

## Unidade II

**Componentes no Java SE**

- Java
- Java SE
- JavaBeans
- Componentes Gráficos

**Java**

- Linguagem de Programação Java
  - Orientada a objetos
  - Possui um conjunto amplo de APIs
  - Multi-plataforma: *Java Virtual Machine (JVM)*
  - Integrada à Internet: *applets, JSP, Servlets, ..*
  - De fácil aprendizagem
  - Bem aceita por programadores e empresas
  - Suportada por vários fabricantes de software
  - Possui suporte para componentes
    - *JavaBeans*
    - *Enterprise JavaBeans*

**Java**

- Java está disponível em três edições:
  - Java ME (*Micro Edition*): para PDAs, celulares e outros dispositivos com pouca memória e poder de processamento limitado
  - Java SE (*Standard Edition*): versão padrão do Java, com tudo que o usuário comum necessita
  - Java EE (*Enterprise Edition*): versão mais completa, para empresas utilizarem em seus servidores

**Java SE**

- A plataforma Java *Standard Edition (Java SE)* oferece suporte a:
  - Serviços gerais como nomeação (JNDI), bancos de dados (JDBC), segurança (*JavaSecurity*), etc.
  - APIs para comunicação e remota:
    - Comunicação local usando pipes
    - Comunicação remota usando sockets
    - Chamadas remotas de métodos usando Java RMI ou CORBA
  - Componentes: *JavaBeans*

**Java SE**

- *Java Naming and Directory Interface (JNDI)*
  - Associa nomes e atributos a objetos Java
  - Permite a procura de objetos por nome ou atributos
- *Java DataBase Connectivity (JDBC)*
  - Permite que aplicações Java efetuem consultas em SQL em bancos de dados relacionais
  - *Drivers JDBC* permitem acesso aos BDs

**Java SE**

- *JavaSecurity*
  - Fornece suporte a criptografia de dados
  - Permite a criação e a manipulação de chaves, certificados e listas de controle de acesso
- Pipes
  - Canais de comunicação locais e unidirecionais
  - Ligam duas threads na mesma máquina virtual
  - Pacote `java.io.*`

## Java SE

- **Sockets**
  - Representam uma porta de comunicação associada a uma aplicação
  - Podem usar vários protocolos: TCP, UDP, etc.
  - Pacote java.net.\*
- **RMI (*Remote Method Invocation*)**
  - Segue o modelo Cliente/Servidor
  - Fornece um suporte simples para RPC
  - Permite que um objeto Java chame métodos de outro objeto Java rodando em outra máquina virtual

## Java SE

- **CORBA (*Common Object Request Broker Architecture*)**
  - Padrão da OMG (*Object Management Group*)
  - Permite efetuar chamadas remotas de métodos em sistemas abertos, distribuídos e heterogêneos
    - Diferentes máquinas, sistemas operacionais e linguagens de programação
  - Fornece um suporte completo para aplicações distribuídas orientadas a objetos

## JavaBeans



- **JavaBeans**
  - São componentes escritos em Java
  - Situados na camada de aplicação
  - Podem ser usados em aplicações, *applets*, servlets, páginas JSP, ...
  - API JavaBeans: java.beans.\*
- **JavaBeans possuem:**
  - Métodos e atributos, como qualquer classe Java
  - Propriedades: modificadas em tempo de projeto

## JavaBeans

- **Comunicação entre Beans**
  - Chamadas de métodos locais
  - Canais de eventos locais
    - Produtor: envia objetos java.util.EventObject
    - Consumidor: implementa java.util.EventListener
  - Não possui suporte nativo para comunicação remota

## JavaBeans

- **JavaBeans seguem os seguintes padrões:**
  - São classes públicas
  - Possuem um construtor sem parâmetros
  - Nomes de métodos para acesso a propriedades e eventos:
    - Propriedade *X* acessada por métodos:
      - *setX()* e *isX()* se *X* for do tipo *boolean*
      - *setX()* e *getX()* para qualquer outro tipo
    - Tratador do evento *Y* registrado com o método *addYListener()* e removido com *removeYListener()*

## JavaBeans

- **Métodos de acesso a propriedades**
  - Atributo (opcional)

```
private Tipo propriedade ;
```
  - **Setter** (para propriedades modificáveis)

```
public void setPropriedade (Tipo propriedade) {
    this.propriedade = propriedade ;
}
```
  - **Getter**

```
public Tipo getPropriedade () {
    return this.propriedade ;
}
```

## JavaBeans

### ■ Evento

```
import java.util.EventObject;
public class MyEvent extends EventObject {
    private Tipo valor;
    public MyEvent (Object source, Tipo valor) {
        super(source);
        this.valor = valor;
    }
    public Tipo getValor () {
        return this.valor;
    }
}
```

## JavaBeans

### ■ Interface de um tratador de eventos

```
import java.util.EventListener;
public interface MyEventListener extends EventListener {
    public void myHandler (MyEvent evt);
    ...
}
```

## JavaBeans

### ■ Lista que armazena tratadores de um evento

```
protected javax.swing.event.EventListenerList my ListenerList =
    new javax.swing.event.EventListenerList();
```

### ■ Método que registra um tratador de evento

```
public void addMyEventListener(MyEventListener listener) {
    my ListenerList.add(MyEventListener.class, listener);
}
```

### ■ Método que remove um tratador de evento

```
public void removeMyEventListener(MyEventListener lnr) {
    my ListenerList.remove (my EventListener.class, lnr);
}
```

## JavaBeans

### ■ Método de disparo de evento

```
void fireMyEvent( Tipo valor) {
    Object[] list = myListenerList.getListenerList();
    // Cada listener ocupa 2 posições na lista:
    // nome da classe e a instância
    for (int i = 0; i < list.length; i += 2) {
        if ((list[i] == MyEventListener.class) {
            ((MyEventListener) list[i+1]).myHandler(
                new MyEvent(this,valor));
        }
    }
}
```

Obs.: invocar o método sempre que o evento ocorrer

## JavaBeans

### ■ Características adicionais dos JavaBeans

- Salvar estado: interface java.io.Serializable
- Controle de concorrência: palavra-chave synchronized; pacote java.util.concurrent.\*
- Segurança: pacote java.security.\*
- Contêineres para acesso à plataforma e a seus serviços: pacote java.beans.beancontext.\*

## JavaBeans

### ■ Detalhes sobre a interface de JavaBeans são obtidos:

- Usando a API java.lang.reflect e buscando pelos nomes de métodos padronizados para JavaBeans
  - get<Atributo>, set<Atributo>
  - add<Evento>Listener, remove<Evento>Listener
- Através da interface java.beans.BeanInfo, que deve ser implementada por uma classe chamada <NomeDoBean>BeanInfo

## JavaBeans

- **Vantagens e Limitações dos JavaBeans**
  - Beans são reutilizáveis e configuráveis
  - São fáceis de usar e de compor com outros Beans
  - São mais fáceis de manter e distribuir que classes
  - Seu desenvolvimento é um pouco mais complexo que o desenvolvimento de classes e *packages* Java

## JavaBeans

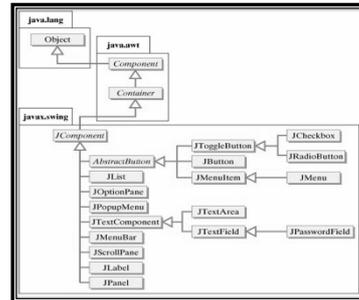
- **Distribuição e Implantação**
  - Beans são distribuídos em arquivos JAR
  - Arquivos JAR devem conter uma descrição do Bean
  - Para implantar o Bean, basta ter o arquivo JAR
  - Depois de implantados, os Beans podem ser configurados e compostos com outros componentes usando ferramentas de desenvolvimento

## Componentes Gráficos

- **Componentes gráficos da API do Java**
  - AWT e Swing possuem JavaBeans gráficos (mas nem todos os JavaBeans são gráficos!)
  - Propriedades alteram a aparência ou o comportamento do componente
  - Eventos são 'contidos': se propagam somente em uma janela/contêiner da interface gráfica

## Componentes Gráficos

- **Componentes do Swing**



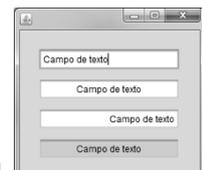
## Componentes Gráficos

- **JLabel: rótulo (texto e/ou imagem)**
  - Principais propriedades:
    - font
    - icon
    - text
    - background, foreground
    - {horizontal,vertical}Alignment
  - Principais eventos:
    - mouse{Clicked, Dragged, Entered, Exited, Moved, Pressed, Released, wheelMoved}



## Componentes Gráficos

- **JTextField: campo de texto**
  - Principais propriedades:
    - font
    - columns
    - text, editable
    - horizontalAlignment
    - background, foreground
  - Principais eventos:
    - actionPerformed, mouse\*
    - key{Typed, Pressed, Released}



## Componentes Gráficos

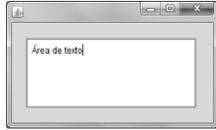
### ■ JTextArea: área de texto

#### ■ Principais propriedades:

- font
- text, editable
- lineWrap
- rows, columns
- background, foreground

#### ■ Principais eventos:

- key\*
- mouse\*



## Componentes Gráficos

### ■ JButton: botão de ação

#### ■ Principais propriedades:

- font
- icon
- text, mnemonic
- background, foreground

#### ■ Principais eventos:

- actionPerformed
- key\*
- mouse\*



## Componentes Gráficos

### ■ JCheckBox: caixa de seleção

### ■ JRadioButton: botão de opção

#### ■ Principais propriedades:

- font, text, mnemonic
- background, foreground
- buttonGroup, selected

#### ■ Principais eventos:

- actionPerformed
- key\*, mouse\*
- stateChanged



## Componentes Gráficos

### ■ JList: lista

#### ■ Principais propriedades:

- font
- background, foreground
- model
- selectionMode

#### ■ Principais eventos:

- key\*, mouse\*
- valueChanged



## Componentes Gráficos

### ■ JComboBox: caixa de combinação

#### ■ Principais propriedades:

- font
- background, foreground
- model
- selectedIndex, selectedItem

#### ■ Principais eventos:

- actionPerformed
- key\*, mouse\*



## Componentes Gráficos

### ■ Contêineres

- JFrame: janela
- JPanel: painel
- JTabbedPane: painel tabulado
- JScrollPane: painel de rolagem
- JToolBar: barra de ferramentas
- etc.