

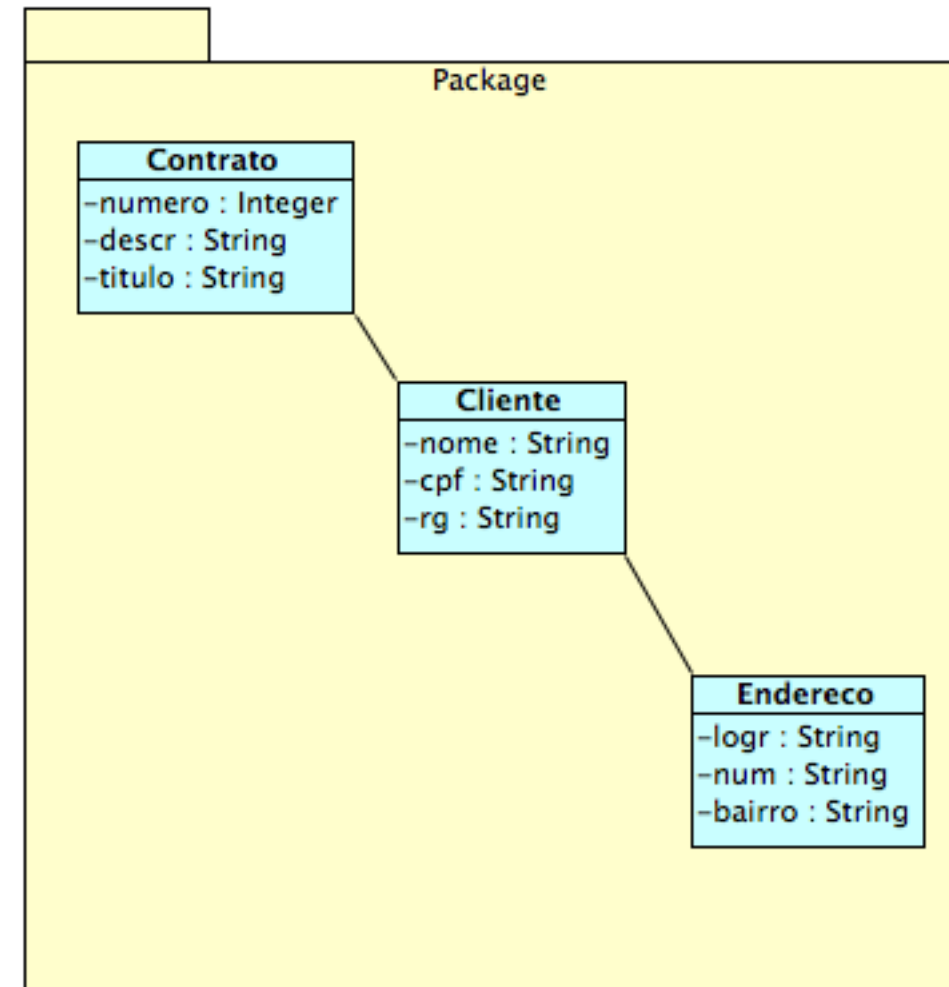


API



Agenda

- Coleções
- Formulários

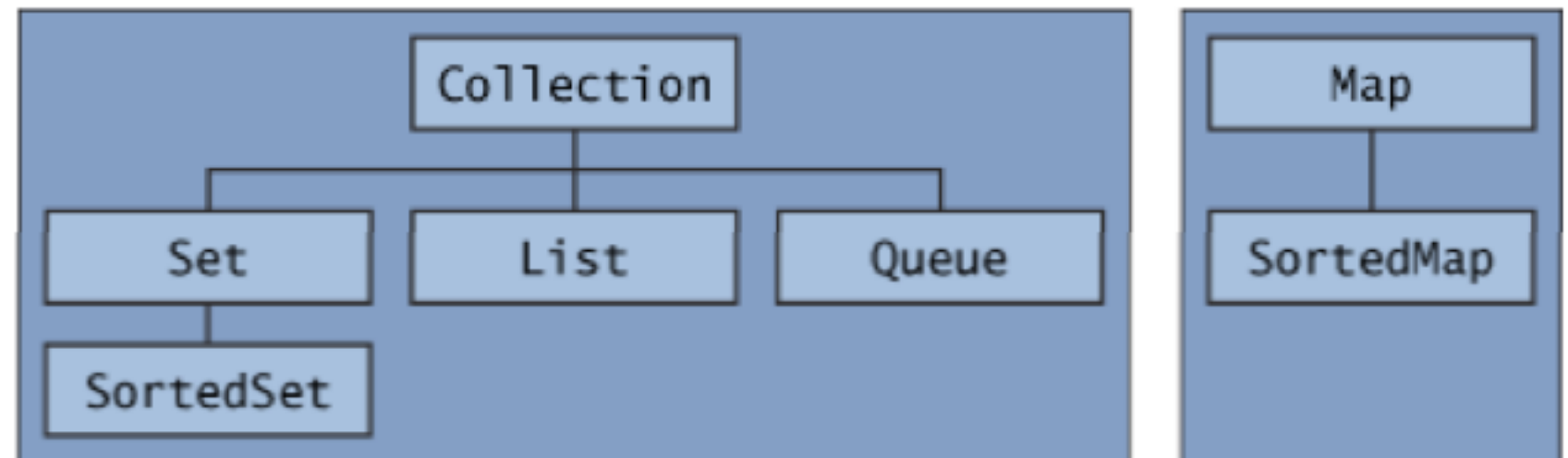




Coleções

O Java Collections Framework é a composição é uma arquitetura unificada para representar e manipular coleções e contém:

- Interfaces
- Implementações
- Algoritmos

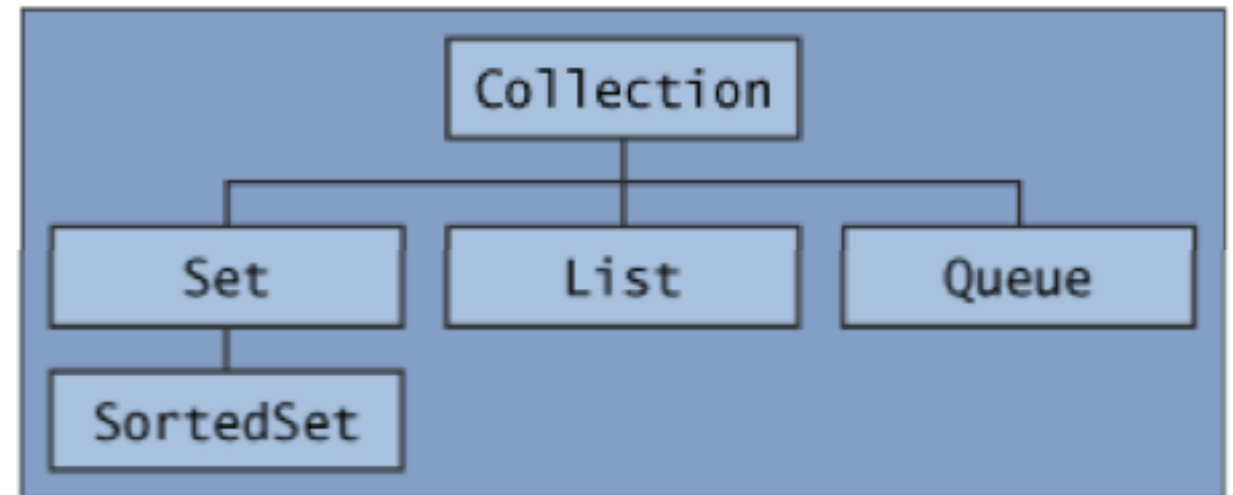




Coleções

Uma Collection representa um grupo de objetos chamados de elementos.

A interface Collection é o último denominador comum para todas as implementações de coleções e é utilizada como argumento quando é necessário o máximo de generalização.



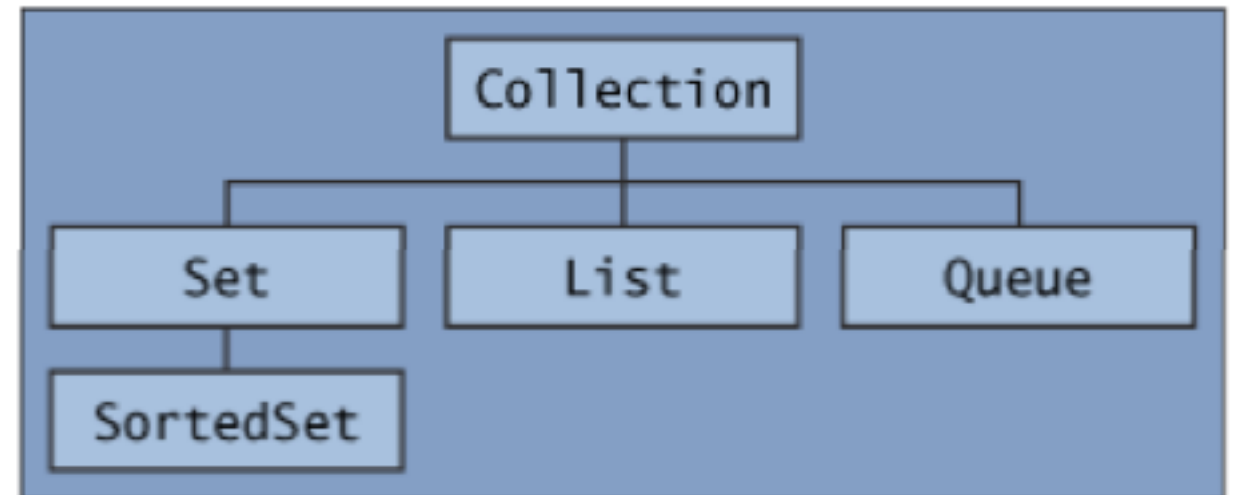


Coleções

Alguns tipo de coleções permitem duplicidade de elementos, e outros não.

Alguns são ordenados e outros não.

As mais específicas sub-interfaces são Set, List e Queue.





Collection

A Interface Collection tem vários métodos declarados e desta forma são comuns a todas as suas implementações, seja List, Set ou Queue.

Lista de alguns métodos de Collections

Métodos	Descrição
size()	Quantos elementos estão contidos na coleção
isEmpty()	true se a coleção estiver vazia
contains()	Verifica se um objeto está na coleção
add()	Adiciona um elemento à coleção
remove()	Remove um elemento à coleção
clear()	Remove todos os elementos de uma coleção
toArray()	Retorna um array dos elementos contidos na coleção
stream()	Retorna um objeto que suporta operações de agregação sequencial ou paralelo com os elementos da coleção



Set

O Set é uma coleção que não pode conter duplicidade de elementos.

Esta interface modela a abstração matemática de um set, tais como um conjunto de cartas em um jogo de truco, as disciplinas de um curso superior.

```
// Declara um set de Strings
Set<String> lista = new TreeSet<>();

// Inclui objetos no set
lista.add("abc");
lista.add("def");
lista.add("fgh");

// Ordena o set
// A implementação TreeSet ordena automaticamente na
// inserção e não permite duplicidade de chave

// Apresenta o set com foreach
for (String txt : lista)
    System.out.println(txt);

// Testa a existência do objeto no set
if(lista.contains("def"))
    // remove o objeto do set
    lista.remove("def");

// Apresenta o set com Stream
lista.stream().forEach(txt -> System.out.println(txt));
```



List

O List é uma coleção ordenada (as vezes chamada de seqüência).

List pode conter elementos duplicados.

Ao adicionarmos um elemento em List é associado um índice a este elemento (sua posição).

As implementações mais utilizadas de List são ArrayList e Vector.

```
// Declara uma lista de Strings
List<String> lista = new ArrayList<>();

// Inclui objetos na lista
lista.add("abc");
lista.add("def");
lista.add("fgh");

// Ordena a lista
Collections.sort(lista);

// Apresenta a lista com foreach
for (String txt : lista)
    System.out.println(txt);

// Testa a existência do objeto na lista
if(lista.contains("def"))
    // remove o objeto da lista
    lista.remove("def");

// Apresenta a lista com Stream
lista.stream().forEach(txt -> System.out.println(txt));
```




Queue

O Queue é uma coleção que armazena elementos em uma FIFO (primeiro que entra é o último que sai).

Numa fila FIFO, todos os novos elementos são adicionados ao final da fila.

Métodos declarados em Queue

Métodos	Descrição
remove() e poll()	Remove e retorna o elemento no topo da fila
element() e peek()	Retorna o elemento no topo da fila sem removê-lo
offer()	Usado somente em Queues com limites, adiciona elementos

```
// Declara uma Fila de Strings
Queue<String> lista = new PriorityQueue<>();

// Inclui objetos na Fila
lista.add("abc");
lista.add("def");
lista.add("fgh");

// Ordena a lista
// A implementação de PriorityQueue ordena
// automaticamente na inserção de objetos

// Apresenta a Fila com foreach
for (String txt : lista)
    System.out.println(txt);

// Testa a existência do objeto na Fila
if(lista.contains("def"))
    // remove o objeto da Fila
    lista.remove("def");

// Apresenta a Fila com Stream
lista.stream().forEach(txt -> System.out.println(txt));

// Retira os objetos da Fila
for (int i = lista.size(); i > 0; i--)
    System.out.println(lista.poll());
```



Map

O Map é um objeto que associa valores a chaves.

Um Map não pode conter chaves duplicadas.

Cada chave só pode estar associada a um valor.

Métodos declarados em Map

Métodos	Descrição
put()	Adiciona chaves e valores ao Map
get()	Retorna o elemento associado a chave informada
containsKey()	Retorna true se a chave existe no Map
containsValue()	Retorna true se o valor existe no Map
values()	Retorna uma Collection dos elementos contidos no Map

```
// Declara um Mapa de Strings
Map<String, String> lista = new TreeMap<>();

// Inclui objetos no Mapa
lista.put("chave1", "abc");
lista.put("chave2", "def");
lista.put("chave3", "fgh");

// Ordena a lista
// A implementação de TreeMap ordena automaticamente
// na inserção de objetos pela chave e não permite
// duplicidade da chave e substitui o valor caso
// isto aconteça

// Apresenta os valores do Mapa com foreach
for (String txt : lista.values())
    System.out.println(txt);

// Testa a existência do objeto no Mapa pela chave
if(lista.containsKey("chave2"))
    // remove o objeto do Mapa pela chave
    lista.remove("chave2");

// Apresenta os valores do Mapa com Stream
lista.values().stream().forEach(txt -> System.out.println(txt));
```



Formulários

As Classes e interfaces localizadas nos pacotes `java.awt` e `javax.swing` são recursos para o desenvolvimento de GUIs (Graphic User Interface).

Uma GUI é desenvolvida utilizando-se componentes, que são objetos visuais que possibilitam ao usuário realizar a interação com o programa por meio do mouse e do teclado.

The image displays four overlapping Java Swing window mockups, each representing a different data entry form:

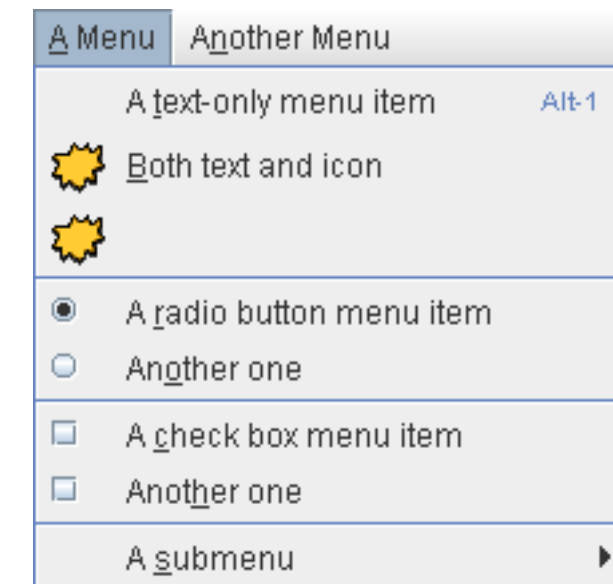
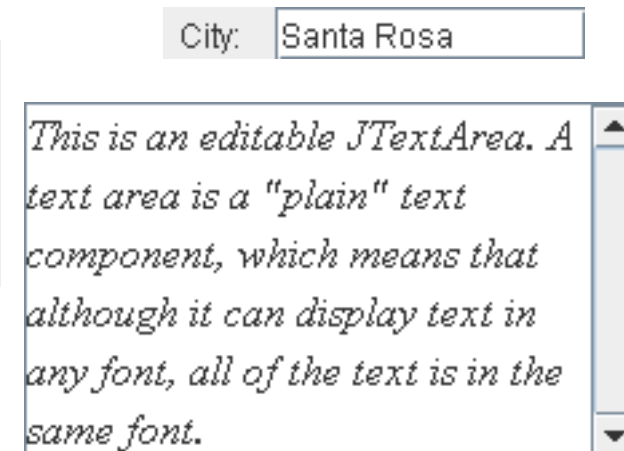
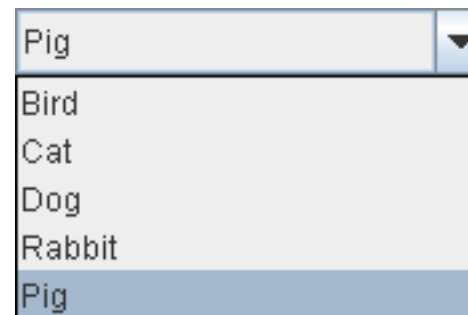
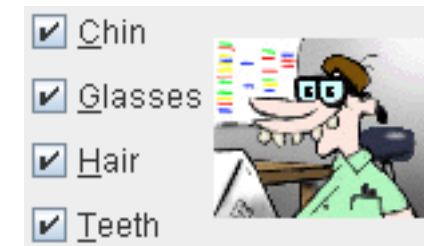
- Cadastro de Contratos:** Features input fields for 'Nº' and 'Ano', a 'Descrição' field, and buttons for 'Inserir', 'Listar', and 'Sair'.
- Cadastro de Produtos:** Includes fields for 'Cod.', 'Nome', 'Descrição', 'Valor', and 'Quantidade'. It contains a table with one row of data and buttons for 'Inserir', 'Salvar', and 'Sair'.
- Cadastro de Endereço:** Contains fields for 'Logr.', 'Nº', 'Compl.', 'Bairro', 'CEP', 'Cidade', and 'Estado', along with 'Salvar' and 'Fechar' buttons.
- Cadastro de Pessoas (partially visible):** Shows fields for 'Nome', 'Fone', 'Endereço', and tabs for 'Física' and 'Jurídica'. It also includes fields for 'RG', 'CPF', and 'E-Mail'.



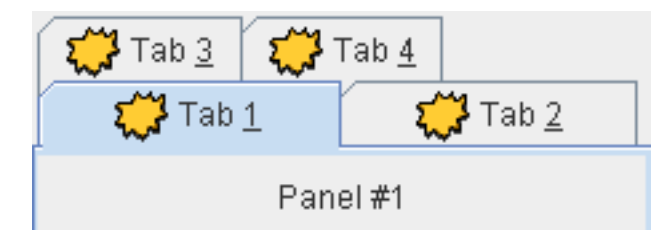
Formulários

Os componentes mais comuns são:

- Rótulos – JLabel
- Botões – JButton
- Caixas de Texto – JTextField
- Áreas de texto – JTextArea
- Caixas de checagem – JCheckBox
- Botões de radio – JRadioButton
- Caixas de seleção - JComboBox
- Listas – JList
- Menus - JMenu
- Tabelas - JTable
- Abas - JTabbedPane



Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34\$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail...	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006



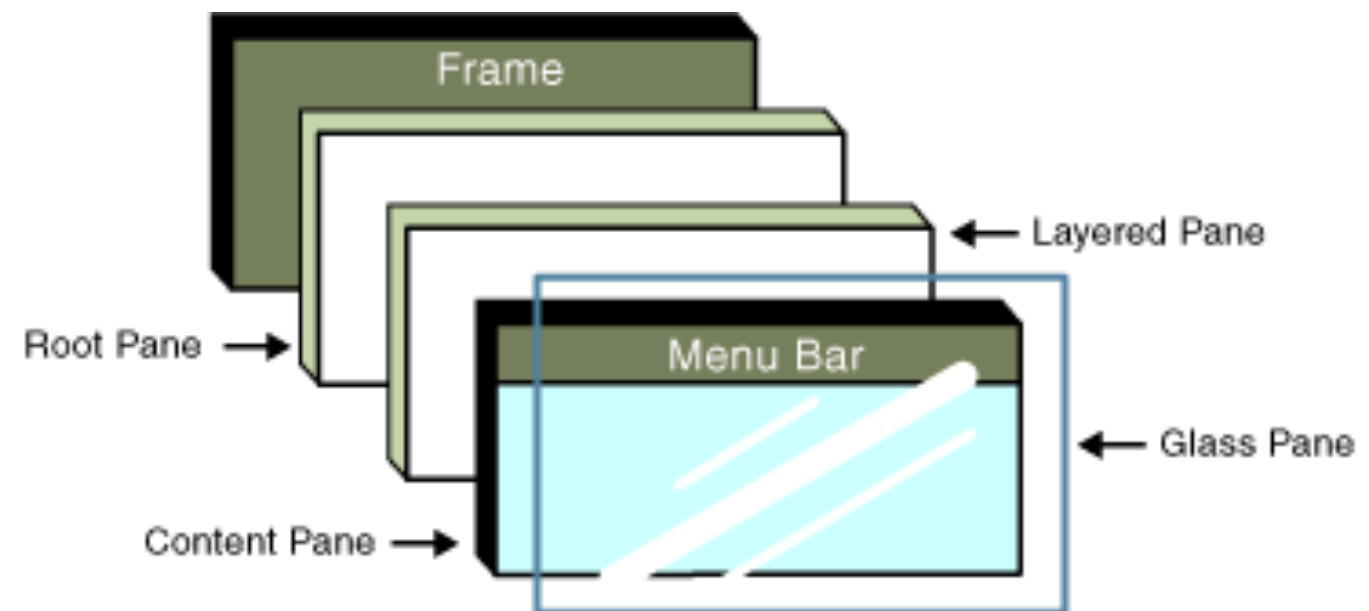


Formulários

Para a construção dos formulários precisamos de um componente base para representar a Janela onde os elementos da interface residirão.

Temos três componentes base:

- JFrame - Janela Principal
- JDialog - Janela Secundária
- JApplet - Área interna num navegador Web



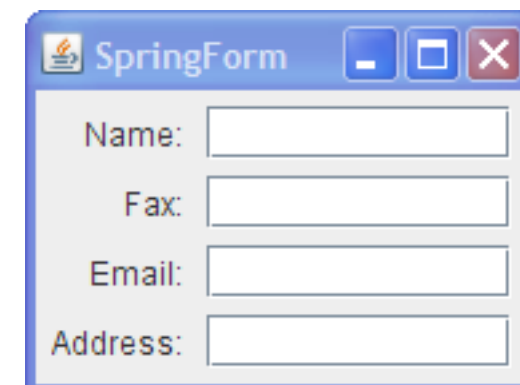
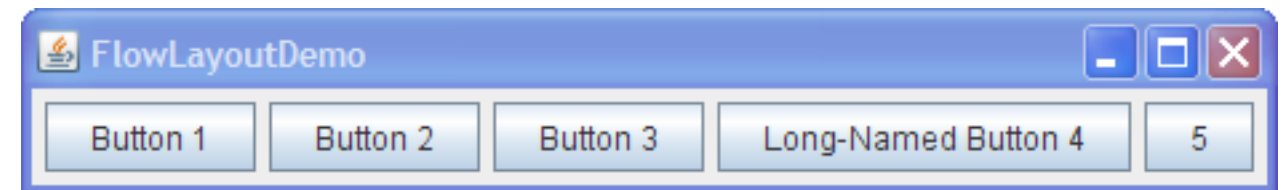
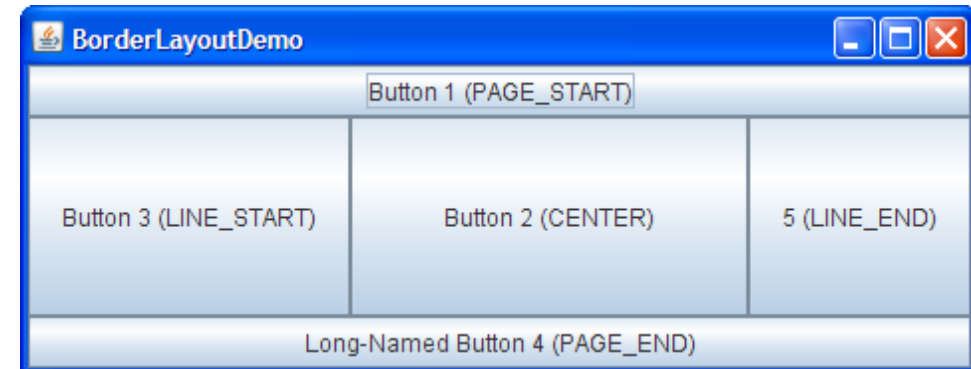


Formulários

Para auxiliar no posicionamento dos elementos da interface no formulário normalmente utilizamos Gerenciadores de Layout.

Alguns gerenciadores de layout incluídos já JVM são:

- BorderLayout
- FlowLayout
- GridLayout
- GroupLayout





Formulários

Os elementos de uma classe que cria um formulário são:

- Uma classe que estende JFrame
- Implementa ActionListener
- Tem como atributos os componentes do formulário
- Seu construtor cria e posiciona os componentes

```
public class Main extends JFrame
    implements ActionListener {
    private JTextField tfNome;
    private JTextField tfEmail;
    private JButton btSalvar;
    private JButton btListar;
    private JButton btSair;

    public Main() {
    }

    @Override
    public void actionPerformed(ActionEvent e) {
    }

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



Formulários

Todos os elementos que geram eventos, tais como botões, devem ser registrados no gestor de ações, o ActionListener.

A classe que implementa o ActionListener deverá tratar seletivamente por todos os eventos gerados pelos elementos registrados.

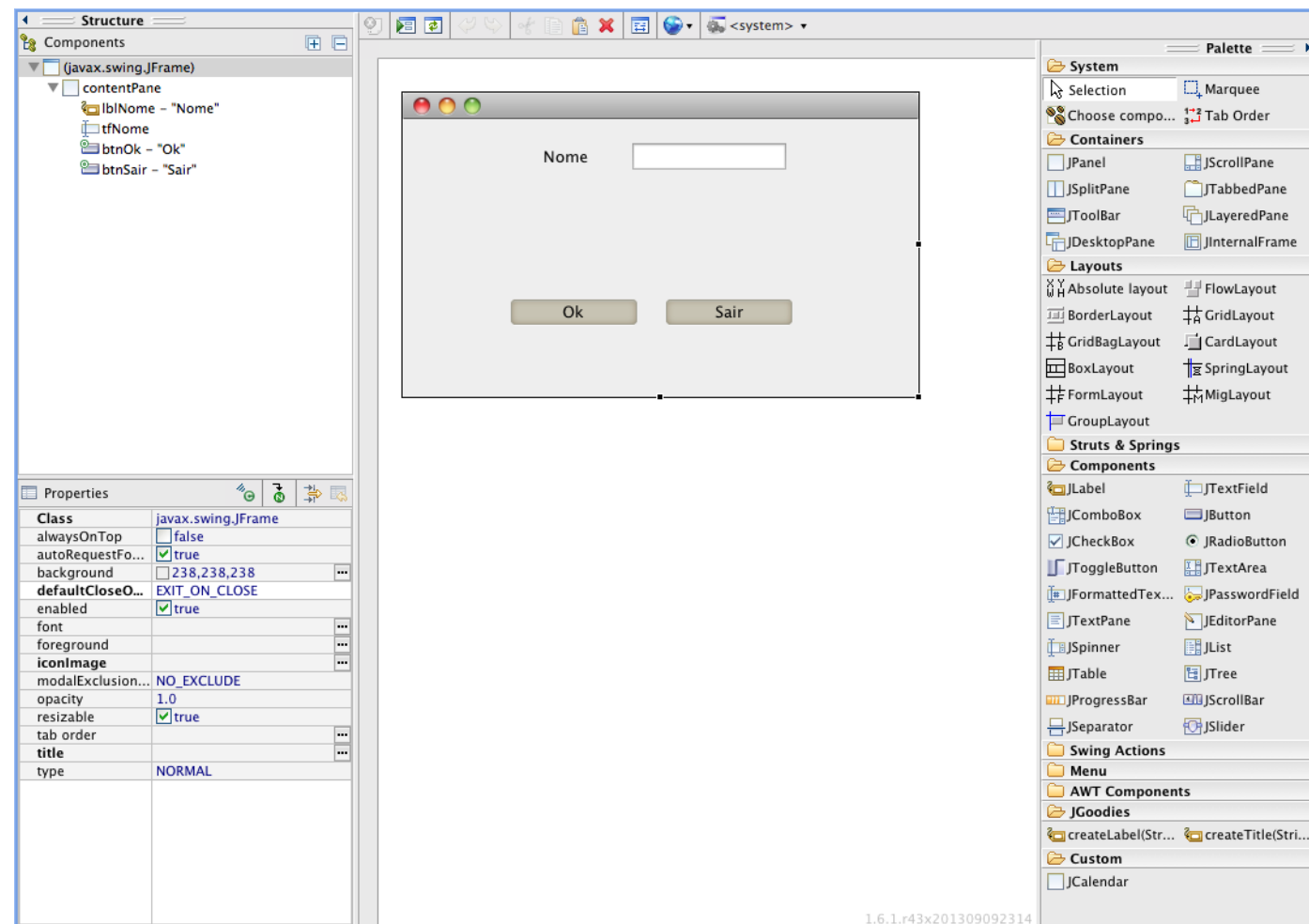
```
@Override
public void actionPerformed(ActionEvent ev) {
    Object botao = ev.getSource();

    if(botao.equals(btSalvar)) {
    } else if (botao.equals(btListar)) {
    } else {
    }
}
```




Formulários

Um formulário pode ser construído utilizando um editor de texto ou um editor especializado, como exemplo o Window Builder do Eclipse.



Referências

- Programando em Java2 - Teoria & Aplicações
Rui Rossi dos Santos - Axcel Books - 2004
- Core Java2 - Volume I - Fundamentos
Cay S. Horstmann & Gary Cornell - The Sun
Microsystems Press - Série Java - 2003
- Java Programming
Nick Clements, Patrice Daux & Gary Williams
- Oracle Corporation - 2000
- A Visual Guide to Swing Components - <http://da2i.univ-lille1.fr/doc/tutorial-java/ui/features/components.html>
- A Visual Guide to Layout Managers - <http://da2i.univ-lille1.fr/doc/tutorial-java/uiwing/layout/visual.html>

