devmedia.com.br

# Criando uma simples Janela Swing

24-33 minutos

---

Este tutorial é bem básico, feito para quem não tem prática na criação de interfaces gráficas em java. Certo, queremos criar uma simples interface com o usuário que seja uma janela padrão (com funções de minimizar, maximizar e fechar) contendo um caixa de texto dentro (digamos que seja um início para o nosso notepad em java :00ps: ). Sigamos o raciocínio: primeiro fazemos a janela. Como em java trabalhamos com objetos em tudo então precisamos de um objeto que [b]seja uma[/b] janela. Então nossa classe de janela vai herdar da classe de janela do Swing (prefira-o ao awt, na maioria dos casos):

```
1 import javax.swing.JFrame;
2 public class MeuNote extends JFrame{
3  . . .
4 }
5
```

Bem, jé temos uma classe que cria objetos que são janelas. Agora precisamos construir a infra-estrutura da janela, modificando-a e adicionando coisas. Isso faz parte do processo de construção da janela e nenhum lugar melhor para isso que no [b]construtor[/b] da classe! Antes uma pequena pausa para explicar um detalhe. As API gráficas do java provêm dois tipos de componentes visuais: componentes propriamente ditos e containers. [list] Um componente é uma peça (widget) que tem alguma função na interface. Um container é um componente que tem a capacidade de abrigar outros componentes (conteiners possuem um método [i]add[/i](), que recebe qualquer componente como parâmetro e o adiciona a sua lista de componentes "abrigados").[/list] Sem entrar em muitos méritos (esse é um tutorial curto), o swing tem uma característica interessante adicional: todos os componentes swing, todos [b]mesmo[/b] são também

containers (possuem um método [i]add[/i]())! :shock: Retomando a nossa trilha agora: nosso próximo objetivo é abrigar uma caixa de texto na janela. Para isso, simplesmente criaríamos um objeto JTextArea (é o componente visual para grandes áreas de texto não formatado (como no notepad do windows) no swing) e o passaríamos para o método [i]add[/i](). Mas há um pulo do gato aí! Até o Java 1.4, não era possível adicionar componentes gráficos [b]diretamente[/b] em um JFrame. Isso pq ele é um container especial que só permite um número limitado de componentes (containers) abrigados nele. Em um JFrame existem alguns containers "invisíveis" sobrepostos. O container mais superficial é chamado de "[color=red:ec9b2e8e19][b]contentPane[/b][/color:ec9b2e8e19]" (painel de conteúdo) e é a superfície na qual os componentes devem ser dispostos. Usar o método [i]add[/i]() do JFrame faria com que nosso componente ficasse embaixo do [color=red:ec9b2e8e19][b]contentPane[/b][/color:ec9b2e8e19], sem efeito visual. Por isso, no J2SE1.4 para trás, o compilador reclama com vc. se isso for feito. Mas então como raios eu adiciono o componente? Simples, obtenha uma referência ao contentPane e use o método [i]add[/i]() desse container:

```
1  import javax.swing.JFrame;
2  import javax.swing.JTextArea;
3  public class MeuNote extends JFrame{
4    /* Componentes devem estar no contexto da
5  instância,
6        para que possam ser acessados em todos
7  os métodos
8        não-estáticos da classe
9    */
10   private JTextArea texto = new JTextArea();
11    public MeuNote(){
12      //Define o título da janela
13      super("Meu Notepad");
14      this.montaJanela();
15    }
16    private void montaJanela(){
       this.getContentPane().add(texto);
```

```
17    }
18  . . .
19 }
20
21
```

No Java 1.5 (J2SE5.0 - Tiger), se vc. usa o método add() do JFrame ([i]this.add(texto)[/i]), o compilador já supõe que vc. quer adicionar um componente no contentPane e gera um bytecode equivalente a [i]this.getContentPane().add(texto)[/i]. Boa notícia, não? Mas lembre-se: só funciona da versão 5.0 do Java em diante. Nosso objetivo está quase pronto, já temos nossa janela. Agora só precisamos exibir a mesma. Uma arquitetura melhor seria criar outra classe executável para usar a nossa janela, inicializando a aplicação criando e exibindo. Mas para sermos simplistas, vamos usar essa classe mesmo, tornando-a uma classe funcional (executável).

```
1  import javax.swing.JFrame;
2  import javax.swing.JTextArea;
3  public class MeuNote extends JFrame{
4    /* Componentes devem estar no contexto da
5  instância,
6       para que possam ser acessados em todos
7  os métodos
8       não-estáticos da classe
9    */
10   private JTextArea texto = new JTextArea();
11    public MeuNote(){
12      //Define o título da janela
13      super("Meu Notepad");
14      this.montaJanela();
15    }
16    private void montaJanela(){
17       this.getContentPane().add(texto);
18    }
19    public static void main(String[] args){
```

```
20        //Cria objeto:
21        MeuNote janela = new MeuNote();
22    }
23 }
24
25
```

Ponha isso pra rodar e vc. vai ter... [b]nada[/b]! Vc. criou o objeto da janela, mas em momento algum disse para torná-lo visivel ao usuário. Acrescente então no método main() a linha:

Ao rodar isso vc. obtém sua janela... Mas ops! ela fica pequenininha lá no canto esquerdo. Vc. tem que definir um tamanho para sua janela! Acrescente a linha:

```
1 /* A medida de tamanho é em pixels por
2 polegada, igual a da resolução da sua tela */
  janela.setSize(640,480);
```

Dica: Faça isso antes de exibir sua janela. Assim a JVM não precisa enviar mensagens ao Sistema Operacional para redimensionar a janela. Defina o tamanho antes (deixando que os componentes dentro da janela se organizem para o tamanho da mesma) e mostre depois! Obs.: Ao invés do setSize(), vc. também pode utilizar o método pack() (sem parâmetros dessa vez) para que ele configure a janela com o melhor parâmetro que abrigue todos os componentes, de acordo com seus tamanhos. Em nosso exemplo não vai adiantar nada, pq não definimos um tamanho (preferencial ou específico) para o JTextArea. Enfim, eis o código que funciona perfeitamente:

```
1 import javax.swing.JFrame;
2 import javax.swing.JTextArea;
3 public class MeuNote extends JFrame{
4   /* Componentes devem estar no contexto da
5 instância,
6       para que possam ser acessados em todos
  os métodos
7       não-estáticos da classe
8   */
9   private JTextArea texto = new JTextArea();
```

```
10    public MeuNote(){
11      //Define o título da janela
12      super("Meu Notepad");
13      this.montaJanela();
14    }
15    private void montaJanela(){
16       this.getContentPane().add(texto);
17    }
18    public static void main(String[] args){
19        //Cria objeto:
20        MeuNote janela = new MeuNote();
21        janela.setSize(640,480);
22        janela.setVisible(true);
23    }
24 }
25
26
27
```

E é o fim de nosso pequeno tutorial. Esse é um ponto de partida para evoluir nossas habilidades com alguns temas mais avançados como: [list]1- Como fazer a minha aplicação interagir com o usuário (tratamento de eventos)? 2- Como colocar uma janela dentro da outra (MDI - Multiple Document Interface)? 3- Como construir um menu e barras de ferramentas? 4- Como tornar meus comandos de aplicação (novo, abrir, salvar, e outros mais complexos) portáveis para outras aplicações que eu fizer? 5- A minha GUI está um pouco feia ou inadequada para usuários acostumados a um determinado Sistema Operacional... Há como mudar a aparência e/ou o comportamento dela (Look And Feel - Aparência e Comportamento)? 6- Como evitar que a janela "congele" quando a aplicação executar uma ação "pesada" (execução multitarefa)? 7- Como eu integro minha aplicação ao "icon tray" do Windows?[/list] Esses e outros tópicos podem ser explorados nos próximos tutoriais. Qual o próximo assunto que vc. gostaria de ver explorado? Opine! E, da próxima vez, considere fazer sua nova aplicação como uma aplicação desktop, oferecendo mais recursos visuais e integração ao seu usuário. Afinal

além de cada vez mais rápido a cada nova versão, java é portável, aproveite isso! :!:

Posts

O assunto que eu gostaria de ver explorado é de MDI e também de telas de modelo para telas de cadastro (com botões incluir, salvar, excluir, atualizar...) e também para telas de pesquisa. Se possível, gostaria de ver alguma técnica para facilitar a obtenação de dados por consulta. Ex: suponha que eu tenha que informar um cliente em determinado cadastro. Então eu tenho que ter um botão localizar cliente. Esse botão irá abrir a tela de pesquisa de cliente (abrir em showmodal) e quando o usuário selecionar o cliente, é retornado para a janela anterior o cliente selecionado. Eu gostaria de saber o que fazer para não ter que repetir sempre o código de instanciar a janela de pequisa, obter o objeto selecionado. Teria alguma coisa?

Olá pessoal! Oi Ronaldo. Para tentar afiar mais suas habilidades em swing e ajudar a atender à demanda do Ronaldo, neste tutorial vamos estender as capacidades do nosso MeuNote com duas novidades: a construção de comandos reutilizáveis, para botões de barra de ferramentas e menu. No próximo artigo vamos expandir nossa aplicação para usar MDI. Como criar comandos reutilizáveis? Uma grande evolução na forma como tratamos os comandos de usuário é encapsular os procedimentos a serem executados em objetos que possuem uma interface comum. Assim poderemos tratar todos os comandos do usuário da mesma forma. Um usuário aperta um botão de salvar e um arquivo é salvo, um usuário aperta um botão e um arquivo é aberto... A forma de acionar a ação é comum (apertar um botão), mas o procedimento executado é diferente. Um padrão de projeto que resolve este tipo de problema é o chamado padrão [b]Command [/b](pesquise por Command - Design Patterns - GoF). O Swing implementa esse padrão através da interface [color=red:9a7cf41c63][b]Action[/b] [/color:9a7cf41c63] e da classe [color=orange:9a7cf41c63] [b]AbstractAction[/b][/color:9a7cf41c63]. Para mostrar em termos simples como a coisa funciona, vamos dotar nosso notepad com uma barra de ferramentas, uma barra de menus e alguns comandos básico de novo, abrir e salvar textos. Primeiramente, vamos construir o novo esqueleto da janela:

```
1  import java.wat.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4  import javax.swing.event.*;
5  //Procure no javadoc cada umas das classes que vc. não
6  conhece neste exemplo. Assim vc. saberá em que pacotes elas
   estão e as conhecerá melhor! ;-)
7  public class MeuNote extends JFrame{
8      //Componentes
9      private JToolBar toolbar = new JToolBar("Ferramentas");
10     private JMenuBar menubar = new JMenuBar();
11     private JMenu arquivo = new JMenu("Arquivo");
12     private JTextArea texto = new JTextArea();
13     //Ações:
14     private Action novo = new NovoAction(this.texto);
15     private Action salvar = new SalvarAction(this.texto);
16     private Action abrir = new AbrirAction();
17     public MeuNote(){
18         super("Meu Notepad");
19         //Desliga automaticamente a aplicação quando o
20 usuário fecha a janela.
21         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22         Container interno = this.getContentPane();
23         this.montaMenu();
24         this.montaToolBar();
25         this.montaGUI();
26     }
27     private void montaMenu(){
28         //JMenuItem pode ser construído a partir de um
29 objeto que implementa a interface Action
30         JMenuItem itemNovo = new JMenuItem(this.novo);
31         JMenuItem itemSalvar = new JMenuItem(this.salvar);
32         JMenuItem itemAbrir = new JMenuItem(this.abrir);
33         this.arquivo.add(itemNovo);
34         this.arquivo.add(itemSalvar);
```

```
35          this.arquivo.add(itemAbrir);
36          this.menubar.add(this.arquivo);
37          this.setJMenuBar(this.menubar);
38      }
39      private void montaToolBar(){
40          //Barras de ferramenta swing também aceitam objetos
41  que implementam Action como parâmetro de construtor
42          this.toolbar.add(this.novo);
43          this.toolbar.add(this.salvar);
44          this.toolbar.add(this.abrir);
45      }
46      private void montaGUI(Container interno){
47          interno.setLayout(new BorderLayout());
48          interno.add(this.toolbar, BorderLayout.NORTH);
49          interno.add(new JScrollPane(this.texto));
50      }
51      public static void main(String args[]){
52          //Vc. sabe o que fazer...
53      }
54  }
55
```

A principal coisa a notarmos nesse código é referente aos menus e a barra de ferramentas. Passamos a eles objetos das classes NovoAction, SalvarAction e AbrirAction. Notamos também que essas classes possuem a interface [b]Action [/b]([i]javax.swing.Action[/i]) em comum. Que tipo de objeto é esse? Em swing, um objeto Action encapsula todo o necessário para gerar um botão ou item de menu. O objeto sabe informar (a quem perguntar) o nome da ação que executa, a imagem de seu botão, a descrição curta ou extensa sobre o trabalho que realiza e, por fim, ele sabe realizar a tarefa para o qual foi desenhado. Tudo isso em um só objeto (ou seja, ele é mais do que um simples ouvinte de evento)! Como criar este tipo de objeto? Vou mostrar aqui como escrever as classes NovoAction e SalvarAction, deixando a última, AbrirAction, como exercício pra você, ok? Bem, uma forma bem mais fácil de escrever uma action é criando uma subclasse de

[b][i]java.swing.AbstractAction[/i][/b]:

```
public class NovoAction extends AbstractAction{
    private JTextArea texto;
    public NovoAction(JTextArea texto){
        //Define o nome da ação
        super("Novo");
        //Define mais algumas características
        this.putValue(Action.SMALL_ICON, new ImageIcon("new.gif"));
        this.putValue(Action.SHORT_DESCRIPTION, "Limpa a área de texto");
        //Consultem a documentação de javax.swing.Action para outras propriedades
    }
    //Definimos aqui o procedimento que será executado quando NovoAction for acionado
    public void actionPerformed(ActionEvent ev){
        this.texto.setText("");
    }
}
```

```
public class SalvarAction extends AbstractAction{
    private JTextArea texto;
    public SalvarAction(JTextArea texto){
        //Define o nome da ação
        super("Salvar");
        //Define mais algumas características
        this.putValue(Action.SMALL_ICON, new ImageIcon("save.gif"));
        this.putValue(Action.SHORT_DESCRIPTION, "Salva arquivo texto");
        //Consultem a documentação de javax.swing.Action para outras propriedades
    }
    //Definimos aqui o procedimento que será executado quando NovoAction for acionado
```

```
15      public void actionPerformed(ActionEvent ev){
16          JFileChooser jfc = new JFileChooser();
17          int resp = jfc.showSaveDialog(this.texto);
18          if(resp != JFileChooser.APPROVE_OPTION) return;
19                              File arquivo =
20  jfc.getSelectedFile();
21                              this.saveFile(arquivo);
22      }
23      //Aqui trabalhamos com classes do java.io
24                  private void saveFile(File f){
25                       try{
26              FileWriter out = new FileWriter(f);
27                              out.write(this.texto.getText());
28                              out.close();
29                       }catch(IOException e){
30                          JOptionPane.showMessageDialog(null,
31  e.getMessage());
32                       }
33      }
}
```

A parte interessante disso tudo é que, se tivermos outro projeto em que seja preciso apagar, salvar ou abrir arquivos de texto em uma JTextArea, poderemos reaproveitar as classes de ação facilmente, pois elas já são razoavelmente auto-suficientes e são pouco dependentes do MeuNote em si (um desafio para você: como aumentar ainda mais o potencial de reutilização dessas actions, de forma que elas não dependam mais da referência a uma JTextArea, mas possam apagar, salvar e abrir arquivos em qualquer lugar?). Quando construir seu projeto, verá que as ações se tornam menus e botões "automagicamente". Quando um menu é clicado, ou um botão da barra de ferramentas é criado, o método actionPerformed da ação correspondente á invocado. Por hora terminamos esse tutorial. Por questão de brevidade, não respondi todas as duvidas possíveis, embora saiba que muitas serão levantadas. Fico no aguardo de vocês para fazerem as perguntas. Esse tutorial vai facilitar o entendimento do próximo artigo, onde vou mostrar a construção de

interfaces MDI. Até lá! :!:

Parabéns pelos tutoriais, estou adorando (eu que não conhecia muito de swing :) ) Apenas duas correções:

```
1 private void saveFile(File f) {
2     FileWriter out = new FileWriter(f);
3     out.write(this.texto.getText());
4     out.close();
5 }
```

É obrigatório o tratamento para a IOException, e o File arquivo = jf[b]b[/b]c.getSelectedFile(); não existe, é jfc, mas é apenas um detalhe! :o Ahh, na MeuNote.java, no método:

```
1 private void montaGUI(Container interno){
2    interno.setLayout(new BorderLayout());
3    interno.add(this.toolbar,
4 BorderLayout.NORTH);
5    interno.add(new JScrollBar(this.texto));
  }
```

A classe JScrollBar está acusando um erro:

The constructor JScrollBar(JTextArea) is undefined

:arrow: [b]Editado novamente:[/b]

```
1 public MeuNote(){
2    super("Meu Notepad");
3    //Desliga automaticamente a aplicação quando o
4 usuário fecha a janela.
5    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6    Container interno = this.getContentPane();
7    this.montaMenu();
8    this.montaToolBar();
9    this.montaGUI();
  }
```

Na invocação do método montaGUI, você deverá passar o interno como parâmetro também :!:

Obrigado pelas correções... Já corrigi o que eu vi . É que eu não tive muito tempo pra escrever a segunda parte e tive

que escrever o código de cabeça... sorry! :oops:

muito legal o tutorial Copérnico :!: só a JScrollBar esta desalinhada, fiz uns testes e consegui arruma-la: interno.add(new JScrollBar(), BorderLayout.EAST); :shock: me puxei pois nunca tinha feito nada com swing :shock: mas nao consegui fazer a classe abre :cry: se alguem fez e quizer posta-la ae pra mim dar uma olhada basica agradeço :D

Eu errei, era JScrollPane, e não JScrollBar... Já que muita gente se interessou, eu vou ver se consigo um tempo pra escrever direito, compilar tudo e por no download... :!:

Visitante. Sua dúvida diz respeito à exibição e atualização de dados em um formulário, certo? Neste tutorial, ainda não chegamos lá ainda, estaremos tratando de outro escopo. Sendo assim, resolvi mover a sua dúvida para outro tópico, onde poderemos escrever especificamente sobre este assunto, ok? :!: Nos encontramos em: [url]http://www.javafree.com.br/forum /viewtopic.php?t=13989[/url] :!:

há alguma diferenca d performance entre usar JInternalFrame ou JFrame??? qual dos dois eh + aconselhável usar???

São usados para propósitos diferentes... JFrame, assim como JWindow e JDialog (e suas contrapartes AWT) é um container "top-level", ou seja, pode ser desenhado diretamente no contexto gráfico do sistema operacional. JInternalFrame é uma janela que só pode ser adicionada dentro de um JDesktopPane. Ela não pode ser desenhada sem ele e é feita especialmente para isso (vc. não pode pôr um JFrame dentro de um JDesktopPane). Para criar uma interface MDI vc. usa um JFrame, um JDesktopPane dentro dele e 1..n JInternalFrames, que aparecerão dentro do JDesktopPane. Como vc. pode ver JFrame e JInternalFrame tem usos inteiramente diferentes, um não pode substituir o outro...

esses action funcionam com textfields???? :arrow:

Salve salve Javaneses. Preliminarmete registro meus agradecimentos pelo tutorial. Apenas alguns ajustes que tive que fazer em meu projeto. Ao clicar nas opções de menu para fazer chamada aos objetos actions, o compilador retornava erro de "NullPointerException". Isso ocorre pq a referencia ao objeto na classe action sempre

referenciava a instancia texto desta propria classe (this.texto....), assim a superclasse não conseguia "enxergar" tal objeto. Solução: Passei como parâmetro para meu construtor das classes actions o meu JTextArea (texto) da superclasse, e em seguida a subclasse.texto recebeu meu objeto superclasse.texto. Veja no código:

```
1  public class NovoAction extends AbstractAction{
2         private JTextArea texto;
3         public NovoAction(JTextArea txt){
4            super("Novo");
5            this.putValue(Action.SMALL_ICON, new
6  ImageIcon("new.gif"));
7            this.putValue(Action.SHORT_DESCRIPTION,
8  "Limpa a área de texto");
9            /*Aqui objeto de instancia texto recebe
10 o objeto
11            * da superclasse texto.
12           */
13           texto = txt;
14        }
15 ...
16 ...
   ...
```

Existem outras maneiras de resolver este problema, como por exemplo austar a referência ao objeto texto da seguinte maneira. [i]NomeDaSuperClasse[/i].this.objeto. Talvez eta não seja a forma mais elegante de resolver o problema, pois você poderá ter problemas de reutilização de código. Por fim, estou submetendo minha classe [i]AbrirAction[/i]:

```
1  package br.est.aca.df.projectSWING;
2  import java.awt.event.ActionEvent;
3  import java.io.*;
4  import java.io.IOException;
5  import javax.swing.AbstractAction;
6  import javax.swing.JFileChooser;
7  import javax.swing.JOptionPane;
```

```
8  import javax.swing.Action;
9  import javax.swing.ImageIcon;
10 import javax.swing.JTextArea;
11 public class AbrirAction extends AbstractAction{
12        private JTextArea texto;
13        private File file;
14        public AbrirAction(JTextArea txt){
15           //Define o nome da ação
16           super("Abrir");
17           //Define mais algumas características
18           this.getValue("Abrir");
19           this.putValue(Action.SMALL_ICON, new ImageIcon("new.
20           this.putValue(Action.SHORT_DESCRIPTION, "Abre arquiv
21 edição");
22           texto = txt;
23        }
24        public void actionPerformed(ActionEvent ev){
25             //private String err="";
26             try{
27               JFileChooser jfc = new JFileChooser();
28               jfc.setFileSelectionMode(jfc.FILES_ONLY);
29               int resp = jfc.showOpenDialog(this.texto);
30               if(resp== jfc.CANCEL_OPTION){
31                 return;
32               }else{
33                 file = jfc.getSelectedFile();
34                 openFile(file);
35               }
36           }
37           catch(Exception e){
38                 JOptionPane.showMessageDialog(null,e.getMessag
39 na abertura de aquivo",JOptionPane.ERROR_MESSAGE);
40           }
41        }
          private void openFile(File f){
```

```
42              try{
43                  FileReader rd = new FileReader(f);
44                  int i = rd.read();
45                  String ret="";
46                  while(i!=-1){
47                      ret = ret+(char)i;
48                      i = rd.read();
49                  }
50                  this.texto.setText(ret);
51              }catch(IOException e){
52                  JOptionPane.showMessageDialog(null, e.getMessa
53              }
54          }
55  }
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
```

```
76
77
78
```

Até a próxima! Adriano Ávila

como q usa a herança d formulários no java???? :arrow:

Da mesma forma que usa herança de qualquer outro objeto. Basta extender a classe usando o comando extends. public class FormularioFilho extends FormularioPai {...}

mas como q eu faco isso usando netbeans???

ai já nao sei. mas normalmente é só mandar criar uma classe qualquer e digitar o código fonte da classe.