JAVA 8

Aplicações, Applets

NESTE MÓDULO É APRESENTADA A ARQUITETURA DOS APPLETS, os métodos de pintura, classes para imagens, sons e gráficos e como usar e criar applets.

Tópicos abordados neste módulo

- Applets
- Métodos e objetos para desenho gráfico

Índice

Objetivos

No final deste módulo você deverá ser capaz de:

- Construir um applet ou converter uma aplicação existente em um applet

- Saber identificar as limitações dos applets
- Usar de forma correta os métodos de pintura e o AWT thread
- Usar um clipe de áudio e importar uma imagem para um applet
- Fazer desenhos em um applet ou aplicação da AWT

8.1. Applets e Gráficos

APPLETS SÃO APLICAÇÕES EM JAVA que são iniciadas a partir de uma página Web. Geralmente usam o contexto gráfico da página para exibir e fazer alguma coisa, mas podem também abrir janelas e escrever na saída padrão (o "Java Console" dos browsers), embora não seja usual.

Applets também têm mais restrições que aplicações por razões de segurança. A maioria dos applets transferidos através da rede não pode:

- Ler, escrever, renomear, apagar, executar ou criar arquivos e diretórios em sistema local.
- Obter informações de arquivos locais como listar conteúdo de diretórios, verificar a existência de um arquivo, saber o tamanho, nome ou data de modificação de um arquivo.
- Criar uma conexão de rede para qualquer máquina a não ser aquela de onde veio a applet.
- Escutar ou aceitar conexões de rede em qualquer porta do sistema local.
- Obter quaisquer informações sobre o usuário que está logado, como userid, etc.
- Definir propriedades do sistema.
- Abortar o interpretador Java (`System.exit()` é ilegal em Applets).
- Usar classes que não pertençam aos oito pacotes da biblioteca padrão Java

Apesar de todas essas restrições, ainda há muito que se pode fazer com applets. Na versão 1.1 de Java, a Sun introduziu um modelo mais flexível de segurança baseado em certificados.

8.2. Como incluir uma applet numa página Web

As páginas que formam a Web são escritas em uma linguagem de descrição de texto chamada HTML. A linguagem HTML consiste de descritores que marcam o texto especificando a forma como o texto será formatado em um browser e marcando a posição onde serão incluídos objetos, como figuras, animações e conteúdo executável.

Descritor <applet>

O descritor `<applet>` é usado pelo HTML 3.2 como forma de incluir conteúdo executável em uma página. A applet aparecerá na parte da página onde for incluído o descritor. Existem atributos para alinhamento e redimensionamento que são idênticos aos utilizados para incluir imagens, através do descritor `<img>`.

A sintaxe mínima do descritor `<applet>` é a seguinte:

```
<applet code="nome_do_arquivo_class"
        height="altura_em_pixels"
        width="largura_em_pixels">
...
[parâmetros e opções para browsers incompatíveis]
...
</applet>
```

Os três atributos `code`, `height` e `width` são os únicos obrigatórios. Se a applet não estiver acessível através de um caminho de subdiretórios em relação ao documento que o referencia, é necessário especificar ainda um argumento `codebase="URL_do_diretório_base"` que contém a URL do lugar onde a applet se encontra. Outro atributo opcional é `name="nome_da_applet"` que rotula a applet com um nome.

Os outros atributos do descritor `<applet>`: `alt`, `align`, `vspace` e `hspace` são os mesmos do descritor HTML `<img>`.

Os parâmetros da applet são passados através do descritor `<param>`, que deve ficar entre `<applet ...>` e `</applet>` (ambos os descritores `<applet>` e `</applet>` devem ser usados, mesmo que não haja parâmetros). `<param>` não tem descritor de fechamento. Recebe dois atributos obrigatórios:

```
<param name="parâmetro" value="valor">
```


`name="parâmetro"` contém o nome de um parâmetro definido pelo programador da applet. `value="valor"` contém um valor que pode ser mudado pelo autor da página Web, mesmo que ele não tenha acesso à fonte da applet.

Qualquer coisa colocada entre `<applet>` e `</applet>`, que não esteja dentro de `<param>` é ignorada por browsers que suportam Java, mas aparecem nos que não a suportam, portanto, é um bom lugar para colocar imagens alternativas, links ou qualquer outra informação que informe à pessoa que lê a página, que ali deveria haver uma applet.

Incluíndo uma applet na página

Para incluir uma applet em uma página, não é necessário saber nada de Java, só um mínimo de HTML. A applet pode ser incluída em qualquer lugar onde possa ser incluída uma imagem.

Na Web há várias localidades que distribuem applets para uso público. Para incluí-los é só chamar a classe principal no descritor `<applet>` e os parâmetros apropriados. Se a applet tiver parâmetros, em geral acompanha alguma informação de como usá-los.

Por exemplo, para incluir a applet `ImageTape` da Sun (vem junto com o JDK. Procure no diretório `java/demo/`) que faz com que um conjunto de imagens role na tela você precisa fazer o seguinte:

- Copie o arquivo `ImageTape.class` para o mesmo diretório onde está sua página HTML que irá chamá-lo.
- Renomeie as imagens da sua animação para `T1.gif`, `T2.gif`, `T3.gif`, etc. (isto é uma convenção do programador da applet). Vamos supor que você tem três imagens GIF de 150x85 pixels de dinossauros: `T1.gif`, `T2.gif` e `T3.gif`.

Crie um diretório onde ficarão as imagens (outra convenção do programador). Vamos chamá-lo de `sauros`.

- Escolha um lugar na sua página Web para colocá-lo, edite a página e acrescente o seguinte código:

```
<applet code="ImageTape.class" width=450 height=85>
  <param name=img value="sauros">
  <param name=nimgs value="3">
  <param name=speed value="10">
</applet>
```


O parâmetro `img` informa o nome do diretório onde estão as imagens `T1`, `T2`, etc. `nimgs` informa quantas são e `speed` diz a velocidade da animação. O tamanho da applet calculamos baseado na colocação das três imagens de 150 pixels lado a lado.

8.3. A classe Applet

O conjunto de classes, interfaces e métodos que compõem o pacote `java.applet` é pequeno. Consiste de uma única classe: Applet, e três interfaces: `AppletContext`, `AppletStub` e `AudioClip`. Applet é uma classe especializada da classe `Panel` que faz parte do pacote `java.awt`.

Applets são subclasses da classe `Applet`. Todo programa que roda como applet, tem na sua declaração "`extends Applet`":

```
public class Nome_da_Classe extends Applet
```

Após estender a classe applet, a nova subclasse deve sobrepor alguns de seus métodos. Nem todos precisam ser definidos explicitamente pelo programador. Em geral, o browser cuida da maioria dos detalhes.

Exemplos **[Java In A NutShell. 2nd. Ed.]**

Os dois exemplos a seguir ambos produzem um programa semelhante, onde se pode rabiscar.

```
// This example is from the book "Java in a Nutshell, Second Edition".
// Written by David Flanagan.  Copyright (c) 1997 O'Reilly & Associates.
// You may distribute this source code for non-commercial purposes only.
// You may study, modify, and use this example for any purpose, as long as
// this notice is retained.  Note that this example is provided "as is",
// WITHOUT WARRANTY of any kind either expressed or implied.

import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class Scribble2 extends Applet
                       implements MouseListener,  MouseMotionListener {
  private int last_x, last_y;

  public void init() {
    // Tell this applet what MouseListener and MouseMotionListener
    // objects to notify when mouse and mouse motion events occur.
```



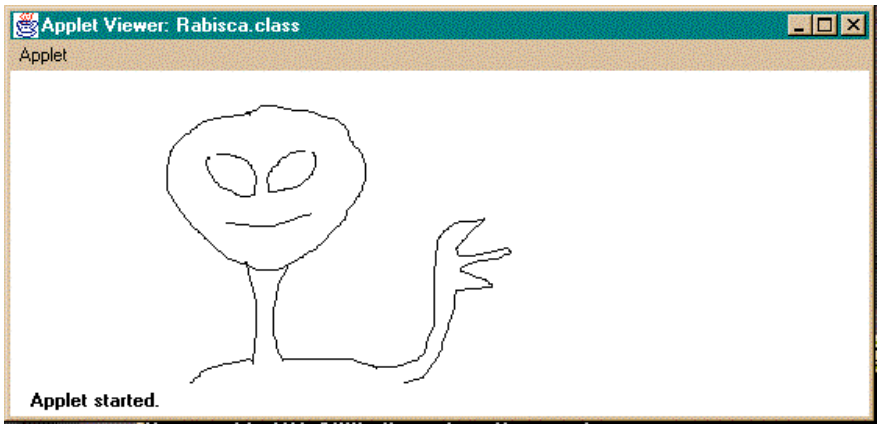
```
    // Since we implement the interfaces ourself, our own methods are called.
    this.addMouseListener(this);
    this.addMouseMotionListener(this);
  }

  // A method from the MouseListener interface.  Invoked when the
  // user presses a mouse button.
  public void mousePressed(MouseEvent e) {
    last_x = e.getX();
    last_y = e.getY();
  }

  // A method from the MouseMotionListener interface.  Invoked when the
  // user drags the mouse with a button pressed.
  public void mouseDragged(MouseEvent e) {
    Graphics g = this.getGraphics();
    int x = e.getX(), y = e.getY();
    g.drawLine(last_x, last_y, x, y);
    last_x = x; last_y = y;
  }

  // The other, unused methods of the MouseListener interface.
  public void mouseReleased(MouseEvent e) {;}
  public void mouseClicked(MouseEvent e) {;}
  public void mouseEntered(MouseEvent e) {;}
  public void mouseExited(MouseEvent e) {;}

  // The other method of the MouseMotionListener interface.
  public void mouseMoved(MouseEvent e) {;}
}
```

Este outro applet faz a mesma coisa que o exemplo anterior mas usando classes internas.[1]:

```
// This example is from the book "Java in a Nutshell, Second Edition".
// Written by David Flanagan.  Copyright (c) 1997 O'Reilly & Associates.
// You may distribute this source code for non-commercial purposes only.
// You may study, modify, and use this example for any purpose, as long as
// this notice is retained.  Note that this example is provided "as is",
// WITHOUT WARRANTY of any kind either expressed or implied.

import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class Scribble3 extends Applet {
  int last_x, last_y;

  public void init() {
    // Define, instantiate and register a MouseListener object.
    this.addMouseListener(new MouseAdapter() {
      public void mousePressed(MouseEvent e) {
```



```
        last_x = e.getX();
        last_y = e.getY();
      }
    });

    // Define, instantiate and register a MouseMotionListener object.
    this.addMouseMotionListener(new MouseMotionAdapter() {
      public void mouseDragged(MouseEvent e) {
        Graphics g = getGraphics();
        int x = e.getX(), y = e.getY();
        g.setColor(Color.black);
        g.drawLine(last_x, last_y, x, y);
        last_x = x; last_y = y;
      }
    });

    // Create a clear button
    Button b = new Button("Clear");
    // Define, instantiate, and register a listener to handle button presses
    b.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {  // clear the scribble
        Graphics g = getGraphics();
        g.setColor(getBackground());
        g.fillRect(0, 0, getSize().width, getSize().height);
      }
    });
    // And add the button to the applet
    this.add(b);
  }
}
```



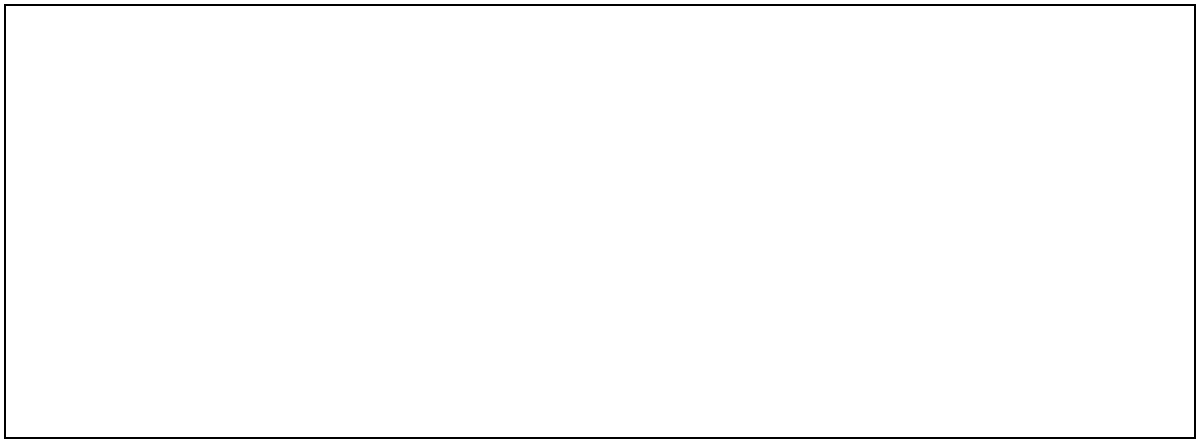
Métodos do ciclo de vida de um Applet

Os principais métodos de qualquer Applet são os que caracterizam o seu ciclo de vida. Applets rodam como `Threads`, e portanto, têm métodos parecidos com os de `Thread`. Os métodos abaixo são chamados automaticamente pelo browser. O programador do applet deve sobrepô-los se desejar que algum deles se comporte

de forma diferente (além desses, o método Component.`paint()` é chamado pelo sistema para desenhar o applet, ou qualquer um dos seus componentes).

- `public void init()`: este método realiza qualquer inicialização necessária na applet.
- `public void start()`: É o método que inicia a execução da applet. Geralmente é usado para iniciar um objeto `Thread` que roda o código principal da applet.
- `public void stop()`: Interrompe a execução da applet.
- `public void destroy()`: libera quaisquer recursos de memória que a applet esteja usando.

A figura abaixo ilustra o que ocorre durante o ciclo de vida de uma applet ao ser chamado pelo browser, visualizado e descartado. Qualquer um dos métodos pode ser sobreposto (de "Hooked on Java", Arthur Van Hoff, et Al, Addison Wesley, 1996).



Métodos de pintura

A pintura e re-pintura da tela são coisas que uma applet (ou qualquer aplicação AWT) típica faz várias vezes durante a sua execução. O método responsável por isso é o método `paint()`, também herdado por `java.awt.Component`. O sistema de execução Java se encarrega de chamar este método quando precisa. Nós, nos encarregamos de definir o que ele vai fazer.

A sua sintaxe básica é

```
public void paint(Graphics g)
```

`paint()` recebe como argumento um objeto do tipo `Graphics`, que é uma classe abstrata que define uma interface independente de plataforma para gráficos

(um contexto gráfico – área da tela do browser reservada para a execução do applet). Depois de criado um objeto `Graphics`, ele pode ser usado para alterar várias características de uma applet na tela. O contexto gráfico é fornecido pelo browser que define seu tamanho através de HTML. O método `paint()` o recebe e através dele pode desenhar na área do browser.

`paint()` é chamado automaticamente pelo sistema sempre que for necessário repintar a tela. Você nunca deve chamá-lo diretamente pois ele é chamado usando um *thread* diferente (o thread da AWT). Chama-lo no thread principal de sua aplicação poderá causar efeitos estranhos e atrapalhará a sua chamada normal, sempre que a tela for encoberta. O método `repaint()`, também disponível em qualquer componente, agenda uma chamada de `paint()` no thread do AWT através do método `update()`, que faz duas coisas: a) limpa a tela e; b) chama `paint()` para redesenhá-la. Os métodos `paint()` e `update()` rodam no mesmo thread (portanto não devem ser chamados diretamente). Se você quiser chamar `paint()` sem limpar a tela (se não quiser que a tela seja redesenhada), sobreponha `update()` da forma:

```
public void update(Graphics g) {
    paint(g);
}
```

Agora o método `repaint()` não mais limpará a tela antes de chamar `paint()`.

O seguinte exemplo (do "Java in a NutShell", mostra a utilização de `paint()` e objetos gráficos, para desenhar, mudar a cor e a fonte do texto.

```
// This example is from the book _Java in a Nutshell_ by David Flanagan.
// Written by David Flanagan.  Copyright (c) 1996 O'Reilly & Associates.
// You may study, use, modify, and distribute this example for any purpose.
// This example is provided WITHOUT WARRANTY either expressed or implied.

import java.applet.*;
import java.awt.*;


public class SecondApplet extends Applet {
    static final String message = "Java Colors";
    private Font font;

    // One-time initialization for the applet
    public void init() {
        font = new Font("Helvetica", Font.BOLD, 48);
```



```
    }

    // Draw the applet whenever necessary.  Do some fancy graphics.
    public void paint(Graphics g) {
        // The pink oval
        g.setColor(Color.pink);
        g.fillOval(10, 10, 330, 100);

        // The red outline.  java doesn't support wide lines, so we
        // try to simulate a 4-pixel wide line by drawing four ovals
        g.setColor(Color.red);
        g.drawOval(10,10, 330, 100);
        g.drawOval(9, 9, 332, 102);
        g.drawOval(8, 8, 334, 104);
        g.drawOval(7, 7, 336, 106);

        // The text
        g.setColor(Color.black);
        g.setFont(font);
        g.drawString(message, 40, 75);
    }
}
```

Métodos de Imagem e Som

O método

```
public Image getImage(URL url)
```

é usado pela `Applet` para recuperar uma imagem a partir de uma URL. Os métodos

```
public void play(URL url)
public AudioClip getAudioClip(URL url)
```

são usados para incluir e executar um clip sonoro em uma página através de uma applet. Para executar o clip uma única vez, usa-se `play()` com a URL do clip, formato "`*.au`" como argumento. Para fazer coisas mais sofisticadas, como fazer o clip reiniciar e repetir várias vezes, usa-se o método `getAudioClip()` associado a um objeto do tipo `AudioClip`, que pode então ser controlado de forma mais eficiente.

O exemplo a seguir usa imagens, `AudioClips` e eventos. É uma applet que mantém uma imagem apagada na tela até que o mouse esteja sobre a applet. Neste momento, a imagem "acende" e mostra suas cores originais. Se o mouse for clicado sobre a imagem, ele buscará uma URL definida por parâmetro:

```
/** Highlight Applet version 2 (uses Image Filters)
```



```
 */

import java.applet.*;
import java.net.*;
import java.awt.image.*;
import java.awt.*;
import java.awt.event.*;


public class HighLight extends Applet implements Runnable, MouseListener {

    Image img, gray;
    URL url;
    String audio;
    Thread sndthread = null;

    public void init() {
        Dimension d = this.size();
        // busca URL da imagem;
        img = getImage(getDocumentBase(),getParameter("SRC"));

        // As três linhas a seguir criam uma cópia cinza da imagem
        ImageFilter f = new GrayFilter();
        ImageProducer producer = new FilteredImageSource(img.getSource(), f);
        gray = this.createImage(producer);

        // define tamanho da applet
        resize(d.width, d.height);

    }

    public void mouseReleased (MouseEvent e) {
        // quando mouse for liberado sobre Applet, acessa URL
        try {
            url = new URL(getDocumentBase(), getParameter("HREF"));
        } catch (MalformedURLException mue) {
            showStatus("Bad URL");
        }
        getAppletContext().showDocument(url);

    }

    public void mouseExited (MouseEvent e) {
        // quando mouse sair da applet, faça-o apagar
        repaint();
        if (sndthread == null) {
            sndthread = new Thread(this);
            sndthread.setPriority(Thread.MIN_PRIORITY);
            sndthread.start();
            audio = getParameter("OUTSND"); // arquivo de som (au)
            sndthread = null;               // opcional
        }
    }
```



```
    public void mouseEntered(MouseEvent e) {
        // when mowse enters applet, show image.
        Graphics g = this.getGraphics(); // new Graphics context
        g.drawImage(img, 0, 0, this);
        if (sndthread == null) {
            sndthread = new Thread(this);
            sndthread.setPriority(Thread.MIN_PRIORITY);
            sndthread.start();
            audio = getParameter("INSND");  // arquivo de som (au)
            sndthread = null;               // opcional
        }
    }

public void mouseClicked(MouseEvent e) {}
public void mousePressed(MouseEvent e) {}


    public void run() {
        play(getCodeBase(), audio);
    }

    public void paint(Graphics g) {
        // pinta imagem inicial de cinza
        g.drawImage(gray, 0, 0, this);
    }
}


// This class from "Java In a NutShell" by David Flanagan,
// O'Reilly and Associates. page 158
//
class GrayFilter extends RGBImageFilter {

    public GrayFilter() {
        canFilterIndexColorModel = true;
    }

    public int filterRGB(int x, int y, int rgb) {
        int a = rgb & 0xff000000;
        int r = ((rgb & 0xff0000) + 0xff0000)/2;
        int g = ((rgb & 0x00ff00) + 0x00ff00)/2;
        int b = ((rgb & 0x0000ff) + 0x0000ff)/2;
        return a | r | g | b;
    }
}
```

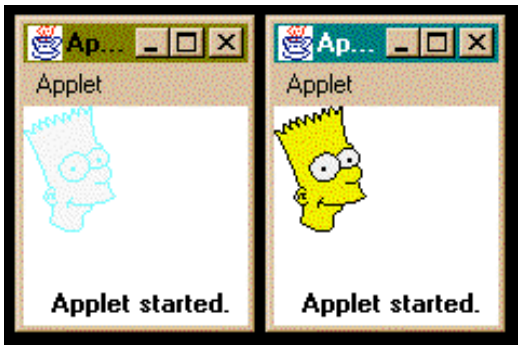
Os parâmetros usados (em `<param name="..." ...>`)são:

- `SRC` = URL da imagem a ser usada
- `HREF` = URL do link a ser acionado ao se clicar na imagem

e os opcionais:

- `INSND` = URL do arquivo de som que tocará quando o mouse entrar.
- `OUTSND` = URL do arquivo de som que tocará quando o mouse entrar.
- A classe `GrayFilter` implementa um método que "apaga" um pouco a imagem. Este método foi utilizado na applet para provocar o efeito apagado na imagem.

A figura abaixo ilustra o resultado (a segunda imagem aparece quando o mouse está sobre a applet):



Métodos para Leitura de Parâmetros HTML

Para ler parâmetros e outras informações a respeito dos arquivos HTML, a applet dispõe dos seguintes métodos:

- `public URL getDocumentBase()`: retorna a URL base do documento que contém a applet.
- `public URL getCodeBase()`: retorna a URL base da applet (do arquivo `.class`)
- `public String getParameter(String nome)`: retorna uma `String` contendo o nome de um parâmetro especificado no descritor `<param>` ou no descritor `<applet>`. Por exemplo, para ler os parâmetros da applet que faz rolar dinossauros na tela (que vimos no início deste capítulo), usaríamos:

```
String velocidade = getParameter("speed");
String imagens = getParameter("img");
String num_imagens = getParameter("nimgs");
```

8.4. Applets e componentes da AWT

Como a classe `Applet` herda toda a implementação da classe `java.awt.Component`, que é a principal classe do pacote `java.awt`, quase tudo o

que se pode fazer em relação à programação da GUI com o AWT, pode-se fazer em uma applet.

Uma exceção são os menus, que só podem ser incluídos em objetos da classe `java.awt.Frame`. `Frame` e `Panel` são duas subclasses descendentes de `Component`, mas `Panel` não descende de `Frame`, portanto Applet não herda a implementação de `Frame`.

No entanto, uma applet pode chamar um objeto `Frame` que abra como uma janela separada do browser, e nesta janela, colocar menus e tudo mais o que puder ser colocado em um `Frame`. Veja a especificação ou uma documentação específica sobre o AWT para maiores detalhes. Outra solução é usar JFC-Swing.

Exercício

1. Transforme uma das aplicações gráficas que você desenvolveu no capítulo anterior em uma Applet (não inclua os menus).
2. Escreva uma applet que desenhe um gráfico de barras a partir de informações passadas como parâmetros no HTML.

8.5. Gráficos

As outras classes do AWT que não abordamos são as que lidam com gráficos. São provavelmente as classes mais importantes já que definem os métodos de desenho de linhas, textos e pintura de imagens. Com elas é possível inclusive construir outros objetos (widgets) independentes de plataforma para a GUI (classe `java.awt.Toolkit`).

Neste curso, não entraremos nos detalhes de programação desta biblioteca gráfica, que certamente dariam um livro inteiro. Apenas apresentaremos a seguir os métodos construtores mais comuns das principais classes.

- **Color:** descreve uma cor. Esta classe contém constantes que definem cores. Para criar um objeto `Color`:

```
Color c = new Color(255, 0, 0);
```

- **Dimension:** um par de variáveis de instância que contém a largura e altura de alguma coisa. Uso típico em um applet, para capturar o seu tamanho:


```
Dimension d = this.size();
int largura = d.width;
int altura = d.height;
```

- **Font:** descreve uma fonte. Para criar um objeto `Font`:

```
Font f = new Font("Sans-Serif", Font.BOLD, 12);
```

- **Point:** armazena as coordenadas X e Y de um ponto bi-dimensional. O método `move()` define as coordenadas e `translate()` adiciona valores específicos às coordenadas. Vários métodos de outros objetos utilizam objetos do tipo `Point`.

```
Point p = new Point(pos_x, pos_y);
```

- **Polygon:** um polígono definido como uma matriz de pontos. Os pontos do polígono podem ser especificados com o método `addPoint(x, y)` ou através do construtor. O método `inside(x, y)` testa se um determinado ponto está contido no polígono.

```
Polygon py =
         new Polygon (x[], y[], num_pontos);
```

- **Rectangle:** define um retângulo pelas coordenadas do seu canto superior esquerdo e uma largura e altura, que podem ser manipulados livremente. Vários métodos de outros objetos utilizam objetos do tipo `Rectangle`. Uso típico:

```
Rectangle p = new Rectangle (x, y, larg, alt);
```

- **Graphics:** É uma classe abstrata que define vários métodos para desenho de linhas, de texto, pintura de imagens, cópia de áreas e recorte gráfico. Um objeto do tipo `Graphics` não pode ser criado diretamente. Precisa ser obtido via um método `getGraphics()` de um `Component` ou `Image`. O uso mais comum é como argumento do método `paint()` de um `Component`:

```
public void paint(Graphics g);
```

  O Java 2 (JDK1.2) aumentou bastante os recursos de `Graphics` na classe `Graphics2D`. Para usar `Graphics2D` basta fazer uma conversão, dentro do método paint, antes do uso:

```
public void paint(Graphics g) {
    Graphics g2 = (Graphics2D) g;
```



```
        g2.setColor(...) ; métodos de Graphics2D
    }
```

- **Image:** é uma classe abstrata que representa uma imagem de uma forma independente de plataforma. Deve ser obtida através de uma chamada `Applet.getImage()` ou `Component.createImage()`. Os métodos mais importantes são: `getGraphics()`, que retorna um objeto gráfico que pode ser usado para desenhar em imagens na memória; e `getSource()`, que retorna um objeto `ImageProducer` que gera a imagem.

  O trecho de código a seguir é usado para pintar uma imagem na memória (usando um objeto `Image`, para armazená-lo). A vantagem de se pintar uma imagem na memória antes de se utilizá-la, está em evitar piscadas em animações, causadas pelo redesenho da tela. Esta técnica se chama "double-buffering":

```
(...)
Dimension d = this.size(); // tam. da applet

// cria objeto Image offscreen (na memória)
Image offscreen = this.createImage(d.width, d.height);

// cria contexto grafico em offscreen
Graphics g = offscreen.getGraphics();

// chama paint() e pinta imagem em offscreen
paint(g);

// cria contexto grafico em Applet (na tela)
g = this.getGraphics();


// desenha imagem em Applet (aparece na tela)
g.drawImage(offscreen, 0, 0, this);
```

