x,y,compX,compY,rx,ry), fillRoundRect(...)  
draw3DRect(x,y,compX,compY,saliente), fill3DRect(...)
```

### ○ Exercício: Desenhar rectângulo.



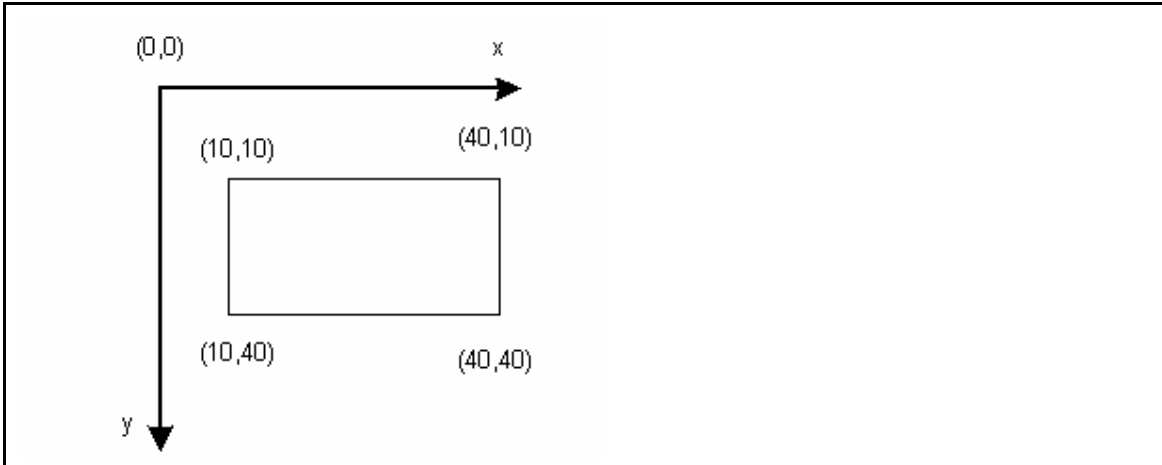
### 5.4.1- Desenhar linhas e rectângulos

#### 5.4.1.1- O JPanel e o contexto gráfico

Quando se cria um objecto da classe JApplet ele recebe um JPanel que ocupa toda a área reservada à applet. A classe JPanel é um “container” que pode receber objectos Swing ou AWT, tais como JLabel e JButton. Estes objectos são abordados no curso de Swing.

O método paint() recebe como parâmetro um objecto da classe Graphics, pertencente ao package java.awt. A imagem resultante da colocação de objectos no JPanel da applet é capturada por esse objecto. Sobre esse contexto gráfico podem depois ser desenhados outros componentes gráficos, como linhas, rectângulos, círculos e arcos. As propriedades com que se está a desenhar, por exemplo a cor da caneta, a cor de fundo e o tipo de letra, ficam também armazenadas nesse objecto. Por isso se diz que ele guarda o **contexto gráfico** da applet.


Para a colocação de um elemento gráfico no rectângulo da applet é necessário utilizar um sistema de coordenadas que referencie as diferentes posições. Em Java todos os objectos gráficos referenciam a sua posição em pixel, tendo como origem o canto superior esquerdo do rectângulo, como mostra a seguinte figura:



#### 5.4.1.2- Desenhar linhas

Para desenhar uma linha sobre o contexto gráfico, utiliza-se o método **drawLine(x1,y1,x2,y2)** pertencente à classe **Graphics**. A linha desenhada vai do ponto (x1,y1) ao ponto (x2,y2). Como o **paint()** é o responsável pelo output, é neste método que normalmente se chama o **drawLine()**.


O exemplo abaixo mostra como se desenha uma linha:



```
import java.awt.Graphics;

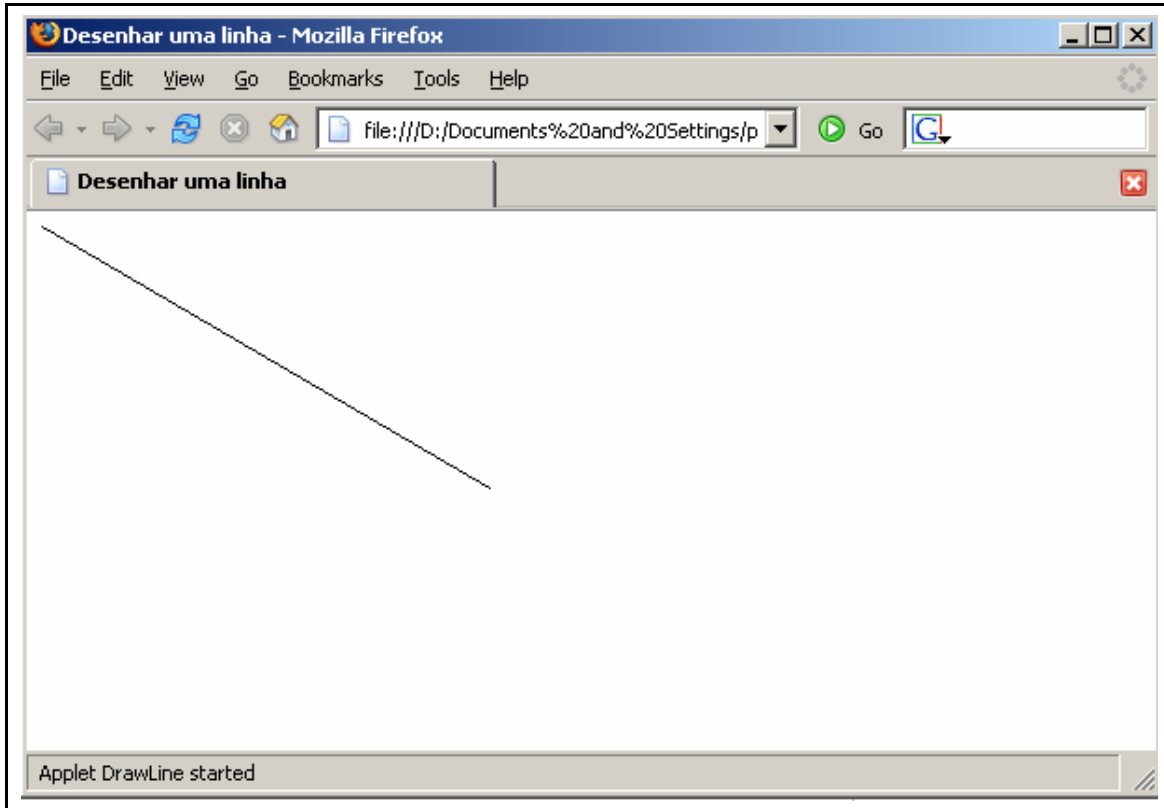
public class DrawLine extends javax.swing.JApplet {
    public void paint(Graphics g) {
        g.drawLine(0, 0, 240, 140);
    }
}
```

O código HTML para testar a applet:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Desenhar uma linha</title>
    <!-- drawline.html -->
</head>
<body bgcolor="Blue">
    <applet code="DrawLine" width=400 height=170>
    </applet>
</body>
</html>
```

O output da applet:



#### 5.4.1.3- Desenhar rectângulos

Para desenhar um rectângulo utilizamos um dos métodos da classe Graphics:-

- **drawRect(x,y,compx,compy)** → desenha um rectângulo simples, com o canto superior esquerdo em (x,y), comprimento compx e altura compy;
- **fillRect(x,y,compx,compy)** → mesmo que o anterior, mas o rectângulo é preenchido;
- **drawRoundRect(x,y,compx,compy,rx,ry)** → desenha um rectângulo de cantos arredondados, com o canto superior esquerdo em (x,y), comprimento compx, altura compy e os cantos arredondados de largura rx e altura ry;
- **fillRoundRect(x,y,compx,compy,rx,ry)** → mesmo que o anterior, mas o rectângulo é preenchido;
- **draw3DRect(x,y,compx,compy,saliente)** → desenha um rectângulo com relevo para fora (saliente=true) ou para dentro (saliente=false). O realce dado à altura do relevo é pequeno, pelo que se torna difícil ver o efeito 3D. Para o acentuar recorre-se ao desenho com cor de fundo;
- **fill3DRect(x,y,compx,compy,saliente)** → mesmo que o anterior mas preenchido;

A applet abaixo desenha os diferentes tipos de rectângulos:

```

import java.awt.Graphics;

public class DrawRect extends javax.swing.JApplet {
    public void paint(Graphics g) {
        g.drawRect(10, 10, 200, 50);
        g.fillRect(220, 10, 200, 50);
        g.drawRoundRect(10, 70, 200, 50, 10, 10);
    }
}

```



```

        g.fillRoundRect(220, 70, 200, 50, 10, 10);

        g.draw3DRect(10, 130, 95, 50, true);
        g.draw3DRect(115, 130, 95, 50, false);
        g.fill3DRect(220, 130, 95, 50, true);
        g.fill3DRect(325, 130, 95, 50, false);

        g.setColor(getBackground());
        g.draw3DRect(10, 190, 95, 50, true);
        g.draw3DRect(115, 190, 95, 50, false);
        g.fill3DRect(220, 190, 95, 50, true);
        g.fill3DRect(325, 190, 95, 50, false);
    }
}

```

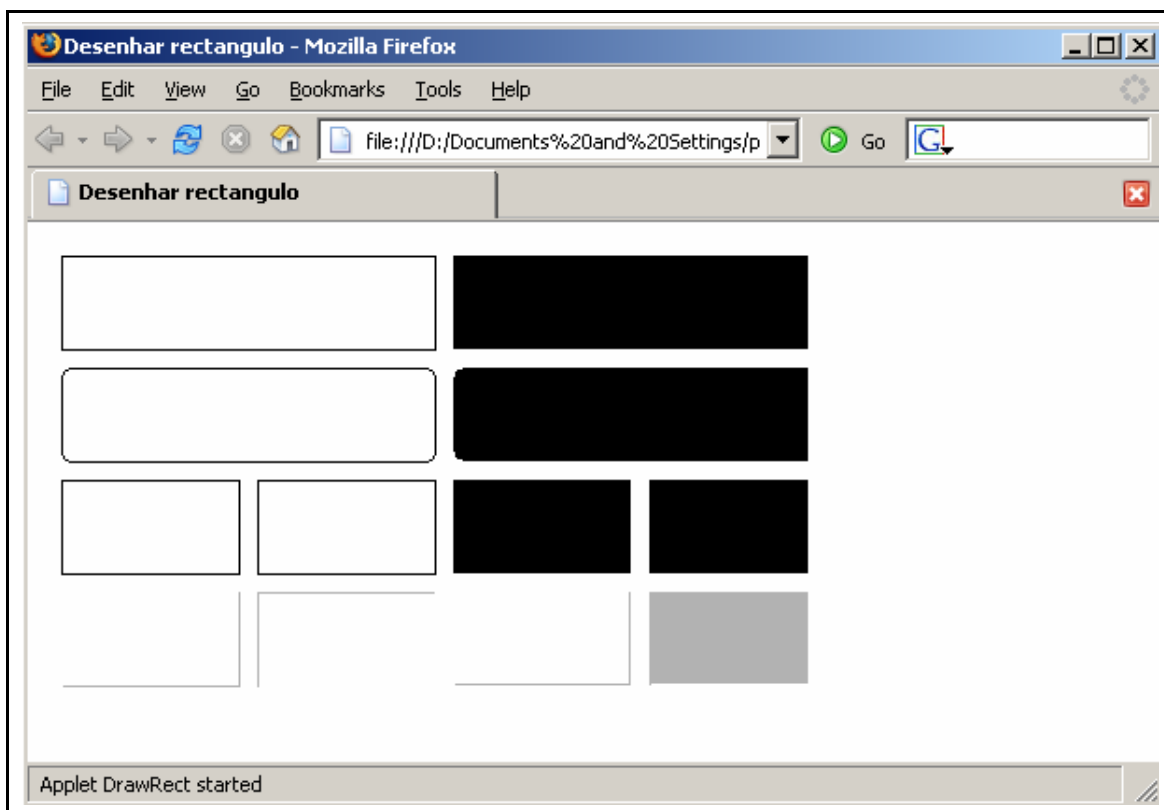
O código HTML:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Desenhar rectangulo</title>
    <!-- drawrect.html -->
</head>
<body>
    <applet code="DrawRect" width=410 height=250>
    </applet>
</body>
</html>

```

O output da applet no browser:



## Desenhar curvas

- Ovais, circunferências e círculos:
  - `drawOval(x,y,compx,compy);`
  - `fillOval(x,y,compx,compy);`
- Curvas e arcos:
  - `drawArc(x,y,compx,compy,ai,af);`
  - `fillArc(x,y,compx,compy,ai,af);`
- **Exercício:** Desenhar curvas.




### 5.4.2- Desenhar curvas

#### 5.4.2.1- Desenhar circunferências e ovais

Os **`drawOval()`** e **`fillOval()`**, pertencentes à classe `Graphics`, permitem desenhar circunferências, círculos e ovais. O método desenha uma oval circunscrita pelo rectângulo que recebe como parâmetro. Se o parâmetro for um quadrado temos um círculo. Caso use `fill` terá uma circunferência.

- **`drawOval(x,y,compx,compy);`**
- **`fillOval(x,y,compx,compy);`**

O código da applet:



```
import java.awt.Graphics;

public class DrawOval extends javax.swing.JApplet {
    public void paint(Graphics g) {
        g.drawRect(10, 10, 100, 100);
        g.drawOval(10, 10, 100, 100);
        g.fillOval(120, 10, 100, 100);
        g.drawOval(10, 120, 100, 50);
        g.fillOval(120, 120, 100, 50);
    }
}
```

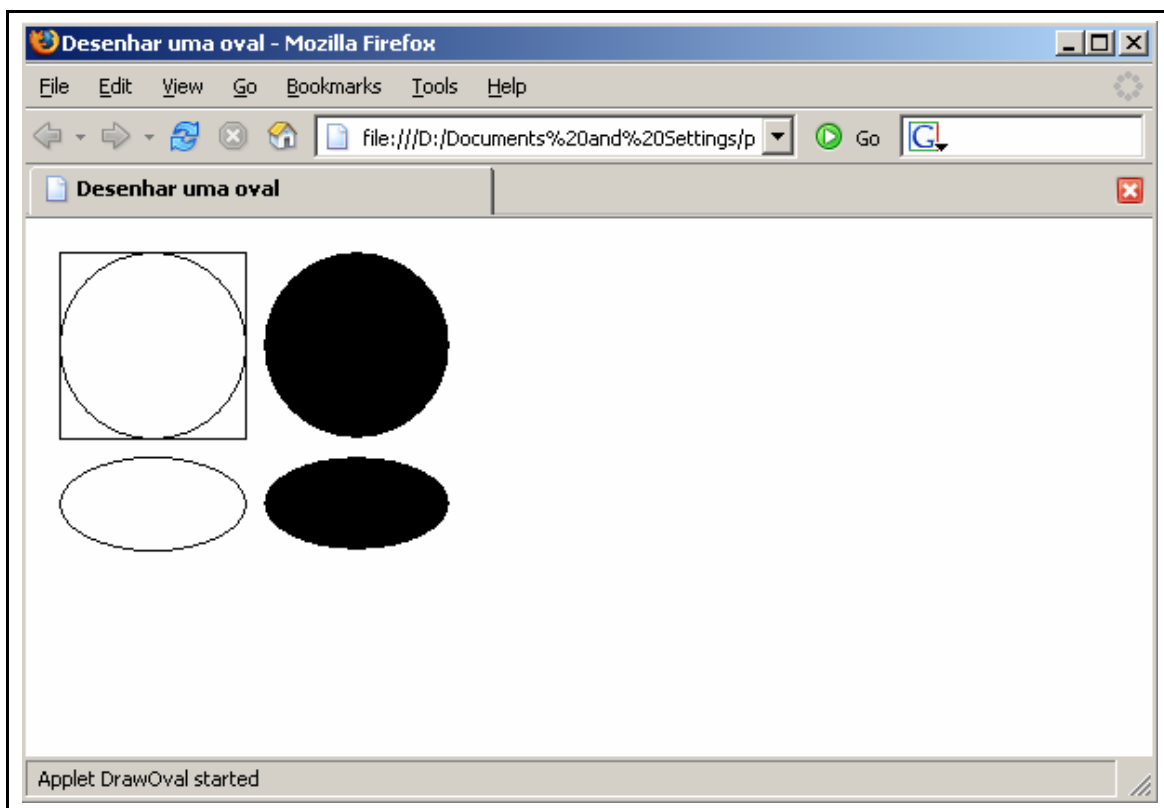
O código HTML:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Desenhar uma oval</title>
  <!-- drawoval.html -->
</head>
<body>
  <applet code="DrawOval" width=230 height=200>
  </applet>
</body>
</html>

```

O output no browser:




#### 5.4.2.2- Desenhar curvas e arcos

Os métodos **drawArc()** e **fillArc()** pertencem à classe **Graphics** e permitem desenhar arcos de circunferência, círculo ou oval. O método desenha um arco contido numa oval, definido entre os ângulos 'ai' (inicial) e 'af' (final), medidos entre 0 e 360°, no sentido directo (contrário aos ponteiros do relógio).

- `drawArc(x,y,compx,compy,ai,af);`
- `fillArc(x,y,compx,compy,ai,af);`

O código da applet:




```
import java.awt.Graphics;

public class DrawArc extends javax.swing.JApplet {
    public void paint(Graphics g) {
        g.drawOval(10, 10, 100, 100);
        g.drawArc(10, 10, 100, 100, 0, 90);
        g.drawArc(110, 10, 100, 100, 0, 90);

        g.drawOval(10, 120, 100, 30);
        g.fillArc(10, 120, 100, 30, 0, 90);
        g.fillArc(110, 120, 100, 30, 0, 90);
    }
}
```

O código HTML:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Desenhar um arco</title>
    <!-- drawarc.html -->
</head>
<body>
    <applet code="DrawArc" width=230 height=200>
    </applet>
</body>
</html>
```

O output no browser:



## Sumário

---

- Ambiente de trabalho;
- *Applet*;
- Desenhar;
- Texto;
- Cor;
- Imagem;
- Som;
- Exercício.



### 5.5- Texto

## Escrever texto

- Escrever:
  - `g.drawString("Texto escrito por drawString()", 10, 30);`
- Alterar o tipo de letra:
  - `Font f = new Font (Nome,Tipo,Tamanho);`
  - **Nome:** "TimesRoman", "Serif", "SansSerif", "Courier", "Arial", "Monospaced", "Terminal", "Helvetica", "Dialog", "Symbol"
  - **Tipo:** `Font.PLAIN`, `Font.BOLD`, `Font.ITALIC`;
  - **Tamanho:** número inteiro entre 8 e 24;
- **Exercício:** *Applet* que escreve texto.



### 5.5.1- Escrever

Para escrever texto na applet utiliza-se o método **drawString(texto,x,y)** pertencente à classe **Graphics**. O argumento "texto" é a string que será escrita. Os números "x" e "y" são as coordenadas do ponto onde começa uma linha horizontal imaginária sobre a qual será escrito o texto. O tipo de letra, tamanho e cor que serão utilizados são os pré definidos.

O código da applet:



```
import java.awt.Graphics;

public class Texto extends javax.swing.JApplet {
    public void paint(Graphics g) {
        g.drawString("Texto escrito por drawString()", 10, 30);
        g.drawString("Texto escrito com linha suporte", 10, 70);
        g.drawLine(10, 70, 200, 70);
    }
}
```

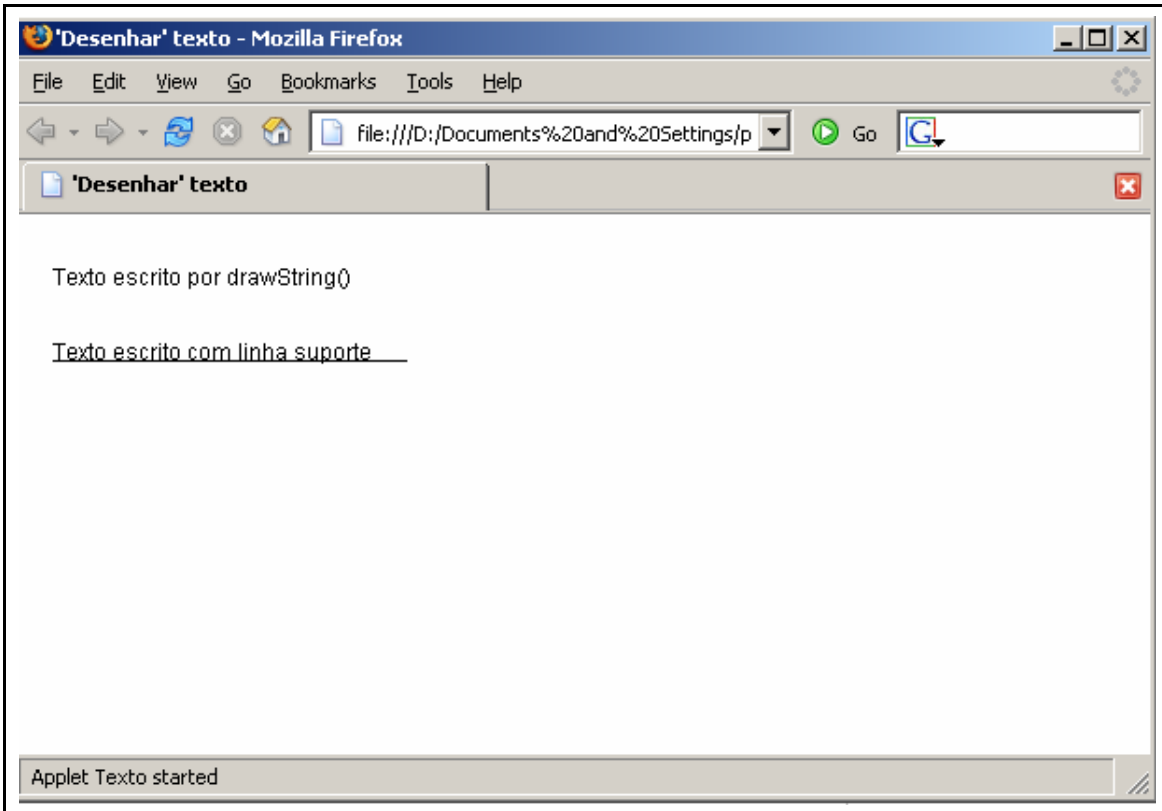
O código HTML:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>'Desenhar' texto</title>
    <!-- texto.html -->
</head>
<body>
    <applet code="Texto" width=200 height=100>
```

```
</applet>
</body>
</html>
```

O output no browser:



### 5.5.2- Alterar o tipo de letra

Para alterar o tipo de letra e seu tamanho é necessário criar um objecto da classe **Font** que pertence ao package `java.awt`. As propriedades deste objecto serão aproveitadas pelo ambiente gráfico para alterar o tipo de letra activo e o seu tamanho. Um dos construtores da classe `Font` recebe os seguintes parâmetros:



```
Font f = new Font ("nome",tipo,tamanho);
```


- “**nome**” pode assumir o valor de uma das seguintes constantes definidas na classe `Font`: `"TimesRoman"`, `"Serif"`, `"SansSerif"`, `"Courier"`, `"Arial"`, `"Monospaced"`, `"Terminal"`, `"Helvetica"`, `"Dialog"`, `"Symbol"`;
- “**tipo**” pode assumir o valor de uma das seguintes constantes: `Font.PLAIN`, `Font.BOLD`, `Font.ITALIC`;
- “**tamanho**” pode ser um número inteiro entre 8 e 24.



```
Font f = new Font("Courier",Font.Bold,12);
```

Para alterar o tipo de letra activo na applet é necessário associar as propriedades do objecto `Font` ao contexto gráfico da applet. Para isso utiliza-se o método `setFont()` da classe `Graphics`, que recebe como parâmetro um objecto do tipo `Font`. A sua sintaxe é: `g.setFont(f)`; sendo `g` o ambiente gráfico e `f` o objecto `Font`.

O código da applet:



```
import java.awt.Graphics;
import java.awt.Font;

public class Fontes extends javax.swing.JApplet {
    public void paint(Graphics g) {
        Font f1 = new Font("TimesRoman", Font.PLAIN, 12);
        Font f2 = new Font("SansSerif", Font.ITALIC, 14);
        Font f3 = new Font("Monospaced", Font.PLAIN, 16);
        Font f4 = new Font("Courier", Font.BOLD, 18);
        Font f5 = new Font("Arial", Font.BOLD, 20);

        g.setFont(f1);
        g.drawString("TimesRoman, Plain, 12", 10, 20);

        g.setFont(f2);
        g.drawString("SansSerif, Italic, 14", 10, 40);

        g.setFont(f3);
        g.drawString("Monospaced, Plain, 16", 10, 70);

        g.setFont(f4);
        g.drawString("Courier, Bold, 18", 10, 110);

        g.setFont(f5);
        g.drawString("Arial, Bold, 20", 10, 150);
    }
}
```

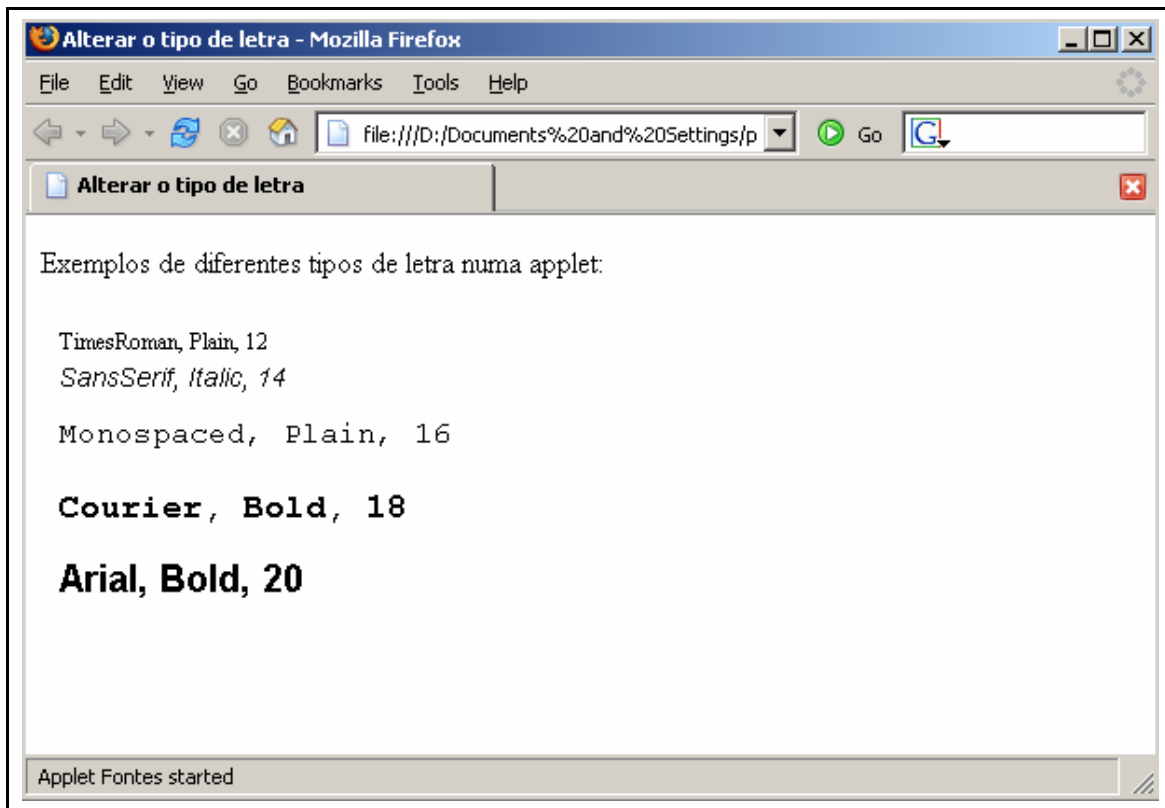
O código HTML:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Alterar o tipo de letra</title>
    <!-- fontes.html -->
</head>
<body>
    <p> Exemplos de diferentes tipos de letra numa applet:</p>
    <applet code="Fontes" width=230 height=200>
    </applet>
</body>
</html>
```



O output no browser:



## Sumário

---

- Ambiente de trabalho;
- *Applet*;
- Desenhar;
- Texto;
- Cor;
- Imagem;
- Som;
- Exercício.



### 5.6- Cor

## Alterar a cor

- Definir:
  - `Color c = new Color(Red, Green, Blue)`
  - `Color.YELLOW`, `Color.BLUE`, `Color.GREEN`, ..., `Color.RED`
- Aplicar no contexto gráfico (Graphics g):
  - `g.setColor(c);` ou `g.setColor(Color.RED);`
  - `g.drawString("Ola", 10, 30);`
- Aplicar no *background*:
  - `this.setBackground(c);`
  - `this.setBackground(Color.RED);`
  - `this` é o `JPanel` da `JApplet`.
- **Exercício:** Escrever com diferentes tipos de letra e cores



### 5.6.1- Alterar a cor

Nos programas Java podemos definir a cor com que se escreve e a cor de fundo. O Java utiliza o método **RGB** para gerar as cores. Uma cor é definida em função da quantidade de cada um dos seus 3 componentes básicos: vermelho (Red), verde (Green) e azul (Blue). A quantidade de cada componente é definida por um número entre 0 e 255. Zero representa ausência desse componente. 255 representa esse componente na máxima força. O número de cores diferentes que é possível representar com este sistema é de  $256 \times 256 \times 256 = 16.777.216$ .

Para definir uma cor é necessário criar um objecto da classe **Color** que pertence ao package `java.awt`.



```
Color c = new Color(R,G,B);
```

Por exemplo:



```
Color c = new Color(0,255,0); //cor verde
Color c = new Color(110,24,48); //cor acastanhada.
```

Na classe `Color` são definidas várias variáveis com os valores RGB das cores mais populares:

Cor	Nome da variável	Valor RGB
Amarelo	<code>Color.YELLOW</code>	255, 255, 0
Azul	<code>Color.BLUE</code>	0, 0, 255

Azul turquesa	Color.CYAN	0,255,255
Branco	Color.WHITE	255,255,255
Cinzentos	Color.GRAY	128,128,128
Cinzentos claro	Color.LIGHTGRAY	192,192,192
Cinzentos escuro	Color.DARKGRAY	64,64,64
Cor de laranja	Color.ORANGE	255,200,0
Cor de rosa	Color.PINK	255,175,175
Violeta	Color.MAGENTA	255,0,255
Preto	Color.BLACK	0,0,0
Verde	Color.GREEN	0,255,0
Vermelho	Color.RED	255,0,0

### 5.6.2- Aplicar no contexto gráfico

Para alterar a cor activa usa-se o método `setColor()`, da classe `Graphics`:



```
Color c=new Color(0,255,0);
g.setColor(c)
```

Usando as variáveis anteriores podemos alterar a cor activa com uma única instrução:



```
g.setColor(Color.GREEN);
```

### 5.6.3- Aplicar no background

A cor de fundo também pode ser alterada recorrendo ao método `setBackground()` da classe `Component`. Este método tem que ser aplicado no `JPanel` incluído na applet e não no contexto gráfico. Note-se que a classe `Graphics` não possui esse método.



```
this.setBackground(Color.CYAN);
```

### 5.6.4- Exemplo

O exemplo a seguir cria uma applet que escreve texto usando diferentes tipos de letra e cor.



```
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;

public class Cores extends javax.swing.JApplet {
    public void paint(Graphics g) {
        //alterar cor de fundo do panel
        this.setBackground(Color.CYAN);

        Font f1 = new Font("TimesRoman", Font.PLAIN, 12);
        Font f2 = new Font("SansSerif", Font.ITALIC, 14);
        Font f3 = new Font("Monospaced", Font.PLAIN, 16);
        Font f4 = new Font("Courier", Font.BOLD, 18);
        Font f5 = new Font("Arial", Font.BOLD, 20);
```

```
g.setColor(Color.RED);
g.setFont(f1);
g.drawString("Times Roman, Plain, 12", 10, 20);


g.setColor(Color.MAGENTA);
g.setFont(f2);
g.drawString("Sans Serif, Italic, 14", 10, 40);

g.setColor(Color.BLUE);
g.setFont(f3);
g.drawString("Monospaced, Plain, 16", 10, 70);

Color c = new Color(255, 200, 0);
g.setColor(c);
g.setFont(f4);
g.drawString("Courier, Bold, 18", 10, 110);

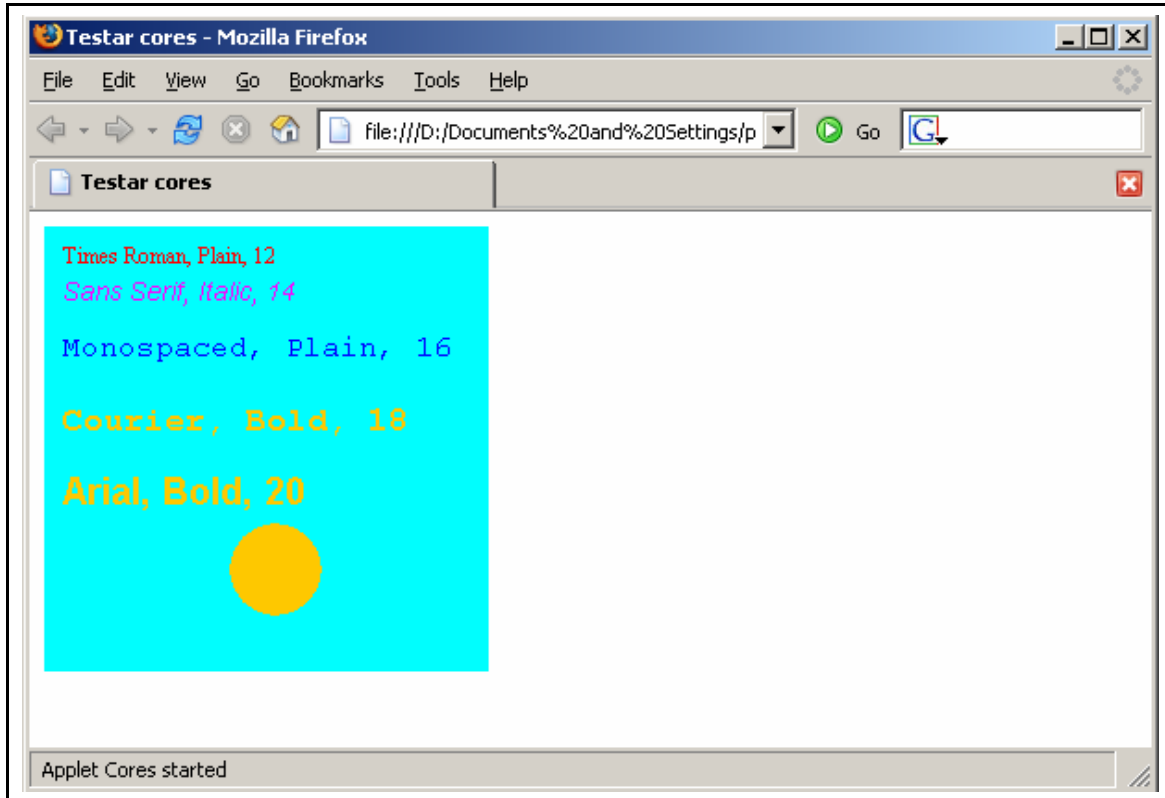
g.fillOval(100, 160, 50, 50);
g.setFont(f5);
g.drawString("Arial, Bold, 20", 10, 150);
    }
}
```

O código HTML:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Testar cores</title>
  <!-- cores.html -->
</head>
<body>
  <applet code="Cores" width=240 height=240>
    </applet>
</body>
</html>
```

O output no browser:



## Sumário

---

- Ambiente de trabalho;
- *Applet*;
- Desenhar;
- Texto;
- Cor;
- Imagem;
- Som;
- Exercício.



### 5.7- Imagem

## Desenhar uma imagem

- Ler a imagem:
  - No init() da applet: **Image i = getImage(argumento);**
- Argumento pode ser:
  - new URL("endereço internet"),
  - getDocumentBase(), "nomeFicheiro"
  - getCodeBase(), "nomeFicheiro"
- No paint():
  - Graphics g; g.drawImage(i, 20, 10, **this**);
  - **this** - Interface ImageObserver - faz *rendering*
- **Exercício: Desenhar imagem.**



### 5.7.1- Desenhar uma imagem

Uma applet Java pode colocar no écran uma imagem armazenada num ficheiro com formato GIF ou JPG. Consegue-se fazer animação quando se colocam no écran séries de imagens, que vistas em sequência, com um intervalo de tempo adequado, dão a ideia de movimento.

#### 5.7.1.1- Ler a imagem

Uma imagem é armazenada num objecto da classe Image, pertencente ao package java.awt. Para atribuir a imagem ao objecto chama-se o método getImage() no init() da applet. As versões mais usadas do getImage() são comparadas na tabela seguinte:

<code>Image i=getImage(new URL("endereço internet"));</code>	Vai buscar a imagem utilizando o respectivo endereço URL (Universal Resource Locator)
<code>Image i=getImage(getDocumentBase(),"nomeFicheiro");</code>	Vai buscar a imagem à mesma directoria onde está a página html
<code>Image i=getImage(getCodeBase(),"nomeFicheiro");</code>	Vai buscar a imagem à directoria onde está a applet (*.class)

A utilização dos métodos getCodeBase() e getDocumentBase() evita a colocação de um URL (Universal Resource Locator), que funciona como um endereço absoluto. Desta forma,




se os ficheiros forem movidos para outra directoria, a applet continua a funcionar, pois as referências são relativas.

### 5.7.1.2- Desenhar

Para desenhar a imagem na área da applet usa-se o método `drawImage()` pertencente à classe `Graphics`, que recebe como parâmetro um objecto da classe `Image`.

### 5.7.2- Exemplo

O exemplo a seguir cria uma applet que lê uma imagem com formato "gif" e a coloca no ecrã. A imagem usada é o logótipo da Sun para o Java.




```
import java.awt.Graphics;
import java.awt.Image;

public class InsImage extends javax.swing.JApplet {
    Image i;

    public void init() {
        i = getImage(getDocumentBase(), "chavena.gif");
    }

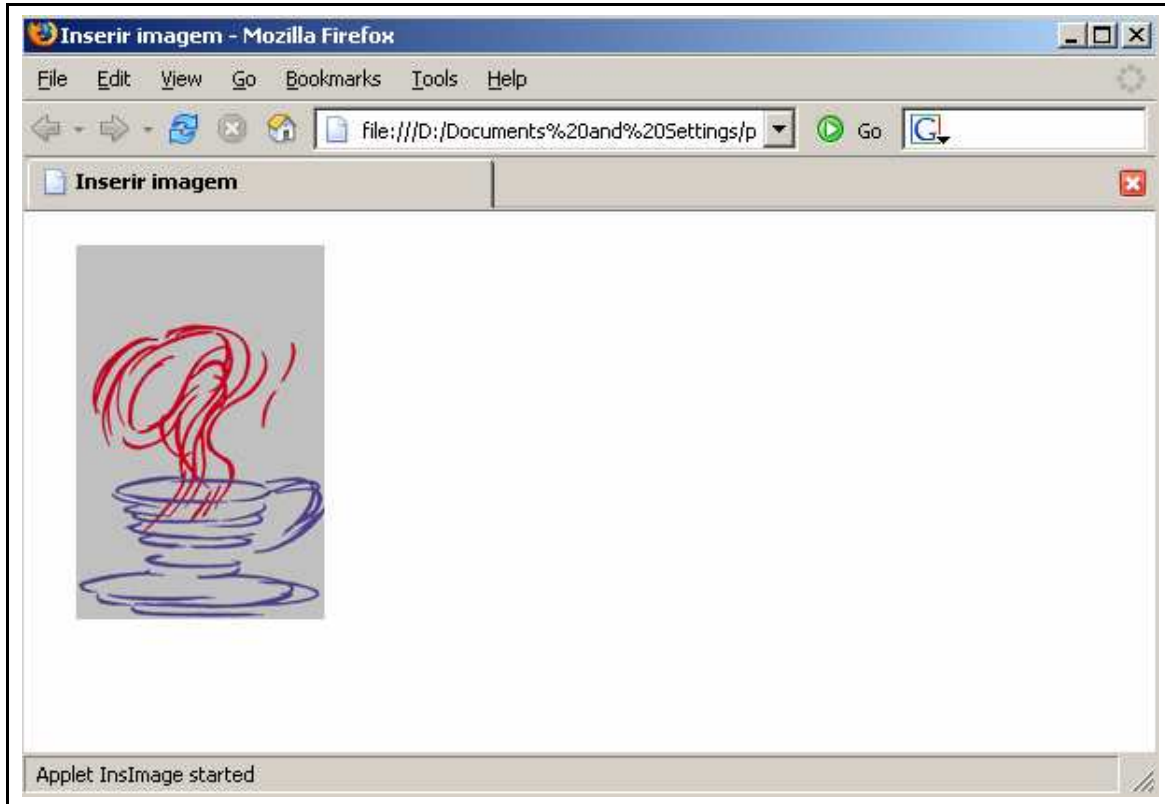
    public void paint(Graphics g) {
        g.drawImage(i, 20, 10, this);
    }
}
```

O código HTML:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Inserir imagem</title>
    <!-- insimagem.html -->
</head>
<body>
    <applet code="InsImage" width=240 height=240>
    </applet>
</body>
</html>
```

O output no browser:



O último parâmetro do método `drawImage()` é um objecto de uma classe que implementa a interface `ImageObserver`. Esta classe faz "*rendering*" da imagem em background antes de a colocar no ecrã.

## Sumário

---

- Ambiente de trabalho;
- *Applet*;
- Desenhar;
- Texto;
- Cor;
- Imagem;
- Som;
- Exercício.



### 5.8- Som

## Tocar som

- Carregar o som no `init()` da *applet*:
  - `AudioClip som = getAudioClip(argumento);`
  - Argumento:
    - `new URL("endereço internet");`
    - `getDocumentBase(), "nomeFicheiro";`
    - `getCodeBase(), "nomeFicheiro";`
- Tocar:
  - `play();`
  - `loop();`
  - `stop();`
- **Exercício:** Tocar o som do comboio.



### 5.8.1- Tocar Som

#### 5.8.1.1- Carregar

O Java consegue ler ficheiros de som no formato “**au**” e armazená-los na classe `AudioClip`, que pertence ao package `java.applet`. Os métodos mais usados para carregar um som são:

<code>getAudioClip(new URL("endereço WWW do ficheiro"));</code>	Vai buscar o ficheiro de som utilizando o respectivo endereço URL (Universal Resource Locator)
<code>getAudioClip(getCodeBase(), "nomeFicheiroSom");</code>	Vai buscar o ficheiro de som à mesma directoria onde está a applet (*.class)
<code>getAudioClip(getDocumentBase(), "nomeFicheiroSom");</code>	Vai buscar o ficheiro de som à directoria onde está a página html


#### 5.8.1.2- Tocar

A classe `AudioClip` possui um conjunto de métodos que dão ao programador controle sobre a reprodução do ficheiro de som:

- **play()** → permite tocar um ficheiro de som importado para um objecto da classe `AudioClip`;
- **loop()** → toca um ficheiro de som em ciclo. Só pára quando for executado o `stop()`;
- **stop()** → pára a reprodução de um ficheiro de som;

### 5.8.2- Exemplo

A applet do exemplo a seguir cria um objecto da classe AudioClip usando o ficheiro de som "comboio.au". O som é reproduzido uma vez com o método play() e em seguida é colocado em loop até à execução do método stop(), que é executado quando ocorre o stop() da applet.



```
import java.applet.AudioClip;
import java.awt.Graphics;


public class Som2 extends javax.swing.JApplet {
    AudioClip som;

    public void start() {
        som = getAudioClip(getCodeBase(), "comboio.au");
        //som.play(); /*toca uma vez*/
        som.loop(); /*toca em loop*/
    }

    public void paint(Graphics g) {
        g.drawString("Tocando 'comboio.au' ...", 10, 30);
    }

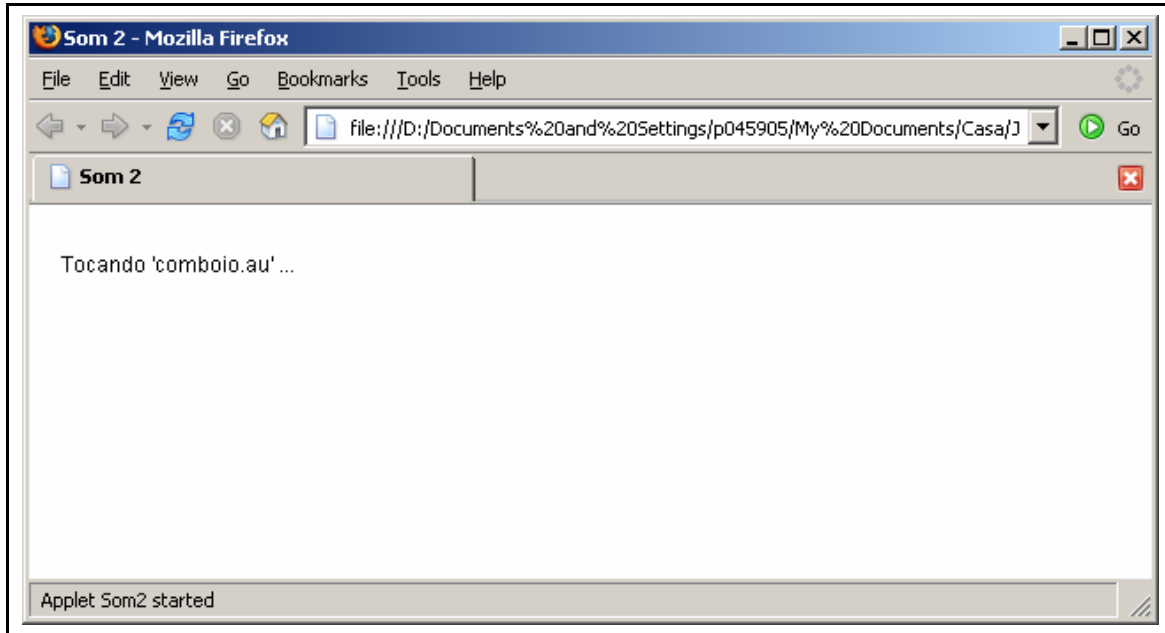
    public void stop() {
        som.stop();
    }
}
```

O código HTML:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Som 2</title>
    <!-- som1.html -->
</head>
<body>
    <applet code="Som2" width=150 height=100>
    </applet>
</body>
</html>
```

O output no browser:



## Sumário

---

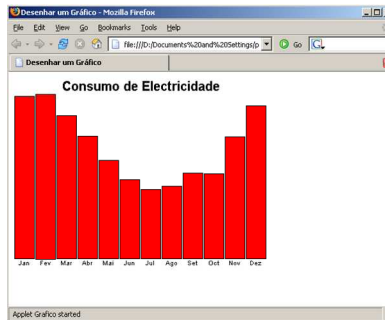
- Ambiente de trabalho;
- *Applet*;
- Desenhar;
- Texto;
- Cor;
- Imagem;
- Som;
- Exercício.



### 5.9- Exercício

## Exercício – Gráfico de barras

- Escrever uma *applet* que desenha um gráfico de barras, recebendo os seguintes parâmetros do HTML:
  - Title: Título do gráfico, que aparece em cima;
  - Values: Número de valores;
  - name\_n: Nome do valor n que será colocado no eixo dos X;
  - value\_n: Valor do parâmetro n que será colocado no eixo dos Y;



Java Web  
© Citeforma

Capítulo 5 - Applets

27

### 5.9.1- Exercício: Gráfico de barras

Escrever uma applet que permita criar um gráfico de barras cujos parâmetros vêm da página HTML. Estes devem ter o seguinte sequência e significado:

Nome do parâmetro	Tipo de dados	Significado
Title	String	Título do gráfico, que aparece em cima
Values	int	Número de valores
name_n	String	Nome do valor n que será colocado no eixo dos XXs. N está compreendido entre 1 e o valor do parâmetro values
value_n	Number	Valor do parâmetro n que será colocado no eixo dos YYs. N está compreendido entre 1 e o valor do parâmetro values

Segue o código da applet:



```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
```



```

/*Versão inicial feita por por Pedro Nunes*/
/*Versão revista por Jose Aser*/
public class Grafico extends javax.swing.JApplet {

    String title;
    double[] values;
    String[] names;
    String v;
    double minValue;
    double maxValue;
    int appletWidth;
    int appletHeight;
    int barWidth;
    Font titleFont;
    Font labelFont;

    @Override
    public void init() {
        // Ler os paramatros a partir da pagina de HTML
        v = getParameter("values");
        if (v == null) {
            System.out.println("O numero de parametros é nulo. A applet abortou");
            return;
        } else {
            int n = Integer.valueOf(v).intValue();
            System.out.println("Numero de valores=" + v);
            values = new double[n];
            names = new String[n];
            title = getParameter("title");
            if (title == null || title.equals("")) {
                title = "Sem nome";
            }
            //System.out.println("Title=" + title);

            //ler os parametros e determinar o maximo e minimo
            minValue = Double.MAX_VALUE;
            maxValue = 0;
            for (int i = 0; i < n; i++) {
                values[i] = Double.valueOf(getParameter("value_" + (i +
1))).doubleValue();
                //converter para numero positivo
                if (values[i] < 0) {
                    values[i] = -1 * values[i];
                }
                //determinar o minimo
                if (minValue > values[i]) {
                    minValue = values[i];
                }
                //determinar o maximo
                if (maxValue < values[i]) {
                    maxValue = values[i];
                }
                System.out.println("Values [" + i + "] = " + values[i]);
                //ler o nome do parametro
                names[i] = getParameter("name_" + (i + 1));
                //System.out.println("Names [" + i + "] = " + names[i]);
            }
            //System.out.println("maxValue = " + maxValue);
            //System.out.println("minValue = " + minValue);

            Dimension d = this.getSize();
            appletWidth = d.width;
            appletHeight = d.height;
            barWidth = appletWidth / values.length;
            //System.out.println("appletWidth=" + appletWidth);
            //System.out.println("appletWidth=" + appletHeight);

```

```

        titleFont = new Font("SansSerif", Font.BOLD, 15);
        labelFont = new Font("SansSerif", Font.PLAIN, 10);

    }

@Override
public void paint(Graphics g) {
    //System.out.println("Values (length) = " + values.length);
    if (values.length == 0) {
        System.out.println("O array de valores esta vazio");
        return;
    }

    //Limpar a area de desenho
    g.setColor(Color.WHITE);
    g.fillRect(0, 0, appletWidth, appletHeight);

    //escrever title
    g.setColor(Color.BLACK);
    FontMetrics titleFontMetrics = g.getFontMetrics(titleFont);
    int titleDraw = titleFontMetrics.getAscent() + 3;
    int titleWidth = titleFontMetrics.stringWidth(title);
    //se a dimensão do titulo não couber no espaco disponivel escrevemos #
    if (titleWidth > appletWidth) {
        title="#";
        titleWidth = titleFontMetrics.stringWidth(title);
    }
    int x = (appletWidth - titleWidth) / 2;
    g.setFont(titleFont);
    g.drawString(title, x, titleDraw);

    //determinar a dimensão da area de desenho.
    //Em cima fica um rectangulo para titulo
    //Em baixo fica rectangulo para labels
    FontMetrics labelFontMetrics = g.getFontMetrics(labelFont);
    int top = titleDraw + titleFontMetrics.getDescent() + 3;
    int botton = appletHeight - labelFontMetrics.getAscent() -
titleFontMetrics.getDescent() - 3 - 3;
    int labelDraw = appletHeight - labelFontMetrics.getDescent() - 3 ;
    double scale = (botton - top) / maxValue;
    //System.out.println("scale="+scale);

    g.setFont(labelFont);
    for (int k = 0; k < values.length; k++) {
        int x1 = k * barWidth + 1;
        int height = (int) (values[k] * scale);
        System.out.println("height=" + height);
        g.setColor(Color.RED);
        g.fillRect(x1, (botton-height), barWidth - 2, height);
        g.setColor(Color.BLACK);
        g.drawRect(x1, (botton-height), barWidth - 2, height);
        int labelWidth = labelFontMetrics.stringWidth(names[k]);
        //se a label for mais comprida que o espaco disponivel escreve #
        if (labelWidth > barWidth) {
            names[k] = "#";
            labelWidth = labelFontMetrics.stringWidth("#");
        }
        x = k * barWidth + (barWidth - labelWidth) / 2;
        g.drawString(names[k], x, labelDraw);
    }
}
}

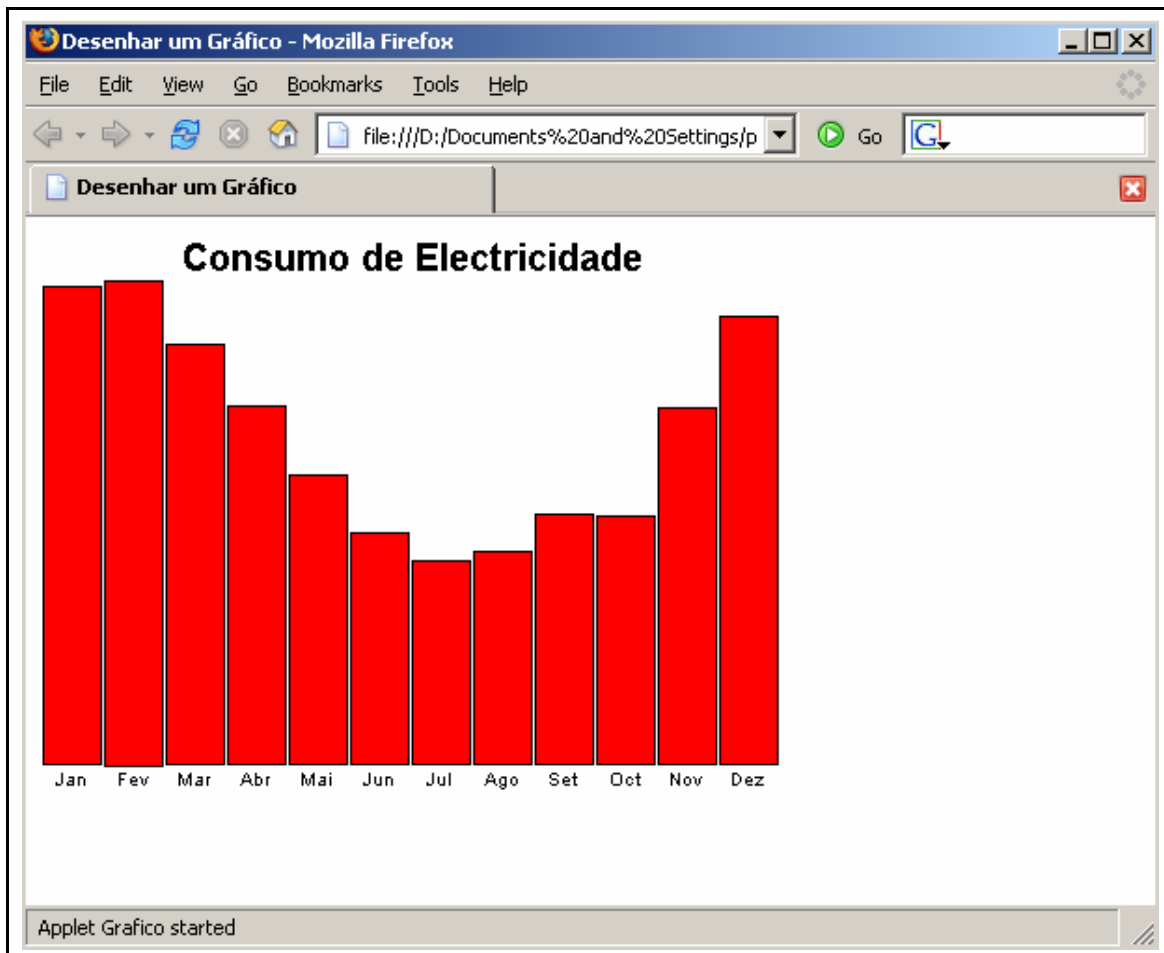
```

O código HTML:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Desenhar um Grafico</TITLE>
  <!-- File: Grafico.html -->
</HEAD>
<BODY>
  <APPLET CODE="Grafico" WIDTH=400 HEIGHT=300>
    <PARAM NAME="title" VALUE="Consumo de Electricidade">
    <PARAM NAME="values" VALUE="12">
    <PARAM NAME="name_1" VALUE="Jan">
    <PARAM NAME="name_2" VALUE="Fev">
    <PARAM NAME="name_3" VALUE="Mar">
    <PARAM NAME="name_4" VALUE="Abr">
    <PARAM NAME="name_5" VALUE="Mai">
    <PARAM NAME="name_6" VALUE="Jun">
    <PARAM NAME="name_7" VALUE="Jul">
    <PARAM NAME="name_8" VALUE="Ago">
    <PARAM NAME="name_9" VALUE="Set">
    <PARAM NAME="name_10" VALUE="Oct">
    <PARAM NAME="name_11" VALUE="Nov">
    <PARAM NAME="name_12" VALUE="Dez">
    <PARAM NAME="value_1" VALUE="368">
    <PARAM NAME="value_2" VALUE="373">
    <PARAM NAME="value_3" VALUE="324">
    <PARAM NAME="value_4" VALUE="276">
    <PARAM NAME="value_5" VALUE="223">
    <PARAM NAME="value_6" VALUE="178">
    <PARAM NAME="value_7" VALUE="157">
    <PARAM NAME="value_8" VALUE="164">
    <PARAM NAME="value_9" VALUE="193">
    <PARAM NAME="value_10" VALUE="191">
    <PARAM NAME="value_11" VALUE="275">
    <PARAM NAME="value_12" VALUE="345">
  </APPLET>
</BODY>
</HTML>
```

O resultado da execução:



## Sumário

---

- Ambiente de trabalho;
- *Applet*;
- Desenhar;
- Texto;
- Cor;
- Imagem;
- Som;
- Exercício.

