Componentes
Parte 02

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Junho de 2022

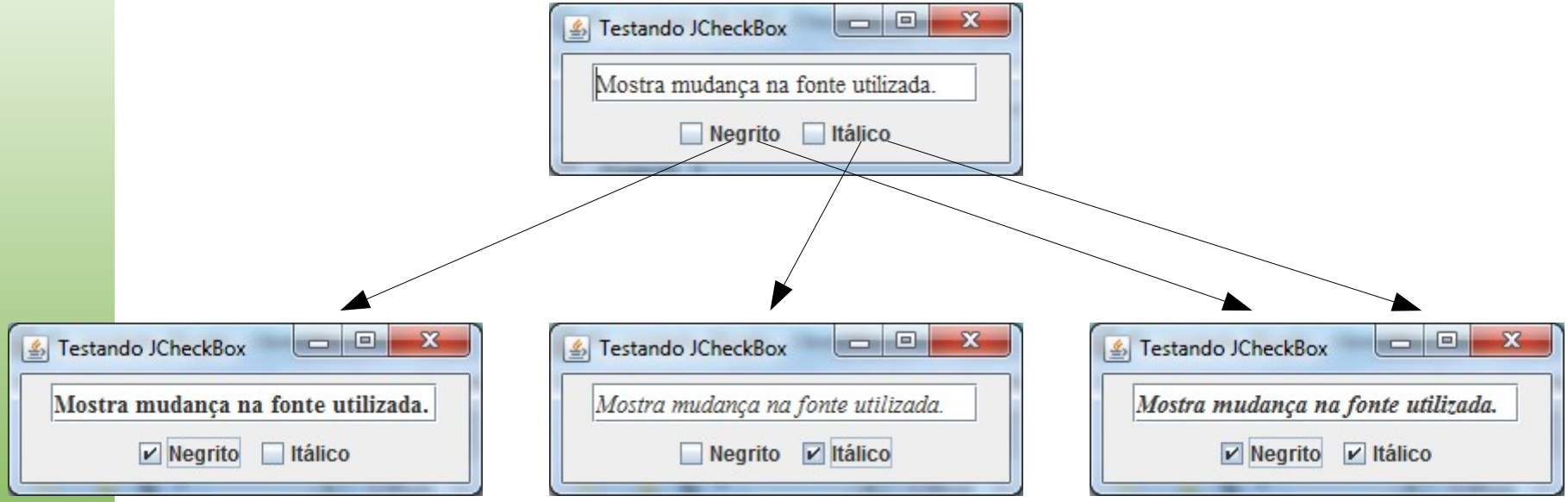
Botões que Mantêm o Estado

- Botões de estado:
  - O Swing contém três tipos de botões de estado:
    - `JToggleButton`,
    - `JCheckBox` e
    - `JRadioButton`.

JCheckBox

- JCheckBox:
  - Contém um rótulo de caixa de seleção que aparece à direita da caixa de seleção por padrão;
  - Gera um `ItemEvent` quando é clicado.
    - `ItemEvents` são tratados por um `ItemListener`;
    - Passado para o método `itemStateChanged`.
  - O método `isSelected` retorna se uma caixa de seleção está selecionada (`true`) ou não (`false`).

Exemplo 01 - JCheckBox



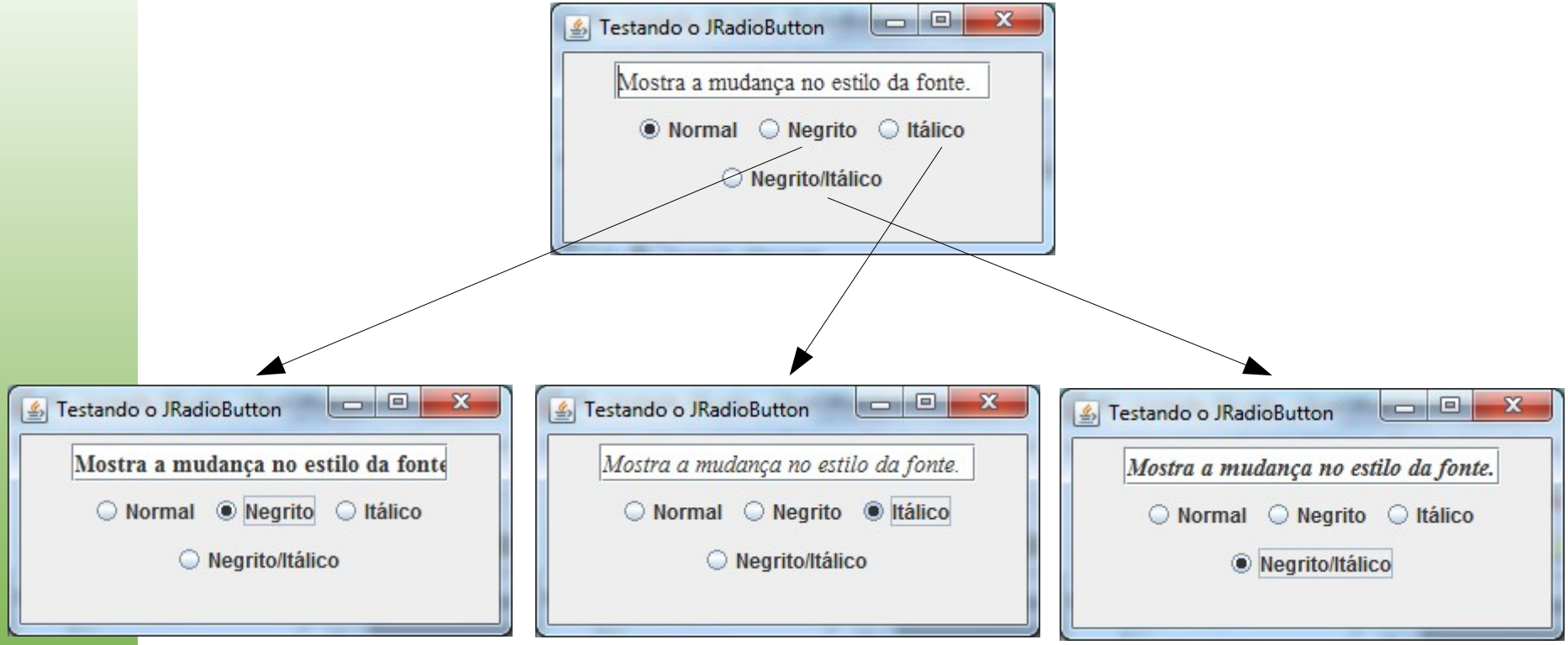
Exemplo 01 - JCheckBox

- Parte 01 - Classe Main.
- Parte 02 - Classe CheckBoxFrame.

JRadioButton

- JRadioButton:
  - Tem dois estados – selecionado e não selecionado;
  - Normalmente aparece em um grupo no qual somente um botão de opção pode ser selecionado de cada vez:
    - Grupo mantido por um objeto ButtonGroup:
      - Declara o método add para adicionar um JRadioButton ao grupo.
  - Normalmente, representa opções mutuamente exclusivas.

Exemplo 02 - JRadioButton



Exemplo 02 - JRadioButton

- Parte 01 - Classe Main.
- Parte 02 - Classe RadioButtonFrame.

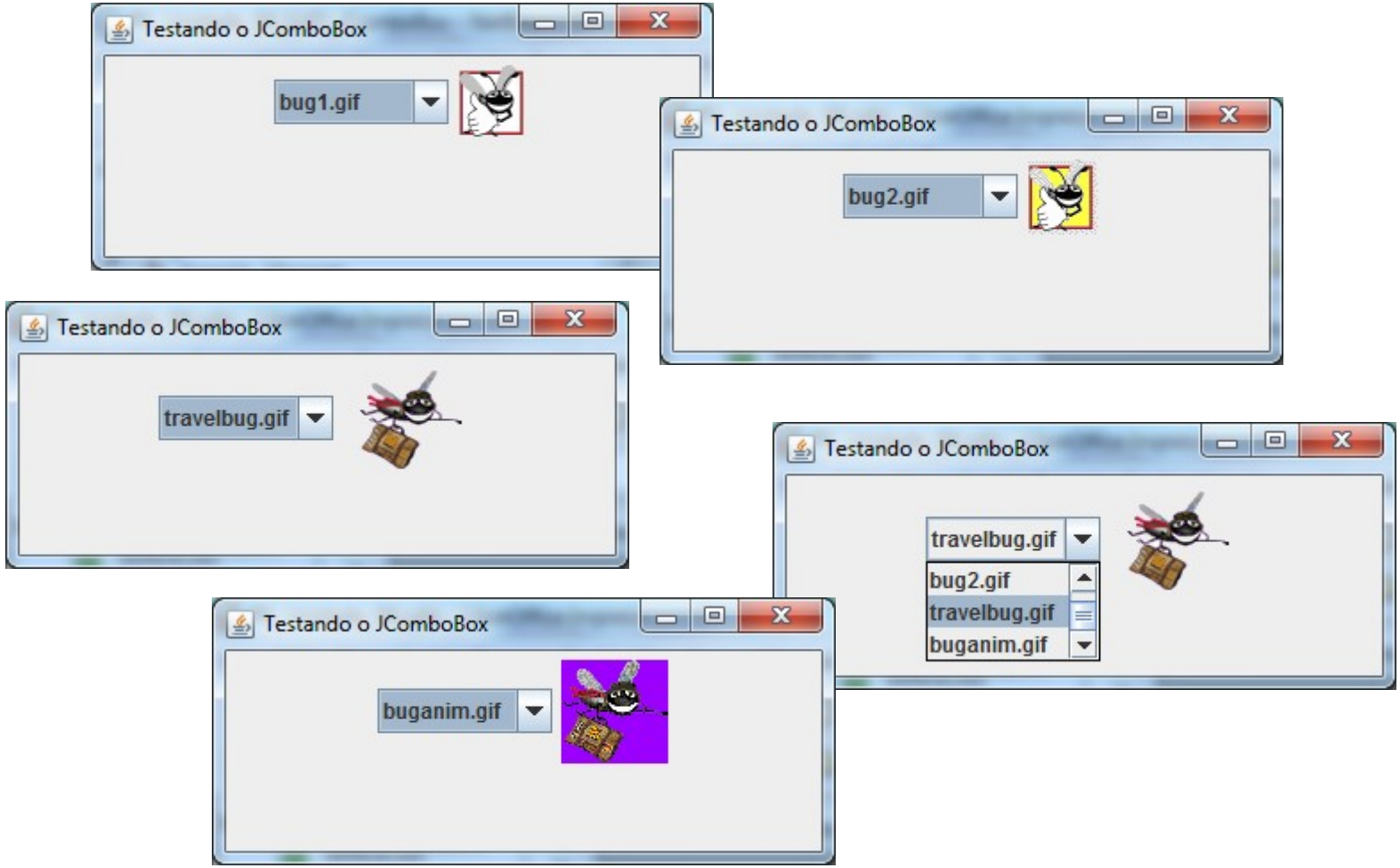
JComboBox

- Caixa de combinação:
  - Às vezes, também chamada lista *drop-down*;
  - Implementada pela classe `JComboBox`;
  - Cada item na lista tem um índice;
  - O método `setMaximumRowCount` configura o número máximo de linhas mostradas de cada vez;
  - O `JComboBox` fornece uma barra de rolagem e setas para cima e para baixo para percorrer a lista.

Classe Interna Anônima

- Classe interna anônima:
  - Forma especial de classe interna que é declarada sem nome;
  - Em geral, aparece dentro de uma chamada de método;
  - Tem acesso limitado a variáveis locais.

Exemplo 03 - JComboBox



Exemplo 03 - JComboBox

- Parte 01 - Classe Main.
- Parte 02 - Classe ComboBoxFrame.

Recordando: Modificador `final`

- O modificador final quando aplicado a uma variável local diz que o valor desta não poderá ser modificado após a sua inicialização.
- Então se fizéssemos algo do tipo:

```
public void contarAte(final int contador) {
    contador = 100; // Erro!
    ...
}
```

- O modificador final pode ser usado em java para definir constantes, apesar de uma das palavras chaves da linguagem ser a palavra `const`, ela não é utilizada.
- Assim você diz que uma variável é uma constante aplicando a ela o modificador `final`. (ARAÚJO, 2008)

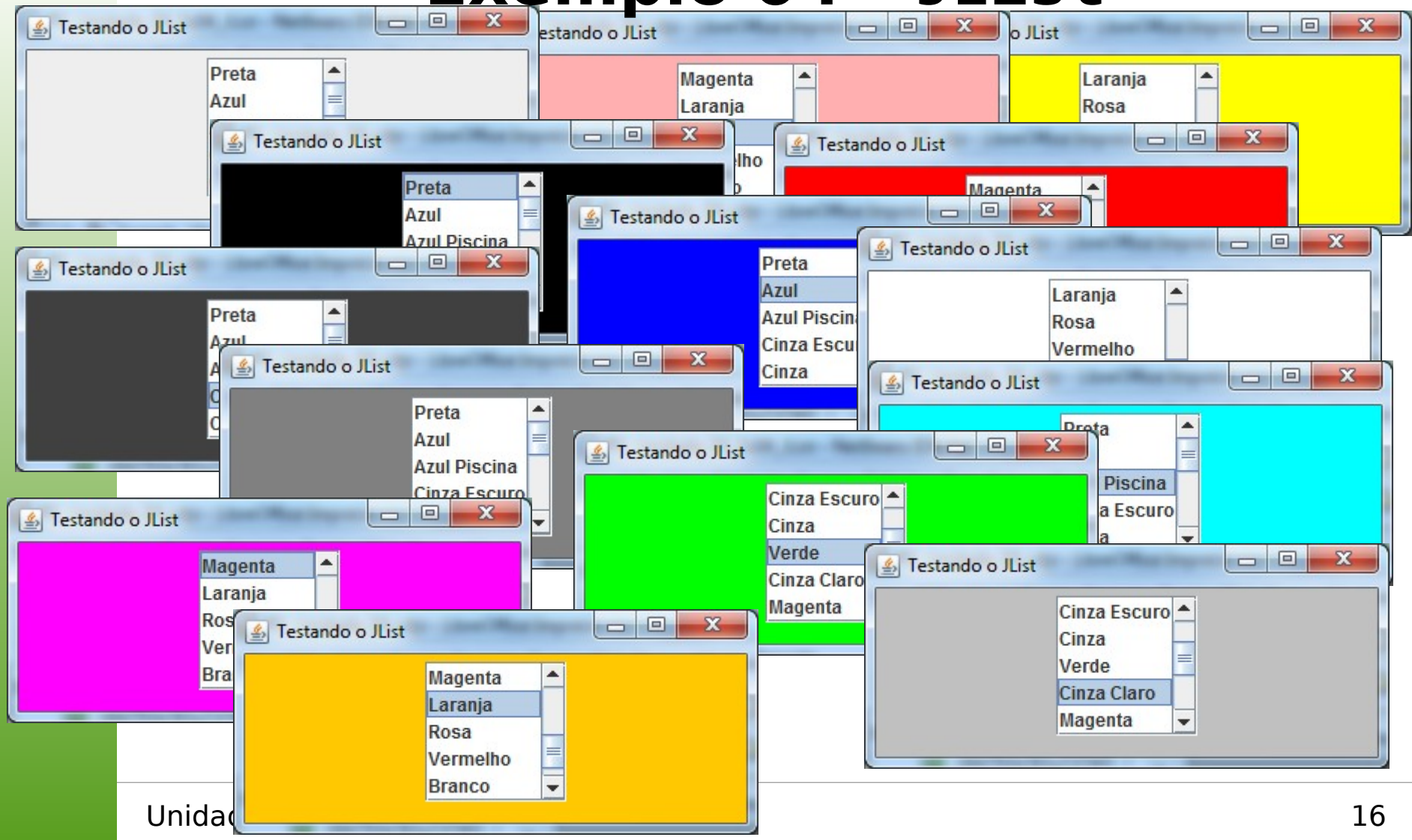
Observação

Uma classe interna anônima (vide exemplo 4) declarada em um método, pode acessar as variáveis de instância e métodos do objeto de classe de primeiro nível que a declararam, bem como as variáveis locais `final` do método, mas não pode acessar variáveis não-`final` do método.

JList

- Lista:
  - Exibe uma série de itens dentre os quais o usuário pode selecionar um ou mais;
  - Implementada pela classe `JList`;
  - Permite listas de seleção única ou listas de múltipla seleção;
  - Um `ListSelectionEvent` ocorre quando um item é selecionado:
    - Tratado por um `ListSelectionListener` e passado para o método `valueChanged`.

Exemplo 04 - JList



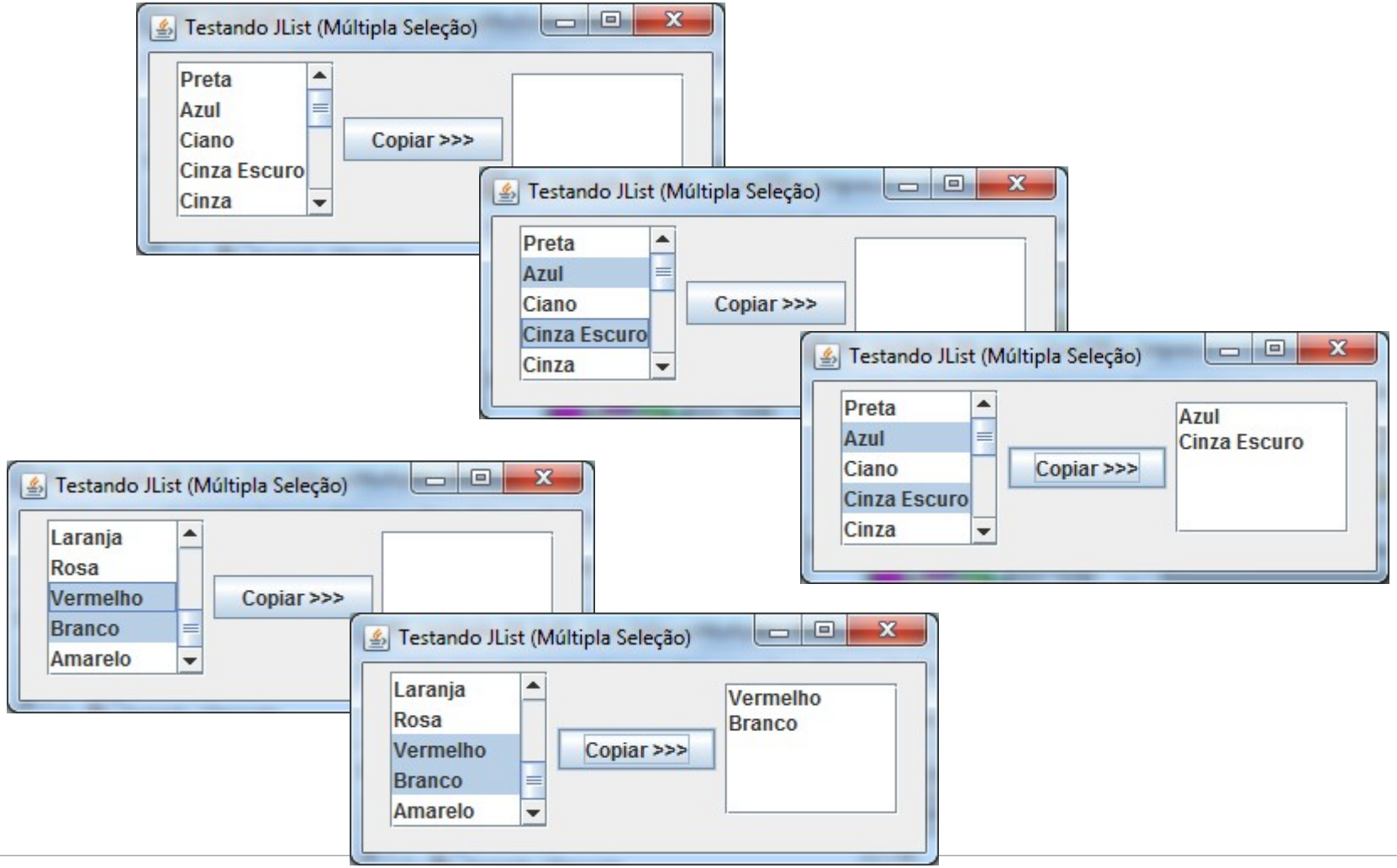
Exemplo 04 - `JList`

- Parte 01 - Classe Main.
- Parte 02 - Classe ListFrame.

Listas de Seleção Múltipla

- Lista de seleção múltipla:
  - Permite que usuários selecionem vários itens;
  - Seleção de um único intervalo que permite apenas um intervalo contínuo de itens;
  - Seleção de múltiplos intervalos que permite que qualquer conjunto de elementos seja selecionado.

Exemplo 05 – JList (Seleção Múltipla)



Listas de Seleção Múltipla

- Exemplo:
  - o código-fonte apresentado no próximo *slide* não está funcionando.
  - Substitua-o por um dos códigos a seguir:

```
private void maneiraDeCopiarItens01(){
  DefaultListModel list = new DefaultListModel();
  for (Object item : jListOrigem.getSelectedValuesList()) {
    if (list.indexOf(item) == -1) {
      list.addElement(item);
    }
  }
  jListDestino.setModel(list);
}

private void maneiraDeCopiarItens02(){
    String[] vet = new String[jListOrigem.getSelectedValuesList().size()];

    for(int i = 0; i < jListOrigem.getSelectedValuesList().size(); i++){
        String item = jListOrigem.getSelectedValuesList().get(i);
        vet[i] = item;
    }
    jListDestino.setListData(vet);
}
```

Exemplo 05 – `JList` (Seleção Múltipla)

- Parte 01 - Classe Main.
- Parte 02 - Classe MultipleSelectionFrame.

Tratamento de Eventos do *Mouse*

- Eventos de *mouse*:
  - Cria um objeto `MouseEvent`;
  - Tratado por `MouseListeners` e `MouseMotionListeners`;
  - `MouseInputListener` combina as duas interfaces;
  - A interface `MouseWheelListener` declara o método `mouseWheelMoved` para tratar `MouseWheelEvents`.

Interface MouseListener e MouseMotionListener

Métodos de interface MouseListener e MouseMotionListener

Métodos de interface ***MouseListener***

`public void mousePressed( MouseEvent event )`

Chamado quando um botão do mouse é pressionado enquanto o cursor de mouse estiver sobre um componente.

`public void mouseClicked( MouseEvent event )`

Chamado quando um botão do mouse é pressionado e liberado enquanto o cursor do mouse pairar sobre um componente. Esse evento é sempre precedido por uma chamada para `mousePressed`.

`public void mouseReleased( MouseEvent event )`

Chamado quando um botão do mouse é liberado depois de ser pressionado. Esse evento sempre é precedido por uma chamada para mousePressed e um ou mais chamadas para `mouseDragged`.

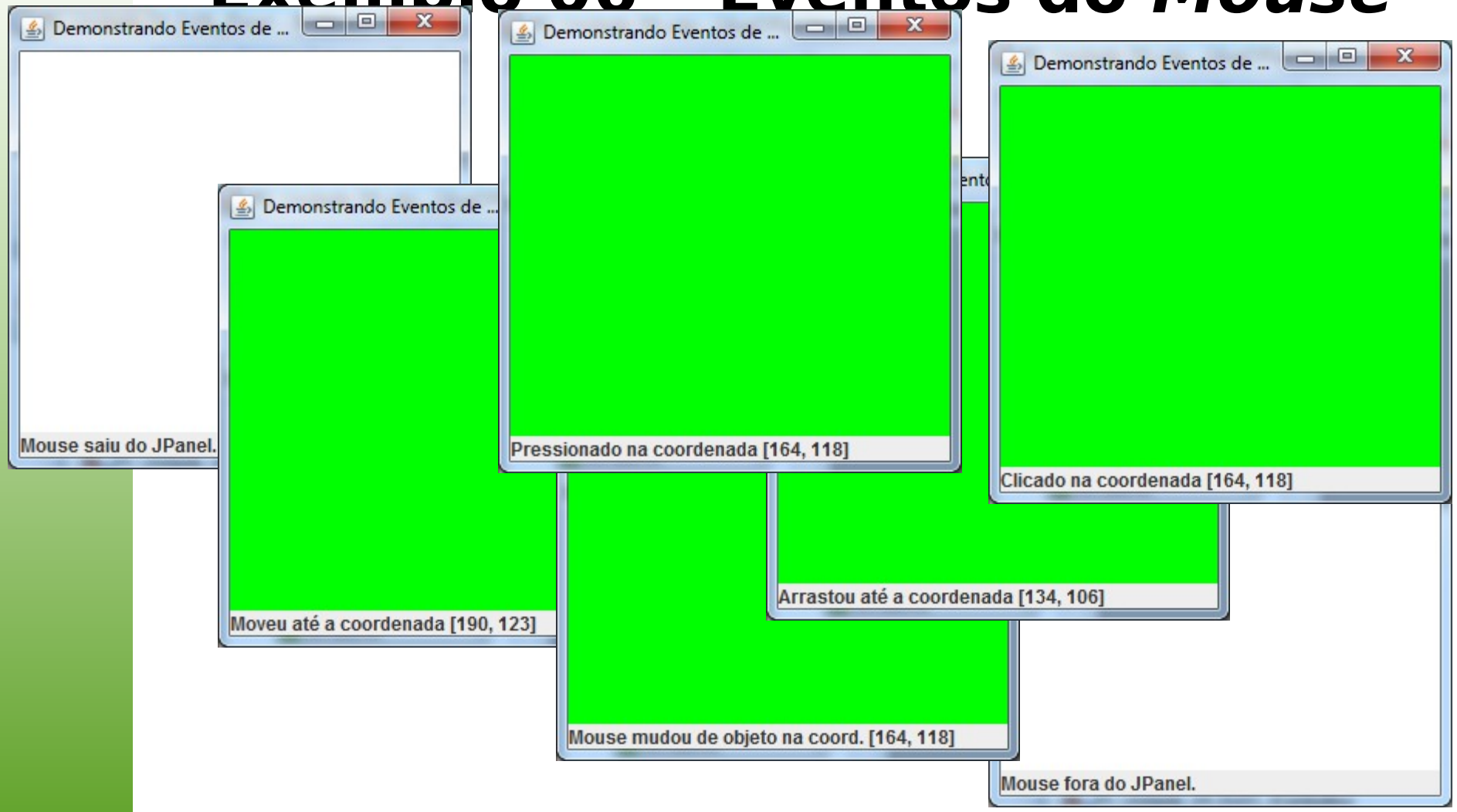
`public void mouseEntered( MouseEvent event )`

Chamado quando o cursor do mouse entra nos limites de um componente.

Interface MouseListener e MouseMotionListener

Métodos de interface MouseListener e MouseMotionListener	
`public void mouseExited( MouseEvent event )`	Chamado quando o cursor do mouse deixa os limites de um componente.
Métodos de interface `MouseMotionListener`	
`public void mouseDragged( MouseEvent event )`	Chamado quando o botão do mouse é pressionado enquanto o cursor de mouse estiver sobre um componente e o mouse é movido enquanto o botão do mouse permanecer pressionado. Esse evento é sempre precedido por uma chamada para `mousePressed`. Todos os eventos de arrastar são enviados para o componente em que o usuário começou a arrastar o mouse.
`public void mouseMoved( MouseEvent event )`	Chamado quando o mouse é movido quando o cursor de mouse estiver sobre um componente. Todos os eventos de movimento são enviados para o componente sobre o qual o mouse atualmente está posicionado.

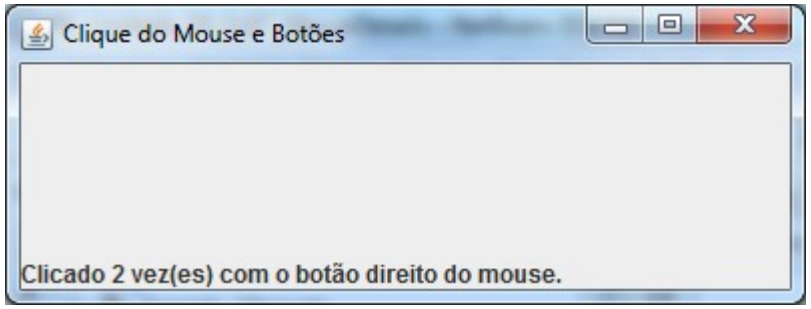
Exemplo 06 – Eventos do *Mouse*

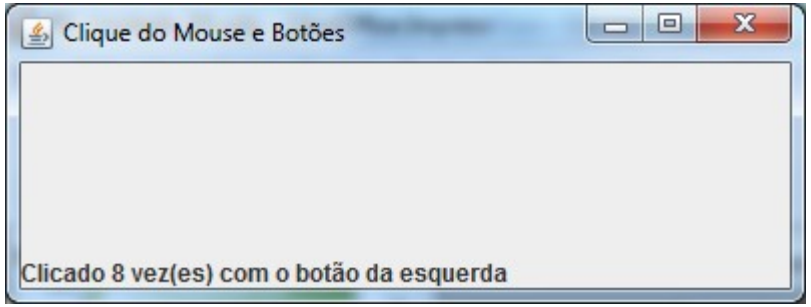


Exemplo 06 – Eventos do *Mouse*

- Parte 01 - Classe Main.
- Parte 02 - Classe MouseTrackerFrame.

Exemplo 07 – Eventos de Clique do *Mouse*





Exemplo 07 – Eventos de Clique do *Mouse*

- Até a versão 8 da JDK, o código a seguir funcionava perfeitamente:

- Parte 01 - Classe Main.

- Parte 02 - Classe MouseDetails.

Exemplo 07 – Eventos de Clique do *Mouse*

- Na versão 11 da JDK, apesar de estar em desuso, o código a seguir ainda funciona:

```
61      private void formMouseClicked(java.awt.event.MouseEvent evt) {
62          int xPos = evt.getX(); // Coordenada X.
63          int yPos = evt.getY(); // Coordenada Y.
64          String details = String.format("Clicado %d vez(es)", evt.getClickCount());
65
            if (evt.getModifiers() == MouseEvent.BUTTON1_MASK)
67              details += " com o botão direito do mouse.";
68          else
69              if(evt.getModifiers() == MouseEvent.BUTTON2_MASK)
70                  details += " com o botão do meio do mouse.";
71              else
72                  if(evt.getModifiers() == MouseEvent.BUTTON3_MASK)
73                      details += " com o botão da esquerda";
74
75          lblStatus.setText(details + "[" + xPos + "," + yPos + "]");
76      }
```

Não se esqueça do: `import java.awt.event.MouseEvent;`

Exemplo 07 – Eventos de Clique do *Mouse*

- Na versão 11 da JDK, o código a seguir funciona:

```
60
61      private void formMouseClicked(java.awt.event.MouseEvent evt) {
62          int xPos = evt.getX(); // Coordenada X.
63          int yPos = evt.getY(); // Coordenada Y.
64          String details = String.format("Clicado %d vez(es)", evt.getClickCount());
65
66          if(SwingUtilities.isRightMouseButton(evt))
67              details += " com o botão direito do mouse.";
68          else
69              if(SwingUtilities.isMiddleMouseButton(evt))
70                  details += " com o botão do meio do mouse.";
71              else
72                  if(SwingUtilities.isLeftMouseButton(evt))
73                      details += " com o botão da esquerda";
74
75          lblStatus.setText(details + "[" + xPos + "," + yPos + "]");
76      }
77
```

Não se esqueça do: `import javax.swing.SwingUtilities;`

Subclasse JPanel para Desenhar com o *Mouse*

- Sobrescrevendo a classe JPanel:
  - Fornece uma área dedicada de desenho.

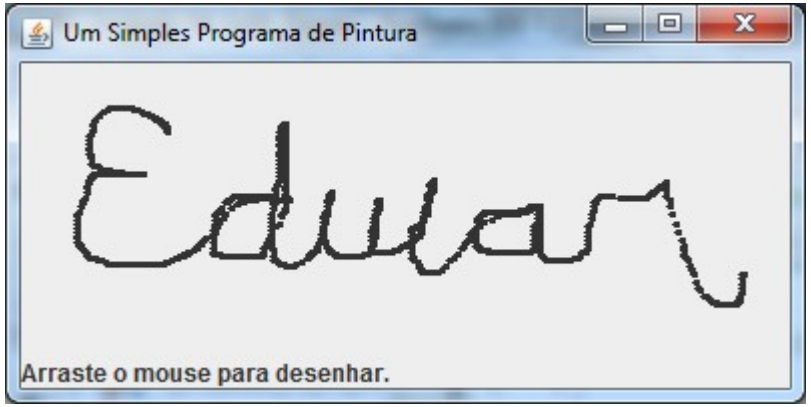
Método paintComponent

- Desenha em um componente Swing;
- ao sobrescrever o método permite-se criar desenhos personalizados;
- deve primeiro chamar o método de superclasse quando sobrescrito.

Definindo a Área de Desenho

- Subclasse personalizada de `JPanel`:
  - Oferece uma área de desenho personalizada;
  - A classe `Graphics` é utilizada para desenhar nos componentes Swing;
  - A classe `Point` representa uma coordenada *x*-*y*.

Exemplo 08 – *Paint Panel*



Exemplo 08 – *Paint Panel*

- Exemplo:
  - o código-fonte apresentado no próximo *slide* não está funcionando.
  - Substitua-o pelo código a seguir:

```
private int ponto = 0;
private Point vetPontos[] = new Point[1000];
...
private void formMouseDragged(java.awt.event.MouseEvent evt) {
  if(ponto < vetPontos.length){
        vetPontos[ponto] = evt.getPoint();
        ponto++;
        repaint();
    }
}

@Override
public void paint(Graphics g){
    super.paint(g);
    for(Point p : vetPontos){
        if(p != null)
            g.fillRect((int)p.getX(), (int)p.getY(), 3, 3);
    }
}
```

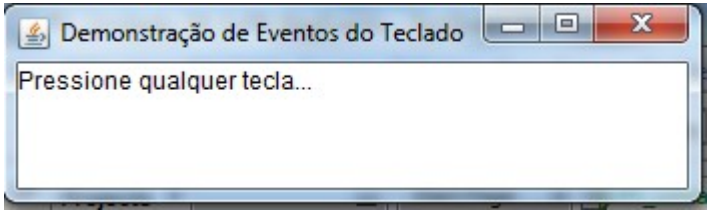
Exemplo 08 – *Paint Panel*

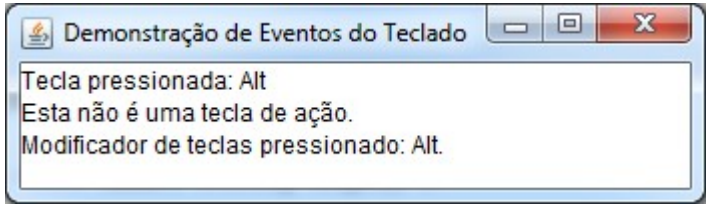
- Parte 01 - Classe Main
- Parte 02 - Classe PaintPanel

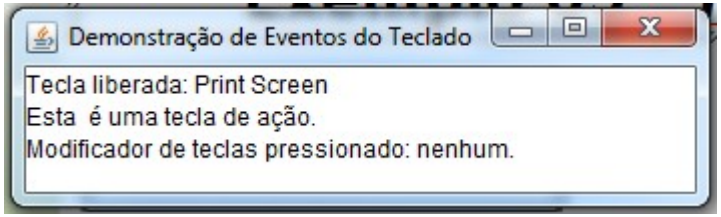
Tratamento de Eventos do Teclado

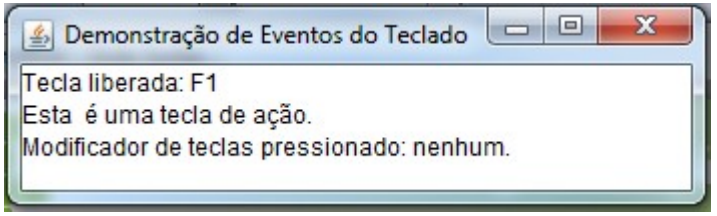
- Interface `KeyListener`:
  - Para tratar eventos de teclado — `KeyEvents`.
  - Declara os métodos `keyPressed`, `keyReleased` e `keyTyped`, sendo que cada um recebe um `KeyEvent` como seu argumento.

Exemplo 09 – *Key Events*









Exemplo 09 – *Key Events*

- Parte 01 - Classe Main
- Parte 02 - Classe KeyEventsPanel

Exercício

- Construa uma classe para manipulação de *strings*. Sua classe deverá possuir os seguintes métodos:
  - Inverter: este método escreve uma *string* original de trás para frente;
  - Tamanho: este método deverá retornar o tamanho de uma *string*;
  - Palíndromo: este método deverá verificar se a *string* é um palíndromo ou não. Ou seja, se a *string* original é idêntica a sua ordem inversa;
  - Vogais: este método deverá retornar o número de vogais existentes em uma *string*;
  - Consoantes: este método deverá retornar o número de consoantes existentes numa *string*;
  - Criptografar: este método deve receber um valor numérico de 1 dígito e somar este valor a cada letra da *string* (Ex.: se n=4 então: a → e, b → f, c → g, ..., w → a, x → b, y → c e z → d);
  - Descriptografar: este método deverá fazer o inverso do método anterior.
- Sua classe deverá receber e devolver uma *string* através dos métodos *get* e *set*;
- Crie uma interface que possibilite ao usuário do *software* trabalhar com a classe criada da forma mais simples e amigável possível. Para isto, faça uso dos componentes, propriedades e métodos estudados nesta unidade.

Exercícios

- Resolver os exercícios das Listas 02 e 03

Bibliografia

- DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**; 6. ed., São Paulo: Pearson, 2005.

- ARAÚNO, Robson. **Declarando Variáveis Locais**. Disponível em: <http://www.javeiro.com/2008/09/declarando-variveis-locais.html> Acesso em: 20 set. 2012.