Tratando eventos em Java

Sumário

- Considerações iniciais
- Elementos de uma aplicação
- Principais tipos de eventos em Java:
  (WindowEvent, MouseEvent, ActionEvent, KeyEvent)
- Capturando os eventos:
  - Definindo tratadores de eventos
    (WindowAdapter, MouseMotionListener, MouseListener, ActionListener)
  - Adicionando tratadores de eventos em uma aplicação

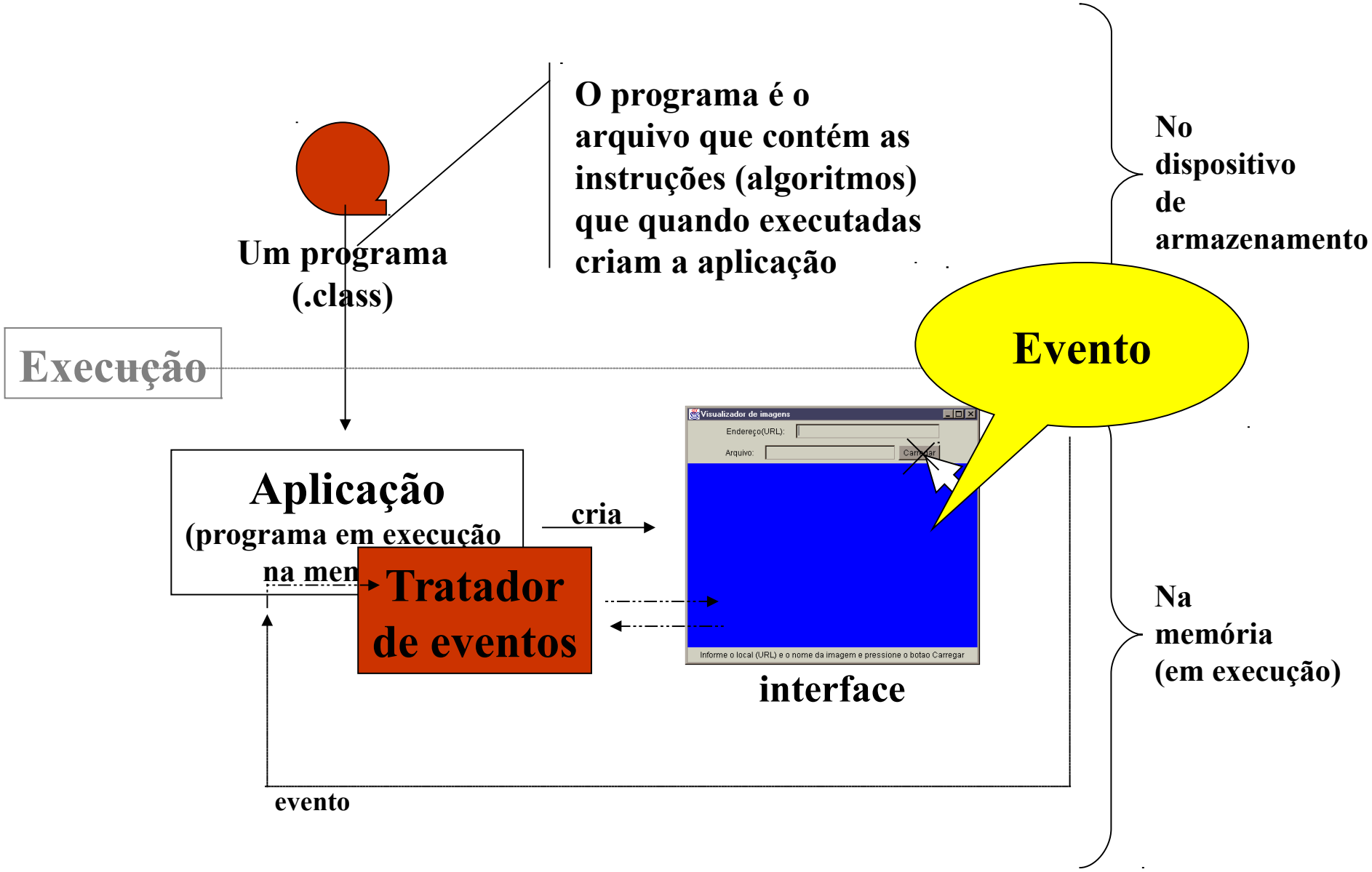
Considerações iniciais

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

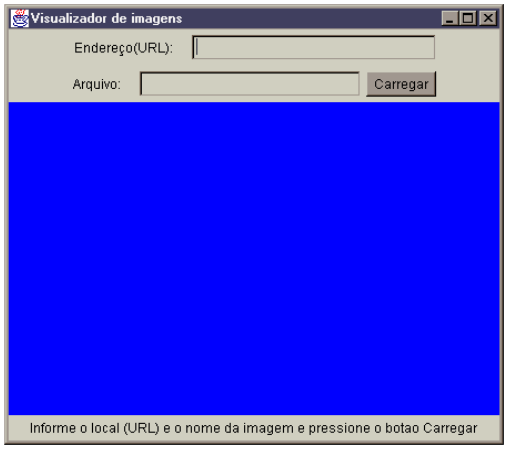

O programa é o arquivo que contém as instruções (algoritmos) que quando executadas criam a aplicação
No dispositivo de armazenamento
Um programa (.class)
Execução
Evento
Aplicação
(programa em execução
na mem
cria
Tratador de eventos
Visualizador de imagens
Endereço(URL):
Arquivo:
Carregar
Informe o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

Programação Orientada a Eventos

- Na programação orientada a eventos, nós precisamos de um loop eterno que fica eternamente esperando por uma entrada do usuário.
- Isto é feito no Java sem nosso conhecimento...
- Só pelo fato de uma GUI estar ativa, este pseudo-loop está rodando.
- Quando um evento ocorre, o gerenciador de janelas (*window manager*) cria um evento e passa para um tratador de eventos definido pelo programador.
- Este processo é chamado de *callback*.
- No final das contas, nosso trabalho é definir os tratadores de eventos...

Principais tipos de eventos

Principais tipos de eventos



Labels

Panel

TextFields

Frame

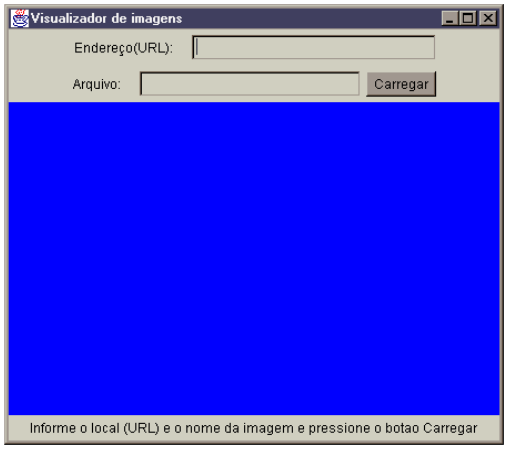
Button

Labels não utilizam muitos eventos.

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

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

Principais tipos de eventos



Labels

Panel

TextFields

Frame

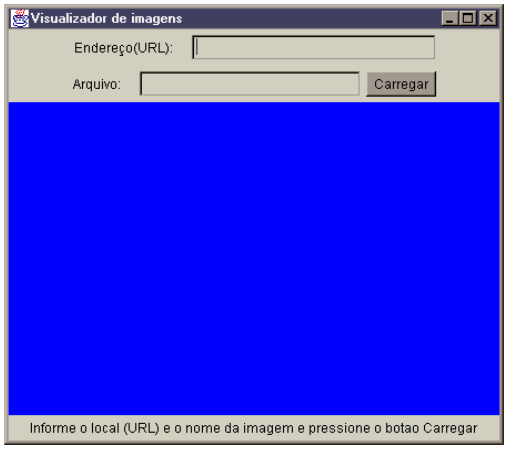
Button

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

Com Panels outros eventos mais interessantes podem ser capturados:

- **Movimento do mouse;**
- **Ação de mouse;**
- **Teclado;**
- **Re-pintura/atualização;**

Principais tipos de eventos



Labels

Panel

TextFields

Frame

Button

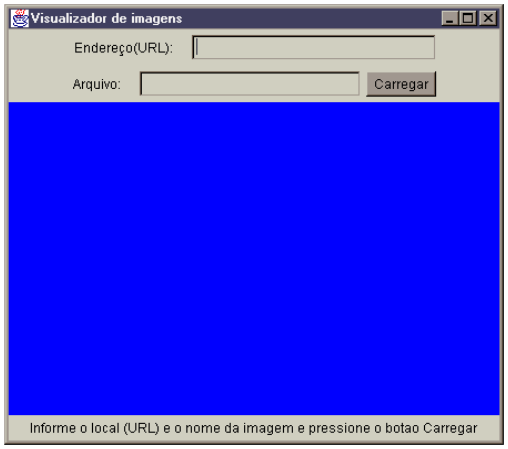
Em Java, cada componente de interface tem seu conjunto específico de eventos.

Como os TextFields são caixas de texto que coletam informações do usuário, 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

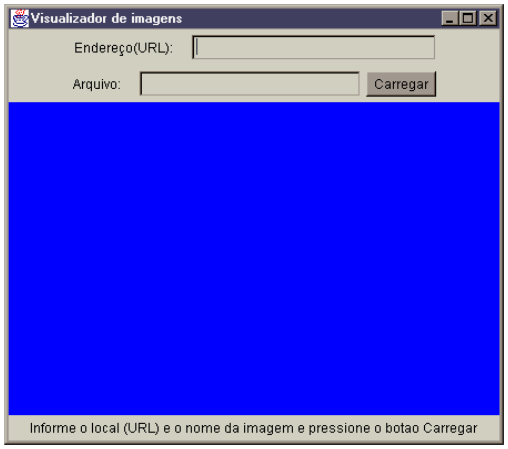
Button

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;**
- **Ação de mouse;**
- **Teclado;**
- **Re-pintura/atualização;**
- **Manipulação de janela;**

Principais tipos de eventos



Labels

Panel

TextFields

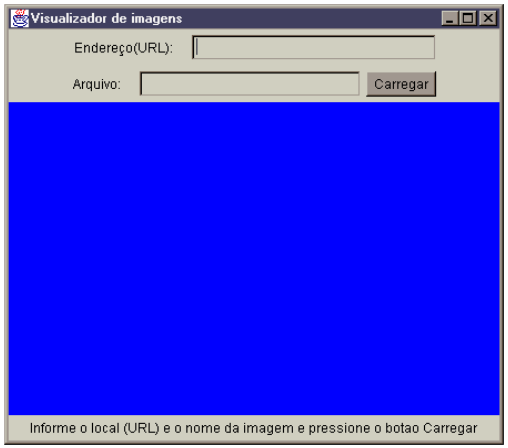
Frame

Button

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

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

Principais tipos de eventos



Labels

Panel

TextFields

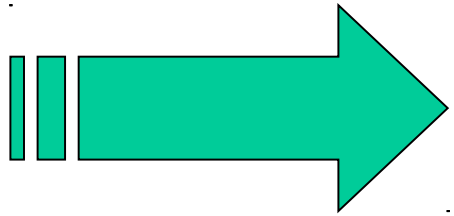
Frame

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)

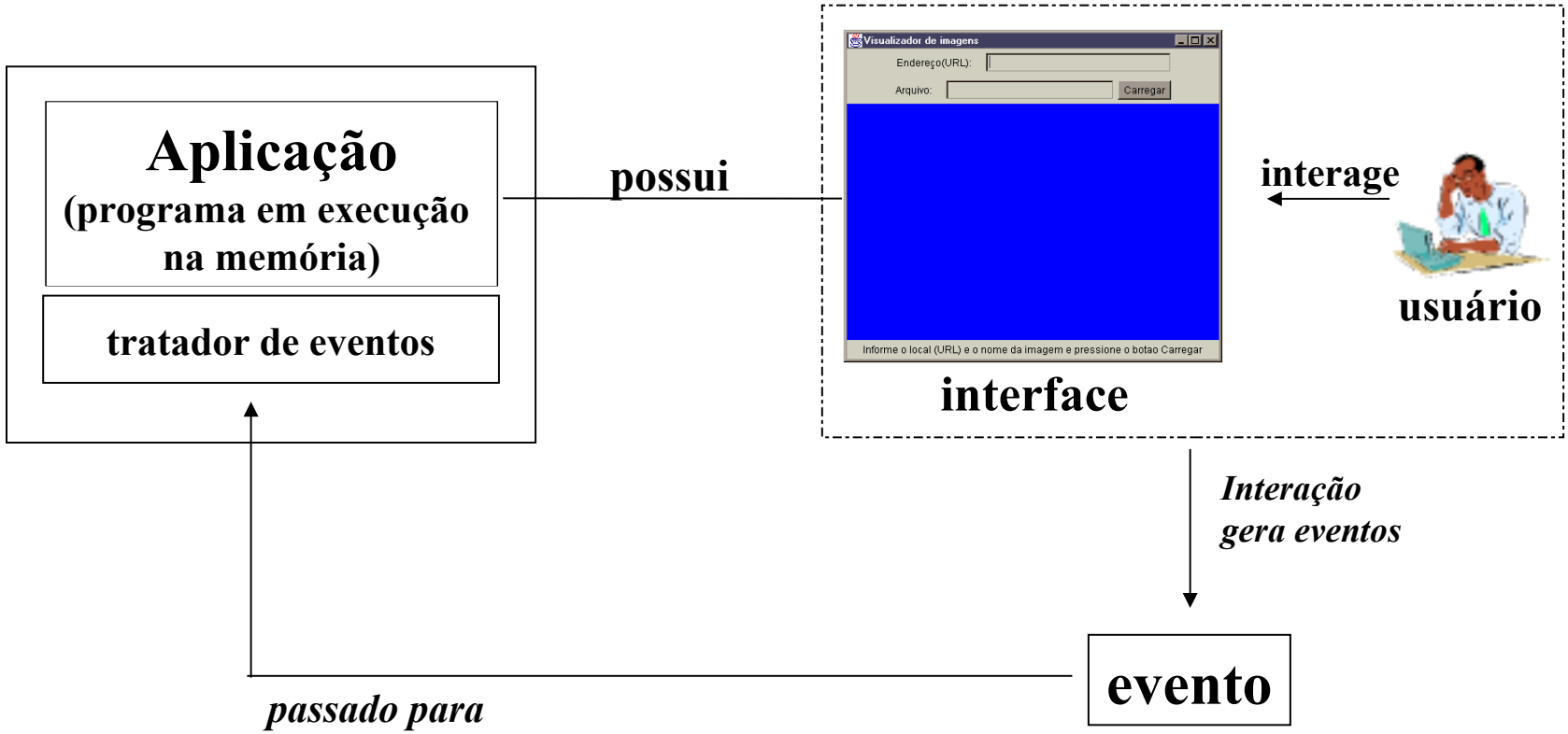
  Gerados quando uma janela é aberta, fechada, maximizada ou minimizada, entre outros.

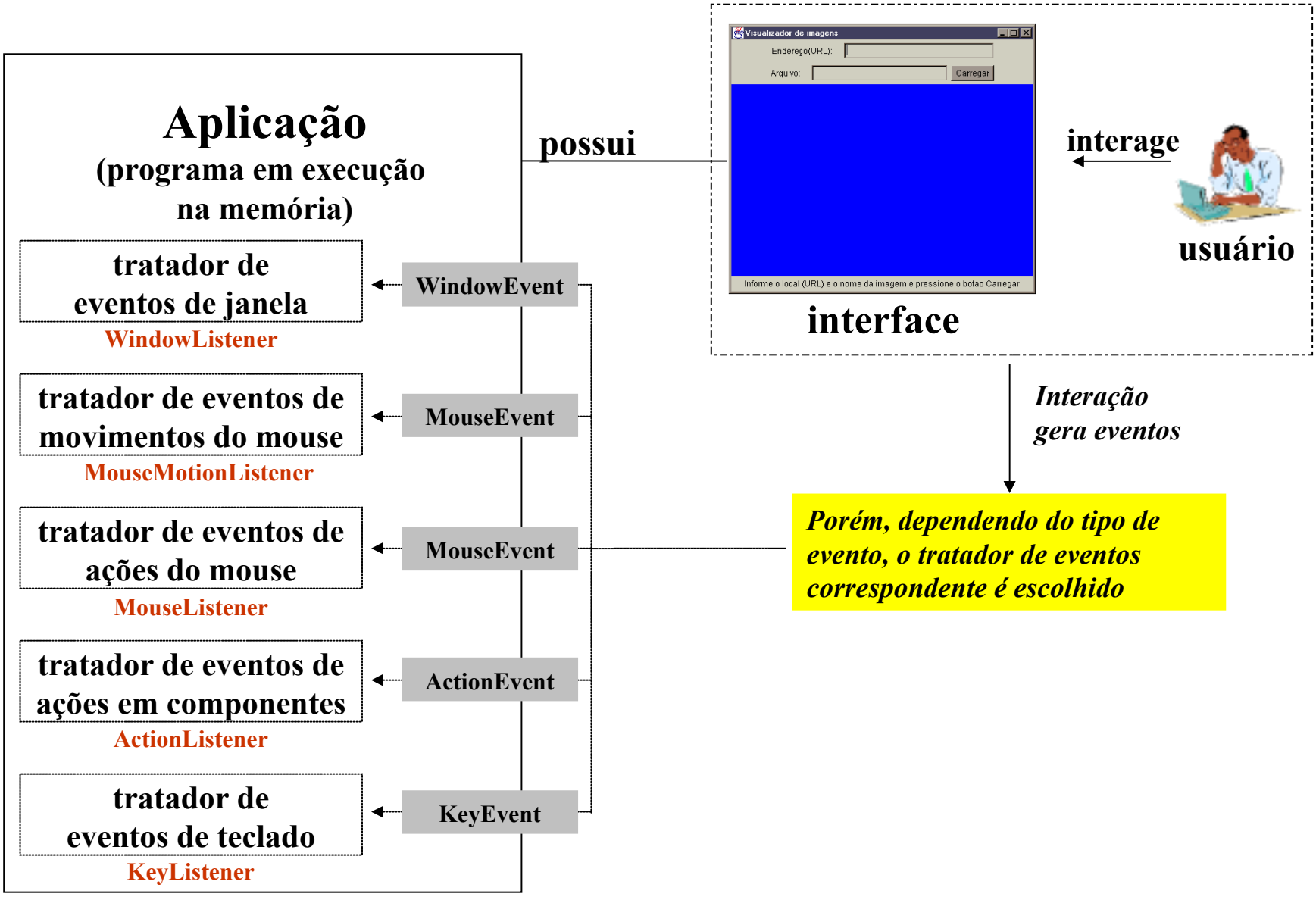
- **Eventos de ações** ocorridas em componentes (ActionEvents)

  Gerados quando um componente sofre uma ação 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 ***ação 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)
tratador de
eventos de janela
WindowListener
tratador de eventos de
movimentos do mouse
MouseMotionListener
tratador de eventos de
ações do mouse
MouseListener
tratador de eventos de
ações em componentes
ActionListener
tratador de
eventos de teclado
KeyListener
WindowEvent
MouseEvent
MouseEvent
ActionEvent
KeyEvent
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
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 ação (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.

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. Clique em uma delas a seguir para descobrir quais são os métodos que elas oferecem e quais eventos cada um deles trata:
  - 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();
   }
}
```

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

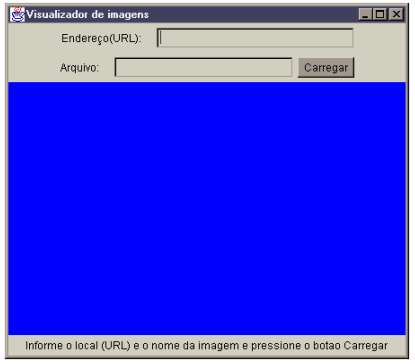
A segunda classe é a que define a interface (o tipo de janela) da aplicação

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!

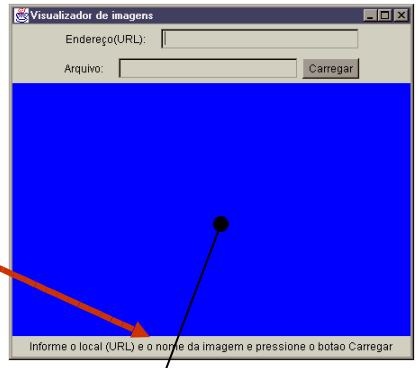
Exemplo de fonte de aplicação que trata eventos:

```
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

Neste tutorial você...

- Aprendeu quais são os elementos de uma Aplicação: o programa, a janela e os tratadores de evento;
- Aprendeu quais são os principais tipos de eventos em Java: WindowEvent, MouseEvent, ActionEvent, KeyEvent;
- Aprendeu a definir classes e objetos que podem ser utilizados nas suas aplicações a fim de capturar e tratar eventos.

Para ver um exemplo
de aplicação que define e
utiliza tratadores de eventos

Tratadores de Eventos

(Listeners e Adapters)

Escutam quando um evento acontece e **recebem** um objeto evento para **tratá-lo** em seus métodos.

Principais Eventos

São enviados aos tratadores de eventos (Listeners e Adapters). Estes os gestionam