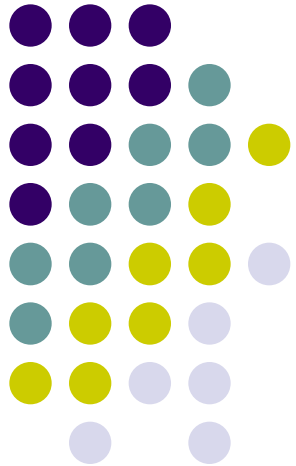
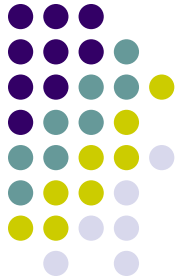
Swing

Alberto Costa Neto
DComp - UFS

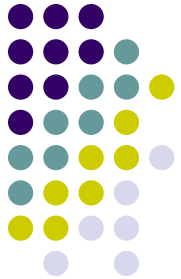


Swing
[Conteúdo]



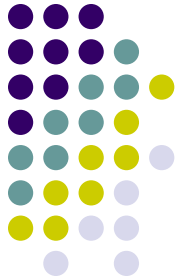
- Introdução
- AWT
- AWT x Swing
- Hierarquia de Componentes
- JLabel e Icon
- JButton
- JCheckBox
- JRadioButton e ButtonGroup
- JTextField e JPasswordField

Swing
[Conteúdo]



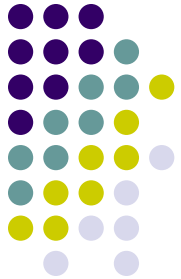
- JFrame
- JPanel
- Toolkit
- Gerenciadores de Layout
- Tratamento de Eventos
- JList
- JComboBox
- JTextArea
- Menu
- Bibliografia

Swing
[Introdução]



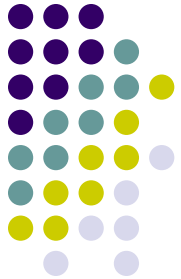
- Desde a versão 1.0, Java já tem suporte a interface gráfica, chamada Abstract Window Toolkit (AWT)
- Na versão 1.1 houve uma melhora da manipulação de eventos (ainda em Swing)
- Swing foi incorporada à versão 1.2 do Java
- Há uma versão que pode ser baixada e executada em ambientes Java 1.1

Swing
[AWT]



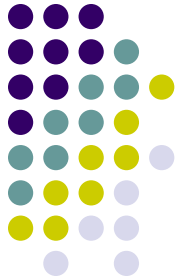
- AWT define componentes abstratos que utilizam as implementações dos componentes de interface gráfica oferecidas pela plataforma em execução
- Tem um bom desempenho
- Adapta-se ao estilo (“Look and Feel”) e comportamento da plataforma

Swing
[AWT]



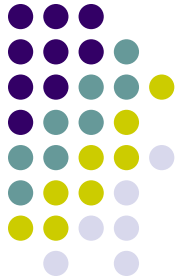
- Depender de componentes da plataforma execução tem desvantagens:
  - Bugs nos componentes nativos
  - Conjunto de componentes torna-se restrito
  - “Write Once, Debug Everywhere”

Swing
[AWT x Swing]



- Os componentes Swing são “pintados” sobre janelas em branco, diferente de AWT
- Swing oferece consistência entre plataformas
  - Componentes têm a mesma aparência e o mesmo comportamento em plataformas diferentes
  - “Write Once, Run Anywhere”
- Swing tem um conjunto mais rico e conveniente de elementos de interface

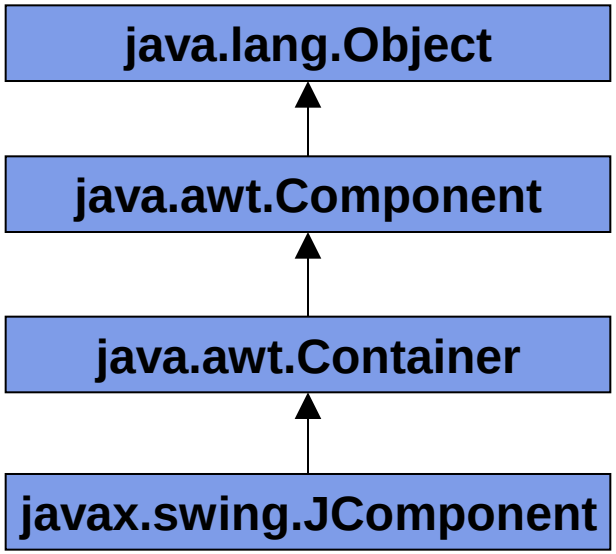
Swing
[AWT x Swing]



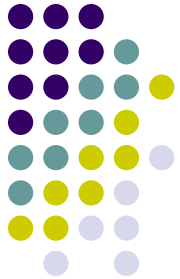
- Swing depende muito pouco da plataforma, tendo menor propensão a bugs
- Uma desvantagem de Swing sobre AWT é o desempenho inferior
- Exemplo
  - SwingSet2 (demo do JDK)

Swing
[Hierarquia de Componentes]

- Component é a superclasse abstrata dos componentes AWT
- Container é um componente que pode conter outros componentes AWT
- JComponent é a classe base para os componentes Swing

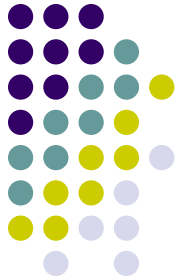


Swing
[JLabel e Icon]



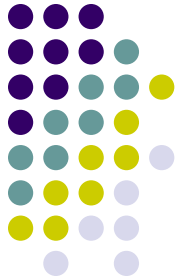
- Subclasse de JComponent que representa um rótulo
- Propriedades (manipuladas usando get/set):
  - Text, Icon, HorizontalAlignment, VerticalAlignment, HorizontalTextPosition e VerticalTextPosition
- Icon e ImageIcon
- Exemplo
  - JLabelDemo

Swing
[JButton]



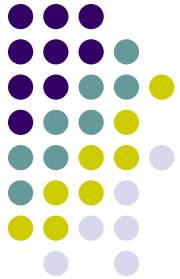
- A classe javax.swing.JButton oferece uma implementação padrão de um botão
- Gera um evento quando o usuário o clica
- setEnable permite (des)habilitá-lo
- setText especifica o texto do botão
- Exemplo
  - JButtonDemo

Swing
[JCheckBox]



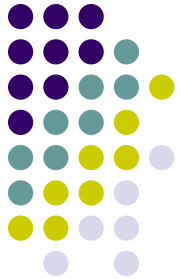
- javax.swing.JCheckBox mostra um item marcado ou desmarcado
- O método isSelected indica se está marcado e setSelected pode (des)marcar
- Tem um ancestral comum com JButton (javax.swing.AbstractButton), tendo atributos e comportamento semelhantes
- Exemplo
  - JCheckBoxDemo

Swing
[JRadioButton e ButtonGroup]



- JRadioButton são usados em grupo, ou seja, somente um botão por vez fica selecionado
- ButtonGroup tem métodos para adicionar, remover e obter os botões do grupo
- Exemplo
  - JRadioButtonDemo

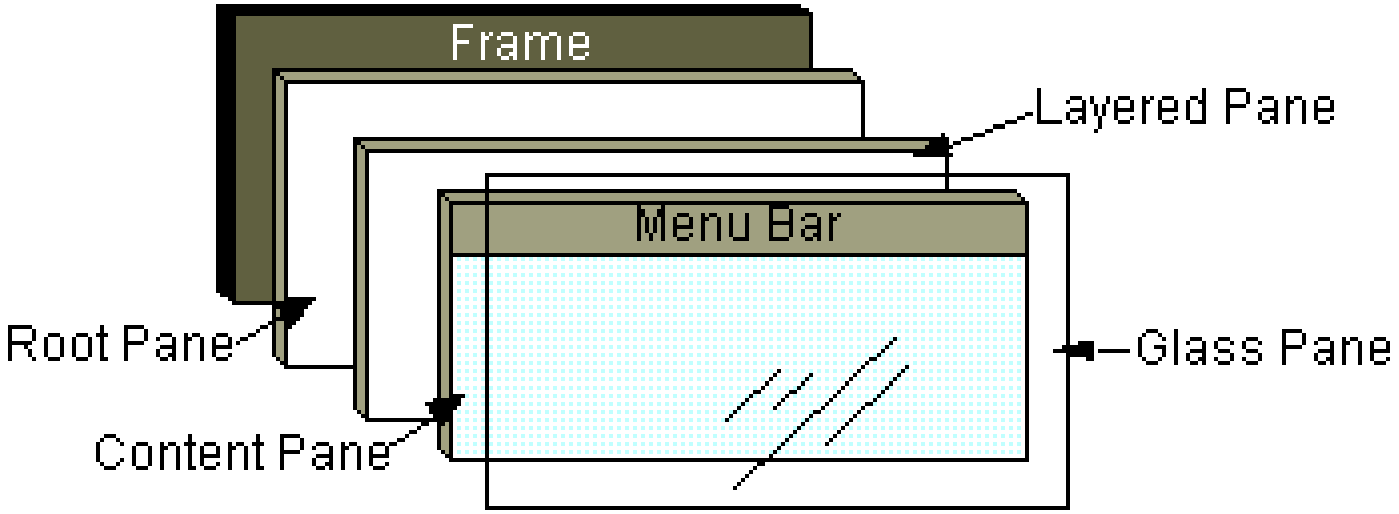
Swing
[JTextField e JPasswordField]



- JTextField permite a edição de uma linha de texto (campo)
- JPasswordField é uma subclasse de JTextField que não exibe os caracteres digitados (usa um caractere de eco)
  - O método getPassword retorna um char[ ] com os caracteres digitados
  - setEchoChar muda o caractere de eco
- Exemplo
  - JTextFieldDemo

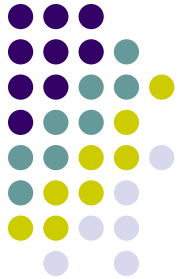
Swing
[JFrame]

- Versão estendida de Frame para Swing
- Não é totalmente compatível com Frame
- Estrutura diferente de Frame



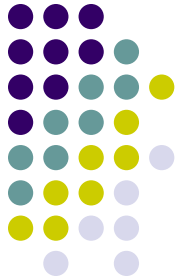
  - O Content Pane é onde os componentes e o menu são adicionados

Swing
[JFrame]



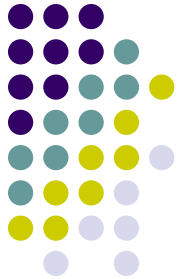
- Principais Métodos
  - getContentPane (retorna Container)
  - pack, getSize, setSize, setResizable
  - isVisible, setVisible, show, isShowing, dispose, hide, toFront e toBack
  - addWindowListener e setDefaultCloseOperation
  - getLocation, setLocation, getLocationOnScreen, getBound e setBounds
  - setTitle e setIconImage

Swing
[JFrame]



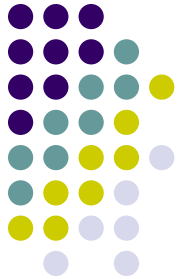
- Adicionar componentes
  - <jframe>.getContentPane().add(<comp>)
  - É possível alterar o container usando o comando
    - <jframe>.setContentPane(<container>)
- Remover componentes
  - <jframe>.getContentPane().remove(<comp>)

Swing
[JPanel]



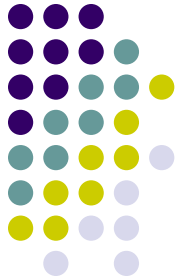
- É um Container genérico que requer poucos recursos
- Ancestral direto de JComponent, que por sua vez é subclasse de java.awt.Container
- Serve para agrupar e organizar componentes
- O gerenciador de layout padrão é FlowLayout
- Principais métodos
  - add, remove, getLayout e setLayout
  - getComponentCount e getComponents

Swing
[Toolkit]



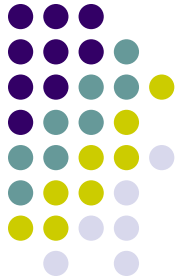
- Segue o padrão de projeto Abstract Factory
- As suas subclasses implementam vários factory methods (createButton, createLabel, etc)
- Principais métodos
  - getDefaultToolkit
  - getScreenSize
  - getImage
  - getScreenResolution
  - getFontList
  - beep

Swing
[Gerenciadores de Layout]



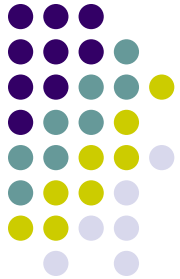
- Swing e AWT trabalham com uma forma diferente de organização de componentes
- Ao adicionar um componente em um Container, não indicamos a posição exata (x e y) onde o mesmo ficará
- A organização é feita pelo gerenciador de layout do Container
- Todo Container tem um gerenciador de layout (objeto que implementa java.awt.LayoutManager)

Swing
[Gerenciadores de Layout]



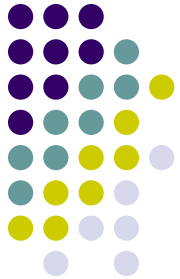
- Isso permite à aplicação adaptar-se mais facilmente às diferentes plataformas
- Pode-se utilizar um gerenciador de layout diferente para cada Container
- Estão disponíveis vários gerenciadores de layout:
  - FlowLayout e GridLayout (mais simples)
  - BorderLayout e CardLayout
  - BoxLayout
  - GridBagLayout (mais flexível)

Swing
[Gerenciadores de Layout]



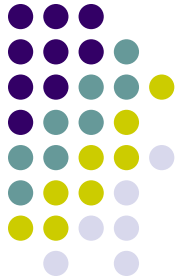
- Mudando o Gerenciador de Layout de um Container
  - Todo Container traz um gerenciador de layout padrão associado
    - FlowLayout para Painéis
    - BorderLayout para Janelas
  - A classe java.awt.Container define o método setLayout, que recebe como parâmetro um objeto que implementa LayoutManager

Swing
[Gerenciadores de Layout]



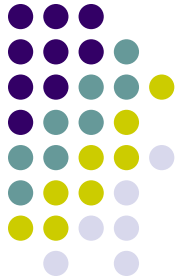
- BorderLayout
  - É gerenciador de layout padrão de janelas
  - Tem 5 áreas para colocar componentes: north (norte), south (sul), east (leste), west (oeste), e center (centro)
  - Todo o espaço extra é colocado na área central
  - É possível definir uma distância mínima entre os componentes tanto na vertical como na horizontal
  - Exemplo
    - BorderLayoutDemo

Swing
[Gerenciadores de Layout]



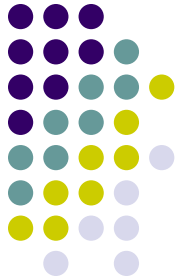
- CardLayout
  - Deve ser usada quando se tem uma área que contém componentes diferentes em momentos diferentes
  - Freqüentemente usados por Combo Box, com o estado da caixa de checagem determinando qual Panel (grupo de componentes) deve ser exibido
  - Permite especificar uma distância mínima vertical e horizontal entre os componentes
  - Exemplo
    - CardLayoutDemo

Swing
[Gerenciadores de Layout]



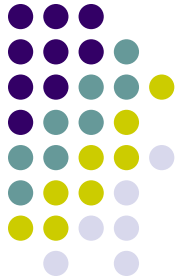
- FlowLayout
  - Padrão para todos os painéis
  - Coloca os componentes da esquerda para a direita (como padrão), iniciando em novas linhas se necessário
  - Pode-se optar também por distribuir os componentes da direita para a esquerda ou do centro para os lados
  - É possível especificar uma distância mínima tanto na vertical como na horizontal entre os componentes
  - Exemplo
    - FlowLayoutDemo

Swing
[Gerenciadores de Layout]



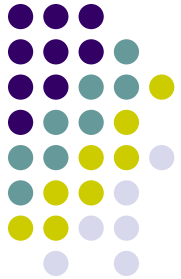
- GridLayout
  - Trabalha com a idéia de um Grid com as dimensões (linhas e colunas) especificadas no seu construtor
  - Coloca os componentes com o mesmo tamanho nas linhas e colunas do grid
  - É possível especificar uma distância mínima entre os componentes na vertical e na horizontal
  - Exemplo
    - GridLayoutDemo

Swing
[Gerenciadores de Layout]



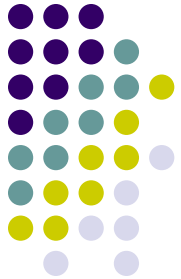
- GridBagLayout
  - É bastante sofisticado e flexível
  - Alinha os componentes colocando-os em células, permitindo que alguns componentes ocupem mais de uma célula
  - As linhas no grid não são necessariamente da mesma altura, o mesmo acontecendo com a largura das colunas
  - Os Componentes adicionados ao Container devem ser associados a uma instância de GridBagConstraints

Swing
[Gerenciadores de Layout]



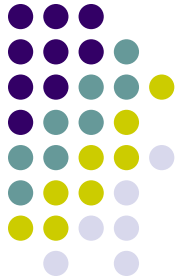
- GridBagConstraints tem atributos de instância que são utilizadas pelo gerenciador de layout para distribuir os componentes no Grid
  - gridx e gridy
    - Especifica a linha e coluna da parte superior esquerda do componente
    - A coluna mais à esquerda tem o endereço gridx=0, e a linha superior tem o endereço gridy=0
    - Pode-se utilizar GridBagConstraints.RELATIVE (o valor padrão) para especificar que o componente seja colocado à direita ou abaixo do último componente adicionado ao Container

Swing
[Gerenciadores de Layout]



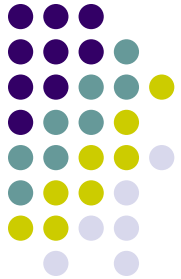
- gridwidth e gridheight
  - Especifica o número de colunas (gridwidth) e linhas (gridheight) na área de exibição do componente
  - Essas restrições especificam o número de células usadas pelo componente e não o número de pixels
  - O valor padrão é 1
  - Pode-se usar GridBagConstraints.REMAINDER para especificar que o componente é o último da linha ou coluna

Swing
[Gerenciadores de Layout]



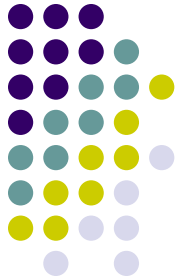
- fill
  - Usado quando a área de exibição do componente é maior que a requerida pelo componente e determina se e como redimensionar o componente
  - Valores possíveis:
    - GridBagConstraints.NONE (padrão)
      GridBagConstraints.HORIZONTAL (ocupa toda a área horizontal disponível, sem modificar a altura)
    - GridBagConstraints.VERTICAL (ocupa toda a área vertical disponível, sem modificar a largura)
    - GridBagConstraints.BOTH (componente preenche a área de exibição completamente)

Swing
[Gerenciadores de Layout]



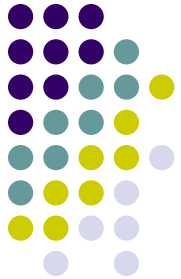
- ipadx e ipady
  - Especifica o espaçamento interno (quanto adicionar ao tamanho mínimo do componente).
  - O valor padrão é 0
  - A largura do componente será no mínimo igual ao seu tamanho mínimo mais ipadx*2 pixels (já que o espaçamento se aplica a ambos os lados do componente)
  - ipady atua de forma semelhante na altura do componente

Swing
[Gerenciadores de Layout]



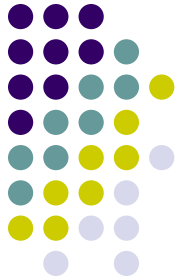
- insets
  - Especifica o espaçamento externo (a quantidade mínima de espaço entre o componente e as arestas da área de exibição)
  - O valor é especificado usando instâncias de java.awt.Insets
  - O padrão é não ter espaçamento externo

Swing
[Gerenciadores de Layout]



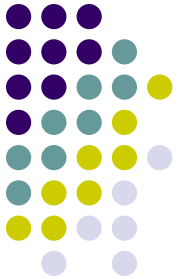
- anchor
  - Usado quando o componente é menor que sua área de exibição para determinar onde (dentro da área) colocar o componente
  - Valores possíveis:
    - GridBagConstraints.CENTER (o padrão)
    - GridBagConstraints.NORTH
    - GridBagConstraints.NORTHEAST
    - GridBagConstraints.EAST
    - GridBagConstraints.SOUTHEAST
    - GridBagConstraints.SOUTH
    - GridBagConstraints.SOUTHWEST
    - GridBagConstraints.WEST
    - GridBagConstraints.NORTHWEST

Swing
[Gerenciadores de Layout]



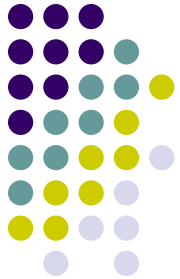
- weightx e weighty
  - Pesos são usados para determinar como distribuir o espaço entre colunas (weightx) e entre linhas (weighty)
  - Geralmente os pesos são valores entre 0.0 e 1.0 (podem ser superiores)
  - Ao redimensionar um Container, os componentes com maiores pesos receberão mais espaço
- Exemplo
  - GridBagLayoutDemo

Swing
[Gerenciadores de Layout]



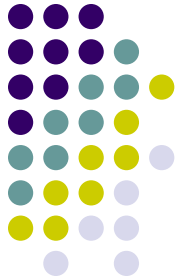
- Outros Gerenciadores
  - SpringLayout
  - BoxLayout

Swing
[Tratamento de Eventos]



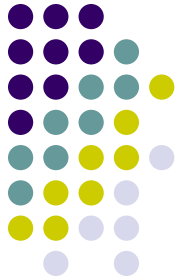
- A partir do Java 1.1 foi introduzido um novo modelo de eventos que serve tanto para o AWT como para Swing
- Cada tipo de evento é representado por uma classe (subclasse de java.util.EventObject)
- Eventos do AWT são subclasses de java.awt.AWTEvent
- Os vários tipos de eventos são definidos no pacote java.awt.event

Swing
[Tratamento de Eventos]



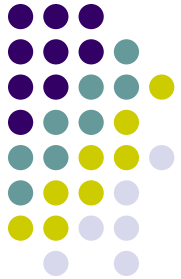
- O modelo de manipulação de eventos 1.1 é baseado no conceito de “event listeners”
- Event Listener
  - Um objeto interessado em receber eventos
- Event Source
  - Objeto que gera eventos e mantém a lista dos *listeners* que estão interessados em serem notificados quando os eventos ocorrerem
  - Provê métodos que permitem adicionar ou remover *listeners* da lista de objetos interessados

Swing
[Tratamento de Eventos]



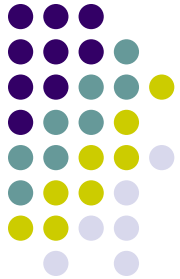
- Quando um *event source* gera um evento, o *event source* notifica todos os objetos *listeners* da ocorrência do evento
- Um *event source* notifica um *event listener* através da chamada de um método passando um objeto evento
- Para que um *event source* possa chamar um método de um *listener*, é preciso que todos os eles implementem os métodos requeridos

Swing
[Tratamento de Eventos]



- Todos os *event listeners* de um tipo de devem implementar uma interface
  - Um *event listener* para eventos ActionEvent deve implementar a interface ActionListener
- O pacote java.awt.event define uma interface de *event listener* para cada tipo de evento
  - Na verdade, para eventos MouseEvent, define duas interfaces: MouseListener e MouseMotionListener
- Todas as interfaces de *event listeners* estendem a interface java.util.EventListener (interface de marcação).

Swing
[Tratamento de Eventos]



- Uma interface event listener pode definir mais de um método
  - A classe como MouseEvent, por exemplo, representa vários tipos diferentes de eventos de mouse, tais como o clique de um botão e a liberação de um botão
  - Esses tipos diferentes de eventos provocam a chamada de métodos diferentes nos event listeners
  - Por convenção, é passado aos métodos de um event listener um único argumento, o evento.
  - O evento contém informações que os listeners utilizam para responder ao evento

Swing
[Tratamento de Eventos]

- Eventos, interface dos Listeners e métodos

Classe do Evento	Interface dos Listeners	Métodos dos Listeners
ActionEvent	ActionListener	actionPerformed()
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged()
ComponentEvent	ComponentListener	componentHidden() componentMoved() componentResized() componentShown()
ContainerEvent	ContainerListener	componentAdded() componentRemoved()
FocusEvent	FocusListener	focusGained() focusLost()

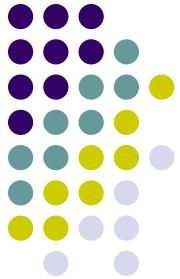
Swing
[Tratamento de Eventos]

Classe do Evento	Interface dos Listeners	Métodos dos Listeners
ItemEvent	ItemListener	itemStateChanged()
KeyEvent	KeyListener	keyPressed() keyReleased() keyTyped()
MouseEvent	MouseListener MouseMotionListener	mouseClicked() mouseEntered() mouseExited() mousePressed() mouseReleased() mouseDragged() mouseMoved()

Swing
[Tratamento de Eventos]

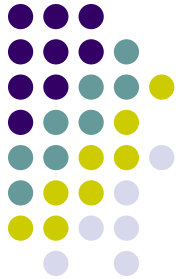
Classe do Evento	Interface dos Listeners	Métodos dos Listeners
TextEvent	TextListener	textValueChanged()
WindowEvent	WindowListener	windowActivated() windowClosed() windowClosing() windowDeactivated() windowDeiconified() windowIconified() windowOpened()

Swing
[Tratamento de Eventos]



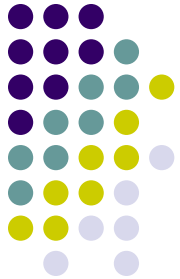
- Classes adaptadoras
  - Para cada interface de event listener que contém mais de um método, java.awt.event define uma classe “adaptadora” que oferece implementações vazias para cada método na interface
  - Interessantes quando se está interessado em implementar apenas um ou alguns dos métodos de uma interface event listener, pois basta estender a classe e sobrepor o(s) método(s) desejado(s)
  - Seguem uma nomenclatura padrão, na qual o nome da classe adaptadora é igual ao nome da interface event listener trocando-se “Listener” por “Adapter”
    - O adaptador para WindowListener é WindowAdapter

Swing
[Tratamento de Eventos]



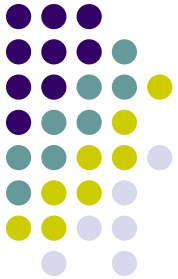
- Registro de Listeners
  - Para que um listener passe a receber eventos de um event source é necessário registrar-se junto ao event source
  - Os métodos para registrar event listeners seguem um padrão. Se o event source gera eventos de um tipo X, ele deve ter
    - um método chamado addXListener() para adicionar
    - método removeXListener() para remover

Swing
[Tratamento de Eventos]



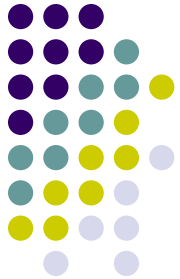
- Eventos gerados por alguns Componentes
  - Component
    - ComponentEvent
  - Container
    - ContainerEvent
  - JFrame
    - WindowEvent, ContainerEvent
  - JButton, JCheckBox e JRadioButton
    - ActionEvent, ChangeEvent e ItemEvent

Swing
[Tratamento de Eventos]



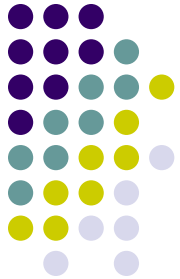
- Exemplo
  - TratamentoEventosDemo

Swing
[JList]



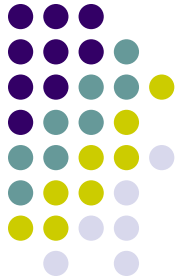
- Permite a seleção de um ou mais itens de uma lista
- Principais Métodos
  - setListData, addListSelectionListener
  - getSelectedIndices, getSelectedValues, getSelectedValue
  - setSelectedIndex, setSelectedIndices, setSelectedValue
  - setSelectionMode (define o modo de seleção usado)
    - ListSelectionModel.SINGLE_SELECTION,
    - ListSelectionModel.SINGLE_INTERVAL_SELECTION,
    - ListSelectionModel.MULTIPLE_INTERVAL_SELECTION
- A classe JScrollPane inclui as barras de rolagem
- Exemplo
  - JListDemo

Swing
[JComboBox]



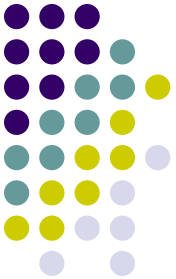
- Combina um botão ou um campo de edição e uma lista
- Pode-se selecionar um valor da lista ou digitar algum valor no campo de edição
- Principais Métodos
  - insertItemAt, addItem, removeItemAt, removeItem, removeAllItens, getItemAt e getItemCount
  - getEditor
  - getSelectedItem, getSelectedIndex e getSelectedObjects
- Exemplo
  - JComboBoxDemo

Swing
[JTextArea]

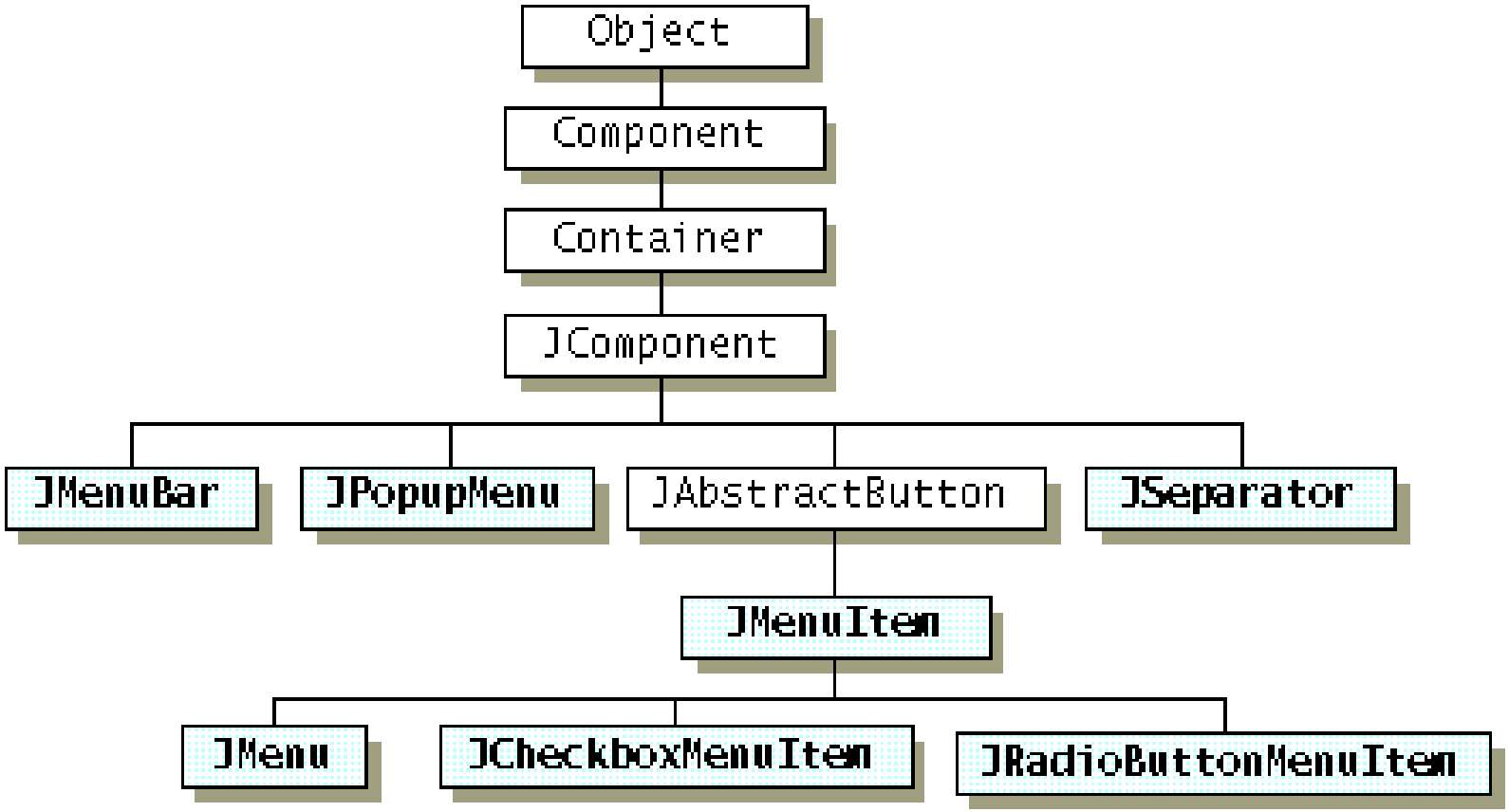


- É uma área que exibe texto plano
- Principais Métodos
  - append, insert, replaceRange, setText
  - read e write
  - getColumns, setColumns, getRows, setRows
  - getLineCount
- Exemplo
  - JTextAreaDemo

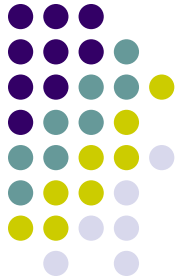
Swing
[Menu]



- Hierarquia de classes para criar menus

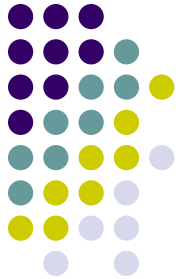


Swing
[Menu]



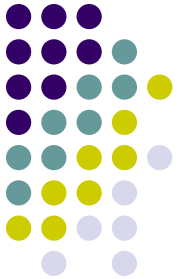
- JMenuBar representa uma barra de menu onde podem ser incluídos menus (JMenu ou JPopupMenu)
- Menus normalmente contêm itens de menu
- Um item de menu é toda subclasse de JMenuItem
- JSeparator permite separar componentes presentes nos menus

Swing
[Menu]



- JCheckBoxMenuItem e JRadioButtonMenuItem são componentes a JCheckBox e JRadioButton que podem ser inseridos em menus
- Exemplo
  - MenuDemo

Swing
[Bibliografia]



- Core Java 2 – Vol 1 (capítulos 7 a 9)
- Core Java 2 – Vol 2 (capítulo 6)
- Deitel (capítulos 12 e 13)