UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

Programação Orientada a Objectos II

Tratamento de eventos em Java

Docente: Ruben Manhiça

Maputo, Agosto de 2016

Conteúdo da Aula

1. Elementos de uma aplicação
2. Tipos de eventos

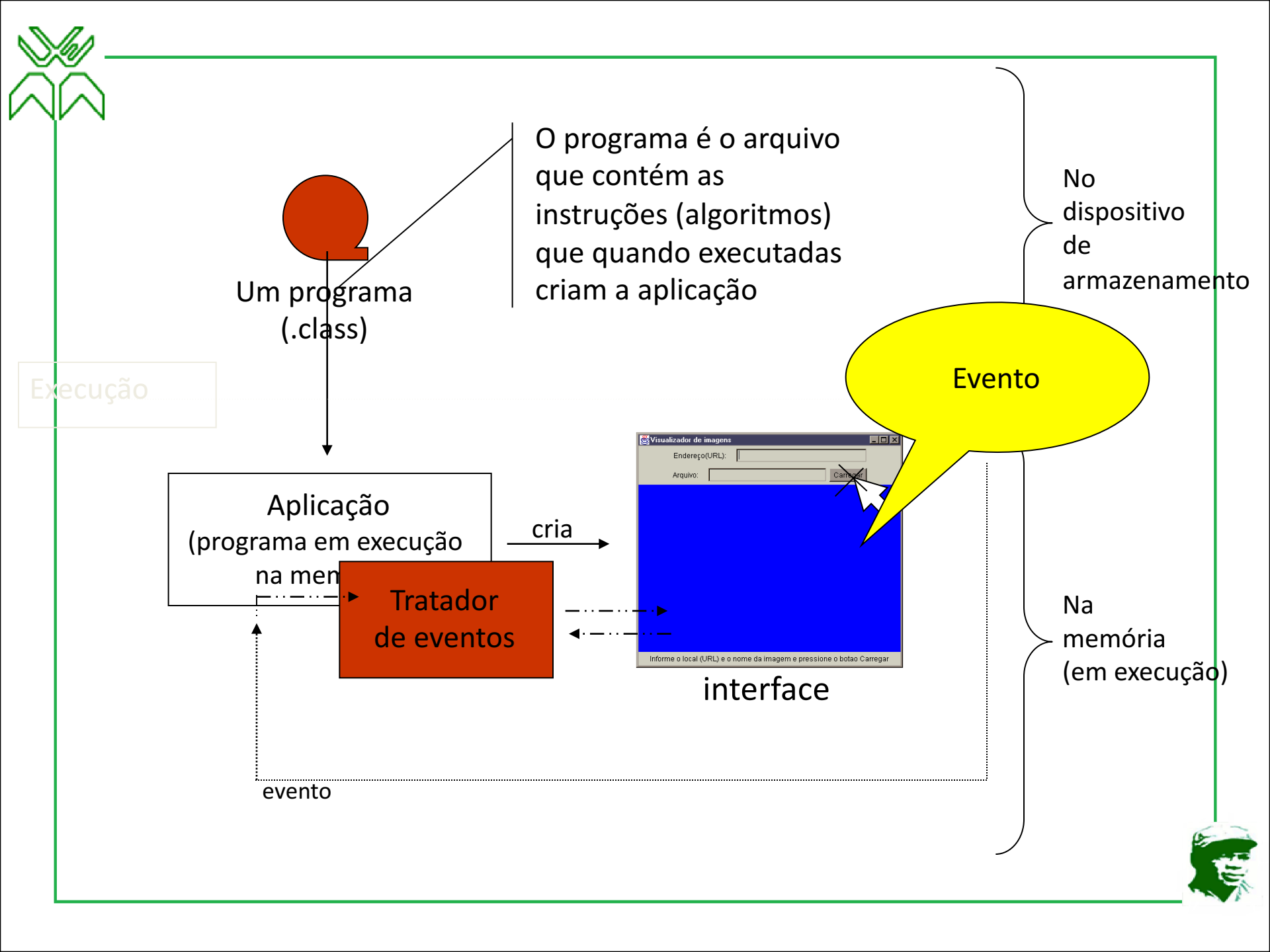
Considerações iniciais

Uma aplicação não é constituída somente de sua interface.

É comum pensarmos que um programa ou aplicação é constituído somente de sua interface, já que é isso o que vemos.

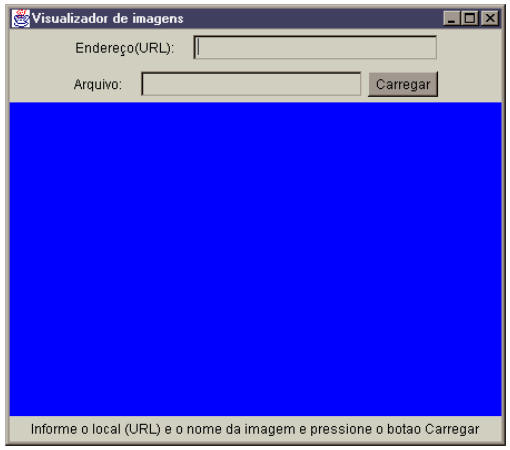
Porém, Como programadores, devemos saber que a interface é somente um dos componentes da aplicação. Ela pode ser definida como sendo o meio pelo qual o programa estabelece comunicação com o usuário (informa e recebe dados e informações).

Elementos de uma aplicação


O programa é o arquivo que contém as instruções (algoritmos) que quando executadas criam a aplicação
Um programa (.class)
Execução
Aplicação (programa em execução na mem
cria
Tratador de eventos
Visualizador de imagens
Endereço(URL):
Arquivo:
Carregar
Informe o local (URL) e o nome da imagem e pressione o botao Carregar
interface
Evento
evento
No dispositivo de armazenamento
Na memória (em execução)

Principais tipos de eventos

Principais tipos de eventos



Labels

Panel

TextFields

Frame

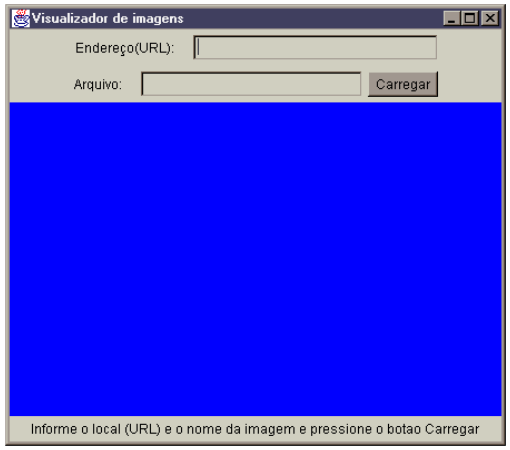
Button

Labels não utilizam muitos eventos.

Um dos únicos eventos que faz sentido deles gerar é o evento que avisa que o mouse está a passar por cima deles.

Em Java, cada componente de interface tem seu conjunto específico de eventos.

Principais tipos de eventos



Labels

Panel

TextFields

Frame

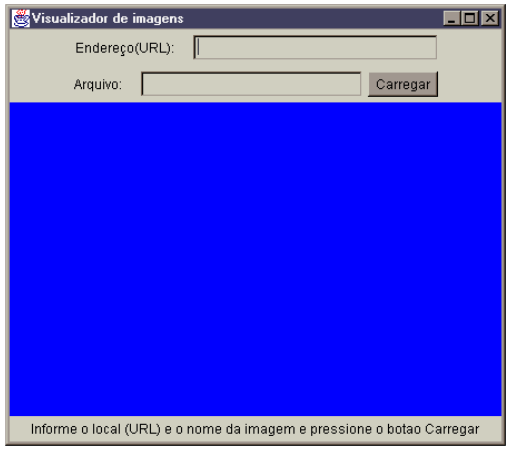
Button

Em Java, cada componente de interface tem seu conjunto específico de eventos.

Com Panels outros eventos mais interessantes podem ser capturados:

- Movimento do mouse;
- Acção de mouse;
- Teclado;
- Re-pintura/actualização;

Principais tipos de eventos



Labels

Panel

TextFields

Frame

Button

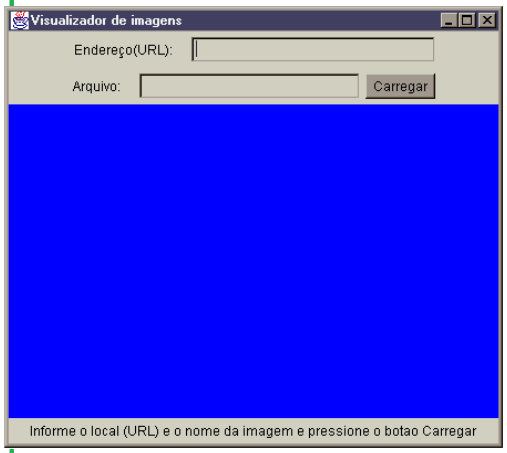
Em Java, cada componente de interface tem seu conjunto específico de eventos.

Como os TextFields são caixas de texto que recolhem informações do utilizador, faz sentido capturar deles os eventos de:

- Teclado;
- Alteração de seu conteúdo;

(ou até mesmo os eventos de movimento ou ação do mouse, em alguns casos)

Principais tipos de eventos



Labels

Panel

TextFields

Frame

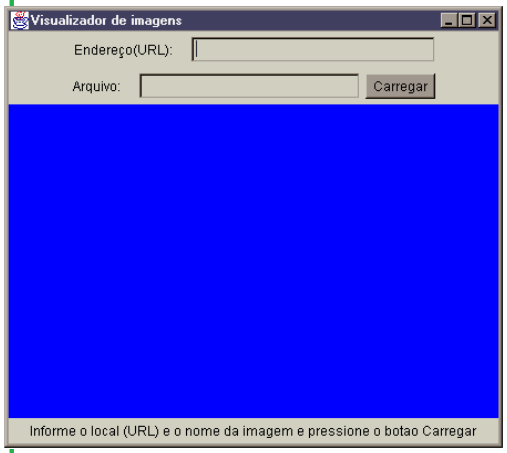
Button

Em Java, cada componente de interface tem seu conjunto específico de eventos.

Os Frames são parecidos com os Painéis. Na verdade eles são painéis com bordas! Logo, além dos eventos de um painel, eles também geram eventos de janelas:

- Movimento do mouse;
- Acção de mouse;
- Teclado;
- Re-pintura/atualização;
- Manipulação de janela;

Principais tipos de eventos



Labels

Panel

TextFields

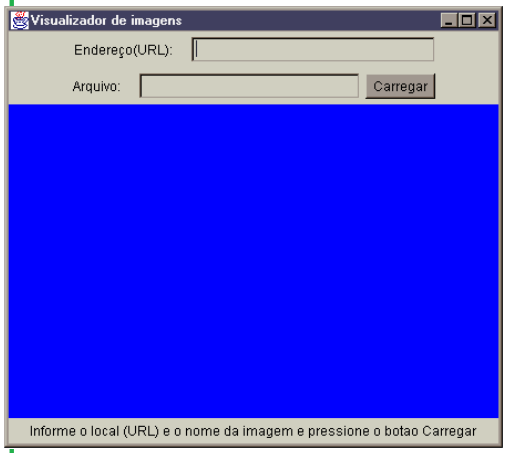
Frame

Button

Em Java, cada componente de interface tem seu conjunto específico de eventos.

Já os Buttons (botões), geram eventos de acção (pressionado, solto...) e de mouse (clicado, passando por cima...)

Principais tipos de eventos



Labels

Panel

TextFields

Frame

Button

Logo, existem diferentes classes de eventos e tratadores de eventos, que devem ser utilizadas de acordo com os componentes e as necessidades que o programador possui.

Em Java, cada componente de interface tem seu conjunto específico de eventos.

Decorar todos os tipos e tratadores de eventos que o Java possui para cada objeto não faz sentido. Ao invés disso, vamos utilizar os tratadores de eventos principais.

Principais tipos de eventos

- **Eventos de janela** (WindowEvents)

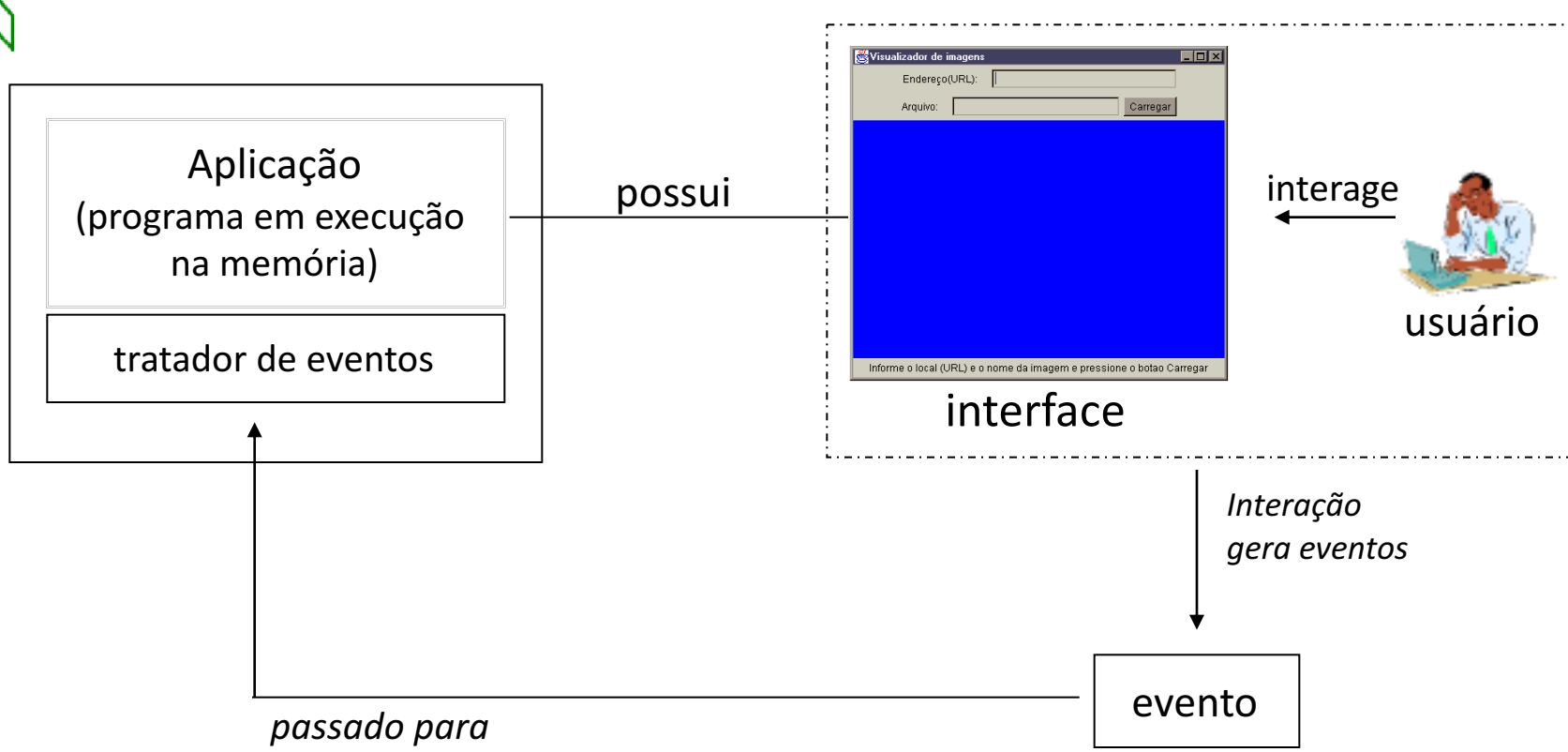
  Gerados quando uma janela é aberta, fechada, maximizada ou minimizada, entre outros.

- **Eventos de acções** ocorridas em componentes (ActionEvents)

  Gerados quando um componente sofre uma acção gerada pelo usuário (seleção de um elemento ou clique do mouse em um botão, por exemplo).

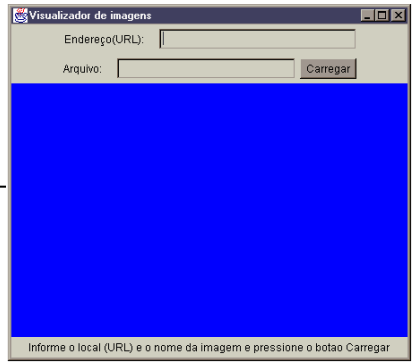
- **Eventos gerados pelo mouse** (MouseEvents)
  - Pelo ***movimento do mouse***;
  - Por uma ***acção do mouse*** (botão clicado, pressionado ou solto);
- Eventos gerados pelo teclado (KeyEvents).

Capturando eventos


Aplicação
(programa em execução
na memória)
tratador de eventos
possui
Visualizador de imagens
Endereço(URL):
Arquivo:
Carregar
Informe o local (URL) e o nome da imagem e pressione o botao Carregar
interface
interage
usuário
Interação
gera eventos
evento
passado para

Aplicação
(programa em execução
na memória)

possui



interface

interage

usuário

tratador de
eventos de janela
WindowListener

WindowEvent

tratador de eventos de
movimentos do mouse
MouseMotionListener

MouseEvent

tratador de eventos de
ações do mouse
MouseListener

MouseEvent

tratador de eventos de
ações em componentes
ActionListener

ActionEvent

tratador de
eventos de teclado
KeyListener

KeyEvent

Interação
gera eventos

Porém, dependendo do tipo de evento, o tratador de eventos correspondente é escolhido

Tipos de tratadores de eventos

Logo, o Java possui 5 tipos básicos de tratadores de eventos que você pode utilizar em seus programas:

- WindowListener - eventos de janelas
- MouseListener - eventos de mouse (clique)
- MouseMotionListener - eventos de movimento de mouse
- ActionListener - eventos de acção (geralmente gerados por botões)
- KeyListener - eventos gerados pelo teclado

Para cada um desses tipos o Java oferece uma Classe ou Interface que você pode utilizar em seus programas. Cada um deles possui uma série diferente de métodos, que tratam eventos específicos.

Tratando eventos

Os eventos não são tratados automaticamente. Para fazer isso você deve seguir os seguintes passos:

- Para cada componente de interface que você criou (janela, botão, painel, caixa de texto...), decida quais são os eventos que você quer tratar (cada componente pode gerar um ou mais tipos de eventos);
- Após, defina uma classe adicional no seu programa que seja capaz de tratar cada um desses eventos. Essa classe, tratadora de eventos, deve ser uma classe filha de uma das classes tratadoras de eventos vistas anteriormente (WindowListener, MouseListener, MouseMotionListener, ActionListener ou KeyListener);
- Finalmente, crie objetos que sejam do tipo da classe tratadora de eventos que você definiu e depois diga para cada componente, qual é o objeto que trata seus eventos.

Tipos de tratadores de eventos:

- Para capturar e tratar os eventos você deve criar objetos de manipulação de eventos. Para cada tipo de evento existe uma classe Java específica para tratá-lo.
- Você só tem que definir uma classe adicional em seu programa que seja filha da classe Java que trata o tipo de evento que você deseja manipular.
- Para os eventos vistos, as classes-pai que você pode utilizar são:
  - Eventos de janela (WindowEvent): WindowListener
  - Eventos de Ação (ActionEvent): ActionListener
  - Eventos de mouse (MouseEvent):
    - MouseMotionListener → para movimentos do mouse
    - MouseListener → para demais ações do mouse
  - Eventos de teclado (KeyEvent): KeyListener

Exemplo

Para exemplificar, vamos criar um tratador de eventos de janela e adicioná-lo à janela da nossa aplicação:

O primeiro passo é criar uma classe que defina o tratador de eventos que queremos. Isso é feito através da adição de uma nova classe no final do programa fonte da nossa aplicação. Vamos então criar uma classe chamada de, por exemplo, TratadorDeEventos, e vamos estende-la de WindowAdapter:

```
class TratadorDeEventos extends WindowAdapter // filha do tratador de eventos de janela
```

Depois, dentro dela, declare e implemente todos os métodos que tratam os eventos que você deseja capturar (neste caso, somente o evento de fechar janela):

```
{ // abre a classe
    public void windowClosing(WindowEvent e)   // método chamado quando o
    {                                                    // usuário clica o botão fechar
       System.exit(0);    // Esse comando faz terminar o programa
    }
 } // Fecha a classe
```

Exemplo

- Cada evento gerado chama um método diferente (e correspondente) para tratá-lo. Todos os eventos de *fechar janela*, por exemplo, chamam o mesmo método: windowClosing().
- Isso significa que, para cada evento que você quer tratar, você deve descobrir qual é o método que o trata e implementá-lo.
- Cada uma das classes-pai (tratadoras de eventos) possui seu conjunto de métodos específico.
  - WindowListener ou WindowAdapter
  - MouseListener ou MouseAdapter
  - MouseMotionListener ou MouseMotionAdapter
  - ActionListener

Obs: A diferença entre um Listener e um Adapter é que o primeiro é uma interface, e você deve implementar (implements) todos os seus métodos. Já o segundo é uma classe-pai pronta, e você pode redefinir somente os métodos para os eventos que lhe interessam

Exemplo

- Depois de ter criado um tratador de eventos, falta dizer para o Java que os objetos criados a partir dessa classe é que vão tratar os eventos.
- Para fazer isso, no programa principal crie um objeto do tipo TratadorDeEventos (que é a classe definida por nós para tratar eventos de janela) e depois adicione esse tratador de eventos ao objeto janela de sua aplicação (no caso isso é feito através do método addWindowListener():

```
public class VImagem2 // Classe que define o programa principal
{
  public static void main(String args[])
  {
    Janela minhaJanela = new Janela();
      // cria objeto tratador de eventos:
    TratadorDeEventos tratador = new TratadorDeEventos();
    // Adiciona o tratador de eventos à janela:
    minhaJanela.addWindowListener(tratador);
    minhaJanela.show(); // SertVisible(true)
  }
}
```

Métodos de adição de tratadores de eventos

- Para cada componente que você deseja tratar eventos, você deve criar e adicionar o tratador de eventos correspondente.
- Cada tipo de adaptador possui o seu método de adição correspondente:

  addWindowListener*(tratador_de_eventos)* → adiciona um tratador de eventos de janela;

  addActionListener*(tratador_de_eventos)* → adiciona um tratador de eventos gerados por ações em componentes (geralmente utilizado em botões);

  addMouseListener*(tratador_de_eventos)* → adiciona um tratador de eventos gerados pelo movimento do mouse;

  addMouseMotionListener*(tratador_de_eventos)* → adiciona um tratador de eventos gerados por ações do mouse;

  addKeyListener*(tratador_de_eventos)* → adiciona um tratador de eventos gerados pelo teclado;

Exemplo de fonte de aplicação que trata eventos:

```
public class Aplicação{
   public static void main(String argumentos[]){
      Janela  jan  = new Janela();
      TratEventosJan trat = new TratEventosJan();
      jan.addWindowListener(TratEventos());
      jan.show();
   }
}

class Janela extends Frame{
   public Janela(){
      setBackground(Color.blue);
      add("Center", new Label("Janela da aplicação"));
   }
}

class TratEventosJan extends WindowAdapter{
   public void windowClosing(WindowEvent evento){
      System.exit(0);
   }
}
```

A primeira classe é a que define a aplicação

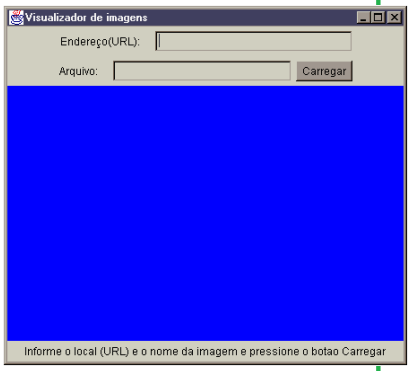
A segunda classe é a que define a interface (o tipo de janela) da aplicação

A última classe é a que define o tratador de eventos de janela da aplicação

Exemplo de fonte de aplicação que trata eventos:

```
public class Aplicação{

   public static void main(String argumentos[]){

      Janela          jan  = new Janela();

      TratEventosJan trat = new TratEventosJan();

      jan.addWindowListener(trat);
      jan.show();

   }

}
```

A primeira classe é responsável por criar a janela da aplicação.




```
class Janela extends Frame{

   public Janela(){

      setBackground(Color.blue);

      add("Center", new Label("Janela da aplicação"));

   }

}
```

A janela é criada de acordo com a classe de janela definida!

```
class TratEventosJan extends WindowAdapter{

   public void windowClosing(WindowEvent evento){

      System.exit(0);

   }

}
```

Exemplo de fonte de aplicação que trata eventos:

```
public class Aplicação{
   public static void main(String argumentos[]){
      Janela          jan  = new Janela();
      TratEventosJan trat = new TratEventosJan();
      jan.addWindowListener(trat);
      jan.show();
   }
}

class Janela extends Frame{
   public Janela(){
      setBackground(Color.blue);
      add("Center", new Label("Janela da aplicação"));
   }
}

class TratEventosJan extends WindowAdapter{
   public void windowClosing(WindowEvent evento){
      System.exit(0);
   }
}
```

Em seguida, é criado o objeto tratador de eventos.

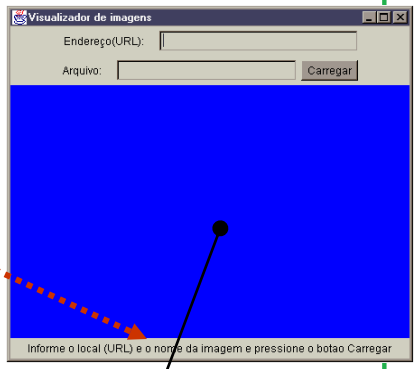
trat
(tratador de eventos)

Ele é criado de acordo com a classe definida para tratar eventos!

Exemplo de fonte de aplicação que trata eventos:

```
public class Aplicação{
   public static void main(String argumentos[]){
      Janela          jan  = new Janela();
      TratEventosJan trat = new TratEventosJan();
      jan.addWindowListener(trat);
      jan.show();
   }
}

class Janela extends Frame{
   public Janela(){
      setBackground(Color.blue);
      add("Center", new Label("Janela da aplicação"));
   }
}

class TratEventosJan extends WindowAdapter{
   public void windowClosing(WindowEvent evento){
      System.exit(0);
   }
}
```

Finalmente, o tratador de eventos é adicionado à janela:



trat
(tratador de eventos)

Assim, todo evento de janela gerado nesta janela é capturado e tratado pelo objeto tratador de eventos.

Eventos gerados por janelas

- Toda janela gera um evento de janela (WindowEvent) quando sofre uma acção do usuário.

Esses são os métodos que você tem que declarar/implementar para capturar eventos de janela em seu programa

Quando a janela surge, ela gera um evento do tipo windowOpened(WindowEvent ev)

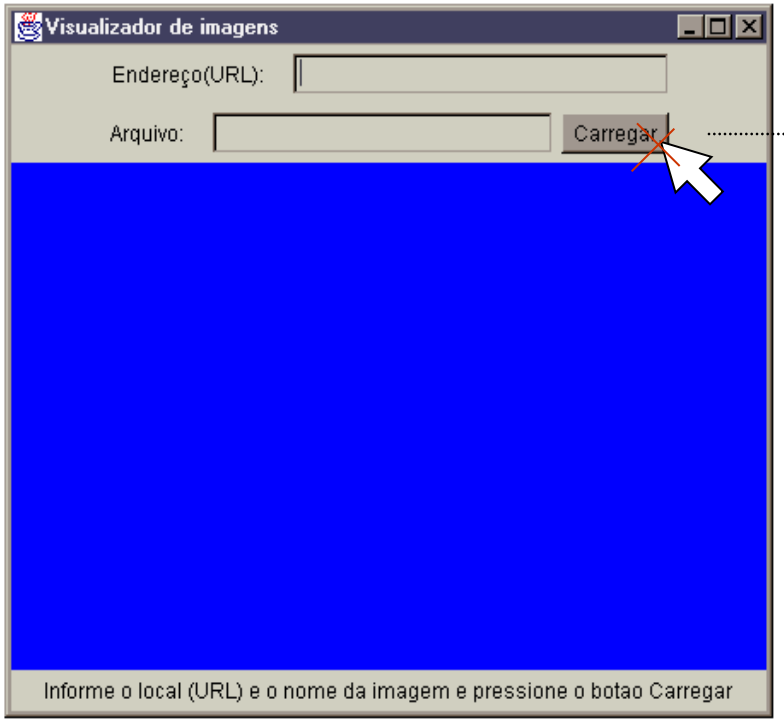
Quando o usuário manda minimizar a janela, ela gera um evento do tipo windowIconified(WindowEvent ev)

Quando o usuário manda maximizar a janela, ela gera um evento do tipo windowDeiconified(WindowEvent ev)

Quando o usuário manda fechar a janela, ela gera um evento do tipo windowClosing(WindowEvent ev)

Eventos gerados por componentes

- Toda interação com um componente de interface gera um evento de ação (ActionEvent). Esses eventos são tratados por uma ActionListener. A ActionListener só possui um método: o actionPerformed():



Quando um botão é pressionado ele gera um evento do tipo:

actionPerformed(ActiveEvent ev)

Você pode identificar quem foi que gerou o evento através do método:

ev.getActionCommand()

Eventos gerados pelo mouse

- Os eventos gerados por alguma acção do mouse (clicar um botão, por exemplo), são tratados pela classe MouseListener ou MouseAdapter e suas classes-filhas, definidas pelo programador. Nestas classes, você deve utilizar os seguintes métodos:
  - mouseClicked(MouseEvent ev) → quando um botão do mouse é *clicado*;
  - mouseEntered(MouseEvent ev) → quando o mouse entra em um componente (passa por cima);
  - mouseExited(MouseEvent ev) → quando o mouse sai de um componente;
  - mousePressed(MouseEvent ev) → quando um botão do mouse é pressionado;
  - mouseReleased(MouseEvent ev) → quando um botão do mouse é solto.

Eventos gerados pelo movimento do mouse

- Os eventos gerados por algum movimento do mouse são tratados pela classe MouseMotionListener ou MouseMotionAdapter e suas classes-filhas, definidas pelo programador. Nestas classes, você deve utilizar os seguintes métodos:

  - mouseMoved(MouseEvent ev) → quando o mouse é movido dentro de um componente ou janela;
  - mouseDraged(MouseEvent ev) → quando o mouse é movido com um botão pressionado dentro de um componente ou janela (para implementar o recurso *arrastar-e-soltar*);

WindowEvent (evento de janela)

- O Evento (objeto) passado ao tratador de eventos de janela é o WindowEvent. Dentro de um método que trata os eventos de janela, você pode utilizar esse objecto para descobrir algumas informações sobre o evento. Alguns de seus métodos são:

  paramString() → retorna uma string que identifica o evento;

  getSource() → Retorna o objecto que gerou esse evento.

MouseEvent (evento de mouse)

- O Evento (objeto) passado a qualquer tratador de eventos de mouse é o mesmo: MouseEvent.
- Dentro de um método que trata o evento de mouse você pode utilizar esse objeto para descobrir algumas informações sobre o evento.
- Alguns de seus métodos são:

getX() → retorna a coordenada x (coluna) onde o evento ocorreu;

getY() → retorna a coordenada y (linha) onde o evento ocorreu;

paramString() → retorna uma string que identifica o evento;

source → é uma referência ao objeto que gerou o evento do mouse.

ActionEvent
(evento de Acção em um componente)

- O Evento (objecto) passado a função de tratamento de eventos de acção é o ActionEvent. Dentro de um método que trata o evento você pode utilizar esse objeto para descobrir algumas informações sobre ele. Alguns de seus métodos são:

  getActionCommand() → retorna o nome do comando gerado (geralmente o nome do componente que gerou esse evento);

  getModifiers() → Indica se alguma tecla especial (SHIFT, CONTROL...) estava pressionada quando o evento ocorreu;

  paramString() → retorna uma string que identifica o evento;

  getSource() → Retorna o objeto que gerou esse evento.

FIM!!!

Duvidas e Questões?