UNIP

UNIVERSIDADE PAULISTA

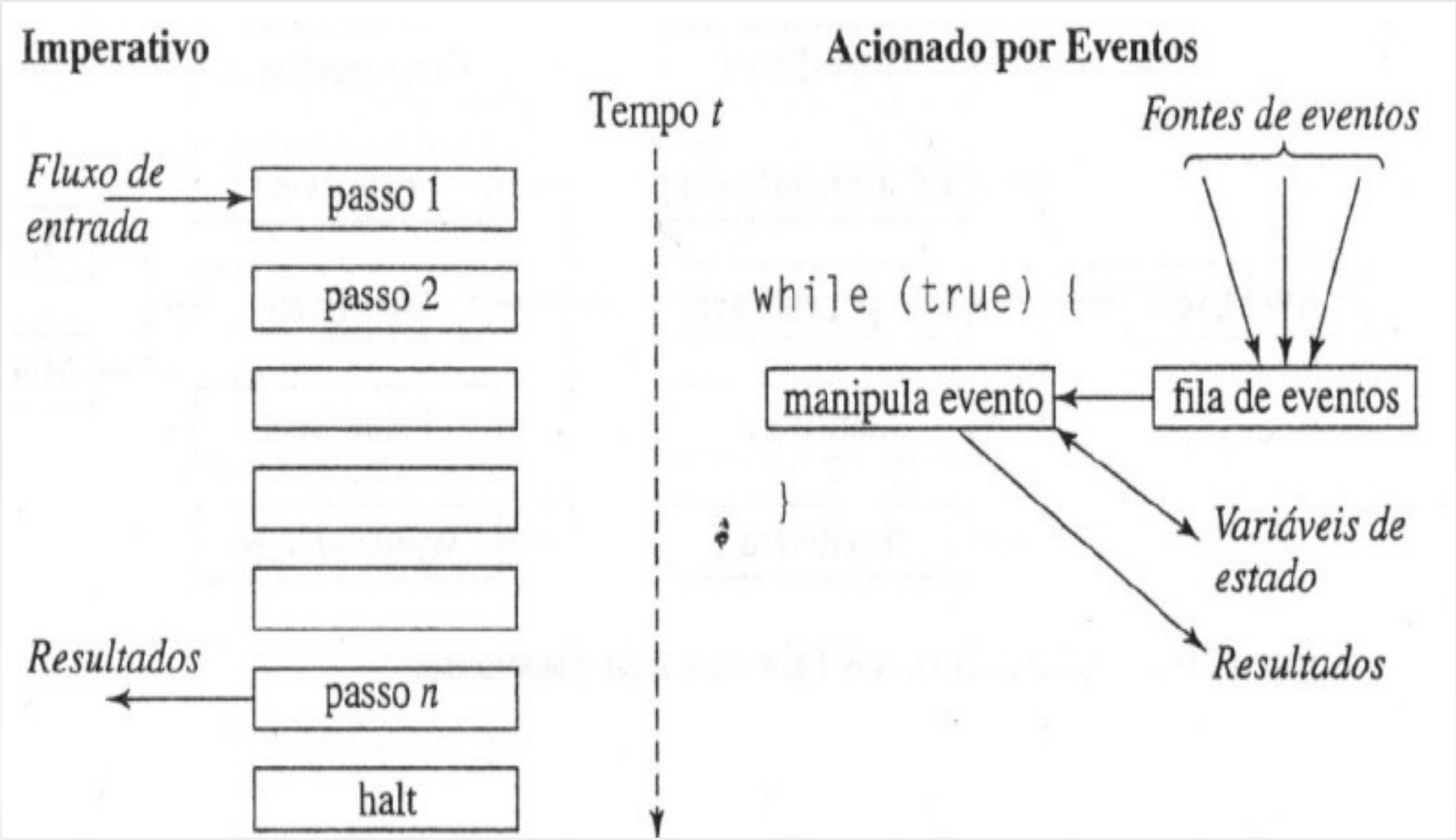
Paradigmas de Linguagens

Paradigma Orientado a Eventos

Professora Sheila Cáceres

Programação Orientada a Eventos

- No modelo imperativo tradicional, o programador determina a ordem de entrada dos dados.
- Em contraste, os programas orientados a eventos **não controlam** a sequência na qual ocorrem eventos de entrada; em vez disso, eles são escritos para **reagir** a qualquer sequência razoável de eventos.
- O sistema em tal paradigma é programado em sua base em um **laço de repetição** de eventos, que **recebem** repetidamente **informação** para processar e disparam uma função de **resposta** de acordo com o evento.
- Ou seja, as entradas governam a sequência de operações executadas pelo programa (programa não determina a ordem da sequência).

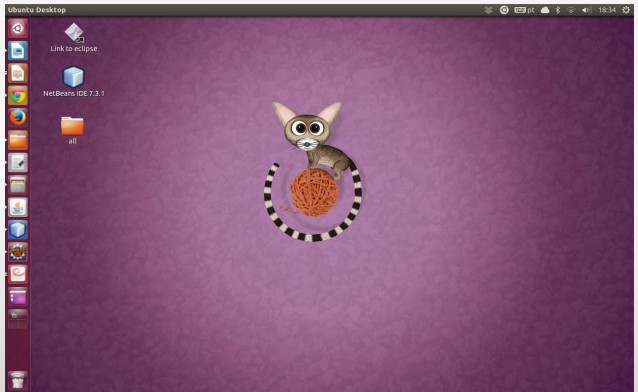

Imperativo
Acionado por Eventos
Tempo t
Fontes de eventos
Fluxo de entrada
passo 1
passo 2
passo n
halt
Resultados
while (true) {
}
manipula evento
fila de eventos
Variáveis de estado
Resultados

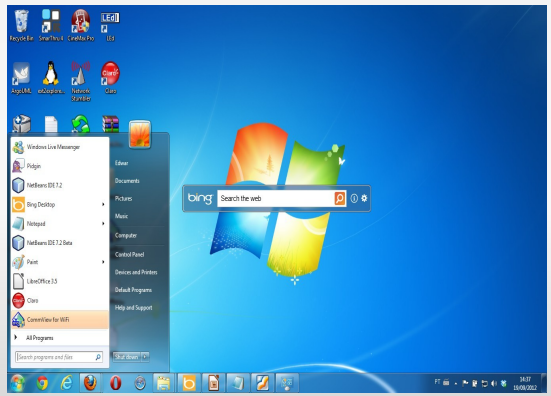
Programas acionados por eventos

- Um programa acionado por eventos tipicamente **não tem um ponto de parada predeterminado**, como chegar ao fim do arquivo lendo os dados.
- A entrada de um programa acionado por eventos vem de **fontes de eventos** autônomas distintas que podem ser sensores de um robô ou botões em uma aplicação interativa.
- Esses eventos ocorrem de forma assíncrona e cada um deles entra em uma fila de eventos para serem tratados
- A medida que o tempo passa, um laço de controle recupera o próximo evento dessa fila e o manipula.

Programas acionados por eventos

- Resumindo, os programas orientados a eventos são aqueles que quando começam, só ficam na espera de ações do usuário (eventos) para tratá-los adequadamente.
- **Exemplos:**
  - Firefox, Google-chrome
  - Editores de texto como ***Word*** ou ***LibreOffice Writer***.
  - Folhas de cálculo como **Excel** ou ***LibreOffice Calc***.





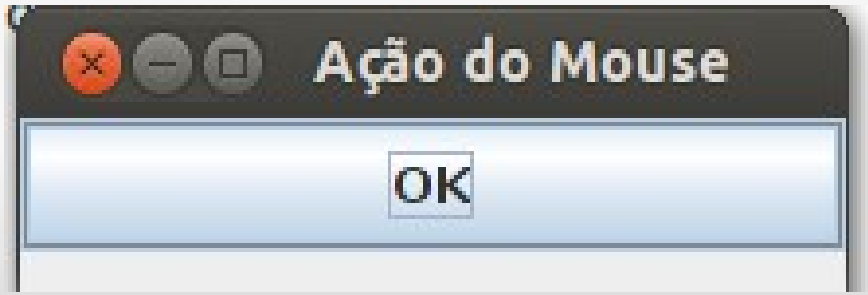
Conceitos Básicos

Eventos

- São as ações que o usuário realiza sobre o programa.
- Exemplos de eventos típicos são
  - Clique do mouse sobre um botão
  - Escolher uma opção de um menu
  - Escrever uma caixa de texto
  - Mover o mouse
- Geralmente cada vez que se produze um evento se cria um objeto.
- Em java os tipos de **eventos** que podem ocorrer são definidos pelas subclasses da classe java.Awt.Event.

Fontes de Eventos (disparadores de eventos)

- São aqueles componentes que accionam os eventos (ex em java: JButton, JLabel, JTextField).
- Em java os objetos que podem ser fontes de eventos são subclasses (ou filhos) da classe JComponent.
- Cada **fonte de evento** em uma interação com o usuário **gera** um objeto **evento**.
- Exemplo, um botão pode ser a fonte ou o disparador de um evento e em java poderia gerar um objeto evento da classe ActionEvent.



Ouvidores (Listeners)

- Servem para reconhecer (ouvir) a ocorrências de eventos particulares acontecendo sobre fontes de eventos.
- Para que um programa possa realizar alguma ação quando um evento foi disparado, primeiro deve detetar o evento com “ouvidores” (listeners).
- Por exemplo, em java para equipar um botão de forma que o programa possa detectar uma ocorrência da seleção daquele botão, o botão precisa chamar seu método ***addActionListener***. Se isso não for feito, os eventos do botão não serão detectados pelo programa.

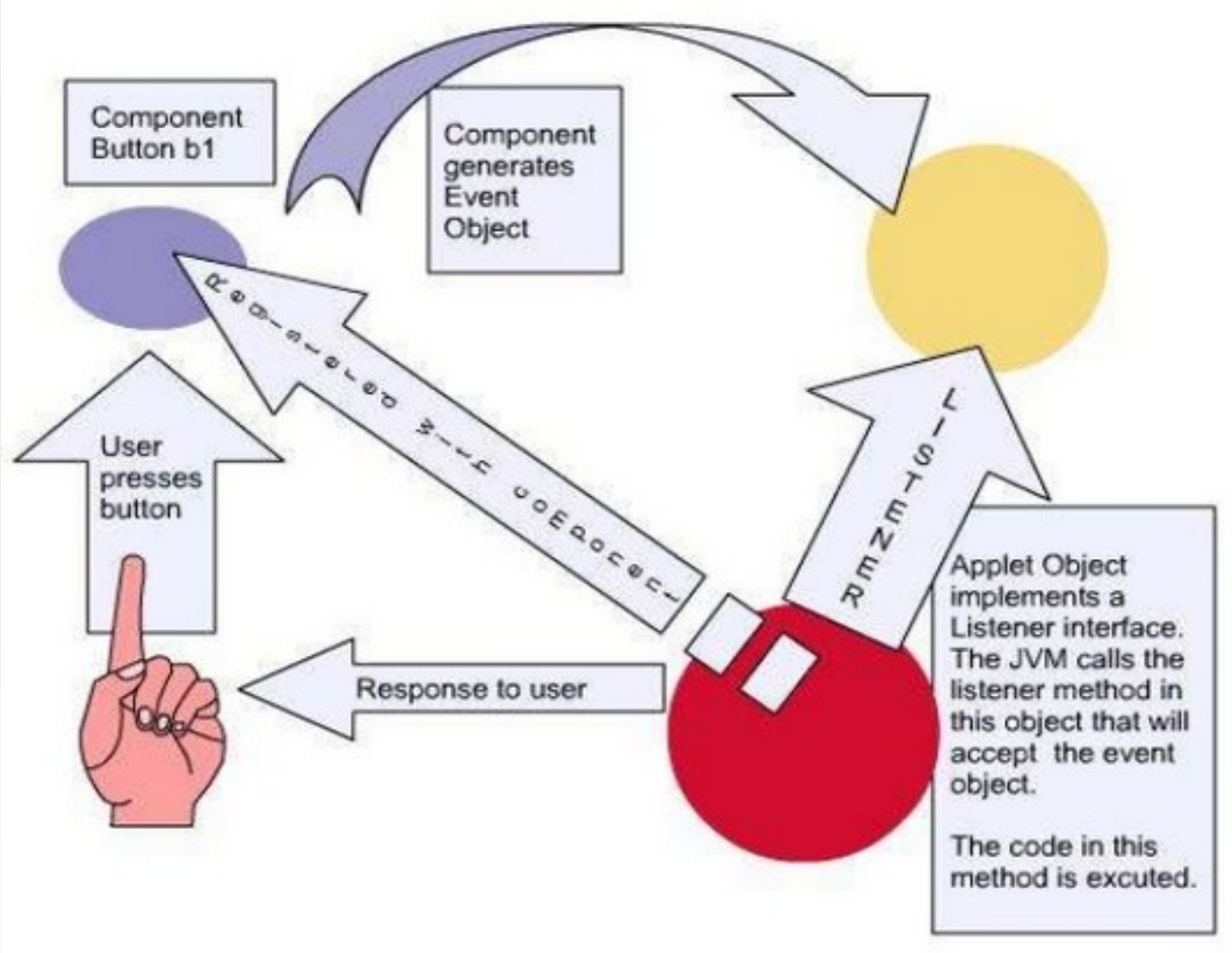
Métodos manipuladores (Handlers)

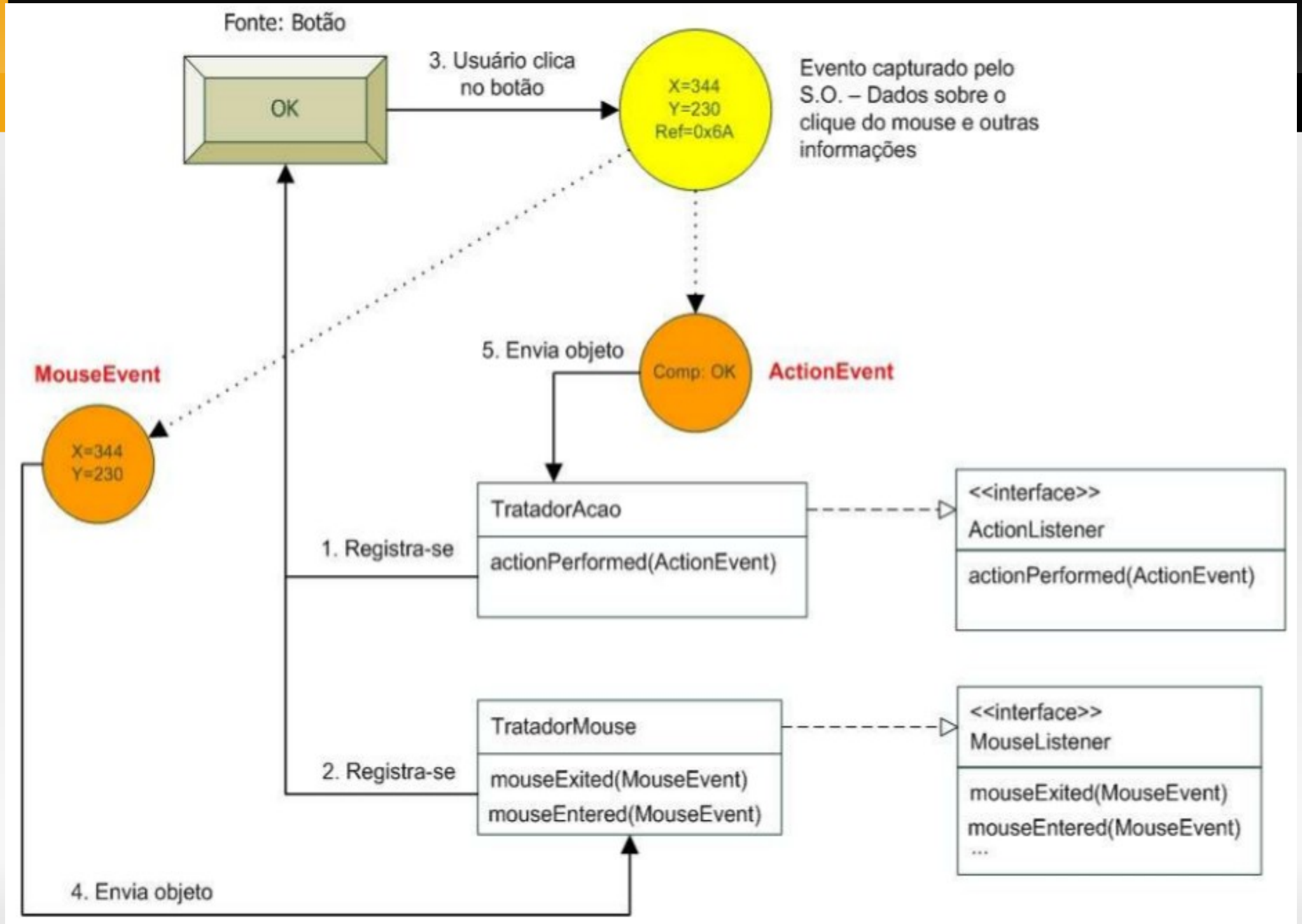
- Para responder aos eventos detectados pelos listeners precisamos implementar métodos especiais chamados de handlers (manipuladores).
- Em java implementamos os handlers dentro das classes listeners.

Alguns eventos, listeners e operações em Java

Eventos

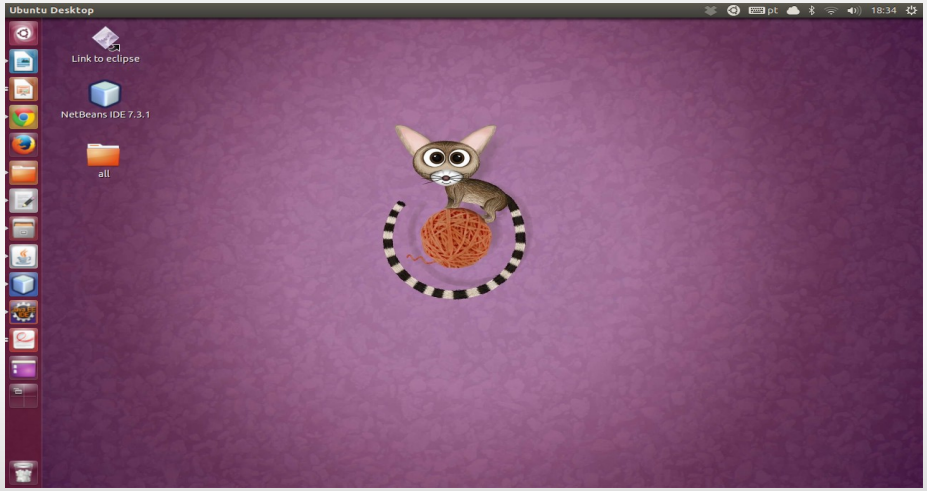
ActionEvent	**ActionListener**	**actionPerformed(ActionEvent)**
ItemEvent	**ItemListener**	**itemStateChanged(ItemEvent)**
KeyEvent	**KeyListener**	**keyPressed(KeyEvent)** **keyReleased(KeyEvent)** **keyTyped(KeyEvent)**
MouseEvent	**MouseListener**	**mouseClicked(MouseEvent)** **mouseEntered(MouseEvent)** **mouseExited(MouseEvent)** **mousePressed(MouseEvent)** **mouseReleased(MouseEvent)**
	MouseMotionListener	**mouseDragged(MouseEvent)** **mouseMoved(MouseEvent)**
TextEvent	**TextListener**	**textValueChanged(TextEvent)**
WindowEvent	**WindowListener**	**windowActivated(WindowEvent)** **windowClosed(WindowEvent)** **windowClosing(WindowEvent)** **windowDeactivated(WindowEvent)** **windowDeiconified(WindowEvent)** **windowIconified(WindowEvent)** **windowOpened(WindowEvent)**


Component
Button b1
Component
generates
Event
Object
User
presses
button
Response to user
Applet Object
implements a
Listener interface.
The JVM calls the
listener method in
this object that will
accept the event
object.
The code in this
method is excuted.


Fonte: Botão
OK
3. Usuário clica no botão
X=344
Y=230
Ref=0x6A
Evento capturado pelo S.O. – Dados sobre o clique do mouse e outras informações
5. Envia objeto
Comp: OK
ActionEvent
MouseEvent
X=344
Y=230
TratadorAcao
actionPerformed(ActionEvent)
1. Registra-se
<<interface>>
ActionListener
actionPerformed(ActionEvent)
TratadorMouse
mouseExited(MouseEvent)
mouseEntered(MouseEvent)
2. Registra-se
<<interface>>
MouseListener
mouseExited(MouseEvent)
mouseEntered(MouseEvent)
...
4. Envia objeto

Interfaces Gráficas de usuário (GUI)

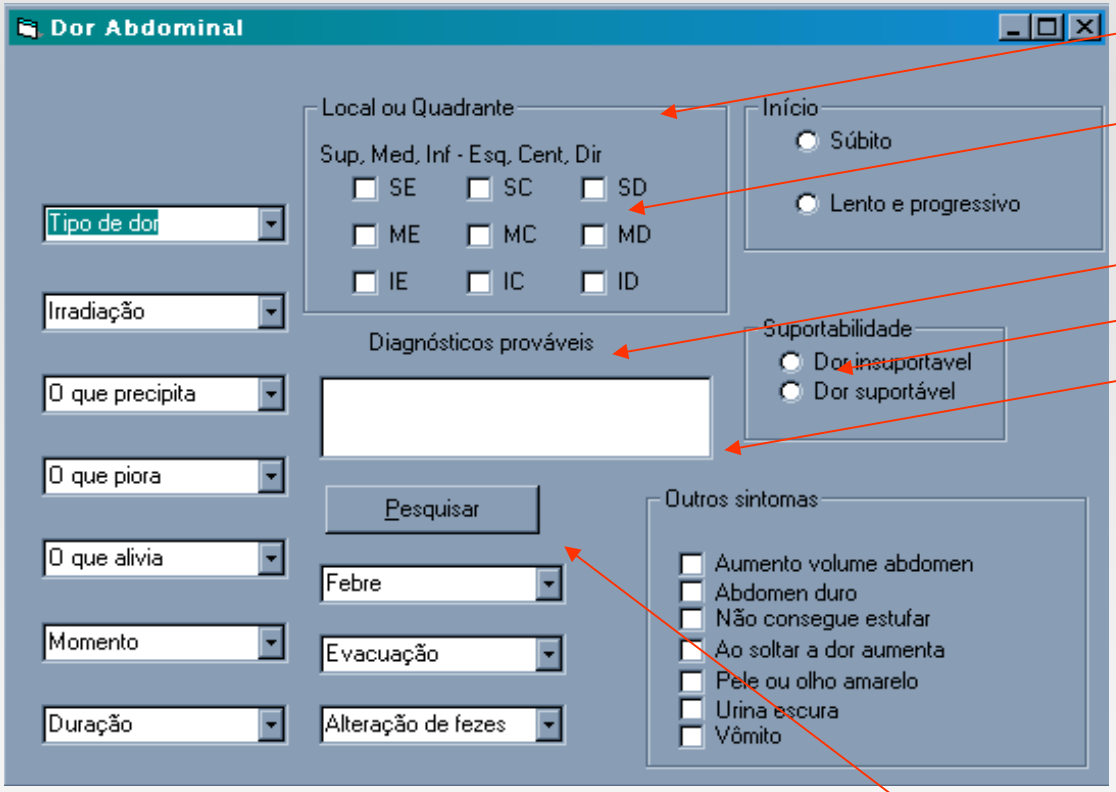
- Também chamadas de GUI (Graphical User Interface)
- Uma aplicação baseada em uma interface gráfica de usuário é um programa que roda dentro da sua própria janela e comunica-se com os usuários utilizando elementos gráficos como botões e menus.
- São projetadas para **reagirem a eventos** em vez de iniciá-los.



Componentes Visuais

- Botões,
- Rótulos,
- Caixas de Texto,
- Caixas de Seleção,
- entre outros.

Componentes Visuais



Painel

Caixa de seleção

Rótulo

Botão de Opção

Caixa de texto

Caixa combinada

Botão de comando

Eventos que podem ser disparados por componentes visuais

Alguns dos muitos eventos que podem ser disparados por componentes visuais são:

- Eventos do Mouse
  - Clique (Click)
  - Movimento (MouseHover, MouseLeave, MouseMove)
- Eventos do Teclado
  - Pressionamento de teclas (Down, Press, Up)
- Eventos da Janela
  - Alteração de tamanho (Resize)

Propriedades dos Componentes Visuais

- São características dos componentes visuais.
- As principais propriedades dos componentes visuais são:
  - Altura (Height): : junto com a largura define a dimensão do componente.
  - Largura (Width): junto com a altura define a dimensão do componente
  - Cor (Color, Backgroundcolor)
  - Fonte (Font)
  - Image (Image)
  - Nome (Name): identificador do componente dentro do programa
- Veja que nem sempre todos os componentes visuais possuem todas as características mencionadas acima.
- Debe-se diferenciar os **componentes visuais** (fontes de eventos) dos **eventos** que esses componentes podem disparar.

Qual é a alternativa que contém uma propriedade que não é válida para um componente visual.

a. Altura
b. Cor
c. Fonte
d. Largura
e. Clique

Vantagens do Paradigma Orientado a Eventos

- Flexibilidade
- Adequação para as interfaces gráficas
- Simplicidade de programação
- Facilidade de desenvolvimento

Flexibilidade

- É um dos tipos de programação mais flexível pois permite ao programador projetar visualmente a forma de acordo com as suas necessidades
- Programam-se objetos sobre o ambiente de programação como uma enorme gama de eventos que podem fazer coisas diferentes quando forem executados
- O programador tem mais controle sobre o que eles querem que o programa faça quando o usuário faz alguma coisa

Adequação para as interfaces gráficas

- O programador pode selecionar diferentes controles e coloca-los em formularios por exemplo.
- Permite que o usuário utilize o programa **sem existir uma sequencia** especifica **certa**, pois tem o controle completo sobre as ações que quer que o programa realize.
- Com pouco código geralmente, um componente (botão por exemplo) pode executar uma determinada ação quando um evento escolhido é percebido pelo programa.

- **Simplicidade de programação**

  Como muitas vezes o programador pode criar visualmente o seu programa, existe uma economia de tempo e esforço (O aspecto visual simplifica as tarefas mais complexas.)

- **Facilidade de desenvolvimento**

  O desenvolvimento orientado a eventos fica facilitado dado que o programador tem que lidar com um controle de cada vez (controles de um formulário são trabalhados de forma independente, apesar de as vezes trabalhar juntos).

Linguagens

- Algumas Linguagens típicas desse paradigma são:
- Delphi
- Visual Basic
- Java

Delphi

- Linguagem orientada a eventos e a objetos.
- Os dados são manipulados pelos objetos que são manipulados por ferramentas presentes no próprio programa.

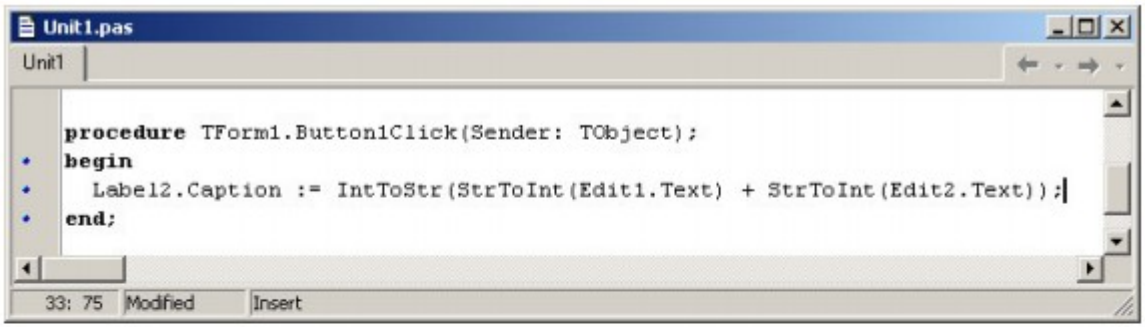
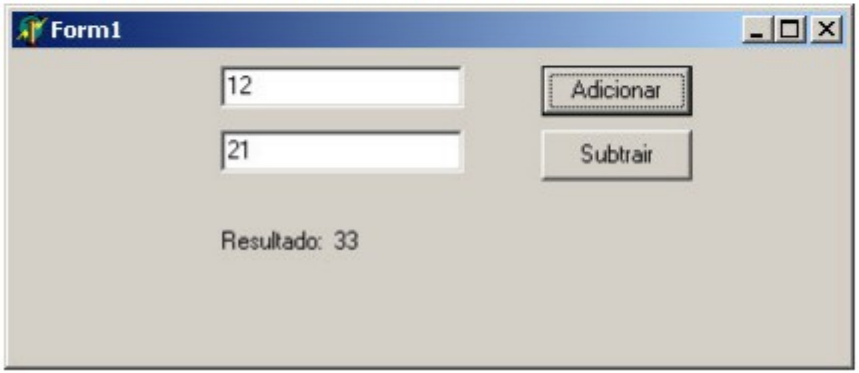


Figura 5. Código digitado no evento do botão.

Tendo como resultado na execução do aplicativo a seguinte imagem:



Java

- Java suporta programação orientada a eventos fornecendo certas classes e alguns métodos que podem ser usados para projetar tal interação.
- Java tem conseguido adaptar o enfoque orientado a **eventos** ao paradigma orientado a **objetos**.
- Java classifica os **eventos** que podem ocorrer ocasionados por **fontes de eventos**, reconhece a ocorrências daqueles eventos com ouvidores (**listeners**) e depois responde a cada evento quando ele ocorrer usando métodos manipuladores (**handlers**) implementados dentro das classes Listeners.
- Exemplos externos

Conclusões

- Constitui um paradigma que tem adquirido uma importância maior devido ao crescimento da demanda de aplicações pela Internet.
- Tem estreita ligação com a programação orientada ao objeto pois geralmente a programação por eventos necessita de objetos para tratar os eventos.

Referências Bibliográficas

O material para a realização desta apresentação foi extraido de:

* Tucker, Linguagens de Programação, principios e paradigmas.

* Principios de Linguagens de programação, Ana Cristina Vieira de Melo, Flavio Soares