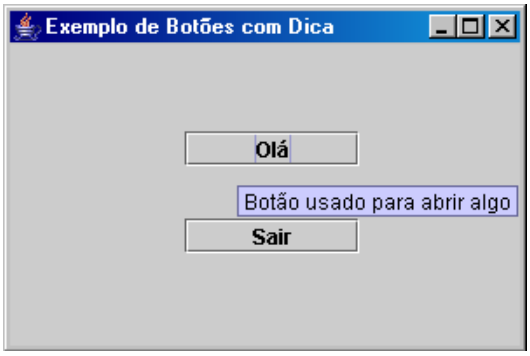
14 – Como adicionar dica aos seus botões

O processo de adicionar dicas aos botões é sempre uma boa idéia definir as dicas de ferramentas para que o usuário, ao posicionar o ponteiro do mouse sobre o controle (Botão), possa ter mais informações sobre a funcionalidade das ferramentas do aplicativo.
Veja na listagem abaixo:

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

public class ExemploBotoesDicas extends JFrame{
        JButton botao1,botao2;
        public ExemploBotoesDicas(){
                super("Exemplo de Botões com Dica");
                Container tela = getContentPane();
                tela.setLayout(null);
                botao1 = new JButton("Olá");
                botao2 = new JButton("Sair");
                botao1.setBounds(100,50,100,20);
                botao2.setBounds(100,100,100,20);
                botao1.setToolTipText("Botão usado para abrir algo");
                botao2.setToolTipText("Botão que será usado para sair");
                tela.add(botao1);
                tela.add(botao2);
                setSize(300,200);
                setVisible(true);
                setLocationRelativeTo(null);
        }

public static void main(String args[]){
        ExemploBotoesDicas app = new ExemploBotoesDicas();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

A dica de ferramenta para os botões foi definida nas linhas:

```
botao1.setToolTipText("Botão usado para abrir algo");
botao2.setToolTipText("Botão que será usado para sair");
```



14.1 – Adicionar cores diferentes as dicas

Se você observou atentamente a dica de ferramenta do exemplo anterior, deve ter percebido que o Java define o azul como cor de fundo e o texto na cor preta. É possível acessar e manipular algumas das propriedades das dicas de ferramenta de modo a adequá-las às cores propostas em seus aplicativos. É uma técnica raramente usada, mas que surte efeitos bem interessantes.

As dicas de ferramenta do Windows tem o amarelo como cor de fundo. Veja como é possível atingir esse efeito em um aplicativo Java:

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

public class ExemploBotoesDicas2 extends JFrame{
        JButton botao1,botao2;
        public ExemploBotoesDicas2(){
                super("Exemplo de Botões com Dica");
                Container tela = getContentPane();
                tela.setLayout(null);
                botao1 = new JButton("Olá");
                botao2 = new JButton("Sair");
                botao1.setBounds(100,50,100,20);
                botao2.setBounds(100,100,100,20);
                botao1.setToolTipText("Botão  usado para abrir algo");
                botao2.setToolTipText("Botão  que será usado para sair");
                UIManager.put("ToolTip.background",SystemColor.info);
                tela.add(botao1);
                tela.add(botao2);
                setSize(300,200);
                setVisible(true);
                setLocationRelativeTo(null);
        }
public static void main(String args[]){
        ExemploBotoesDicas2 app = new ExemploBotoesDicas2();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

E para alterar a cor da letra da dica, basta você digitar:

```
UIManager.put("ToolTip.foreground",Color.blue);
```

15 – Tipos de diálogos fornecidos pela classe JOptionPane

Vamos usar constantemente a classe JOptionPane para exibir caixas de mensagens mostrando resultados de operações. Nestes tópicos vamos estudar os diferentes tipos de diálogos, assim como seus construtores mais importantes.

Os diálogos de mensagens fornecidos pela classe JOptionPane são do tipo modal, ou seja, quando o diálogo é exibido, tudo que estiver sendo executado no aplicativo é suspenso até que a janela do diálogo seja fechada. Isso é importante, uma vez que dependemos da resposta do usuário para continuar o processamento normal do programa.

Os diálogos da classe JOptionPane podem ser exibidos com chamadas aos seguintes métodos estáticos:

Método	Uso
showConfirmDialog	Usado para pedir confirmação, ou seja, o usuário tem a opção de responder “sim”, “não” ou “cancelar”.
showInputDialog	Usado para obter entrada para o programa.
showMessageDialog	Usado para exibir mensagem sobre a execução de alguma operação.

15.1 – Como entender os diálogos de confirmação

Diálogos de confirmação são usados para perguntar ao usuário se uma determinada operação deve ser iniciada, continuada ou interrompida. Aplicativos em Java geralmente usam esse tipo de diálogo para confirmar a gravação de arquivos, fechamento de janelas, etc.

Exemplo:

```
JOptionPane.showConfirmDialog(null,”Fechar?”,”Fechar”,JOptionPane.YES_NO_OPTION);
```

Este método retorna um valor inteiro correspondente à opção escolhida pelo usuário. Em diálogos de confirmação esse retorno pode ser um dos três valores seguintes:

YES_OPTION – indica que o usuário escolheu a opção “Yes” ou “Sim”.

NO_OPTION – indica que o usuário escolheu a opção “No” ou “Não”.

CANCEL_OPTION – indica que o usuário escolheu a opção “Cancel” ou “Cancelar”.
Os parâmetros para o método são:

```
JOptionPane.showConfirmDialog(parentComponent,”message”,”title”,optionType);
```

parentComponent – determina a janela na qual o diálogo será exibido. Geralmente forncemos o valor null para esse parâmetro.

message – o texto a ser exibido no diálogo.

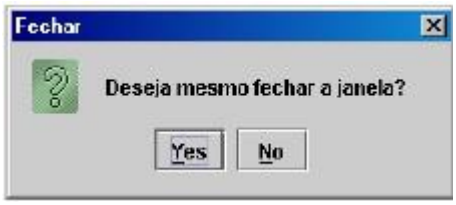
title – o texto para a barra de título do diálogo.

optionType – valor inteiro equivalente às opções disponíveis para o tipo de diálogo, ou seja, YES_NO_OPTION, ou YES_NO_CANCEL_OPTION.

A listagem abaixo mostra como os diálogos de confirmação podem ser usados. O Aplicativo pergunta se o usuário deseja realmente fechar a janela, caso você clique no yes a janela será fecada, caso contrário o janela não será fechada.

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

public class ExemploBotaoSair1 extends JFrame{
        JButton botao;
        public ExemploBotaoSair1(){
                super("Exemplo de Botões com Diálogos de confirmação");
                Container tela = getContentPane();
                tela.setLayout(null);
                botao = new JButton("Sair");
                botao.setBounds(100,50,100,20);
                botao.setToolTipText("Botão que finaliza a janela");
                tela.add(botao);
                botao.addActionListener(
                        new ActionListener(){
                            public void actionPerformed(ActionEvent e) {
                                int opcao;
                                opção=JOptionPane.showConfirmDialog(null,
                                "Deseja mesmo fechar a janela?",
                                "Fechar",JOptionPane.YES_NO_OPTION);
                                if (opcao==JOptionPane.YES_OPTION)
                                    System.exit(0);
                            }});
                setSize(300,200);
                setVisible(true);
                setLocationRelativeTo(null);
        }
public static void main(String args[]){
        ExemploBotaoSair1 app = new ExemploBotaoSair1();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Execute o aplicativo clique no botão sair e veja a mensagem do programa:



Se você clicar em Yes a janela será fechada e caso contrário ela continuará. Passos para colocar o botão a janela e dicas de botões já foram vistos anteriomente, e para chamar o caixa de confirmação basta você clicar no sair, e o evento será executado, método para chamar o evento também já foi visto.

Veja o que tem de diferente:

```
botao.addActionListener(
        new ActionListener(){
                public void actionPerformed(ActionEvent e){
                        int opcao;
                        opcao=JOptionPane.showConfirmDialog(null,"Deseja
                                mesmo fechar a janela?","Fechar",
                               JOptionPane.YES_NO_OPTION);
                        if (opcao==JOptionPane.YES_OPTION)
                           System.exit(0); }});
```

Foi declarada uma variável do tipo inteira com o nome de opção. Essa variável vai receber a resposta do JOptionPane, e vai armazenar o valor que será escolhido na caixa de confirmação. E o comando IF, que vai fazer a comparação se o conteúdo da variável opção for igual JOptionPane.YES_OPTION, ele retorna a verdadeiro e chama o método System.exite(0) e finaliza a janela.

15.2 – Traduzir os botões de diálogos

Seria interessante exibir os textos dos botões em português? A classe JOptionPane fornece um método chamado showOptionDialog que permite personalizar o texto dos botões de diálogos. Não tem muita diferença entre o showConfirmDialog e showOptionDialog.

Vejamos a listagem:

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

public class ExemploTraduzir extends JFrame{
        JButton botao;
        public ExemploTraduzir(){
                super("Exemplo de Botões Traduzidos");
                Container tela = getContentPane();
                tela.setLayout(null);
                botao = new JButton("Sair");
                botao.setBounds(100,50,100,20);
                botao.setToolTipText("Botão que finaliza a janela");
                tela.add(botao);
                botao.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent e){
                                int opcao;
```

```
                                    Object[] botoes = {"Sim","Não"};
                                    opcao = JOptionPane.showOptionDialog(null,"Deseja  mesmo fechar
 a janela?","Fechar",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE,null,bot
oes,botoes[0]);
                                    if (opcao==JOptionPane.YES_OPTION)
                                    System.exit(0);       }});
                    setSize(300,200);
                    setVisible(true);
                    setLocationRelativeTo(null);}

public static void main(String args[]){
          ExemploTraduzir app = new ExemploTraduzir();
          app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);      } }
```

O que foi modificado e o que há de diferente?

Os botões de diálogos traduzidos foram criados em português como itens de um vetor. O vetor com essas duas opções, só apareceram dois botões, caso você queira o cancelar basta adcionar a lista do vetor = {“Sim”,”Não”,”Cancelar”}. Object [ ] botoes = {"Sim","Não"};

Em seguida a variável opcao que é do tipo inteira receberá o resultado da chamada ao método showOptionDialog: Variável inteira: int opcao;

Vetor com os botões traduzidos: Object[] botoes = {"Sim","Não"};

Variável opção que esta recebendo o resultado do showOptionDialog:
opcao=JOptionPane.showOptionDialog(

null, – determina a janela na qual o diálogo será exibido. Geralmente forncemos o valor null para esse parâmetro.

"Deseja mesmo fechar a janela?", – o texto a ser exibido no diálogo.

"Fechar", – o texto para a barra de título do diálogo.

JOptionPane.YES_NO_OPTION, - valor inteiro equivalente às opções disponíveis para o tipo de diálogo, ou seja, YES_NO_OPTION, ou YES_NO_CANCEL_OPTION .

JOptionPane.QUESTION_MESSAGE, determina o tipo da mensagem a ser exibida, exemplo pergunta, erro, informação, etc.

null, - ícone a ser exibido no diálogo, ícone personalizado.

botoes,botoes[0]); - Um vetor de objetos, geralmente do tipo string, representando as possiveis escolhas, ou seja, os botões para o diálogo.

15.3 – Como obter dados por meio de diálogos do tipo prompt

Diálogos do tipo prompt são usados quando precisamos recebe entrada de dados, geralmente fornecidos pelo usuário do aplicativo.

A forma mais básicade um diálogo do tipo prompt é obtida pelo método : showInputDialog com a seguinte sintaxe:

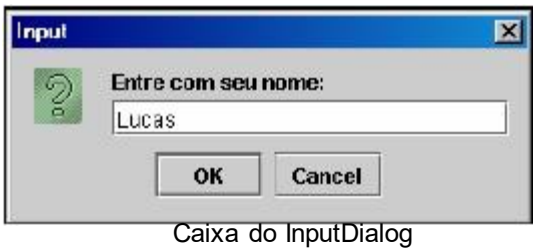
```
showInputDialog(mensagem);
```

Quando o usuário digita o conteúdo solicitado e pressiona o botão OK, o conteúdo da caixa é atribuído a variável nome. E o conteúdo dessa variável será escrito dentro de um rótulo na janela. Se o botão cancel for acionado acontecerá um erro no aplicativo.

Como você pode perceber boa parte da caixa
de diálogo esta em inglês. Infelizmente não é possível traduzir esse diálogo sem nos aventuramos em criar uma classe personalizada.

Veja a listagem:

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

public class ExemploInputDialog  extends JFrame{
        JButton botao;
        String nome;
        JLabel rotulo;
        public ExemploInputDialog(){
                super("Exemplo de Input Dialog");
                Container tela = getContentPane();
                tela.setLayout(null);
                nome = JOptionPane.showInputDialog("Entre  com seu nome: ");
                rotulo = new JLabel("");
                botao = new JButton("Sair");
                rotulo.setBounds(10,40,350,20);
                botao.setBounds(100,90,100,20);
                botao.setToolTipText("Botão  que finaliza a janela");
                rotulo.setText("O  nome digitado  foi: "+nome.toUpperCase());
                tela.add(rotulo);
                tela.add(botao);
                botao.addActionListener(new  ActionListener(){
                        public void actionPerformed(ActionEvent  e){
                                int opcao;
                                Object[] botoes = {"Sim","Não"};
                                opcao = JOptionPane.showOptionDialog(null,"Deseja
                                mesmo fechar a janela?","Fechar",
                                JOptionPane.YES_NO_OPTION,
                                JOptionPane.QUESTION_MESSAGE,
                                null,botoes,botoes[0]);
                                if (opcao==JOptionPane.YES_OPTION)
                                    System.exit(0);
                        }});
                setSize(300,200);
                setVisible(true);
                setLocationRelativeTo(null);
        }

public static void  main(String  args[]){
        ExemploInputDialog  app = new ExemploInputDialog();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```



Caixa do InputDialog

Quando você clicar no Ok.
Veja o resultado, o nome vai aparecer todo em maiúsculo

15.4 – Como fornecer os valores a serem selecionados em um diálogo

O uso de diálogos prompt para obter entrada do usuário é muito conveniente, mas há situações em que o usuário deve apenas selecionar um dos valores fornecidos pelo aplicativo. Para essas situações você pode usar o seguinte método da classe JOptionPane.

Veja a lista:

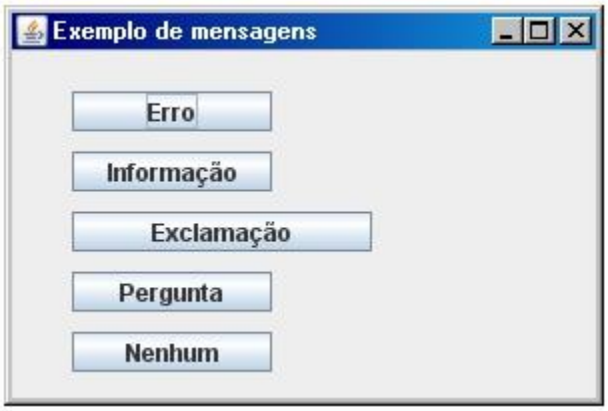
```
Import javax.swing.JOptionPane;

public class ExemploInputDialog1{
   public static void main(String args[]){
                Object linguagens [] = {"Java","Delphi","C++","Visual Basic"};
                Object opcao = JOptionPane.showInputDialog(null,"Qual sua linguagem favorita?
" ,"Enquete",JOptionPane.QUESTION_MESSAGE,null,linguagens,linguagens[0]);
               JOptionPane.showMessageDialog(null,"Você escolheu: "+opcao);
                System.exit(0);
  }
}
```

15.5 – Entendimento e uso do método showMessageDialog.

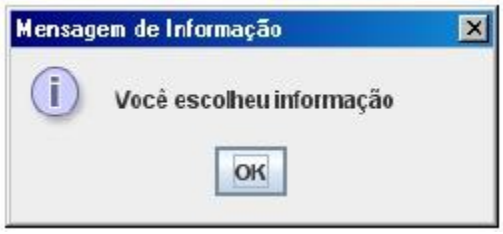
Como o leitor já conhece todos os parâmetros, nos determinamos apenas no parâmetro messaType. É aqui que definimos o tipo de mensagem que será exibido. Os valores possíveis são: ERROR_MESSAGE, INFORMATION_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE. Cada um desses valores produz uma mensagem diferente, como mostra o seguinte aplicativo:

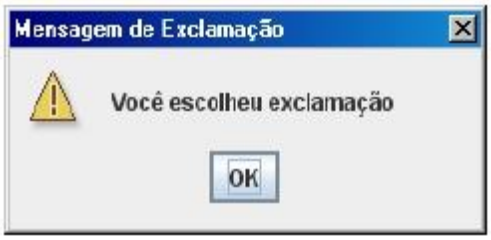
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class ExemploDeMessagens extends JFrame{
   JButton erro, informacao, exclamacao, pergunta, nenhum;
   public ExemploDeMessagens(){
      super("Exemplo de mensagens");
      Container tela = getContentPane();
      tela.setLayout(null);
      erro = new JButton("Erro");
      informacao = new JButton("Informação");
      exclamacao = new JButton("Exclamação");
      pergunta = new JButton("Pergunta");
      nenhum = new JButton("Nenhum");
      erro.setBounds(30,20,100,20);
      informacao.setBounds(30,50,100,20);
      exclamacao.setBounds(30,80,150,20);
      pergunta.setBounds(30,110,100,20);
      nenhum.setBounds(30,140,100,20);
      tela.add(erro);
      tela.add(informacao);
      tela.add(exclamacao);
      tela.add(pergunta);
      tela.add(nenhum);

      erro.addActionListener(new ActionListener(){public void actionPerformed(ActionEvent e){
                  JOptionPane.showMessageDialog(null,"Você escolheu erro","Mensagem de
                  Erro",JOptionPane.ERROR_MESSAGE,null);}})

      informacao.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){
                  JOptionPane.showMessageDialog(null,"Você escolheu informação","Mensagem de
                  Informação",JOptionPane.INFORMATION_MESSAGE,null);}});

      exclamacao.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){
                 JOptionPane.showMessageDialog(null,"Você escolheu exclamação","Mensagem de
                 Exclamação",JOptionPane.WARNING_MESSAGE,null);}});

      pergunta.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){
                  JOptionPane.showMessageDialog(null,"Você escolheu pergunta","Mensagem de
                  Pergunta",JOptionPane.QUESTION_MESSAGE,null);}});

       nenhum.addActionListener(new ActionListener(){ public void actionPerformed(ActionEvent e){
                  JOptionPane.showMessageDialog(null,"Você escolheu
                   nenhum","Mensagem",JOptionPane.PLAIN_MESSAGE,null);}});

    setSize(300,200);
    setVisible(true);
    setLocationRelativeTo(null);
   }

public static void main(String args[]){
   ExemploDeMessagens app = new ExemploDeMessagens();
   app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

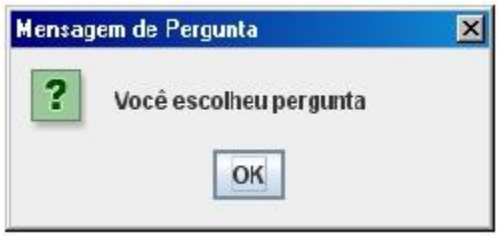
Tela inicial do aplicativo



Quando você clicar no botão a mensagem correspondente vai aparecer.







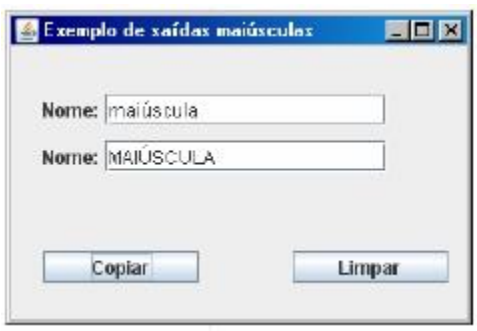
Veja um trecho de código que produz a segunda mensagem:

JOptionPane.showMessageDialog(null,"Você escolheu pergunta","Mensagem de Pergunta",JOptionPane.QUESTION_MESSAGE,null);}});

16 – Maiúsculas e Minúsculas

Nada como você entrar com os dados em minúsculos e obter sua saída em maiúsculos, ou vice versa. Para colocar a saída em maiúsculo utilize o método: toUpperCase(). Observe o aplicativo abaixo:

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

public class ExemploMaiusculas extends JFrame{
        JButton copiar,limpar;
        JLabel rotulo1,rotulo2;
        JTextField texto1,texto2;
        public ExemploMaiusculas(){
                super("Exemplo de saídas maiúsculas");
                Container tela = getContentPane();
                tela.setLayout(null);
                rotulo1 = new JLabel("Nome:  ");
                rotulo2 = new JLabel("Nome:  ");
                texto1 = new JTextField(20);
                texto2 = new JTextField(20);
                copiar = new JButton("Copiar");
                limpar = new JButton("Limpar");
                rotulo1.setBounds(20,30,50,20);
                rotulo2.setBounds(20,60,50,20);
                texto1.setBounds(60,30,180,20);
                texto2.setBounds(60,60,180,20);
                copiar.setBounds(20,130,100,20);
                limpar.setBounds(180,130,100,20);
                copiar.addActionListener(new  ActionListener(){
                 public void actionPerformed(ActionEvent  e){
                    texto2.setText(texto1.getText().toUpperCase());}});
                 limpar.addActionListener(new  ActionListener(){
                 public void actionPerformed(ActionEvent  e){
                    texto1.setText("");
                    texto2.setText("");
                    texto1.requestFocus();}});
                tela.add(rotulo1);
                tela.add(rotulo2);
                tela.add(texto1);
                tela.add(texto2);
                tela.add(copiar);
                tela.add(limpar);
                setSize(300,200);
                setVisible(true);
                setLocationRelativeTo(null);  }

public static void main(String args[]){
        ExemploMaiusculas app = new ExemploMaiusculas();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
        }
```



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

public class ExemploMinusculas extends JFrame{
        JButton copiar,limpar;
        JLabel rotulo1,rotulo2;
        JTextField texto1,texto2;
        public ExemploMinusculas(){
                super("Exemplo de saídas minúsculas");
                Container tela = getContentPane();
                tela.setLayout(null);
                rotulo1 = new JLabel("Nome:  ");
                rotulo2 = new JLabel("Nome:  ");
                texto1 = new JTextField(20);
                texto2 = new JTextField(20);
                copiar = new JButton("Copiar");
                limpar = new JButton("Limpar");
                rotulo1.setBounds(20,30,50,20);
                rotulo2.setBounds(20,60,50,20);
                texto1.setBounds(60,30,180,20);
                texto2.setBounds(60,60,180,20);
                copiar.setBounds(20,130,100,20);
                limpar.setBounds(180,130,100,20);
                copiar.addActionListener(new  ActionListener(){
                public void actionPerformed(ActionEvent  e){
                    String texto;
                    texto = texto1.getText().toUpperCase();
                    texto1.setText(texto);
                   texto2.setText(texto1.getText().toLowerCase());}});
                limpar.addActionListener(new  ActionListener(){
                public void actionPerformed(ActionEvent  e){
                   texto1.setText("");
                   texto2.setText("");
                   texto1.requestFocus();}});
                tela.add(rotulo1);
                tela.add(rotulo2);
                tela.add(texto1);
                tela.add(texto2);
                tela.add(copiar);
                tela.add(limpar);
                setSize(300,200);
                setVisible(true);
                setLocationRelativeTo(null);}

public static void main(String args[]){
        ExemploMinusculas app = new ExemploMinusculas();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);}  }
```

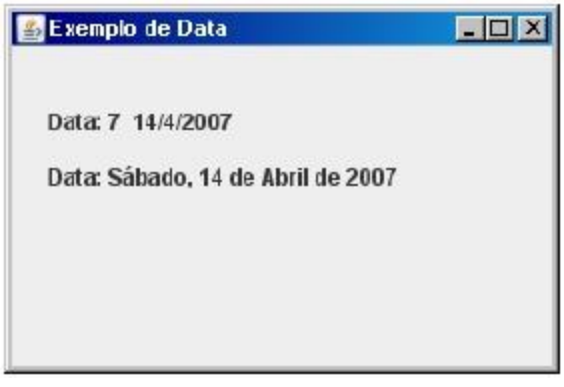
Para colocar em minúsculos, basta você chamar o método toLowerCase(), se você digitar a letra minúscula, quando você clicar em copiar, o texto da primeira caixa vai ficar maiúsculas e a segunda fica minúsculas. E o botão limpar serve para limpar as duas caixas e coloca o foco para a primeira caixa.

17 – Adicionando Data nos seus aplicativos

O aplicativo abaixo mostra dos tipos de data, somente com números e a outra com o dia da semana e os meses escritos com o nome.

Vamos ver o aplicativo:

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

public class Data extends JFrame{
  JLabel rotulo,rotulo2;
  int ds,dia,mes,ano;
  Calendar data;
  String diasemana[]={"Domingo","Segunda - Feira","Terça - Feira","Quarta - Feira",
                      "Quinta - Feira","Sexta - Feira","Sábado"};
  String meses[]={"Janeiro","Fevereiro","Março","Abril","Maio","Junho",
                  "Julho","Agosto","Setembro","Outubro","Novembro","Dezembro"};
  public Data(){
    super("Exemplo de Data");
    Container tela = getContentPane();
    tela.setLayout(null);
    rotulo = new JLabel("");
    rotulo2 = new JLabel("");
    rotulo.setBounds(20,30,280,20);
    rotulo2.setBounds(20,60,280,20);
    data = Calendar.getInstance();
    ds = data.get(Calendar.DAY_OF_WEEK);
    dia = data.get(Calendar.DAY_OF_MONTH);
    mes = data.get(Calendar.MONTH);
    ano = data.get(Calendar.YEAR);
    rotulo.setText("Data: "+ds+" "+dia+"/"+(mes+1)+"/"+ano);
    rotulo2.setText("Data: "+diasemana[ds-1]+", "+dia+" de "+meses[mes]+" de "+ano);
    tela.add(rotulo);
    tela.add(rotulo2);
    setSize(300, 200);
    setVisible(true);
    setLocationRelativeTo(null);   }

  public static void main(String args[]){
    Data app = new Data();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);   } }
```



18 – Adicionando Horas ao seu aplicativo

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.DecimalFormat;
import javax.swing.Timer;

public class Hora extends JFrame{
  JLabel rotulo;
  int hh,mm,ss,h;
  Calendar hora;
  DecimalFormat formato;
  public Hora(){
    super("Exemplo de Hora");
    Container tela = getContentPane();
    tela.setLayout(null);
    rotulo = new JLabel("");
    rotulo.setBounds(20,30,280,20);
    ActionListener tarefa = (new ActionListener(){
          public void actionPerformed(ActionEvent e){
                HORAS();
          }
    });
    javax.swing.Timer time = new javax.swing.Timer(1000,tarefa);
    time.start();
    tela.add(rotulo);
    setSize(300, 200);
    setVisible(true);
    setLocationRelativeTo(null);
  }
  public static void main(String args[]){
    Hora app = new Hora();
    app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
  private void HORAS(){
    hora = Calendar.getInstance();
    hh = hora.get(Calendar.HOUR_OF_DAY);
    mm = hora.get(Calendar.MINUTE);
    ss = hora.get(Calendar.SECOND);
    formato = new DecimalFormat("00");
    rotulo.setText("Hora: "+formatar(hh%12)+":"+formatar(mm)+":"+formatar(ss));
  }
  private String formatar(int num){
          formato = new DecimalFormat("00");
          return formato.format(num);
  }
}
```

Vamos criar uma classe privada com o nome de HORAS vamos definir os parâmetros para hora, minutos e segundos.

Depois criar um outra classe privada com o nome de formatar para que você possa formatar a hora com dois dígitos, e coloca a hora no rótulo com uma string.

Criar uma ação com o nome de tarefa, que vai chamar a classe privada HORAS.

Por último chamar a biblioteca com o nome de Timer, e atribuir o tempo de um segundo que é representado por mil. E chamar o método time.start(), para que o relógio possa movimentar os segundos.

Aplicativo bem interessante execute veja como funciona.