

A partir de uma interface gráfica pronta, a aplicação fica à espera das ações do usuário, que irá gerar um **evento**. Quando um evento acontece, o programa responde executando um método que está associado à este evento (***event-handling method***).

Veja alguns tipos comuns de eventos:

- eventos de container (inserção ou remoção de componente)
- eventos de foco (mouse entrou ou saiu de um componente)
- eventos de entrada: teclado e mouse
- eventos de janela: open, close, resize, minimize, etc
- eventos de ação: notificam a ação de um componente específico (ex: clique em um botão)
- eventos de ajuste: movimento de um scrollbar, por exemplo
- eventos de item: seleção de um elemento em uma lista, checkbox, etc
- eventos de texto: alteração do texto em um JTextArea, JTextField, etc

COMO TRATAR OS EVENTOS

Em Java, os eventos são representados por objetos. Quando um evento ocorre, o sistema recolhe todas as informações relevantes para o evento e constrói um objeto para conter essa informação.

Diferentes tipos de eventos são representados por objetos pertencentes a classes diferentes. Por exemplo, quando o usuário pressiona um dos botões de um mouse, um objeto pertencente a uma classe chamada ***MouseEvent*** é construído.

Exemplo: ***MouseListener*** - interface para eventos de mouse

- **`mouseClicked(MouseEvent e)`** - chamado quando o botão do mouse é clicado (e solto) sobre um componente;
- **`mousePressed(MouseEvent e)`** - chamado quando o botão do mouse é clicado sobre um componente;
- **`mouseReleased(MouseEvent e)`** - chamado quando o botão do mouse é solto sobre um componente;
- **`mouseEntered(MouseEvent e)`** - chamado quando o mouse “entra” na área de um componente;
- **`mouseExited(MouseEvent e)`** - chamado quando o mouse deixa a área de um componente;

O objeto contém informação, tais como a fonte do evento (qual foi o componente que o usuário clicou), as coordenadas (x, y) do ponto no componente onde ocorreu o clique, e qual botão do mouse foi pressionado. Quando o usuário pressiona uma tecla do teclado, um **KeyEvent** é criado. Após o objeto de evento ser construído, ele é passado como um parâmetro para a sub-rotina específica. Nesta sub-rotina, o programador diz quais são as instruções que devem ser executadas quando o evento ocorrer.

Sendo assim, este é o *looping* que acontece internamente (e automaticamente) em uma aplicação GUI:

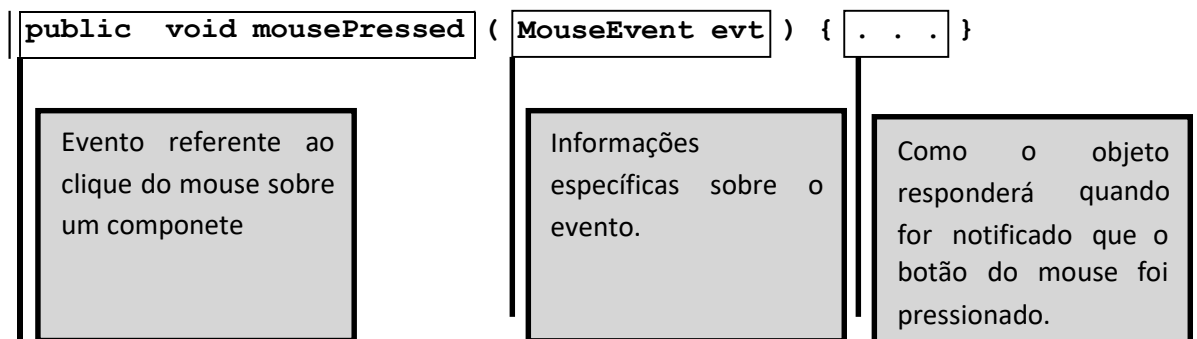
```
Enquanto o programa ainda estiver em execução:
    Esperar a ocorrência do próximo evento
    Chamar a sub-rotina para manipular este evento
```

Se uma aplicação está interessada em um evento específico (por exemplo, clique em um botão), deve solicitar ao sistema para “escutar” o evento. Para que um componente ou container possa “escutar” eventos, é preciso implementar um **listener**.

Um **listener** é um objeto que inclui um ou mais métodos de manipulação de eventos e que tem a responsabilidade de responder a um evento.

Os **listeners** são implementados através de **interfaces**. Uma interface define um conjunto de métodos que uma classe deve implementar mas não define como esses métodos devem ser implementados. Uma interface serve para implementar funcionalidades diversas: o conjunto de funcionalidades que um dispositivo oferece ao mundo exterior (seja ela uma classe, um componente ou subsistema). **ESTE ASSUNTO SERÁ ESTUDADO DEPOIS EM DETALHES. NO MOMENTO USAREMOS UMA INTERFACE SEM APLICAR SEUS CONCEITOS PROPRIAMENTE DITOS.**

Por exemplo, se um objeto é construído para servir de ouvinte para eventos do mouse (MouseEvent), então ele deve conter o seguinte método (entre vários outros):



Etapas a serem seguidas:

1. Importar a classe responsável pela manipulação de eventos:

`import java.awt.event.*;`
2. Declarar que uma classe implementa a interface “de escuta” (**listener**) adequado. Por exemplo: **MouseListener**.
3. Fornecer as definições dessa classe para as sub-rotinas da interface.
4. Associar o objeto **listener** ao componente que irá gerar os eventos, chamando, por exemplo, o método **addMouseListener()** no componente.

```
import java.awt.Component;  
import java.awt.event.*;
```

1

```
public class RepaintOnClick implements MouseListener {  
  
    public void mousePressed(MouseEvent evt) {  
        Component source = (Component)evt.getSource();  
        source.repaint();  
    }  
  
    public void mouseClicked(MouseEvent evt) { }  
  
    public void mouseReleased(MouseEvent evt) { }  
  
    public void mouseEntered(MouseEvent evt) { }  
  
    public void mouseExited(MouseEvent evt) { }  
  
}
```

3

2

```
RepaintOnClick listener = new RepaintOnClick();  
panel.addMouseListener(listener);
```

4

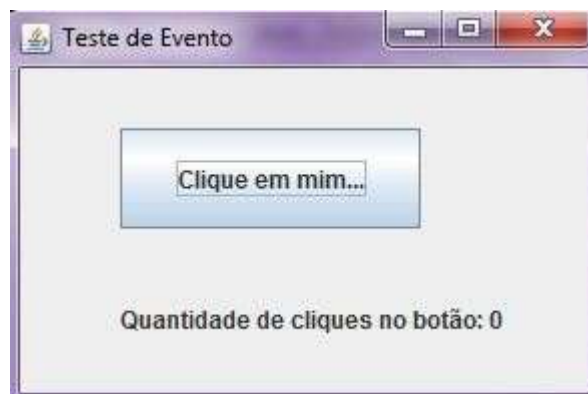
EXEMPLO DE EVENTOS:

- Usuário clica em um botão: **ActionListener**
- Usuário fecha um frame: **WindowListener**
- Usuário pressiona um botão do mouse: **MouseListener**
- Usuário move o mouse: **MouseMotionListener**
- Componentes se tornam visíveis: **ComponentListener**

ActionEvents: tipo de evento mais simples e comum no Swing que representa uma ação qualquer ocorrendo em um componente da GUI. Criado por:

- cliques em botão
- mudanças em checkboxes
- cliques de menu
- digitar [Enter] em uma textbox

EXEMPLO 1 – Contando os cliques do mouse...



Nesta aplicação, a cada vez que o usuário clica sobre o botão, a mensagem é atualizada (contador de cliques).

```
import javax.swing.*;
import java.awt.event.*;

class Gui extends JFrame implements ActionListener {

    int cont=0;
    JButton but = new JButton("Clique em mim...");
    JLabel texto = new JLabel("Quantidade de cliques no botão: " + cont);

    public Gui () {
        setTitle("Teste de Evento");
        setResizable(true);
        setSize(300, 200);
        setLayout(null);
        but.setBounds(50, 30, 150, 50);
```

```

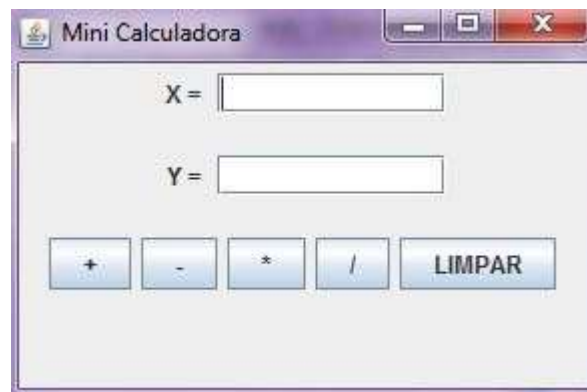
        add(but);
        but.addActionListener(this);
        texto.setBounds(50,100,250,50);
        add(texto);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        Gui janela = new Gui();
        janela.setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        cont++;
        texto.setText("Quantidade de cliques no botão: " + cont);
    }
}

```

EXEMPLO 2 – Calculadora simples...



```

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

class Gui extends JFrame implements ActionListener {

    JTextField xInput;
    JTextField yInput;
    JLabel answer;
    JButton bt1;
    JButton bt2;
    JButton bt3;
    JButton bt4;
    JButton bt5;
    JPanel respPanel;

    public Gui () {

```

```

setTitle("Mini Calculadora");
setResizable(true);
setSize(300, 200);
setLayout(new GridLayout(4,1,3,3));
BorderFactory.createEmptyBorder(5,5,5,5);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

xInput = new JTextField(10);
JPanel xPanel = new JPanel();
xPanel.add(new JLabel(" X = "));
xPanel.add(xInput);
add(xPanel);

yInput = new JTextField(10);
JPanel yPanel = new JPanel();
yPanel.add(new JLabel(" Y = "));
yPanel.add(yInput);
add(yPanel);

bt1 = new JButton("+");
bt2 = new JButton("-");
bt3 = new JButton("*");
bt4 = new JButton("/");
bt5 = new JButton("LIMPAR");
JPanel btPanel = new JPanel();
btPanel.setLayout(new FlowLayout());
btPanel.add(bt1);
btPanel.add(bt2);
btPanel.add(bt3);
btPanel.add(bt4);
btPanel.add(bt5);
add(btPanel);

bt1.addActionListener(this);
bt2.addActionListener(this);
bt3.addActionListener(this);
bt4.addActionListener(this);
bt5.addActionListener(this);

answer = new JLabel();
respPanel = new JPanel();
respPanel.add(answer);
add(respPanel);
}

public void actionPerformed(ActionEvent evt) {
    double x=0;
    double y=0;
    double resp=0;
    flag=0; // Sinalizador para erro na digitação (0=OK / -1=erro)

    try {

```

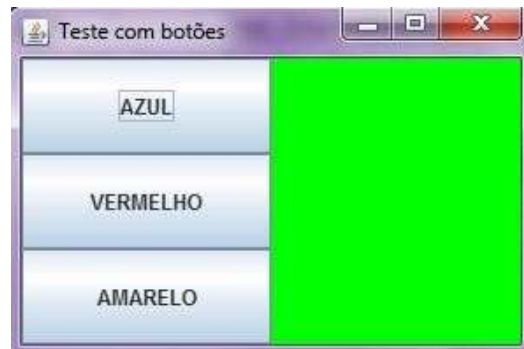
```

        String xStr = xInput.getText();
        x = Double.parseDouble(xStr);
    }
    catch (NumberFormatException e) {
        // Valor digitado não é numérico
        answer.setText("Valor Ilegal para X.");
        xInput.requestFocus();
        flag = -1;
    }
    try {
        String yStr = yInput.getText();
        y = Double.parseDouble(yStr);
    }
    catch (NumberFormatException e) {
        // Valor digitado não é numérico
        answer.setText("Valor Ilegal para Y.");
        yInput.requestFocus();
        flag = -1;
    }
    if (flag == 0){        // Senão houve erro na digitação
        String op = evt.getActionCommand();
        if (op.equals("+")){
            resp = x+y;
            answer.setText("RESPOSTA = "+ String.valueOf(resp));
        }
        else if (op.equals("-")){
            resp = x-y;
            answer.setText("RESPOSTA = "+ String.valueOf(resp));
        }
        else if (op.equals("*")) {
            resp = x*y;
            answer.setText("RESPOSTA = "+ String.valueOf(resp));
        }
        else if (op.equals("/"))
            if (y == 0)
                answer.setText("IMPOSSÍVEL DIVISÃO POR ZERO");
            else {
                resp = x/y;
                answer.setText("RESPOSTA = "+ String.valueOf(resp));
            }
            else {
                xInput.setText(" ");
                yInput.setText(" ");
                answer.setText(" ");
                xInput.requestFocus();
            }
        }
    }

    public static void main(String[] args) {
        Gui janela = new Gui();
        janela.setVisible(true);
    }
}

```

EXEMPLO 3 – ESCOLHA UMA COR



Nesta aplicação, dependendo do botão que o usuário clicar, a cor do lado direito se altera.

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

class Gui extends JFrame implements ActionListener {
    JButton bt1;
    JButton bt2;
    JButton bt3;
    JPanel dir;

    public Gui () {
        setTitle("Teste com botões");
        setResizable(true);
        setSize(300, 200);
        setLayout(new GridLayout(1,2));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel esq = new JPanel();
        add(esq);

        dir = new JPanel();
        dir.setBackground(Color.GREEN);
        add(dir);

        bt1 = new JButton("AZUL");
        bt2 = new JButton("VERMELHO");
        bt3 = new JButton("AMARELO");

        esq.setLayout(new GridLayout(3,1));
        esq.add(bt1);
        esq.add(bt2);
        esq.add(bt3);

        bt1.addActionListener(this);
        bt2.addActionListener(this);
        bt3.addActionListener(this);
    }
}
```



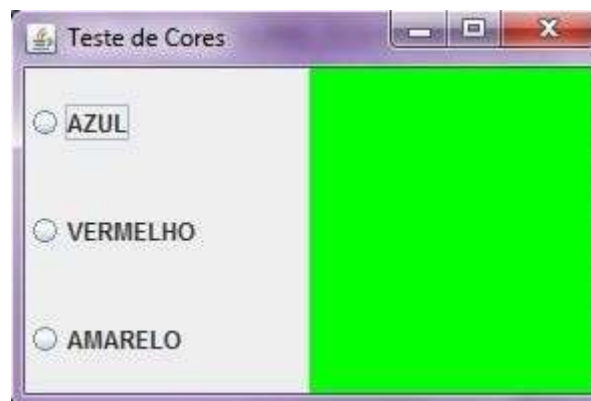
```

    public void actionPerformed(ActionEvent evt) {
        if (evt.getSource()==bt1)
            dir.setBackground(Color.BLUE);
        else
            if (evt.getSource()==bt2)
                dir.setBackground(Color.RED);
            else
                if (evt.getSource()==bt3)
                    dir.setBackground(Color.YELLOW);
    }

    public static void main(String[] args) {
        Gui janela = new Gui();
        janela.setVisible(true);
    }
}

```

EXEMPLO 4 – ESCOLHA UMA COR (Versão 2)



```

import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

class Gui extends JFrame implements ActionListener {
    JRadioButton btAzul;
    JRadioButton btVermelho;
    JRadioButton btAmarelo;
    JPanel dir;

    public Gui () {
        setTitle("Teste de Cores");
        setResizable(true);
        setSize(300, 200);
        setLayout(new GridLayout(1,2));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel esq = new JPanel();
        add(esq);
    }
}

```

```

    dir = new JPanel();
    dir.setBackground(Color.GREEN);
    add(dir);

    btAzul      = new JRadioButton("AZUL" , false);
    btVermelho  = new JRadioButton("VERMELHO" , false);
    btAmarelo   = new JRadioButton("AMARELO", false);
    ButtonGroup bgroup = new ButtonGroup();
    bgroup.add(btAzul);
    bgroup.add(btVermelho);
    bgroup.add(btAmarelo);

    esq.setLayout(new GridLayout(3,1));
    esq.add(btAzul);
    esq.add(btVermelho);
    esq.add(btAmarelo);

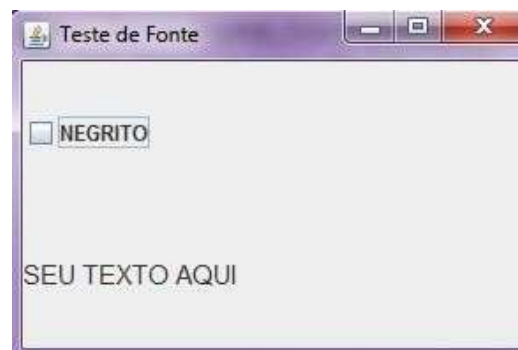
    btAzul.addActionListener(this);
    btVermelho.addActionListener(this);
    btAmarelo.addActionListener(this);
}

public void actionPerformed(ActionEvent evt) {
    if (btAzul.isSelected())
        dir.setBackground(Color.BLUE);
    else
        if (btVermelho.isSelected())
            dir.setBackground(Color.RED);
        else
            if (btAmarelo.isSelected())
                dir.setBackground(Color.YELLOW);
}

public static void main(String[] args) {
    Gui janela = new Gui();
    janela.setVisible(true);
}
}

```

EXEMPLO 5 – FORMATANDO UM TEXTO



Nesta aplicação, se o usuário clicar no CheckBox NEGRITO, a fonte muda automaticamente.

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

class Gui extends JFrame implements ActionListener {
    JCheckBox cb1;
    JLabel lb1;
    JPanel dir;

    public Gui () {
        setTitle("Teste de Fonte");
        setResizable(true);
        setSize(300, 200);
        setLayout(new GridLayout(1,2));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JPanel esq = new JPanel();
        add(esq);

        dir = new JPanel();
        add(dir);

        cb1 = new JCheckBox ("NEGRITO");
        cb1.addActionListener(this);

        esq.setLayout(new GridLayout(2,1));
        esq.add(cb1);

        lb1 = new JLabel("SEU TEXTO AQUI");
        lb1.setFont(new Font("Arial",Font.PLAIN,15));
        esq.add(lb1);
    }

    public void actionPerformed(ActionEvent evt) {
        if (cb1.isSelected())
            lb1.setFont(new Font("Arial",Font.BOLD,15));
        else
            lb1.setFont(new Font("Arial",Font.PLAIN,15));
    }

    public static void main(String[] args) {
        Gui janela = new Gui();
        janela.setVisible(true);
    }
}
```