



# Interativa

## Aplicações de Linguagens de Programação Orientada a Objetos

**Autor:** Prof. Ricardo da Costa Veras

**Colaboradoras:** Profa. Vanessa dos Santos Lessa

Profa. Larissa Rodrigues Damiani

## Professor conteudista: Ricardo da Costa Veras

Mestre em Engenharia da Informação pela UFABC (Universidade Federal do ABC) – 2018; especialista em Orientação a Objetos pela Fiap (Faculdade de Informática e Administração Paulista) – 2004 e graduado em Engenharia Elétrica com ênfase em Eletrônica pela EEM (Escola de Engenharia Mauá) – 1994. Desde 2009 é professor da UNIP (Universidade Paulista) e ministra aulas para os cursos de Ciência da Computação, Sistemas de Informação e Engenharia Básica. Possui experiência na área de Tecnologia da Informação desde 1995, tendo trabalhado com consultoria e prestação de serviços de Análise e Desenvolvimento de Sistemas em diversas empresas.

### Dados Internacionais de Catalogação na Publicação (CIP)

V476a

Veras, Ricardo da Costa.

Aplicações de Linguagens de Programação Orientada a Objetos /  
Ricardo da Costa Veras. – São Paulo: Editora Sol, 2022.

116 p., il.

Nota: este volume está publicado nos Cadernos de Estudos e  
Pesquisas da UNIP, Série Didática, ISSN 1517-9230.

1. Layout. 2. Evento. 3. Relatório. I. Título.

CDU 681.3.062

U516.17 – 22

Prof. Dr. João Carlos Di Genio  
**Reitor**

Profa. Sandra Miessa  
**Reitora em Exercício**

Profa. Dra. Marília Ancona Lopez  
**Vice-Reitora de Graduação**

Profa. Dra. Marina Ancona Lopez Soligo  
**Vice-Reitora de Pós-Graduação e Pesquisa**

Profa. Dra. Claudia Meucci Andreatini  
**Vice-Reitora de Administração**

Prof. Dr. Paschoal Laercio Armonia  
**Vice-Reitor de Extensão**

Prof. Fábio Romeu de Carvalho  
**Vice-Reitor de Planejamento e Finanças**

Profa. Melânia Dalla Torre  
**Vice-Reitora de Unidades do Interior**

### **Unip Interativa**

Profa. Elisabete Brihy  
Prof. Marcelo Vannini  
Prof. Dr. Luiz Felipe Scabar  
Prof. Ivan Daliberto Frugoli

### **Material Didático**

Comissão editorial:

Profa. Dra. Christiane Mazur Doi  
Profa. Dra. Angélica L. Carlini  
Profa. Dra. Ronilda Ribeiro

Apoio:

Profa. Cláudia Regina Baptista  
Profa. Deise Alcantara Carreiro

Projeto gráfico:

Prof. Alexandre Ponzetto

Revisão:

Kleber Souza  
Auriana Malaquias



# Sumário

## Aplicações de Linguagens de Programação Orientada a Objetos

APRESENTAÇÃO .....	7
INTRODUÇÃO .....	8

### Unidade I

1 GUI .....	11
1.1 O AWT.....	12
1.1.1 Componentes principais do AWT .....	13
1.2 O Swing .....	15
1.2.1 Componentes principais do Swing.....	15
1.2.2 Criando a primeira tela.....	17
2 APROFUNDAMENTO EM COMPONENTES COMO LAYOUT, MENUS E TABELAS .....	19
2.1 Layouts .....	19
2.2 O layout nulo .....	20
2.2.1 Sequência para criação de telas com layout nulo .....	22
2.3 O Border Layout.....	22
2.4 O Grid Layout.....	23
2.5 O Card Layout .....	24
2.6 Trabalhando com menus.....	26
2.6.1 Inserindo os elementos de menus .....	27
2.7 Trabalhando com tabelas .....	29
2.7.1 Elementos e características da construção de uma tabela .....	30

### Unidade II

3 EVENTOS .....	37
3.1 Utilizando-se dos eventos (controlando-os).....	39
3.1.1 Passos (na programação) para uso dos eventos.....	40
3.2 Os eventos de ação .....	42
3.3 Os eventos de janela .....	44
4 JDBC .....	46
4.1 Os conectores/drivers.....	47
4.2 Conectando a um banco de dados do MySQL .....	49
4.3 Lendo informações do BD .....	51
4.4 Alterando informações no BD.....	55

### **Unidade III**

5 DESIGN PATTERNS (PADRÕES DE PROJETOS) .....	62
5.1 DTO/VO .....	63
5.2 MVC .....	64
5.3 DAO (Data Access Object) .....	66
6 O HIBERNATE .....	66
6.1 Persistindo dados com o Hibernate .....	71

### **Unidade IV**

7 INTRODUÇÃO A APLICAÇÕES WEB – JSP E JSTL .....	78
7.1 Servidores de aplicação .....	78
7.2 Um pouco de HTML .....	79
7.3 Salvando uma página como JSP .....	82
7.3.1 JSP e seus elementos .....	82
7.3.2 Fazendo funcionar uma JSP .....	84
7.4 O JSTL .....	85
7.4.1 JSTL – core "c:catch": tratando erros .....	93
8 RELATÓRIOS: GERANDO RELATÓRIOS COM O MS EXCEL .....	96

## APRESENTAÇÃO

Prezada(o) aluna(o),

Por meio do presente livro-texto, colocaremos em prática o que vimos na disciplina de Linguagem de Programação Orientada a Objetos, porém introduziremos alguns novos conceitos importantes e direcionados a certas estruturas de programação, assim como para a construção de sistemas, conteúdos esses a serem aplicados na fase de desenvolvimento.

Também vamos ver e utilizar efetivamente elementos e conceitos de orientação a objetos (OO) já estudados anteriormente, como:

- Classes e objetos.
- Métodos e atributos.
- Encapsulamento.
- Herança.
- Método construtor.
- Implementação de interfaces.
- Polimorfismo de classes e de métodos.
- Tratamento de exceções.

Nosso objetivo será aplicar esses conceitos na construção de programas para desktop (inicialmente), além de abordarmos como gerar sistemas mais organizados, seguindo regras e convenções (e padrões) criadas para este fim, e utilizarmos tais conteúdos na criação de sistemas para web.

Assim, serão apresentados alguns conceitos que, apesar de serem ainda básicos, já possibilitarão a geração de sistemas completos, com acesso a dados externos (em banco de dados) e manipulação eficaz das informações.

Aproveite bem este curso, e garanta o seu sucesso.

Bons estudos!

## INTRODUÇÃO

Quando estudávamos a teoria inicial de orientação a objetos (OO), muitas vezes não entendíamos ao certo o porquê de precisarmos aprender todos aqueles conceitos, já que uma linguagem Java trabalha basicamente com três elementos: classes, métodos e atributos.

A ideia principal é que com eles construiremos sistemas mais seguros, confiáveis, eficazes, e que também nos ajudarão na utilização de alguns frameworks, o que permitirá uma compreensão mais rápida e eficaz de tudo o que envolve a construção dos sistemas com que se trabalha profissionalmente. Também temos que lembrar que um sistema não se constrói sozinho, e não se menciona aqui que ele poderia fazê-lo, mas que um sistema não é gerado por apenas um desenvolvedor, e sim por uma equipe na qual cada um trabalha com seu elemento (conhecimento) de domínio e todos estão voltados ao mesmo fim.

Portanto, é importante que a equipe seja comunicativa e saiba dividir seus conhecimentos, independentemente de eles serem técnicos ou sobre o negócio com que se está trabalhando. Somente assim um sistema será criado no melhor tempo e sob as melhores condições. O trabalho com acesso a banco de dados envolve uma cooperação mútua entre pelo menos três grupos de TI (tecnologia da informação) de uma empresa: os analistas, que procuram compreender e transformar em linguagem técnica as necessidades do cliente, de acordo com as plataformas de trabalho disponíveis na corporação; os desenvolvedores que, a partir das ideias dos analistas, elaboram e criam sistemas concretos, gerando a melhor solução no melhor tempo possível; o DBA (DataBase Administrator) que cuida e administra os bancos de dados a fim de que seja possível acessá-los da forma mais rápida e direta possível.

Portanto, é importante que o espírito de equipe esteja sempre presente, já que na maioria das vezes trabalha-se sob a pressão do tempo ou das necessidades do cliente.

Veremos a construção de sistemas que permitem a interação com o usuário através de telas equivalentes a formulários, assim como a persistência dos dados em banco de dados. Mostraremos também as informações solicitadas pelo cliente em forma de relatórios (a serem gerados em Excel), como organizar nosso sistema a fim de torná-lo eficaz e mais fácil de receber manutenção, e como fazê-lo funcionar via web.

### Aplicações desktop e aplicações web

Ao criar aplicações nas quais se pretende que haja interação com o usuário, o desenvolvedor utilizará, em seu código, recursos que possibilitam essa interação, além de pensar a lógica correta para aplicar esses recursos seguindo as regras definidas pelo cliente, prevendo sempre que o usuário talvez não possua (necessariamente) conhecimentos sobre informática (ou tecnologia da informação), mas apenas sobre a sua própria área de negócio.

É importante ainda, antes de codificar um sistema, saber se a interação do usuário com o sistema deverá acontecer via browser (se funcionará via web), ou via sistemas próprios que precisam rodar na máquina do usuário.



A aplicação desktop é aquela construída para funcionar no próprio micro do usuário, utilizando-se dos recursos de interação disponíveis a partir do sistema operacional da máquina. Felizmente, a linguagem Java possui a característica de ser multiplataforma, permitindo que, ao se construir um sistema em uma plataforma, ele funcione em qualquer outro micro que possua a máquina virtual do Java instalada (a JVM). A aplicação desktop é acionada e manipulada totalmente a partir de um browser no micro do usuário. Neste caso, os recursos visuais e de interação utilizados têm como base a linguagem HTML e devem seguir as regras da W3C, a fim de garantir que ela funcione na maior parte dos browsers (ao menos naqueles homologados pela W3C).

Os sistemas construídos para rodar via desktop, muitas vezes, não precisam necessariamente de um servidor para funcionar. Já os sistemas construídos com Java, para funcionarem via browser, necessitam estar em um servidor de aplicações com serviço http, responsável pelas solicitações de acesso dos usuários.

A ideia deste curso é mostrar como gerar telas, com os diversos componentes disponíveis, que permitem a interação com o usuário (como os campos de texto, os labels, os botões etc.).

### Servidores de aplicação

Em geral, um servidor de aplicação Java é (corporativamente) uma máquina (um servidor) que tem instalado um software servidor que permite a vários usuários acessarem serviços (programas web) gerados em uma linguagem Web-Dinâmica (a JSP, por exemplo), de forma que o serviço fica disponível a qualquer momento para vários usuários simultaneamente. No mercado existem alguns servidores de aplicação (gratuitos e/ou proprietários), como o WebSphere (da IBM), JBoss (da Red Hat JBoss Middleware), o GlassFish Server, entre outros.

Neste curso, abordaremos a utilização de aplicações web que utilizam o servidor Http e aplicações Apache Tomcat, um software que pode ser baixado gratuitamente e instalado em seu micro, transformando-o em um servidor web de aplicações Java.

### Sobre as fases de um desenvolvimento de software

Ao trabalharmos sob determinado sistema, devemos compreender que seu processo de criação possui fases até a sua implantação. De forma geral, segundo o que chamamos de RUP (Processo Unificado da Rational), as fases de desenvolvimento de sistemas são divididas basicamente em quatro etapas:

- **Fase de concepção:** aqui o cliente apresenta as suas necessidades e é onde há um levantamento de tudo o que o sistema necessitará realizar como solução àquelas demandas.
- **Fase de elaboração:** nela será complementada fase de concepção, porém com uma visão mais técnica, verificando-se as disponibilidades do ambiente de desenvolvimento e de servidor da empresa.

- **Fase de construção:** aqui a equipe técnica trabalhará na construção (codificação) do sistema propriamente dito, elaborando seus códigos e disponibilizando os acessos necessários às informações.
- **Fase de transição:** nela o sistema criado será colocado em testes para o usuário final, sendo depois posto em funcionamento, algo que chamamos de ambiente de produção (implantação do sistema).

Compreende-se, portanto, que este não é um processo a ser realizado de forma individual, e por isso precisamos aprender a trabalhar em equipe, organizando a construção de nosso código e lembrando que ele poderá ser complementado (implementado) por outros profissionais, e que tais indivíduos deverão ser capazes de compreender a estrutura e a lógica utilizada em sua construção. Portanto, estudaremos alguns padrões de desenvolvimento, ou de estruturas de construção de softwares (os design patterns) que permitem essa organização, assim como a compreensão geral do sistema (por parte da equipe de desenvolvimento do sistema).

### Sobre a organização deste livro-texto

Este livro-texto está dividido em quatro unidades, cujo conteúdo possui a seguinte estruturação:

- **Unidade I:** serão apresentados os componentes básicos de programação desktop em Java (AWT e Swing) na criação de telas de formulários que permitem a interação do sistema com o usuário.
- **Unidade II:** será exibido como podemos trabalhar com os eventos mais comuns de interação, bem como aprenderemos a forma de acessar dados externos provindos de um banco de dados (no caso, o MySQL).
- **Unidade III:** serão explicados os conceitos básicos sobre alguns dos padrões de desenvolvimento (design patterns), o porquê, e como são utilizados na construção de sistemas, assim como utilizar um framework de acesso a dados de um banco de dados: o Hibernate.
- **Unidade IV:** será mostrado como podemos extrair relatórios a partir das informações existentes em um banco de dados, e iniciaremos o estudo da criação de programas para web, aprendendo seu funcionamento básico, e alguns recursos importantes para a sua funcionalidade.

## Unidade I

### 1 GUI

Ao pensarmos na criação de sistemas desktop, ou seja, elaboração de programas que funcionarão diretamente no micro pessoal do usuário (e não em um servidor web a ser acessado via browser), este programa também deverá possibilitar a interação do usuário com o sistema, prevendo que ele vai inserir informações, ou solicitar aquelas que o sistema possa fornecer.

Para essa interação, supõe-se que o sistema seja capaz de funcionar a partir de eventos, que são as reações do sistema às ações do usuário sobre os componentes disponíveis na tela (quando o indivíduo clica sobre um botão, ou escreve um texto em um campo texto, ou quando seleciona itens de uma lista etc.). Cada uma dessas ações é considerada um evento que deve ser compreendido (ou traduzido) pelo sistema.

Para isso, deve-se gerar uma GUI (graphical user interface, ou interface gráfica com o usuário), que é um ambiente gráfico sobre o qual o usuário consegue interagir com o sistema.

Nas primeiras versões do Java, as interfaces gráficas eram programadas com base nos componentes do AWT (Abstract Window Toolkit), um conjunto de ferramentas e componentes de janela disponíveis ao desenvolvedor para que ele pudesse (e ainda possa) criar os ambientes gráficos de interação.

Com a evolução dos computadores, evoluíram também os componentes de interação, de forma que surgiu um novo conjunto de componentes com a biblioteca Swing, que proporcionou relativamente uma maior liberdade de disposição dos componentes na tela, uma aparência mais agradável em relação às telas de interação, com recursos estendidos, e muitas outras facilidades e melhorias.

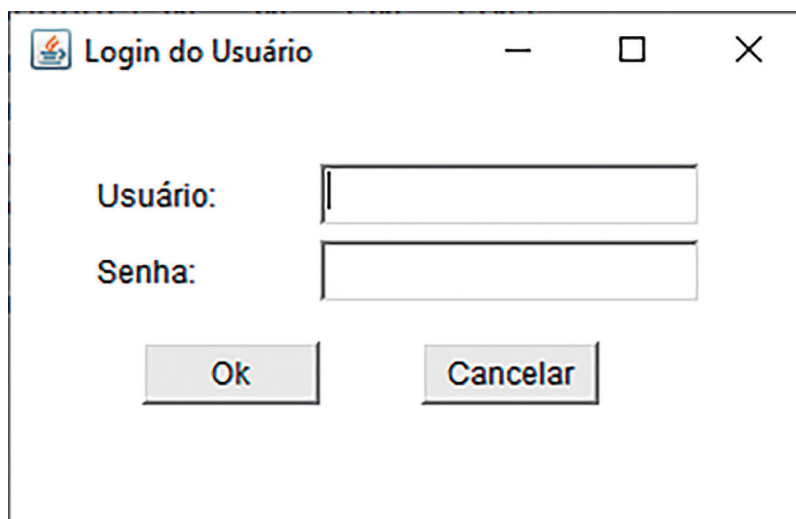


Figura 1 – Tela de login construída a partir da biblioteca AWT

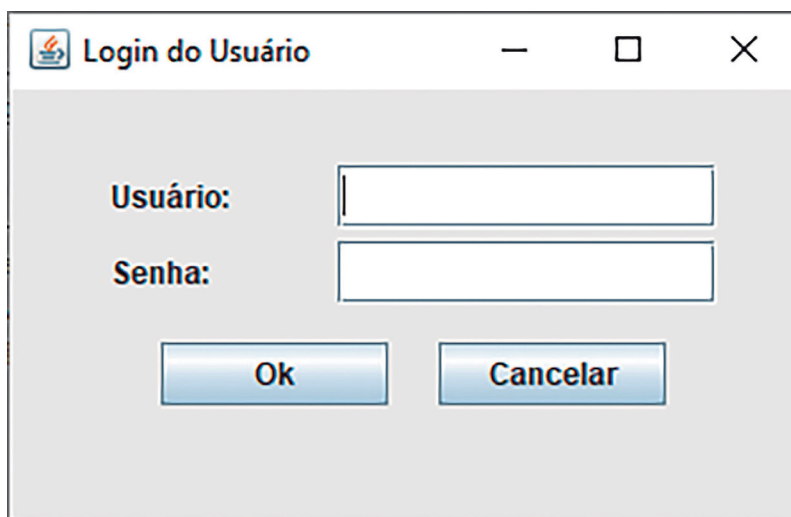


Figura 2 – Tela de login construída a partir da biblioteca Swing

Pode-se observar nas figuras anteriores que a tela que utiliza os componentes da biblioteca AWT lembra versões mais antigas do Windows, enquanto a tela gerada com os componentes da biblioteca Swing tem uma aparência mais atual.

Mesmo com a evolução das telas e dos componentes, alguns itens do AWT ainda são necessários na construção de telas interativas (por exemplo, os componentes que definem alguns layouts comuns de janela ou classes que tratam de eventos básicos).

### Importante

Este curso foi idealizado supondo-se que a(o) aluna(o) irá utilizar-se dos seguintes softwares:

- Eclipse® como IDE para a programação relacionada ao Java;
- MySQL® como banco de dados (BD) relacional;
- Apache Tomcat® como servidor Http e de aplicações.

A utilização de outros softwares, com finalidades equivalentes às descritas anteriormente, pode ser realizada, porém como não está prevista, não será comentada ou explicada.

## 1.1 O AWT

O AWT (Abstract Window Toolkit) é um conjunto (uma biblioteca) de ferramentas utilizado para a construção de janelas interativas.

Quando a linguagem Java foi lançada, uma das características mais marcantes dessa linguagem era o fato de ela ser multiplataforma (o que ainda é), o que significa que um programa gerado em uma plataforma Windows deve rodar em outra, como o MacOS ou o Linux, de forma a não ter diferença de

funcionamento, porém cada um com a sua interface (aparência) característica da plataforma. Na verdade, o AWT se utilizava dos componentes GUI nativos de cada plataforma para gerar as telas definidas na programação do sistema.

Essa biblioteca de componentes e ferramentas existe no pacote `java.awt` desde a versão 1.0 da linguagem.

## 1.1.1 Componentes principais do AWT

Apresentam-se aqui alguns dos principais componentes do AWT. Com eles, é possível construir a maioria dos formulários de interação básica, a partir de alguns componentes, como:

### Frame (ou janela)

Este componente (o frame) representa uma janela com seus elementos básicos (bordas, espaço interno, barra de títulos, além dos três botões padrão no canto superior direito – minimização, maximização e fechamento da janela). Ele é considerado um container cuja característica é poder receber internamente outros componentes.

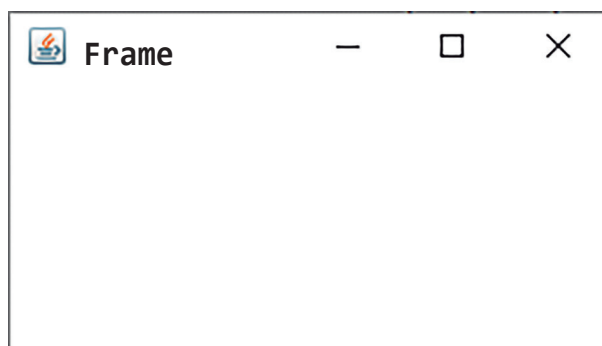


Figura 3 – Frame (vazio) gerado a partir da biblioteca AWT

### Panel (ou painel)

O panel (painel) é um componente que serve como uma janela interna, porém sem a barra de títulos ou os botões padrão de manipulação. Ele também é considerado um container, pois recebe outros componentes em seu interior.



Figura 4 – Panel gerado a partir da biblioteca AWT



## Observação

Não é possível abrir na tela (deixar visível) diretamente apenas um panel (sem que ele esteja inserido em frame). Apenas é possível visualizá-lo a partir de (dentro de) um frame, o que significa que sua visibilidade dependerá também da visibilidade do frame ao qual ele está inserido, direta ou indiretamente.

### Label (ou rótulo)

O label é todo texto que precede um campo para explicar e definir qual significado do conteúdo deve ser nele inserido. Em geral, qualquer texto no interior de uma janela-formulário em que não é possível permitir alteração de seu valor diretamente na tela do usuário (através do teclado) pode ser considerado um label (ou seja, criado a partir de um label).

### TextField (ou campo de texto simples)

O textfield é um campo de texto simples, mais conhecido como caixa de texto, que aceita apenas texto linear (no qual não é possível gerar mais de uma linha de texto).

### Button (ou botão)

O button é um botão de formulário que, quando acionado (por exemplo, ao clicar sobre ele com o mouse), inicia a execução de algum método (determinado evento).

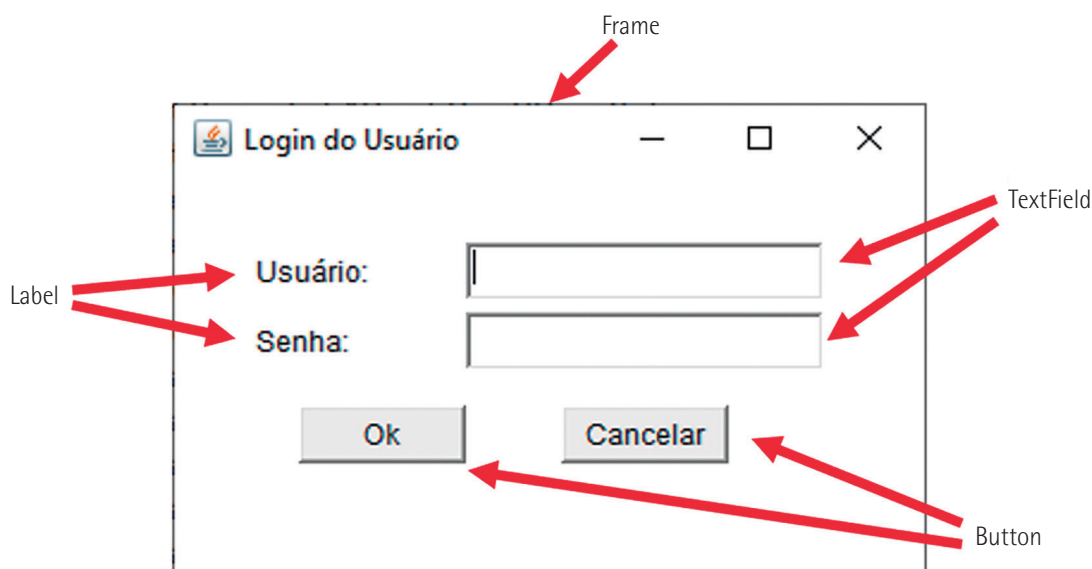


Figura 5 – Possíveis componentes gerados a partir da biblioteca AWT

## 1.2 O Swing

A biblioteca AWT utiliza componentes nativos do sistema operacional para construir suas telas. Isto fazia com que em cada plataforma as telas geradas tivessem diferenças, às vezes significativas, que prejudicavam o seu funcionamento ou sua visualização.

Com a evolução da tecnologia digital, a partir do lançamento da linguagem Java, novas funcionalidades foram adicionadas à biblioteca disponibilizada, de acordo com as necessidades dos sistemas que eram desenvolvidas.

Com o tempo, os desenvolvedores Java criaram então uma biblioteca de componentes específicos (o Swing), que não usava mais componentes gráficos nativos ao sistema operacional, passando a conter seus próprios componentes.

Essa evolução no visual dos componentes fez com que a tela vista em uma plataforma fosse a mesma em qualquer plataforma, homogeneizando os sistemas, deixando-os com igual aparência em todo micro que possuísse instalada uma máquina virtual Java (JVM). Além disso, utilizar a biblioteca Swing tornou mais rápida a renderização das telas (mecanismo de exibição de elementos gráficos) pelo compilador.



### Saiba mais

Para aprender mais sobre componentes GUI, leia o capítulo 11 da obra a seguir:

DEITEL, P.; DEITEL, H. *Java: como programar*. São Paulo: Pearson Education do Brasil, 2017.

### 1.2.1 Componentes principais do Swing

Para uma parte dos componentes básicos, a diferença da sintaxe dos nomes das classes dos componentes do Swing em relação às aquelas dos equivalentes do AWT é a presença da letra "J" antecedendo o seu nome, por exemplo:

- JFrame do Swing (equivalente ao frame do AWT);
- JPanel do Swing (equivalente ao panel do AWT);
- JLabel do Swing (equivalente ao label do AWT);
- JTextField do Swing (equivalente ao textfield do AWT);
- JButton do Swing (equivalente ao button do AWT).

Vale lembrar que nem toda classe de componente do Swing possui a característica de levar o mesmo nome do componente do AWT antecedido pela letra "J", como os componentes de seleção a seguir:

- JComboBox do Swing (equivalente ao choice do AWT);
- JCheckBox do Swing (equivalente ao checkbox do AWT) no qual a diferença está também na letra b da palavra "Box";
- JRadioButton do Swing (equivalente ao checkbox do AWT quando utilizado com o CheckboxGroup da mesma biblioteca).

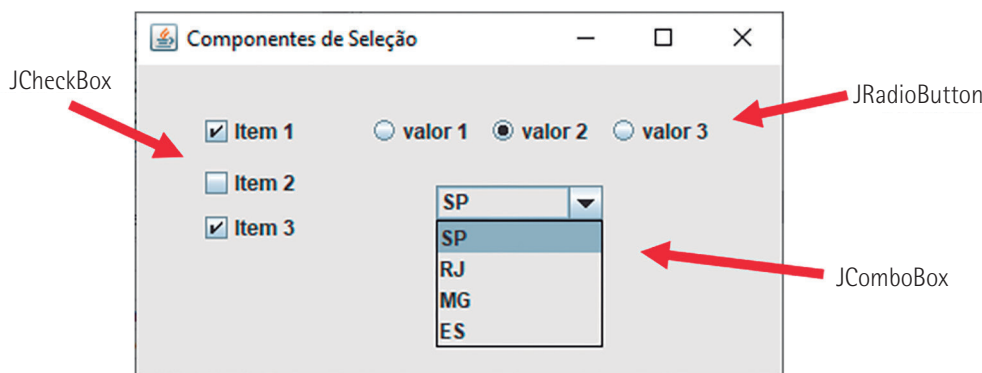


Figura 6 – Componentes de seleção gerados a partir da biblioteca Swing

Deve-se considerar também que a biblioteca Swing é mais completa, havendo componentes que não existem diretamente na lista da biblioteca AWT, tais como:

- JPasswordField: para campos de senha, ocultando o texto nela escrito, utilizando símbolos (normalmente um ponto • ou um asterisco \*);
- JEditorPane e JTextPane: campos de textos que podem ser estilizados (ou formatados) separadamente, assim como mostram textos descritos na linguagem HTML;
- Entre outros recursos e/ou componentes.



### Lembrete

Vimos que todo componente é uma classe Java, de forma que toda classe pode conter atributos (que definem as suas características e o local onde serão guardadas as informações dos objetos), métodos (que definem as ações que podem ou não ser efetivamente acionadas a partir do objeto em memória) e os métodos construtores (que estabelecem como as instâncias podem ser criadas na memória).



## 1.2.2 Criando a primeira tela

Um conceito essencial que devemos compreender é a ideia de que, com a herança, uma subclasse passa a ser a superclasse: se uma classe `Aluno` herda a classe `Pessoa`, podemos dizer que `Aluno` é `Pessoa`; se uma classe `Fusca` herda a classe `Carro`, então `Fusca` é `Carro`.

Assim, para que uma classe `Tela` (que será nossa primeira janela) se torne efetivamente um frame (uma janela), basta que ela herde a classe equivalente da biblioteca AWT (`Frame`) ou Swing (`JFrame`), e este será sempre o nosso primeiro passo na criação de uma janela.

```
import javax.swing.JFrame;

public class Tela extends JFrame {

}
```

A classe `Tela` do exemplo anterior já pode ser considerada efetivamente uma janela (window), já que ela herda a classe `JFrame` por meio do método `setVisible(...)` herdado da classe `JFrame`, ela já poderá ser visualizada.

O exemplo a seguir é o de um programa que, quando acionado, gera uma janela de 400 pixels de comprimento por 300 pixels de altura, distanciado a 200 pixels da lateral esquerda da tela do monitor, e a 100 pixels da parte superior da tela do monitor.

```
import javax.swing.JFrame;
public class Tela extends JFrame {
    public static void main(String[] args) {
        Tela tp = new Tela();
    }
    public Tela() {
        this.montarTela();
    }
    public void montarTela() {
        this.setBounds(200, 100, 400, 300);
        this.setVisible(true);
    }
}
```

De acordo com o exemplo citado, ao acionarmos a opção de menu "Run As – Java Application" do Eclipse para o seu programa, que possui o método "main", o compilador vai instanciar a classe `Tela`, de modo a rodar seu método construtor. Este, por sua vez, vai acionar o método "montarTela()" que possui duas ações:

- posicionar e dimensionar a janela (com o método "setBounds(...)");
- deixar a tela visível (com o método "setVisible(...)").

## Método setBounds(...)

Este método é utilizado para posicionar e dimensionar um componente na tela (ou em um container, o que será explicado posteriormente).

Sua sintaxe é:

```
componente.setBounds(posX, posY, tamHor, tamVer);
```



Figura 7 – Representação de um JFrame dimensionado

onde:

- **posX**: distância horizontal (em pixels) do canto superior esquerdo da margem do componente até a margem esquerda do container ao qual está inserido (no caso do frame da figura anterior, a referência é a própria tela do computador).
- **posY**: distância vertical (em pixels) do canto superior esquerdo da margem do componente até a margem superior do container ao qual está inserido (no caso do frame da figura anterior, a referência é a própria tela do computador).
- **tamHor**: comprimento (tamanho) horizontal (em pixels) do componente.
- **tamVer**: comprimento (tamanho) vertical (em pixels) do componente.

Este único método (`setBounds(posX, posY, tamHor, tamVer)`) pode substituir os dois outros que também poderiam ter sido utilizados:

- método `setLocation(posX, posY)`;
- método `setSize(tamHor, tamVer)`.

onde o método `setLocation(...)` apenas posiciona o componente na tela (ou em seu container), e o método `setSize(...)` somente define as dimensões do componente.

## Método `setVisible(...)`

Este método torna visível ou não, dependendo do valor inserido como parâmetro (que pode ser **true**, ou **false**), um componente na tela (ou em um container, elemento que será explicado posteriormente). Em seu parâmetro, colocamos o valor **true** se queremos que o componente esteja visível, e **false** se desejamos que ele não fique visível.

Para o frame, o valor default, ou seja, aquele que é atribuído automaticamente a este método (`setVisible(...)`), quando não o fazemos explicitamente, é o valor **false**, e por isso temos que colocar o comando (chamando este método) com o parâmetro **true** para o frame. Já para os componentes internos do frame, o valor default da característica de visibilidade é **true**, e, portanto, não é necessário redefini-lo ao longo do código.

Sua sintaxe é:

```
componente.setVisible(<valor_logico>);
```

onde o valor lógico pode ser **true** (verdadeiro) ou **false** (falso).



## Observação

Normalmente inserimos o comando (`this.setVisible(true)`) ao final do método que monta a tela após configurarmos e posicionarmos todos os componentes que aparecerão nela, e ao longo dos códigos do método construtor da classe.

## 2 APROFUNDAMENTO EM COMPONENTES COMO LAYOUT, MENUS E TABELAS

### 2.1 Layouts

O container é um componente que pode receber internamente outros (inclusive outros containers). O layout define como os componentes internos poderão estar dispostos em um container. A maioria dos layouts utilizados faz parte da biblioteca AWT.

Para trabalharmos com a criação de telas e formulários, o Java possui uma série de layouts específicos, com características que definem como devemos trabalhar (disponibilizar) os seus componentes internos.

Tanto o frame (ou o `JFrame`) quanto o panel (ou o `JPanel`) possuem um layout padrão, que para o frame é o Border Layout, e para o panel é o Flow Layout. Cada um desses layouts tem características próprias que serão descritas a seguir.

## 2.2 O layout nulo

Veremos agora que os layouts seguem padrões e configurações específicas que obrigam a disposição de componentes segundo as regras de cada um deles.

Para que possamos inserir componentes em qualquer posição da tela, e com qualquer tamanho, devemos definir o layout como nulo (**null**) da seguinte forma:

```
componente.setLayout (null);
```

Utilizando essa definição, é possível posicionar cada um dos componentes com o método `setBounds(posX, posY, tamHor, tamVer)`, que apenas indicará onde, e com qual tamanho, ele estará localizado na janela (tendo a própria janela como referência para as posições).

### Exemplo de aplicação

Como exemplo, vamos criar a seguinte tela (janela) de senha:

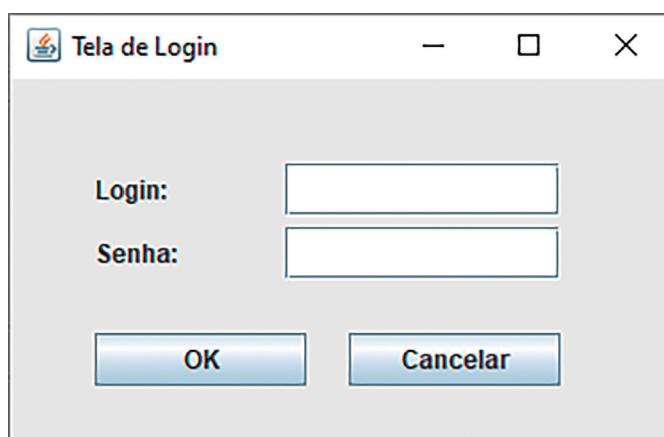


Figura 8 – Representação da tela de login

A qual pode ser gerada a partir do seguinte programa:

```
// Criação da Classe TelaLogin (Janela de Login)
import javax.swing.*;
// Lembrando que para nossa Classe "ser" uma Janela, ela tem que
// ...herdar um Frame (ou um JFrame)
public class TelaLogin extends JFrame {
    // Declarando Componentes como Atributos da Classe
    JLabel l1 = new JLabel("Login:");
    JTextField tf1 = new JTextField("");
    JLabel l2 = new JLabel("Senha:");
    JPasswordField tf2 = new JPasswordField ();
    JButton b1 = new JButton("OK");
    JButton b2 = new JButton("Cancelar");
    // O método main apenas instancia a Classe
    public static void main(String[] args) {
        TelaLogin t1 = new TelaLogin();
    }
}
```

```
}  
// O método construtor "constrói a Tela"  
public TelaLogin() {  
    this.setBounds(200, 100, 330, 210);  
    // Para aceitar componentes em localizações diversas:  
    this.setLayout(null);  
    this.setTitle("Tela de Login");  
    // Dimensionando cada um dos Componentes  
    l1.setBounds(40, 40, 80, 25);  
    l2.setBounds(40, 70, 80, 25);  
    tf1.setBounds(130, 40, 130, 25);  
    tf2.setBounds(130, 70, 130, 25);  
    b1.setBounds(40, 120, 100, 25);  
    b2.setBounds(160, 120, 100, 25);  
    // Adicionando os componentes à Janela (Tela)  
    this.add(l1);  
    this.add(l2);  
    this.add(tf1);  
    this.add(tf2);  
    this.add(b1);  
    this.add(b2);  
    // Por último, deixamos a tela visível  
    this.setVisible(true);  
}  
}
```



## Lembrete

O termo `this` utilizado no exemplo prévio indica o uso do elemento (seja um atributo ou um método) da própria classe, de forma que, apesar dos métodos acionados não existirem diretamente na própria classe, eles constam na classe herdada (JFrame).

Observações:

- A tela gerada pelo programa mencionado não nenhuma tem funcionalidade, já que este será um assunto a ser tratado em outro item posteriormente.
- Tanto na instância do label quanto na do button, pode-se inserir como parâmetro o texto que ele conterá:

```
JLabel lab = new JLabel("texto do label");
```

```
JButton bot = new JButton("texto do botão");
```

- A classe `JPasswordField` representa um campo de texto simples, porém com caracteres escondidos, sendo utilizado para a entrada de senhas de usuários.

## 2.2.1 Sequência para criação de telas com layout nulo

Consta adiante uma sequência de passos para a criação de uma janela com layout nulo:

- Criar a classe que vai representar a janela.
- Fazer com que essa classe herde a classe Frame (se for seguir a biblioteca AWT) ou JFrame (se for seguir a biblioteca Swing).
- Colocar todos os componentes que farão parte da janela, como atributos dessa classe (principalmente os campos de valores textuais ou de seleção).
- Elaborar um método de construção da janela e de seus componentes (pode ser o próprio método construtor da classe, ou um outro que será chamado por ele) e que deverá:
  - Definir a localização e o tamanho da janela.
  - Estabelecer o layout da janela (como nulo).
  - Determinar a localização e o tamanho de cada um de seus componentes.
  - Adicionar cada um dos componentes à janela (senão eles não aparecem).
  - Deixar a janela visível.

## 2.3 O Border Layout

O Border Layout é o layout padrão do Frame (AWT), ou do JFrame (Swing). Esse layout divide a janela em 5 partes: norte, sul, leste, oeste e central, segundo a figura a seguir:

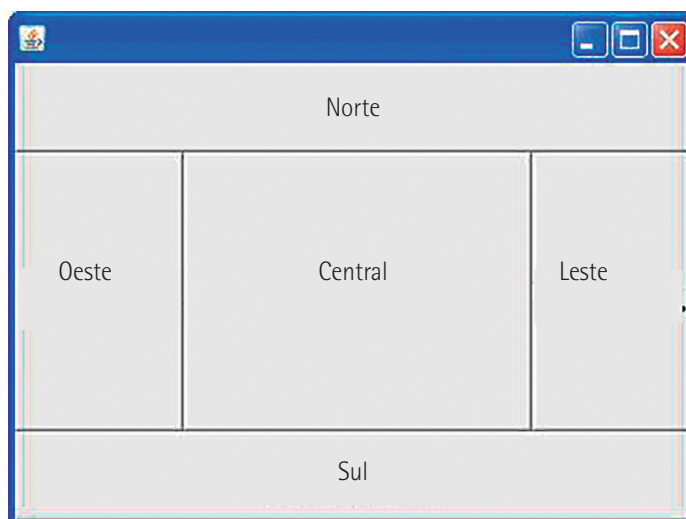


Figura 9 – Representação de um Border Layout

Em um Border Layout, não é necessário utilizar todas as partes das divisões nele existentes, de forma que se pode usar apenas algumas delas.

Em cada parte do layout, somente é possível adicionar um (e apenas um) componente. No entanto, o componente adicionado pode ser um container (Por exemplo: um Panel ou JPanel) dentro do qual dará para inserir vários outros.

Para adicionar um componente, precisamos utilizar o método `add(...)`, definindo em qual parte do Border Layout ele deverá ser inserido. Caso não se defina a parte onde o componente será adicionado, o compilador automaticamente o colocará na parte central (center).

Assim, para adicionar um componente, faz-se:

```
container.add(componente, BorderLayout.PARTE);
```

Por exemplo:

```
this.add(botOk, BorderLayout.SOUTH);
```

(o comando anterior adicionará o objeto botOK – botão OK – na parte sul do Border Layout do frame na própria janela)

## 2.4 O Grid Layout

O Grid Layout é um layout que divide a tela (seja um frame ou um panel) em partes de igual tamanho (ou dimensão), dispostos em linhas e colunas.



Figura 10 – Representação de um Grid Layout de 3 linhas por 4 colunas

Em cada parte do layout, somente é possível adicionar um (e apenas um) componente (que poderá ser um container, o qual pode conter vários outros componentes). Para adicionar um componente nas suas partes, existe uma sequência de adição, fazendo-se:

```
container.add(componente);
```

Por exemplo:

```
this.add(botOk);
```

Nesta linha de código (prévia), queremos adicionar um botão (que já deve ter tido sua configuração definida em linhas de código anteriores) à própria tela (**this**), quando a classe em questão é (através de herança) um frame ou panel.

A sequência de adição de componentes para um GridLayout deve ser: da esquerda para a direita, de cima para baixo (como na sequência numérica da figura de exemplo deste layout, mostrada previamente).

### 2.5 O Card Layout

O Card Layout é um layout que permite gerar várias telas de formulários, relatórios etc., mostrando a cada momento apenas uma das telas (aquela que vai ser utilizada). Ele viabiliza a troca de telas, de forma a permitir gerar um sistema completo, com suas várias telas de acesso para persistência de dados.

Nesse tipo de layout, um painel com layout Card Layout recebe vários outros painéis com layouts diversos.

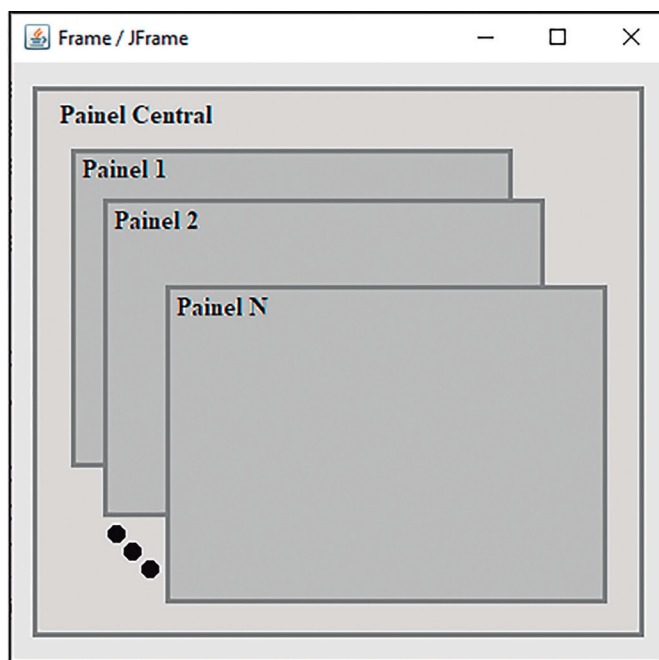


Figura 11 – Representação de um Card Layout



Para utilizar esse layout, geralmente inserimos (adicionamos) na parte central do frame (com seu Border Layout) um painel com Card Layout, definido como Painei Central. Neste Painei Central inserimos (adicionamos) vários outros painéis com layouts diversos, e que estarão visíveis à medida que se solicite sua visualização.

Não há uma sequência específica de inserção de painéis em um Card Layout, apenas devemos estar atentos ao fato de que o primeiro deles estará visível ao abrirmos a tela (ao acionarmos o sistema), isso ocorrerá com o primeiro painel adicionado ao Painei Central ao longo do código.

Exemplo:

Imagine que o objeto `pc` seja o Painei Central (adicionado ao centro do frame). A ele vamos adicionar 4 painéis representando quatro telas diferentes, de forma que, como o painel "p1" será o primeiro inserido, ele será o primeiro visível ao abrirmos (acionarmos) o nosso sistema.

```
// Declarando os Painéis
Panel pc = new Panel(); // Representando o Painei Central
Panel p1 = new Panel(); // Representando o Painei 1
Panel p2 = new Panel(); // Representando o Painei 2
Panel p3 = new Panel(); // Representando o Painei 3
Panel p4 = new Panel(); // Representando o Painei 4
...
this.add(pc); // adicionando o Painei Central (pc) no
              // ...centro (CENTER) do Border Layout
pc.setLayout(new CardLayout); // definindo o layout //...do painel central como
CardLayout
pc.add(p1, "p1");           // será o primeiro painel
                           //...a ser visualizado
pc.add(p2, "p2");
pc.add(p3, "p3");
pc.add(p4, "p4");
...
```

Perceba que ao adicionarmos um painel ao Painei Central, inserimos uma string como 2º parâmetro do método `add(...)`. Como sugestão, daremos o mesmo nome do objeto que está sendo adicionado, mas que na verdade pode ser qualquer nome, desde que cada painel tenha uma denominação. Esta string será necessária quando formos utilizar o comando de alteração de painel (troca de painel), e com isso alternar as telas do sistema.

Para mostrar determinado painel específico, a classe `CardLayout` traz métodos que alteram o painel que está sendo mostrado.

O método `show` da classe `CardLayout` permite indicar o Painei Central (o qual já contém todos os painéis adicionados a ele) e o nome (string) do painel específico que deverá ser mostrado a partir daquele momento.

```
objCardLayout.show(paineiCentral, nomeStringDoPainei);
```



## Observação

O objeto da classe `CardLayout`, que representará o layout da tela (que é um `CardLayout`), deverá ser montado (criado) a partir da captura do layout do Painel Central:

```
CardLayout objCardLayout = (CardLayout) (painelCentral.  
getLayout());
```

O casting realizado ao capturar o layout do Painel Central será o seguinte:

```
CardLayout cL = (CardLayout) (painelCentral.getLayout());  
  
cL.show(painelCentral, "nomeDoPainelASerMostrado");
```

Outros métodos da classe `CardLayout` também permitem visualizar os painéis inseridos no Painel Central, mas estes dependerão da sequência de adicionamento dada a eles, como: `first(painelCentral)`, `last(painelCentral)`, `next(painelCentral)`, `previous(painelCentral)`, que indicam, respectivamente, primeiro, último, próximo e anterior.

## 2.6 Trabalhando com menus

Os menus são componentes que permitem a navegação e o acionamento dos diversos recursos disponíveis em um sistema. Eles podem permanecer na parte superior de uma janela (barra de menus) ou aparecer como opções de ação à medida que clicamos com o botão direito do mouse (menus de contexto).

Vamos mostrar agora como gerar uma barra de menus, como dispor seu conteúdo, e como trabalhar sua funcionalidade. Para a criarmos na parte superior de uma janela, e seu conjunto de elementos, utilizamos basicamente quatro classes:

- **MenuBar OU JMenuBar:** que representam a barra superior de menus das bibliotecas AWT e Swing, respectivamente.
- **Menu OU JMenu:** que representam os elementos de menus, aqueles que fornecem novas opções de menus.
- **MenuItem OU JMenuItem:** que representam os elementos de menus que fornecem ou acionam alguma ação direta.
- **JSeparator:** que representa o separador de categorias de menus. Essa classe existe apenas na biblioteca Swing. Para adicionarmos um separador de categorias utilizando a biblioteca AWT, usamos os métodos `addSeparator()`, ou `insertSeparator(...)`, da classe `Menu`.

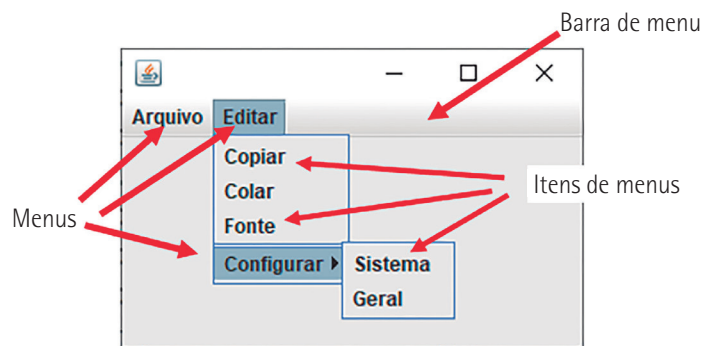


Figura 12 – Representação de uma janela com elementos de menus utilizando-se da biblioteca Swing

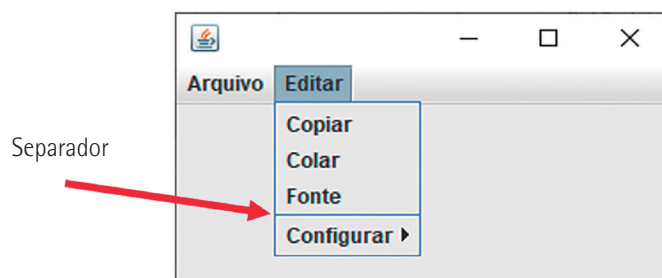


Figura 13 – Representação de uma janela com um separador de categorias de menus



## Saiba mais

Para aprender mais sobre menus e itens de menus, verifique o tutorial disponibilizado e administrado pela Oracle no seguinte link:

ORACLE. *The Java™ Tutorials* [s.d.]. Disponível em: <https://bit.ly/3S2sTL2>. Acesso em: 25 jul. 2022.

### 2.6.1 Inserindo os elementos de menus

De modo geral, os elementos de menus devem ser adicionados de acordo com a sua ordem de aparecimento, seja na barra de menus (da esquerda para a direita), seja em algum menu (de cima para baixo).

A barra de menus precisa ser definida no frame (ou no JFrame) através do método `setMenuBar(...)` ou do método `setJMenuBar(...)`, dependendo de qual biblioteca se está utilizando (se AWT ou se Swing).

Por exemplo: `this.setJMenuBar(objetoJMenuBar);`

Na barra de menus, adicionamos os elementos de menus que primeiro serão visualizados no formulário.

Por exemplo: `objetoJMenuBar.add(objetoJMenu);`

Em cada menu, adicionamos os elementos de menus (Menus e/ou MenuItem's) na sua ordem de aparecimento (de cima para baixo).

Por exemplo: `objetoJMenu.add(elementoMenu);`

Assim como adicionamos elementos, podemos adicionar uma linha de separação:

Por exemplo: `objetoJMenu.add(new JSeparator());`

## Exemplo de aplicação

Como exemplo, vamos gerar o código que monta a seguinte janela de menus:

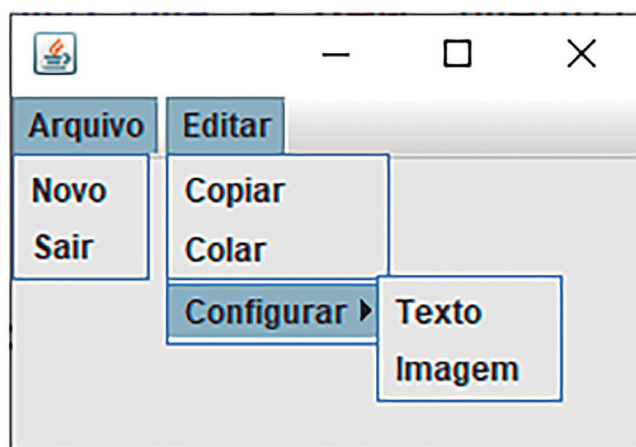


Figura 14 – Janela de exemplo de elementos de menus

Para construir esta janela, basta gerar o seguinte código (classe `TelaComMenus`):

```
// Importações necessárias
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JSeparator;
// Classe TelaComMenus
public class TelaComMenus extends JFrame {
    // A Barra de Menu
    JMenuBar mb = new JMenuBar();
    // Os Menus da Janela
    JMenu mArq = new JMenu("Arquivo");
    JMenu mEdit = new JMenu("Editar");
    JMenu mConfig = new JMenu("Configurar");
    // Os Itens de Menu da Janela
    JMenuItem miNovo = new JMenuItem("Novo");
    JMenuItem miSair = new JMenuItem("Sair");
    JMenuItem miCopiar = new JMenuItem("Copiar");
```

```
JMenuItem miColar = new JMenuItem("Colar");
JMenuItem miCTexto = new JMenuItem("Texto");
JMenuItem miCImg = new JMenuItem("Imagem");
// Separador
JSeparator mSep = new JSeparator();
// Método main para instanciar a Classe
public static void main(String[] args) {
    TelaComMenus tp = new TelaComMenus();
}
// Método construtor que constrói a Tela e seus Menus
public TelaComMenus() {
    this.setBounds(100, 100, 250, 170);
    // "setando" a barra de menu na janela
    // ... (para que a exista)
    this.setJMenuBar(mb);
    // adicionando os menus na barra de menus
    mb.add(mArq);
    mb.add(mEdit);
    // adicionando os elementos do menu "Arquivo"
    mArq.add(miNovo);
    mArq.add(miSair);
    // adicionando os elementos do menu "Editar"
    mEdit.add(miCopiar);
    mEdit.add(miColar);
    mEdit.add(mSep);
    mEdit.add(mConfig);
    // adicionando os elementos do menu "Configurar"
    mConfig.add(miCTexto);
    mConfig.add(miCImg);
    //-----
    this.setVisible(true);
}
}
```



## Lembrete

Na montagem da tela, a última ação é deixar a janela visível. Isso acontece para que não ocorra problemas na ocasião da construção e disponibilização dos componentes nas telas geradas em memória.

## 2.7 Trabalhando com tabelas

Uma forma de trabalhar com tabelas em Java é utilizar a estrutura do componente `JTable`, que possibilita inúmeras funcionalidades práticas ao trabalho com dados lidos de BD.

Para criarmos uma tabela de dados em Java, basicamente utilizamos três classes, duas delas definem efetivamente a estrutura e a disponibilização das informações na forma de tabela, enquanto a outra permite a visualização das várias informações da tabela, mesmo que não caibam em apenas uma tela. A barra de rolagem é um componente geral de telas que possibilita a visualização de

elementos que não estão visíveis por ocuparem um espaço maior do que o definido para amostrá-los, ou por se localizarem abaixo ou acima da linha visível da tela.

Sendo assim, as três classes principais utilizadas na confecção de tabelas são:

- **Classe `JTable`**: representa a estrutura da tabela.
- **Classe `DefaultTableModel`**: representa os itens da tabela (modelo), ou ainda, os elementos internos da tabela.
- **Classe `JScrollPane`**: representa um painel com a barra de rolagem, já que a `JTable`, por si mesma, não possui tal barra, e é muito comum precisarmos dela para ver todas as informações disponíveis na tabela.



### Observação

A classe `JScrollPane` representa um painel com barra de rolagem, ou seja, é um container que recebe internamente outros componentes (ou containers), e disponibiliza os eixos de barra de rolagem para permitir a visualização dos elementos que ficam fora do campo de visão definido pelo tamanho da janela (ou da área estabelecida para amostragem dos elementos).

### 2.7.1 Elementos e características da construção de uma tabela

Apresentaremos agora como devemos preparar a instância das classes que permitem a construção de uma tabela.

No parâmetro do construtor do modelo (`DefaultTableModel`) inclui-se um array de strings com os títulos de cada coluna.

```
objDTM = new DefaultTableModel(arrayStringTit, 0);
```

Observação: `objDTM` é o nome do objeto que representa a classe `DefaultTableModel`. Após a sua instância, pode-se adicionar novas colunas com o método `objDTM.addColumn("TítuloDaColuna")`;

Cada linha de informações (dos dados) da tabela deve ser adicionada ao modelo utilizando-se o método `addRow(...)`, inserindo em seu parâmetro o array de strings com as informações da linha (um array com a mesma quantidade de informações que de títulos):

```
objDTM.addRow(arrayStringLinha);
```

Na instância da tabela (`JTable`), ou ainda no parâmetro de seu construtor, inclui-se o modelo (`DefaultTableModel`) criado.

```
objJT = new JTable(objDTM);
```

Observação: `objJT` é o nome do objeto que representa a classe `JTable`.

Para inserir a tabela ao painel de barra de rolagem, podemos utilizar dois possíveis caminhos:

- Na instância do painel da barra de rolagem, ou ainda, no parâmetro de seu construtor (`JScrollPane`), inclui-se o objeto representante da estrutura da tabela (`JTable`):

```
objJScrollPane = new JScrollPane(objJT);
```

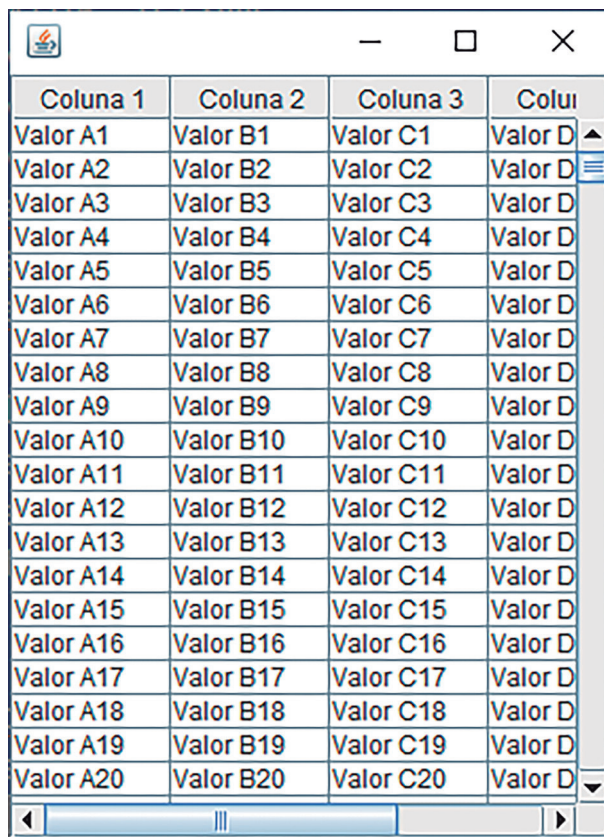
- Utilizando o método `setViewportView(...)` da classe `JScrollPane` (em seu objeto já instanciado), e recebendo como parâmetro a estrutura da tabela (`JTable`):

```
objJScrollPane.setViewportView(objJT);
```

Observação: `objJScrollPane` é o nome do objeto que representa a classe `JScrollPane`.

## Exemplo de aplicação

Agora veremos o exemplo da janela com uma tabela:



Coluna 1	Coluna 2	Coluna 3	Coluna 4
Valor A1	Valor B1	Valor C1	Valor D1
Valor A2	Valor B2	Valor C2	Valor D2
Valor A3	Valor B3	Valor C3	Valor D3
Valor A4	Valor B4	Valor C4	Valor D4
Valor A5	Valor B5	Valor C5	Valor D5
Valor A6	Valor B6	Valor C6	Valor D6
Valor A7	Valor B7	Valor C7	Valor D7
Valor A8	Valor B8	Valor C8	Valor D8
Valor A9	Valor B9	Valor C9	Valor D9
Valor A10	Valor B10	Valor C10	Valor D10
Valor A11	Valor B11	Valor C11	Valor D11
Valor A12	Valor B12	Valor C12	Valor D12
Valor A13	Valor B13	Valor C13	Valor D13
Valor A14	Valor B14	Valor C14	Valor D14
Valor A15	Valor B15	Valor C15	Valor D15
Valor A16	Valor B16	Valor C16	Valor D16
Valor A17	Valor B17	Valor C17	Valor D17
Valor A18	Valor B18	Valor C18	Valor D18
Valor A19	Valor B19	Valor C19	Valor D19
Valor A20	Valor B20	Valor C20	Valor D20

Figura 15 – Janela de exemplo de tabela com 400 linhas de dados e barra de rolagem

```
//Importações necessárias
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
//A Classe Tabela
public class Tabela extends JFrame{
    //Atributos
    JTable jt; //Estrutura da tabela
    DefaultTableModel dtm; //Objeto que permite manipulação
    JScrollPane jspane; //Painel com Barra de Rolagem
    //Método main para instanciar a Classe
    public static void main(String[] args) {
        Tabela tb = new Tabela();
    }
    //Método Construtor - Construindo a Janela com uma Tabela
    public Tabela() {
        //Abrindo a janela bem no centro da tela
        this.setSize(300, 400);
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(this.EXIT_ON_CLOSE);
        //Construindo o array de títulos
        String[] arrTitulo = {"Coluna 1", "Coluna 2", "Coluna 3",
                                "Coluna 4", "Coluna 5"};
        //Instanciando os elementos da tabela
        dtm = new DefaultTableModel(arrTitulo, 0);
        jt = new JTable(dtm);
        jspane = new JScrollPane(jt);
        this.add(jspane); // Inserido na parte central da Janela
        //.....
        //Desvinculando o tamanho das células com o da Janela
        jt.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
        //Mostrando a Barra de rolagem horizontal
        jspane.setHorizontalScrollBarPolicy(
            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        //.....
        //Construindo o array de dados para 400 linhas de tabela
        String[] arrDados = new String[5];
        for (int x = 1; x <= 400; x++) {
            arrDados[0] = "Valor A" + x;
            arrDados[1] = "Valor B" + x;
            arrDados[2] = "Valor C" + x;
            arrDados[3] = "Valor D" + x;
            arrDados[4] = "Valor E" + x;
            //Adicionando uma linha de dados na tabela
            dtm.addRow(arrDados);
        }
        //Deixando a Janela visível
        this.setVisible(true);
    }
}
```





## Resumo

Nesta unidade, vimos que as interfaces gráficas nos possibilitam construir sistemas que permitem a interação do usuário com o sistema construído. Na construção das interfaces, existem duas grandes bibliotecas (conjuntos de classes do Java) que o desenvolvedor pode utilizar, em que encontra componentes usuais, já com suas funcionalidades próprias, além dos elementos que possibilitam o funcionamento geral da interação, como os possíveis eventos a ser acionados, dependendo da ação do usuário.

Entendemos que cada elemento visual é considerado um componente, sendo a utilização e a disponibilização desses componentes os campos de texto, os campos de senha, os labels, os botões, assim como o frame, ou o panel, que são considerados containers, já que podem receber internamente outros componentes. Desta forma, a fim de verificar como os componentes seriam inseridos ou disponibilizados nesses containers, observamos as definições dos possíveis layouts que cada container poderia assumir.

Para muitos elementos básicos utilizados na construção das interfaces gráficas, existem algumas funcionalidades ligadas a eles (que podem ser acionadas pelo usuário), as quais chamamos de eventos, mas que na linguagem Java usam outros elementos básicos conceituais, como as interfaces e seus métodos abstratos.

Por fim, outros elementos de tela que apresentamos foram os componentes de janela, as tabelas e os menus, componentes esses que têm uma particularidade em sua funcionalidade. Para as tabelas, trabalha-se basicamente com três classes, sendo a Table aquela que a define como um todo, a DefaultTableModel, que permite a manipulação dos dados da tabela, e a JScrollPane, que é um painel (container) que receberá a tabela como um todo e tem a barra de rolagem para melhor visualização das informações dela. Quanto aos elementos de menus, exibimos as três classes básicas: a barra de menu (MenuBar), que representa a barra superior de menus de uma janela; os menus (Menu), que nas janelas fornecem outras opções de menus; e os itens de menus (MenuItem), que ao serem acionados geram (ou iniciam) alguma ação no sistema.



## Exercícios

**Questão 1.** (Marinha/2018) Leia o texto a seguir, a respeito da biblioteca Swing.

Embora os programas de console sejam ótimos no ensino dos aspectos básicos de Java e em alguns tipos de programas, como em códigos no lado do servidor, na vida real, a maioria dos aplicativos é baseada em uma interface gráfica de usuário (GUI). Uma das GUI Java mais amplamente usadas é aquela baseada em Swing.

O Swing define uma coleção de classes e interfaces que dá suporte a um rico conjunto de componentes visuais, como botões, campos de texto, painéis de rolagem, caixas de seleção, árvores e tabelas, para citar alguns. Coletivamente, esses controles podem ser usados na construção de interfaces gráficas poderosas e, ainda assim, fáceis de usar. Devido ao seu uso disseminado, a biblioteca Swing é algo que todos os programadores de Java devem conhecer.

O Swing não existia nos primórdios da linguagem Java, ele foi uma resposta às deficiências presentes no subsistema de GUI original da linguagem: Abstract Window Toolkit (AWT).

Adaptada de: SCHILDT, H. *Java para iniciantes: crie, compile e execute programas Java rapidamente*. Porto Alegre: Bookman, 2015.

A GUI apresenta um mecanismo amigável ao usuário para interagir com um aplicativo. Entre os componentes GUI Swing, do pacote javax, qual fornece uma lista drop-down de itens, a partir da qual o usuário pode fazer uma seleção clicando em um item ou, possivelmente, digitando na caixa?

A) JButton.

B) JCheckBox.

C) JPanel.

D) JComboBox.

E) JLabel.

Resposta correta: alternativa D.

### Análise das alternativas

A) Alternativa incorreta.

Justificativa: a classe JButton fornece um dos controles mais usados de Swing, que é o botão de ação (uma instância de JButton).

O JButton herda a classe abstrata AbstractButton, que define a funcionalidade comum a todos os botões. Os botões de ação de Swing podem conter texto, imagem ou ambos. O JButton do Swing é equivalente ao button do AWT.

B) Alternativa incorreta.

Justificativa: a classe JCheckBox fornece a caixa de seleção. Em Swing, uma caixa de seleção é um objeto de tipo JCheckBox. O JCheckBox herda o AbstractButton e o JToggleButton. Portanto, caixa de seleção é, essencialmente, um tipo específico de botão.

C) Alternativa incorreta.

Justificativa: a classe JPanel fornece um tipo de container, que é um componente que pode receber, internamente, outros (inclusive outros containers). Podemos dizer, portanto, que um container serve para agrupar outros componentes.

D) Alternativa correta.

Justificativa: a classe JComboBox fornece um menu suspenso, componente que combina um botão e uma lista do tipo drop-down. O usuário, então, deverá ser capaz de selecionar um dos valores dessa lista.

E) Alternativa incorreta.

Justificativa: a classe JLabel fornece um rótulo, que é um componente que exibe informações. O rótulo é o componente mais simples de Swing, visto que é passivo, ou seja, não responde às entradas do usuário.

**Questão 2.** (FCC/2017, adaptada) Considere a classe Java a seguir.

```
import java.awt.*;
import javax.swing.*;

public class Tela extends JI{

    public Tela() {

        .II.;

        setSize(500, 300);
        setLocation(300, 200);
    }

    public static void main(String[] args) {
        Tela t = new Tela();
        t.setVisible(true);
    }
}
```

Figura 16

Adaptada de: <https://bit.ly/3RsVOap>. Acesso em: 8 set. 2022.

A instrução da lacuna I corresponde a uma classe que oferece um container de nível superior. A instrução da lacuna II define um layout com três linhas e duas colunas.

As lacunas I e II são, correta e respectivamente, preenchidas com:

A) JPanel e `setLayout(new GridBagLayout(3, 2))`.

B) JPanel e `setLayout(new FlowLayout(3, 2))`.

C) JFrame e `setLayout(new GridLayout(3, 2))`.

D) JFrame e `setLayout(new BorderLayout(3, 2))`.

E) JFrame e `setLayout(new GridBagLayout(3, 2))`.

Resposta correta: alternativa C.

### Análise da questão

A biblioteca Swing define dois tipos de containers.

Primeiramente, temos aqueles de nível superior, sendo JFrame um deles. Esses containers herdam as classes Component e Container, de AWT. Conforme o nome sugere, um container de nível superior deve estar no topo de uma hierarquia de contenção. Portanto, ele não fica contido em nenhum outro container. Além disso, toda hierarquia de contenção deve começar com um container de nível superior. O mais usado para aplicativos é o próprio JFrame. Logo, a instrução da lacuna I corresponde à JFrame.

O segundo tipo de container com suporte em Swing é de container leve, como o JPanel. Esse tipo herda JComponent. Geralmente os containers leves são usados para organizar e gerenciar coletivamente grupos de componentes relacionados, porque um container leve pode estar contido dentro de outro.

Em relação à instrução da lacuna II, precisamos, ali, definir um layout com três linhas e duas colunas. A classe GridLayout dispõe os componentes de um container em uma grade retangular. Ela divide a tela em partes de igual tamanho, dispostas em linhas e colunas. O construtor `GridLayout(int linhas, int colunas)` cria um GridLayout com o número especificado de linhas e colunas, sem nenhum gap entre os componentes.

O método `setLayout()` permite que seja ajustado o layout de um container, seja para BorderLayout, seja GridLayout ou qualquer outro desejado. Logo, para definir um layout com 3 linhas e 2 colunas, temos que o trecho II será dado por:

```
setLayout(new GridLayout(3, 2))
```