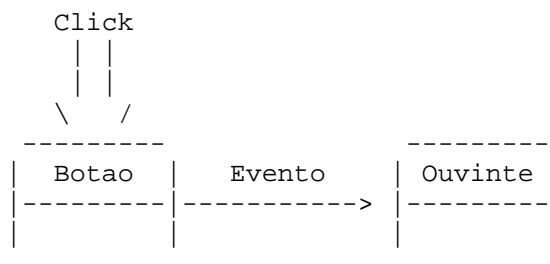


Tratamento de eventos

- Componentes da GUI geram eventos
- Eventos informam ações ocorridas com os componentes. Exemplo:
 - \$ Clique de um botão
 - \$ ENTER em um campo texto
 - \$ Click em um Menu

Quando uma ação ocorre em um componente, ele envia um evento para um objeto "ouvinte" (Listener)



Um objeto "ouvinte" deve implementar uma "interface" que define a assinatura do método que será executado a partir do "componente".

Exemplo:

```
class OuvinteBotao implements ActionListener {
    public void actionPerformed(ActionEvent evento) {
        //Aqui colocamos o que deve ser feito
    }
}
```

Para que esse método possa ser encontrado, devemos informar ao "componente", qual será o objeto "ouvinte" para uma determinada ação. Exemplo:

```
JButton botao = new JButton("Ok");

OuvinteBotao ouvinte = new OuvinteBotao();

botao.addActionListener(ouvinte);
```

Dessa forma, quando o botão for pressionado, o "componente" (do botão) sabe que deve executar o método "actionPerformed" no objeto "ouvinte".

Segue abaixo uma lista de ações bastante comuns em componentes:

| Fato ocorrido | Evento | Interface do ouvinte | Método da interface | Método add |
|----------------------------------|-------------|----------------------|------------------------------|-------------------|
| Click em JButton | ActionEvent | ActionListener | actionPerformed(ActionEvent) | addActionListener |
| ENTER em JTextField | " | " | " | " |
| Click em menu | " | " | " | " |
| Click do mouse | MouseEvent | MouseListener | mouseClicked(MouseEvent) | addMouseListener |
| Alteração texto de um JTextField | CaretEvent | CaretListener | caretUpdate(CaretEvent) | addCaretListener |

Utilização de classes internas anônimas

Classes internas anônimas são muito usadas para declarar manipuladores de eventos da GUI. Seu uso evita a criação separada de classes adicionais, deixando todo esse trabalho para o próprio compilador. Segue abaixo um exemplo de declaração de um manipulador de evento utilizando uma classe anônima.

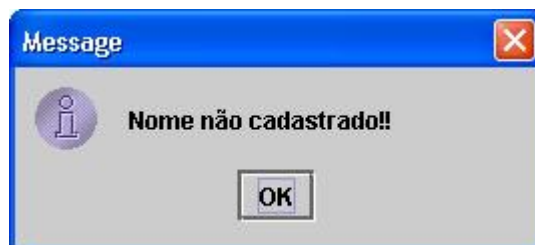
```
botao.addActionListener( new ActionListener() {  
    public void actionPerformed(ActionEvent evt) {  
        pressionouIncluir();  
    }  
});
```

Perceba que o operador "new" cria um objeto do tipo "ActionListener", cujo corpo da classe é definido logo em seguida (entre chaves verdes). Dentro da classe anônima (diz-se anônima porque não foi definido seu nome), temos a declaração do método "actionPerformed". Perceba também que o parênteses (em vermelho) do parâmetro de "addActionListener" envolve toda a declaração da classe anônima.

Essa classe anônima será automaticamente criada em tempo de compilação, e receberá o nome da classe onde está inserida, acrescentado do caracter "\$" e um número de sequência. Suponha que o manipulador de evento acima tenha sido criado dentro de uma classe chamada "FormPrincipal". Ao compilar o programa surgirá um arquivo chamado "FormPrincipal\$1.class". Se outros manipuladores de evento forem criados dentro da mesma classe, receberão o mesmo nome porém com o número de sequência incrementado. Ex: "FormPrincipal\$2.class", "FormPrincipal\$3.class" e assim por diante.

Caixas de diálogo

A biblioteca do Java possui uma classe chamada JOptionPane que permite facilmente apresentar caixas de diálogo para o usuário. Uma das formas mais simples de utilização desta classe é através do método estático "showMessageDialog", que permite apresentar uma mensagem para o usuário através de uma janela titulada como "Message". Exemplo:



A seguinte instrução pode ser usada para apresentar a mensagem acima:

```
JOptionPane.showMessageDialog(this,"Nome não cadastrado!!");
```

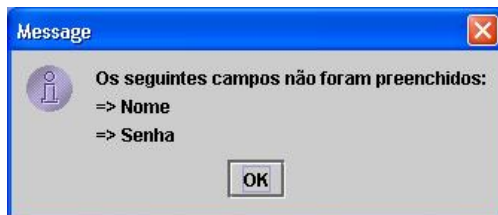
Exercício

- 1) Construa a seguinte janela:

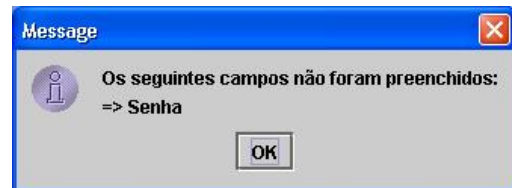


A screenshot of a Windows-style window titled "Janela Principal". It contains two text input fields, one labeled "Nome" and one labeled "Senha". Below these fields are three buttons: "Confirma", "Limpa", and "Sair".

- 2) Ao pressionar o botão "Limpa" deve-se limpar o conteúdo dos dois campos.
- 3) Ao pressionar o botão "Sair" a aplicação deve ser encerrada.
- 4) Apresentar uma mensagem caso o usuário pressione "Confirma" sem ter preenchido o conteúdo dos campos "Nome" ou "Senha". Nesta mensagem deve ser informado que campo não foi preenchido. Exemplo:

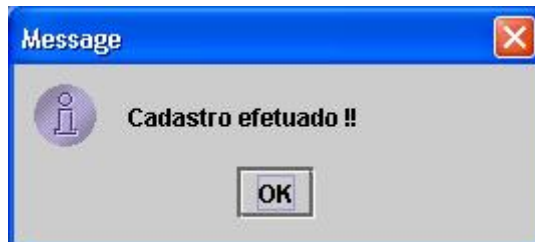


A screenshot of a "Message" dialog box. It contains an information icon, the text "Os seguintes campos não foram preenchidos:", and two lines of text: "=> Nome" and "=> Senha". There is an "OK" button at the bottom.



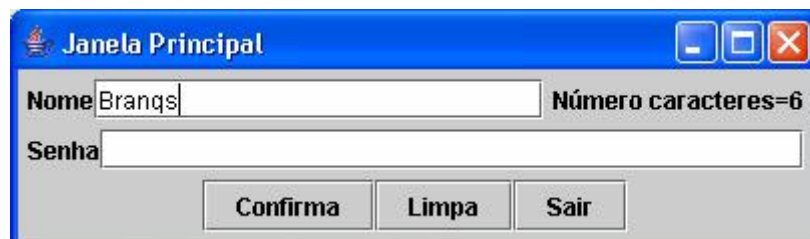
A screenshot of a "Message" dialog box. It contains an information icon, the text "Os seguintes campos não foram preenchidos:", and one line of text: "=> Senha". There is an "OK" button at the bottom.

- 5) Apresentar as mesmas mensagens acima caso o usuário pressione a tecla "ENTER" sem ter preenchido algum campo.
- 6) Ao pressionar a tecla "ENTER" ou o botão "Confirma" com os dois campos preenchidos corretamente, deve-se limpar os campos e apresentar a seguinte mensagem:



A screenshot of a "Message" dialog box. It contains an information icon, the text "Cadastro efetuado !!", and an "OK" button at the bottom.

- 7) Criar um campo que informe o número de caracteres utilizados no campo "Nome" durante a digitação. Exemplo:



A screenshot of the "Janela Principal" window. The "Nome" field now contains the text "Branqs" and a label "Número caracteres=6" is displayed to its right. The "Senha" field is empty. The "Confirma", "Limpa", and "Sair" buttons are still present.