INTERFACE GRÁFICA COM O USUÁRIO

COMPONENTES

1. JLabel – é um rótulo, ou seja, um componente para exibir na tela uma única linha de texto para leitura, é possível exibir junto com o texto também uma imagem. Os rótulos raramente tem o seu conteúdo alterado.

1.1 Médodo setToolTipText() que exibe automaticamente uma mensagem quando passa o mouse sobre o rótulo
Ex: jLabel1. setToolTipText(“Este é o Label 1”);

1.2 Método setIcon() adiciona uma imagem ao Label. Dá para usar os métodos setHorizontalTextPosition() e setVerticalTextPosition() para alinhar o ícone em relação ao texto do rótulo.

Ex:

```
ImageIcon imagem = new ImageIcon("cliente.png");
jLabel1.setIcon(imagem);
jLabel1.setHorizontalTextPosition(SwingConstants.LEFT);   //o texto fica à esquerda da imagem
jLabel1.setHorizontalTextPosition(SwingConstants.RIGHT);   //o texto fica à direita
jLabel1.setHorizontalTextPosition(SwingConstants.CENTER);   //o texto fica em cima da imagem

jLabel1.setVerticalTextPosition(SwingConstants.BOTTOM);   //o texto fica embaixo
jLabel1.setVerticalTextPosition(SwingConstants.TOP);   //o texto fica em cima
jLabel1.setVerticalTextPosition(SwingConstants.CENTER);   //o texto fica no centro
```

Dica: site para baixar ícones gratuitamente:
https://icons8.com/

2. JtextField (campo de texto) e JpasswordField (campo de senha) são campos para digitar texto em uma única linha ou apenas exibir um texto. O JpasswordField mostra que foram digitados caracteres porém oculta os caracteres digitados pois serve para digitar senhas.
O JpasswordField acrecenta uma série de métodos específicos para tratar senhas.
Esses componentes
2.1 Método setEditable(false) faz com que o usuário não tenha permissão de editar o campo de texto.
2.2 Método setEnabled(false) faz com que o campo de texto fique desabilitado.
2.3 Tratamento de eventos
Quando o usuário digita Enter em um campo de texto um evento de ação ocorre. Se um ouvinte de evento é registrado, o evento é processado e os dados dos campos de texto podem ser utilizados no programa.

Para tratar os eventos de um campo de texto é necessário criar um objeto da classe TextFieldHandler que será um ouvinte dos eventos do campo de texto, ou seja, ele que irá fazer o tratamento dos eventos. Irá executar algo quando o usuário der Enter, por exemplo.

Ex: criamos uma classe (dentro da outra classe) que é a tratadora de eventos, a classe que trata eventos sempre precisa implementar a interface ActionListener

```
//classe definida para tratamento de eventos
```

```
private class TextFieldHandler implements ActionListener{
    public void actionPerformed(ActionEvent e){
        String s = "";
        if (e.getSource()==jTextField1){
            s = "texto campo nome:" + e.getActionCommand();
        }else{
            if (e.getSource()==jPasswordField1){
                s = "texto senha:" + e.getActionCommand();
            }
        }
        JOptionPane.showMessageDialog(null, s);
    }
}
```

Após criar a classe que chamamos de TextFieldHandler precisamos instanciar um objeto desta classe (chamamos de handler) e precisamos adicionar esse objeto (ouvinte de eventos) ao jTextField e ao jPasswordField, de acordo com o exemplo abaixo:

```
TextFieldHandler handler = new TextFieldHandler();  //instanciar um objeto ouvinte (handler)
jTextField1.addActionListener(handler);          //adicionamos o ouvinte no jTextField
jPasswordField1.addActionListener(handler);
```

3. Jbutton é um botão em que o usuário clica para acionar uma ação específica.
Podemos colocar além do rótulo de texto também ícone nos botões. Fazemos isso com o método setIcon() e ainda podemos colocar ícones roláveis que só aparecem quando passarmos com o mouse sobre o botão.

```
Ex:    ImageIcon imagem = new ImageIcon("cliente.png");
       jButton1.setIcon(imagem);
       jButton2.setRolloverIcon(imagem);
```

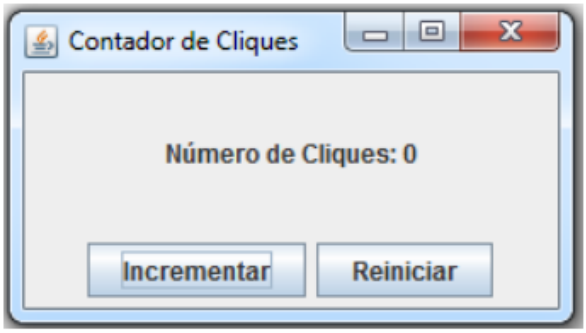
Os botões, assim como os jTextoField também geram eventos. Então, tratamos os eventos de forma parecida com os campos de texto. De acordo com o exemplo abaixo:
Ex: criamos uma classe (dentro da outra classe) que é a tratadora de eventos, a classe que trata eventos sempre precisa implementar a interface ActionListener

```
//classe interna para tratamento de eventos nos botões
   private class ButtonHandler implements ActionListener{
       public void actionPerformed(ActionEvent e){
           JOptionPane.showMessageDialog(null,
                "Você pressionou:" +e.getActionCommand());
       }
   }
```

Após criar a classe que chamamos de ButtonHandler precisamos instanciar um objeto desta classe (chamamos de h) e precisamos adicionar esse objeto (ouvinte de eventos) aos jButtons, de acordo com o exemplo abaixo:

```
ButtonHandler h = new ButtonHandler();
jButton1.addActionListener(h);
jButton2.addActionListener(h);
```

Exercício: Criar uma aplicação que conte o número de cliques em um botão, de acordo com a janela à seguir:



4. JCheckBox (caixa de seleção) – é um botão de estado que pode ter apenas dois valores que seriam ativado/desativado ou verdadeiro/falso.

ItemEvent- quanto o usuário clica em um CheckBox é gerado um ItemEvent que pode ser tratado por um ItemListener que deve definir o método itemStateChanged. Esse método é chamado quando um usuário clica na CheckBox.

Ex: CheckBoxHandler ch = new CheckBoxHandler();
negrito.addItemListener(ch);
italico.addItemListener(ch);

```
//definir a classe CheckBoxHandler que tratará os eventos da CheckBox
private class CheckBoxHandler implements ItemListener{
    private int valNegrito = Font.PLAIN;
    private int valItalico = Font.PLAIN;
    public void itemStateChanged(ItemEvent e){
        if (e.getSource()==italico){
            if (e.getStateChange()== ItemEvent.SELECTED){
                valItalico = Font.ITALIC;
            }else {
                valItalico = Font.PLAIN;
            }
        }
        if (e.getSource()==negrito){
            if (e.getStateChange()== ItemEvent.SELECTED){
                valNegrito = Font.BOLD;
            }else {
                valNegrito = Font.PLAIN;
            }
        }
        jTextField2.setFont(
          new Font("TimesRoman",valNegrito+valItalico,14));
        jTextField2.repaint();
    }
}
```