Trabalhando com gráficos (java.awt)
Enviado por Quarta, agosto 14 @ 17:39:35 EST por **Erro! A referência de hyperlink não é válida.**

http://www.portaljava.com/home/modules.php?name=Search&query=&topic=2Tutorial criado e enviado por:
Waldeck Schützer e Sadao Massago
Departamento de Matemática
Universidade Federal de São Carlos - UFSCar

AWT AWT (Abstract Window Toolkit) é biblioteca da classe java que implementa interface do applet através de janelas. Note que, o applet pode ser implementado para ser executado fora da página web e por isso, AWT pode ser usado para implementar interface de aplicativos escritos em java. Nos veremos ainda, como desenvolver applets que podem ser executador fora da página.

Introdução ao AWT

AWT é um conjunto de recursos gráficos comumente conhecidos pelos sistemas de interface usando janelas. Nos podemos criar janelas, botões, janela de diálogos, etc.
Estas ferramentas possui uma iherarquia, que será explicado mais em diante. Por enquanto, apenas aprenderemos como usar o AWT que está baseado

Tutorial enviado por
Waldeck Schützer e Sadao Massago
Departamento de Matemática
Universidade Federal de São Carlos - UFSCar no interface orientado a enventos.
Programação orientado a eventos é uma técnica de programação na qual, costruimos o programa através de regras (métodos) a ser executados cada vez que o evento diferente ocorrer. Por exemplo, precionar teclado, mover mouse, clicar no botão, receber dados pela rede, todos são eventos. Cada vez que um desses eventos ocorrer, o método correspondente é ativado.
O elemento de AWT são na maioria, componentes. isto quer dizer que eles podem ser adicionados numa janela (frame) ou painel (panel). Para começar, precisamos de uma janela ou uma área em branco para poder colocar os botões e menus.
O applet ja possui sua área própria que é um retângulo sem nada, denominado de **panel**. Então, ele pode começar a colocar (adicionar) botões e outros componentes. Se quiser sub-dividir, basta acrescentar panel's.
Quando ocorrer qualquer evento no componente do AWT, o método denominado de **action** é chamado. O que nos precisamos fazer é implementar o método action para identificar o evento ocorrido e manipular corretamente.
Em vez de ficar discutindo monte de coisas, vamos para parte prática.
Suponhe que queremos adicionar botões no panel do applet (na parte de

escrevendo o applet, a área de trabalho de applet dentro da página HTML foi referenciado como sendo janelas, mas referência correta é painel).

Botão

Botão é uma classe denominado de Button e possui um nome. O evento do botão é cliqui do mouse sobre ele, ou seja, quando mouse é clicado sobre ele, chama o método **action**.
Ele possui três métodos principais:

```
public Button() // construtor. Cria o botão
public Button(String label) // Cria o botão, já dando nome para ele.
public void setLabel(String label) // Atribui nome para botão
public String getLabel() // Retorna o nome do botão
```

Para inserir algo no componente de AWT, utiliza o método add associado a ele. Se o add for chamado sem especificação do objeto, é considerado objeto corrente. Agora, vamos construir um applet simples que altera a cor do panel do applet, clicando sobre botão.

```
import java.awt.*;
import java.applet.*;
public class Botao extends Applet {
  public void init(){
    add(new Button("Azul"));
    add(new Button("Vermelho"));
  }
  public void boolean action(Event evento, Object quem) {
    if(evento.target instanceof Button) {
      String label = (String)quem;
      if(label == "Azul")
        setBackground(Color.blue);
      else if (label == "Vermelho")
        setBackground(Color.red));
      else
        return false;
      repaint();
      return true; // não chama mais nenhum action seguinte
    }
    else
      return false; // evento não foi de botão. Próximo action é chamado.
  }
}
```

Vamos esclarecer as coisas. Primeiro, o método init() que é executado quando o applet inicia, adiciona dois botões no panel da applet, com nome "Azul" e "Vermelho" respectivamente.
Quando o botão for clicado, ocorre o evento no botão, que não possui regras (métodos) associados. Quando o objeto não possui método para cuidar de

evento (caso do botão acima), o evento será passado para a elemento que contém ele. No caso do exemplo acima, o botão foi adicionado no panel da applet e logo, o action da applet é chamado.
Os métodos de tratamento do evento é:
public boolean action(Event evento, Object quem)
onde o parâmetro evento contém informações sobre evento e quem é o objeto na qual o evento ocorreu.
Note que, o evento de mouse e teclado discutido no escrevendo o applet é um evento específico do panel e não inclui nenhum evento tais como clicar botão, selecionar ítem, entrar com texto, etc. Todos esses eventos serão tratados pelo método action.
Primeiro, o action verifica se o evento foi gerado pelo botão. Para isso, basta verificar se o evento.target é uma instância de Button. isto é feito pelo (evento.target instanceof Button). Caso for botão, podemos tomar decisão de acordo com o botão clicado. Para saber qual dos botões foi precionado, converte o objeto quem em String. note que quem é o objeto (botão) que foi clicado. Quando converte para nome, será nome do botão que é armazenado no variável label e é verificado se é "azul" ou "vermelho". Em seguida, muda as cores do fundo. A classe Color possui várias definições de cores usuais, assim como permite criar cores a partir do modelo RGB ou HSI.
Agora, note que retorna false quando o evento não é do botão ou não for botão específico. Quando o método que cuida do evento retornar true, o AWT assume que o evento foi devidamente tratado. quando o false é retornado, o método de tratamento do evento (mouse, teclado ou action) da classe que adicionou ele será chamado. Por exemplo, se o Applet fizer parte de algum outro componente (alguma janela, por exemplo), se action do applet retornar false, o action da janela é chamado. se o action do applet retornar true, o método da janela não será chamado. isto fica mais evidênte na medida que for acostumando com o interface através de AWT e começar adicionar componentes um dentro do outro.

Rótulo:

Um rótulo é denominado de label. A utilidade dele é apresentar textos dentro do panel ou frame.
A classe Label possui três construtores:
public Label()
public Label(String labeltext)
public label(String Labeltext,. int align);
O Label pode ser construido com ou sem o texto e alinhamento especificado. Para alinhamento, existe três tipos definidos na classe label: RIGHT (direita), LEFT (esquerda), CENTER (centrado). Não esqueça que, para acessar o valor definido na classe, devemos colocar o nome da classe antes do valor, tais como Labal.LEFT.
Possui dois métodos:
public void setText(String texto);
public String getText();

Estes métodos atribuem (ou modificam) e retornam respectivamente, o valor a ser mostrado no Label.

```
import java.awt.*;
import java.applet.*;
public class Rotulo extends Applet {
  Label texto = Label("Alo, pessoal!", Label.LEFT);
  boolean original = true;
  public void init(){
    add(new Button("mudar o rotulo"));
  }
  public void boolean action(Event evento, Object quem) {
    if(evento.target instanceof Button) {
      if(original)
         texto.setText("Sou applet de rótulo");
      else
         texto.setText("Alo, pessoa1");
      original = !original;
      repaint();
      return true;
    }
    else
      return false; // evento não foi de botão. Próximo action é chamado.
  }
}
```

Note que não verificamos o nome do botão. Isto pode ser feito, porque tem apenas um botão e logo, se algum botão for clicado, deve ser ele.

Caixa de checagem

Caixa de ticagem é uma espécie de label, com opção de ligar/desliga. Ele apresenta um quadradinho seguido de label, e o usuário pode ligar ou desligar esta opção. Ele é denominado de CheckBox e possui três construtores:

```
public Checkboxl()
public Checkbox(String labeltext)
public Checkbox(String Labeltext, CheckboxGroup g, boolean ini)
```

O terceiro construtor parece estranho, mas isto é devido ao fato de Checkbox é usado também para criar botões de rádio.

O primeiro e segundo construtor tem nada a esplicar. O terceiro construtor é usado, quando deseja criar ChekBox já ligado ou desligao pelo default. O segundo parâmetro CheckboxGroup, por enquanto, esquece. Passe o new Checkboxgoup() no lugar dele que instanciará um novo checkboxgroup.

existe o método denominado

```
public boolean getState()
```

que retorna true se o CheckBox estiver ativado e false caso contrário.

Cada vez que o usuário clicar sobre checkbox para ligar ou desligar, o evento ocorre e action é chamado.

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

```
public class MyCheckBox extends Applet {
  Checkbox Check1 = new CheckBox("gosto de java", new CheckboxGroup(),
true),
        , Check2 = new CheckBox("iniciante em java", new CheckboxGroup(),
true);
  public void init() {
    add(Check1);
    add(Check2);
  }
  public void boolean action(Event evento, Object quem) {
    if(evento.target instanceof CheckBox) {
      CheckBox checkboxAtual = (CheckBox)evento.target;
      if(CheckboxAtual.getLabel() == "gosto de java") { // primeiro Checkbox
        if(CheckboxAtual.getState()) { // ligado:
          // O que vai fazer, se ligar
        }
        else {
          // oque fazer, caso contrário.
        }
        return true;
       }
       else if(CheckboxAtual.getLabel() == "iniciante em java") {
        if(CheckboxAtual.getState()) {
            // o que fazer
         }
         else {
           // o que fazer
         }
         return true;
      }
      return false;
  }
}
```

No exemplo acima, armazenamos o evento.target no variável. Sempre que faz uso repetido, esta técnica facilita o processo.

Normalmente, não implementamos o método action para tratar de CheckBox, toda vez que clicar sobre ele. Programação correta é incluir um botão de confirmação e checar o estado dos CheckBox somente quando o botão for clicado. Portanto, o exemplo acima serve apenas como exemplo para ilustrar o fato de que o checkbox gera evento e é possível tratá-lo.

Verifique no exemplo abaixo, como tudo fica mais simples, se utilizar um botão de confirmação (esta é a progrmação usual de CheckBox).

```
import java.awt.*;
import java.applet.*;
public class MyCheckBox extends Applet {
  Checkbox Check1 = new CheckBox("gosto de java", new CheckboxGroup(),
true),
```

```
       , Check2 = new Checkbox("iniciante em java", new CheckboxGroup(),
true;
  public void init() {
    add(Check1);
    add(Check2);
    add(new Button("OK"));
  }
  public void boolean action(Event evento, Object quem) {
    if(evento.target instanceof Button) { // único botão é de confirmação
      if(Check1.getState()) { // ligado: ele gosta de java
        // O que vai fazer, se ligar
      }
      else {  // desligado: não gosta de java
        // oque fazer, caso contrário.
      }
      if(Check2.getState()) { // ligado: iniciante em java
        // o que fazer
      }
      else { // desligado: não é iniciante em java
       // o que fazer
      }
      return true;
    }
    return false;
  }
}
```

Botão de Rádio.

Botão de rádio, denominado de Radio Button é um conjunto de checkbox e é implementado, usando o CheckBox. Neste conjunto, apenas um ítem estará ligado. É uma espécie de seleção de opção. Ao clicar sobre um deles, desativa automaticamente o outro. Para cliar o botão de rádio, criamos um objeto do tipo CheckboxGroup e ir adicionando Checkbox nele, usando o terceiro construtor do checkbox:
public Checkbox(String texto, CheckboxGroup g, boolean ini)
a classe CheckboxGroup possui um me'todo
public Checkbox getCurrent()
que retorna o checkBox escolhido atualmente (ítem selecionado).
Não esqueça que, bom estilo de programação para usar Checkbox ou Radio Button é criar um botão extra de confirmação e checar o estado de Checkbox ou RadioButton somente quando for clicado sobre botão de confirmação.

```
import java.awt.*;
import java.applet.*;
public class MyCheckBox extends Applet {
  Label texto = new Label();
  CheckboxGroup Escolha = new CheckboxGroup();
  public void init() {
```

```
    add(new CheckBox("Linguagem favorito é Java, Escolha, true));
    add(new CheckBox("Linguagem favorito é C++, Escolha, false));
    add(new CheckBox("Linguagem favorito é Pascal, Escolha, false));
    add(new CheckBox("Linguagem favorito é Basic, Escolha, false));
    add(new CheckBox("Linguagem favorito é Fortran, Escolha, false));
    add(new Button("Confirma"));
    add(new Button("Cancela"));
    add(texto);
  }
  public boolean action(Event evento, Object quem) {
    if(evento.target instanceof Button) {
       if((String)quem == "Confirma") {
         CheckBox atual = Escolha.getCurrent();
         texto.setText(atual.getLabel());
         repaint();
         return true;
        }
     }
     return false;
  }
}
```

Caixa de Escolha

A caixa de escolha é uma caixa que permite escolher ítens entre vários. Ele exibe o ítem selecionado na caixa até que altere a seleção. neste sentido, é semelhante ao botào de rádio.
Ele possui como método básico

```
public suncronized void additem(String intem)
      throws NullPointerException
public void select(String intem)
public String getSelecteditem()
```

que servem respectivamente para adicionar ítem, escolher a seleção atual e obter o ítem selecionado atualmente.
Existem outros métodos associados a ele, mas os três métodos acima já são suficientes para aplicação em geral. caso deseja, consulte o livro sobre outros métodos.
Assim como CheckBox e Radio Buttom, Choice gera evento quando uma seleção é realizado. Na aplicação em geral, cada vez que efetua a escolha diferente, é efetuado a operação correspondênte, mas podemos usar botão extra como no caso de Checkbox e Radio Buttom.
Veremos um exemplo de caixa de seleção para escolher a cor do fundo do panel de applet:

```
import java.awt.*;
import java.applet.*;
public class MyChoice extends Applet {
  Choice cor = new Choice();
  public void init() {
```

```
    cor.addItem("Vermelho");
    cor.addItem("Azul");
    cor.addItem("Verde");
    cor.additem("Amarelo");
    cor.addItem("Branco");
    cor.select("Branco");
    setBackground(Color.White);
    add(cor);
  }
  public boolean action(Event evento, Object quem) {
    if(evento.target instanceof Choice) {
      if(cor.getSelecteditem() == "Vermelho")
        setBackground(Color.Red);
      else if(cor.getSelecteditem() == "Azul")
        setBackground(Color.Blue);
      else if(cor.getSelecteditem() == "Verde")
        setBackground(Color.Green);
      else if(cor.getSelecteditem() == "Amarelo")
        setBackground(Color.yellow);
      else if(cor.getSelecteditem() == "Branco")
        setBackground(Color.White);
      else
        return false;
      return true;
     }
     return false;
    }
  }
}
```

Lista (de rolagem)

A lista de rolagem é quase igual à caixa de seleção. Ele exibe os ítens para ser selecionado, porém, pode conter número grande de dados e pode rolar a lista atrvés da barra de rolagem ao lado dele. Os ítens serão mostrados numa área designada para ele. Na caixa de seleção, apenas um ítem pode ser selecionado, no entanto, na lista, pode ser configurado para selecionar apenas um ítem ou permitir multiplas seleções.
Construtores são:
public List()
public list(int linhas, boolean MultiplaEscolha);
No primeiro construtor, a list é da escolha simplas. No segundo construtor, podemos indicar o número de ítens que será exibido por vez e dizer, se a escolha é apenas um ítem ou multipla escolha.
Para adicionar, ítens, usa-se o método:
public syncronized void addItem("String item);
Para saber os ítens selecionados, podemos usar os métodos:
public syncronized String getSelectedItem();

public syncronized String getSelectedItens();
Respectivamente para lista de escolha simples e escolha multipal.
Uma observação importante é que, não chama o método action quando selecionamos elementos da lista. Portanto, devemos sempre criar um botão de confirmação. Mas caso queira manipular evento quando a seleção ocorre (por exemplo, escolha de fonte no editor), podemos usar o método
public boolean handleEvent(Event event);
Veremos um exemplo simples:

```
import java.awt.*;
import java.applet.*;
public class MyList extends Applet {
  List lista = new List("3, false);
  public void init() {
     lista.addItem("Vermelho");
     lista.addItem("Azul");
     lista.addItem("Verde");
     lista.additem("Amarelo");
     lista.addItem("Branco");
     lista.select("Branco");
     setBackground(Color.White);
     add(new Label("Escolha as cores preferidas");
     add(lista);
     add(new Button("OK");
 }
 public boolean action(Event evento, Object quem) {
    if(evento.target instanceof Button) {
       if((String)quem == "OK")) { // confirmado
          String cor = lista.getSelecteditens();
       if(cor == "Vermelho")
          setBackground(Color.Red);
       else if(cor == "Azul")
          setBackground(Color.Blue);
       else if(cor == "Verde")
          setBackground(Color.Green);
       else if(cor == "Amarelo")
          setBackground(Color.yellow);
       else if(cor == "Branco")
          setBackground(Color.White);
       return true;
     }
     return false;
    }
 }
}
```

A lista possui vários métodos para manipulação bem controlado, porém, não trataremos aqui. O interessado em usar a lista, deve procurar no livro ou manual para ter menor desperdício e maior controle na utilização de listas.

Campo de Textos e Área de Textos

O campo de texto denominado de TextField é uma área retangular onde possa escrever ou digitar uma linha de texto e é fundamental para criar entrada/saida de dados no AWT.
Textfield possui quatro construtores:
public Textfield();
public TextField(int numcols);
public TextField(String texto);
public TextField(String texto, int numcols);
onde numcols é o número de colunas e texto é o texto default.
A área de texto é análogo ao campo de texto, porém, é uma região retangular na qual permite editar texto em várias linhas. O seus construtores são:
public TextArea();
public Textarea(int numcols, numrows);
public Textarea(String texto);
public TextArea(String texto, int numcols, numrows);
onde texto é o texto default, numcols, e numrows são número de linhas e colunas da área de texto.
Essencialmente, a diferênça entre Textfield e TextArea , é a área de edição.
Seus métodos são similares:
Para associar um texto para ser editado, usa-se o método
public setText(String texto);
e o método
public String getText();
retorna o texto editado.
O campo de texto ou área de texto pode ser editábel (entrada) ou não editável (saída). Para configurar a editabilidade, usa-se o método
public void setEditable(boolean editavel);
Se estiver usando apenas para exibir dados, basta chamar setEditable com parâmetro false. A configuração padrão de editabilidade é true.
Como no caso de lista, TextField e TextArea dispõe de uma variedade de métodos para melhor comtrole. O interessado deve consultar o manual.
Também, como na lista, o método action não é chamado. Seu evento pode ser manuseado pelo método handleEvent, ou pelo botão ou menu de confirmação.
Na prática, quando deseja implementar um editor, implementamos o evento de mouse diretamente para TextArea, em vez de implementar o handleEvent().
A seguir, um exemplo simples que usa o Textfield para entrada/saída de dados:

```
import java.awt.*;
import java.applet.*;
public class TextField extends Applet {
  String texto;
  TextField entrada(40), saida(40);
 public void init() {
     saida.setEditable(false);
```

```
    add(new Label("Entrada")); add(entrada);
    add(new Label("Saida")); add(saida);
    add(new Button("Confirma"));
 }
 public boolean action(Event evento, Object quem) {
   if(evento.target instanceof Button) {
     if((String)quem == "Confirma")) { // confirmado
        texto = entrada.getText();
        saida.setText(texto);
        repaint(); // não esquecer
        return true;
      }
    }
    return false;
   }
 }
}
```

Note que, a entrada de sados no AWT é feito através de TextField e TextArea, ambos como sendo String. Quando deseja entrar ou sair om dados numéricos, devemos converter. portanto, devemos saber a conversão básica:
(new Integer(n)).toString() converte o inteiro n para String
(new Double(x)).toString() converte o double x para String
(new Integer(s)).intValue() converte o String s para inteiro
Integer. parseInt(s) converte o String s para inteiro
(new Double(s)).doubleValue() converte o String s para double
Double.parseDouble(s) converte o String s para double

Scroll Bar e Canvas

O uso de Scroll Bar (barra de rolamento) requer um pouco de maturidade no AWT e não será tratado aqui. Canvas, não será necessário para iniciante em AWT e também não será tratado.

Frames

Frame é uma janela e nele, podemos adicionar panel (dentro do panel, podemos adicionar qualquer componentes).
A utilidade maior de um frame é criar applet que possa ser executado fora da página HTML (aplicativo) e criar janelas de comunicação. Seu construtor pode ou não receber o título da janela.
public Frame();
public Frame(String titulo);
O Frame possui diversos métodos, mas veremos apenas o essencial. O interessado deve consultar o manual.
public voidd show();
public void hide();
Um Frame não será mostrado quando é criado, pois pode demorar para criar,

acrescentando componentes nele. Para que a janela seja visível 9e funcione), devemos chamar o método show(). Uma vez que terminou de utilizar a janela, devemos chamar o método hide() para desativar. Note que, o método hide apenas esconde a janela até que outro show() seja chamado.
Alguns métodos úteis são:
public void pack();
public setResizable(boolean redimensionavel);
onde pack() é o método que ajusta o tamanho da janela para tamanho mínimo necessário e setResizable() configura a janela para janela de tamanho ajustável ou não. Quando o parâmetro é true, é de tamanho ajustável e quando é false, é de tamanho fixo.
Quando não vai usar mais o frame, devemos chamar o método
public void dispose();
para liberar o recurso que está sendo usado pelo frame. O método hide() esconde o frame, mas não libera os recursos usados.

Para criar applet que possa ser executado fora da página HTML, implementamos o método main() e através dele, criamos um Frame, adiciona a classe principal da applet e executar o init, e em seguida, executa-se o método show() do Frame. Por exemplo, se o applet chamar MeuApplet, o método main() associado nele será mais ou menos o seguinte:

```
import java.awt.*;
import java.applet.*;
public class Botao extends Applet {
  public void init(){
    add(new Button("Azul"));
    add(new Button("Vermelho"));
  }
  public void boolean action(Event evento, Object quem) {
    if(evento.target instanceof Button) {
      String label = (String)quem;
      if(label == "Azul")
        setBackground(Color.blue);
      else if (label == "Vermelho")
        setBackground(Color.red));
      else
        return false;
      repaint();
      return true; // não chama mais nenhum action seguinte
    }
    else
      return false; // evento não foi de botão. Próximo action é chamado.
  }
  public void main(String args[]) {
    Frame janela = new Frame("Applet de Botão");
    Applet principal = new Botao();
    principal.init();
```

```
    principal.start();
    janela.setLayort(new FlowLayort());
    janela.add(principal);
    janela.pack(); janela.show();
  }
}
```

como exemplo:

```
public void main(String args[]) {
  // cria a janela
  Frame janela = new Frame("Meu Applet");
  // cria o applet
  Applet principal = new MeuApplet();
  // inicializa o applet através de init() e start(): nessa ordem.
  principal.init();
  principal.start();
  // configura o Layort para Flowlayort.
  // um Frame deve ter Layout, obrigatoriamente.
  janela.setLayort(new FlowLayort());
  // Adiciona o applet na janela
  janela.add(principal);
  janela.pack(); // tamanho mínimo necessário
  janela.show(); // ativa a janela
}
```

Menu

Menu possui dois construtores
public Menu()
public Menu(String titulo)
onde o segundo construtor atribui nome para menu. Os ítens do menu denominado de MenuÍten pode ser adicionado (se for menu com vários ítens).
MunuIten possui dosi construtores:
Os ítens do Menu é construido pelos construtores