Se você parar para reparar, um aplicativo gráfico é como se fosse como um cardápio de um restaurante. Você chega, faz seus pedidos e suas ordens são atendidas.
Cada pessoa vai fazer pedidos diferentes, tanto dos pratos como do número de coisas ordenadas.

Da mesma maneira é uma aplicação gráfica em Java.
Um programa nada mais é que uma série de opções, onde cada usuário vai fazer o que quiser, clicando em botões, apertando teclas do teclado, rolando a barra de informações, marcando, selecionando, escrevendo, minimizando, fechando e uma infinidade de possibilidades.

Cada vez que o usuário faz uma destas coisas, dizemos que foi realizado um evento.
Ou seja, um click, o mouse passou em alguma região e algo mudou, ele escreveu algo, deu enter etc etc.

O que a GUI (*Graphic User Interface*) faz é nada mais que tratar estes eventos.
Se ele apertar isso, acontece aquele, Se digitar isso, aquilo abre.
Se clicar aqui, aquilo vai fechar. Se apertar enter, vai pra próxima janela etc etc.
Ou seja, um aplicativo gráfico é uma maneira do usuário realizar pedidos e comandos de uma maneira bem mais simples e intuitiva.
O usuário realiza o evento, e uma ação ocorre.

Como tratar eventos - A interface *ActionListener* e o método *actionPerformed*

Existe uma classe em Java que será a responsável pelo tratamento de eventos.

Ou seja, é nela que vamos identificar o evento que ocorreu e é nela que vamos definir que ações nossos aplicativos devem executar quando tal evento ocorrer.

A classe *ActionListener* é uma interface, ou seja, é classe composta apenas de métodos abstratos.

E isso é até óbvio, pois a classe não tem como saber que tipos de eventos vamos tratar em um aplicativo, muito menos vai saber que tipo ação queremos que nosso software tome quando um evento ocorrer.

Listener pode ser traduzido como 'ouvinte'.

E isso faz sentido, pois essa interface é que vai ficar 'esperando' algo ocorrer. É como se ela ficasse em um loop infinito, o tempo todo testando:

"Ocorreu algo agora? E agora? Foi um evento? É um evento? E agora? E agora?..."

Ou seja, a interface fica na 'escuta', na espreita até o usuário fazer alguma interação com o aplicativo e um evento ocorrer.

A interface *ActionListener* possui somente um método. Então temos apenas um método obrigatório para implementar, que é o método *actionPerformed*, e ele é o responsável por tomar alguma ação caso algum evento ocorra.

Assim, a interface fica na espera de algum evento, e caso ocorra ele é imediatamente passado para o método *actionPerformed*. É dentro deste método que iremos definir as ações que ocorrem.

Para fazer uso dessas funcionalidades, devemos importar:

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

Event Handling - Criando um tratador de eventos

Agora que já temos noção do que é a interface *ActionListener* e seu método *actionPerformed*, vamos aprender como, de fato, tratar um evento.

O tratamento é feito por um objeto da classe *ActionListener*. Há várias maneiras de se criar esse objeto na prática, vamos aprender mais no próximo tutorial, mas neste exemplo vamos fazer de uma maneira bem didática.

Como já estudamos JButton, vamos fazer um aplicativo que irá mostrar uma caixa de diálogo quando clicamos no botão ok ou no botão cancelar.

Para iniciar, vamos criar nossa classe que irá implementar a *ActionListener*, vamos chamar de "ButtonHandler", (handle é manusear).

Assim, o esqueleto de nosso tratador é:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ButtonHandler implements ActionListener {

        public void actionPerformed(ActionEvent evento) {


        }

}
```

Note que que o método *actionPerformed* recebe um argumento 'evento' do tipo *ActionEvent*.

Este argumento 'evento' vai armazenar a natureza do evento, ele sabe especifica e exatamente que componente ocorreu o evento.

Tal componente está armazenado no método *getSource()* deste objeto.

No nosso exemplo, vamos criar dois botões chamados "ok" e "cancela".

Esses botões foram declarados na classe "Botao.java", que usamos no exemplo do artigo passado.

Assim, quando formos criar o *tratador* (handler) pros botões, precisamos passar estes dois botões para nossa classe "ButtonHandler", e esta classe recebe eles por meio do construtor.

Para saber que botão foi clicado, basta fazer testes condicionais para saber o que está guardado em:

evento.getSource()

Dependendo do botão que foi clicado, iremos exibir uma mensagem.

Uma dizendo que o "OK" foi clicado, e o outro exibe a mensagem dizendo que o botão "CANCELAR" foi pressionado.

As mensagens serão exibidas através das *Dialog Boxes*(Caixas de diálogo).

Assim, nossa classe "ButtonHandler" que irá tratar os eventos dos botões é:

ButtonHandler.java

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JOptionPane;

public class ButtonHandler implements ActionListener {
        private JButton ok, cancela;

        public ButtonHandler(JButton ok, JButton cancela){
                this.ok = ok;
                this.cancela = cancela;
        }

        public void actionPerformed(ActionEvent evento) {
                if(evento.getSource() == ok)
```

```
                                JOptionPane.showMessageDialog(null, "O
botão OK foi clicado");

                if(evento.getSource() == cancela)
                        JOptionPane.showMessageDialog(null, "O botão
CANCELA foi clicado");
        }

}
```

O método *addActionListener* - Adicionando um tratador de eventos aos componentes

Pronto, nosso tratador de eventos dos botões, ou handler, foi construído, que é nossa classe "ButtonHandler". Vamos criar um objeto dessa classe, e chamar de "handler":

ButtonHandler handler = new JButton(ok,cancela);

Pronto, agora temos um tratador de botões do tipo 'ok' e 'cancela', que é o objeto *handler*.

Porém, podem existir vários componentes, e para cada um deles há a possibilidade de existir um tratador de evento diferente.

Clicar em um botão é diferente de escrever um texto numa caixa de texto. Numa você digita, e na outra componente você clica.

Assim, para cada componente que vamos criar temos que definir que tratador de eventos aquele objeto vai usar.

Isso é definido passando um objeto do tipo *ActionListener* para o método **addActionListener**, existente nos componentes.

Como queremos tratar apenas os botões 'ok' e 'cancela', e o tratador deles é o mesmo, fazemos:

ok.addActionListener(handler);

cancela.addActionListener(handler);

O código de nossa classe "Botao", que é um JFrame com dois JButtons fica:

Botao.java

```
import java.awt.FlowLayout;
import javax.swing.JFrame;
```

```
import javax.swing.JButton;

public class Botao extends JFrame{
        private JButton ok = new JButton("OK");
        private JButton cancela = new JButton("Cancela");
        private ButtonHandler handler;

        public Botao(){
                super("Criando botões");
                setLayout(new FlowLayout());
                handler=new ButtonHandler(ok, cancela);

                ok.addActionListener(handler);
                add(ok);

                cancela.addActionListener(handler);
                add(cancela);
        }

}
```

E a classe principal é:

Main.java

```
import javax.swing.JFrame;

public class Main {
        public static void main(String[] args) {
                Botao botoes = new Botao();


botoes.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                botoes.setSize(350,70);
                botoes.setVisible(true);
        }

}
```

Se rodarmos o projeto, teremos os seguintes resultados sempre que clicarmos em "OK" ou "CANCELAR":

TURMA 2111A	LINGUAGEM DE PROGRAMAÇÃO II	Data 29-02-16