Eventos

Roteiro

- Frames
- Formatting Output
- Event Handling
- Entering Data Using Fields in a Frame
- Creating a Data Entry Field
- Using a Field
- Reading Data in an Event Handler
- Handling Multiple Button Events

Using A Frame for Output

- **Import** classes and declare fields
- **Instantiate** frame objects and specify some properties
- **Put** display objects into the frame
- **Make** the frame visible on the screen

Import Classes & Declare Fields

- Import package containing the `JFrame` class

```
import javax.swing.*;
```

- Declare a variable of class `JFrame`

```
JFrame outFrame;
```

- Declare a variable of the class `Container`

```
Container outPane;
```

Instantiate Frame Object and Specify Properties

- Instantiate a `JFrame` object

```
outFrame = new JFrame();
```

- Get a content pane object and assign its address to `Container` variable

```
outPane =
   outFrame.getContentPane();
```

- Specify action on window closing

```
outFrame.setDefaultCloseOperation(
  JFrame.EXIT_ON_CLOSE);
```

- Specify size of `JFrame` object in terms of pixels (the individual dots that make up an image on the screen)

```
outFrame.setSize(300,200);
```

- Specify layout manager

```
outPane.setLayout(new FlowLayout());
```

- Layout manager A class that manages the placement of display elements within a content pane

Add Output to Content Pane

- Container class A class into which you can add other elements
- A content pane is a container class
- Add a label to content pane

```
outPane.add(new JLabel("Hi"));
```

Make Frame Visible

```
outFrame.setVisible(true);
```

What happens if you call `setVisible` with `false` as the argument?

Formatting output

- `GridLayout` is another layout manager
- `GirdLayout` allows you to specify how the objects are arranged in the pane

```
outPane.setLayout(new GridLayout(5,2));
```

Aligning of Text

```
dataPane.add(new Label("Name", JLabel.LEFT);
dataPane.add(new Label("Date", JLabel.CENTER);
dataPane.add(new Label("Time", JLabel.RIGHT);
```

What are `LEFT`, `RIGHT`, and `CENTER`?

Event-Driven Programming

- **Event-driven programming** The user's interaction with a GUI component is an event that can be processed by the program
- A GUI component (such as a button) is told where the **event-handling method** is defined that will be called when an event involving that component occurs
- Components are told by a method such as `addActionListener` that takes the method name as an argument

Event Definitions

- **Event** An action, such as a mouse click, that takes place asynchronously with respect to the execution of the application
- **Event handling** The process of responding to events
- **Event listener** An object that contains event handler methods
- **Event source** An object that generates an event

Event Definitions Continued

- **Event handler** A method that responds to an event; part of the event listener that is invoked by the event source object
- **Firing an event** An event source generates an event
- **Registering a listener** Adding an event listener to an event source's list of interested listeners
- **Button** A component that can be added to a frame that fires an event when the user clicks it

Processing an Action Event

- `Button` objects generate "action events" that can be processed by any `ActionListener` object
- Any class that implements `interface` `ActionListener` must provide a definition of method `actionPerformed`
- Once an `ActionListener` object is registered to "handle" a button's action event, its method `actionPerformed` is called automatically whenever that button's action event occurs

Delegation Event Model

- Means the use of event listeners in event handling. The handling of an event is delegated to a particular object in the program
- When there are several buttons in an application, they can all have the same event handler, or they can have different event handlers
- If several buttons have the same event handler, you can use a *selection* statement in method `actionPerformed` to determine which button fired an event

Button, Button, Who has the Button...

- **Declare a variable of the JButton class**

```
JButton done;
```

- **Instantiate a JButton object and assign its address to the variable**

```
done = new JButton("Enter");
// Word "Enter" appears on the button
```

- **Add the object to the frame's content pane**

```
outPane.add(done);
```

Button Event Listeners

- **A listener is a class that implements the `ActionListener` interface**

```
private static
  class Handler implements ActionListener
```

- **The class must contain a method with this heading**

```
public void actionPerformed(ActionEvent event)
{ // code to handle the event }
```

only thing that can change is the parameter name

Registering the Listener

- Registering the Listener Letting the event source (the button) know that the code to handle the event is in the listener
- Declare a variable of the listener class

```
Handler buttonHandler;
```

- Instantiate an object of the listener class

```
buttonHandler = new Handler();
```

- Register the listener with the button

```
done.addActionListener(buttonHandler);
```

Tasks of a Listener

- Listeners are invoked automatically when the event for which they are listening occurs
- A button is pressed and the listener for the button jumps into action
- Possible tasks completed in the listener
  - input data
  - update data
  - report on some internal action

Entering Data

- **Field** A component of a frame in which the user can type a value; the user must first place the cursor in the field by clicking inside the field
- **Dialog** Technique where the user enters data in a field and lets the program know the data is ready to be read by a separate action such as clicking a button

Creating a Data Entry Field

- Declare a variable of the appropriate field class

```
JTextField inField;
```

- Instantiate an object of the class

```
inField = new JTextField(6);
```

or

```
inField = new JTextField("Hello!", 6)
```

- Add the objet to the content pane

```
outPane.add(inField);
```

Reading in a Listener

```
// When the Enter button is pressed this listener
// jumps into action
private static class Handler implements ActionListener
{
  public void actionPerformed(ActionEvent anEvent)
  {
    String inLine;
    inLine = inField.getText();
    System.out.println(inLine);
  }
}
```

What Enter button? Where is `inField` declared?

Organization of Event Application

```
public class CopyString
{
  public static class Handler …
  {
    public void actionPerformed(…){}
  }
  private static JTextField inField;
  public static void main(String[] args)
  {
    JButton done;...
  }
}
```

application class

listener class

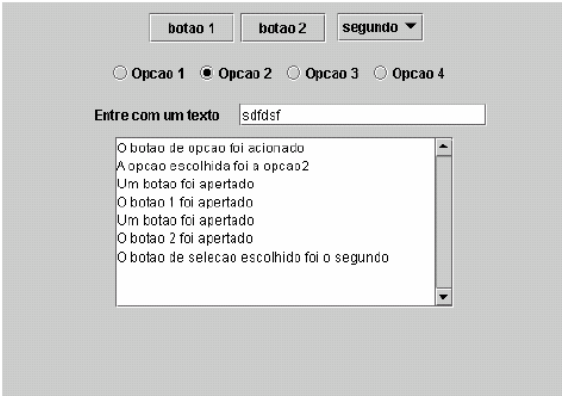
handler method

declare components used in Handler

declare and instantiate other components

Exemplo 1 - ActionListner

- Prof. Aruquia
- Entradas de dados
  - Botoes
  - Check-box
  - Radio




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

public class AppletActionListener extends JApplet implements ActionListener {

   JButton botao1, botao2;
   JPanel grupoBotao12;
   JComboBox botaoDeSelecao;
   String[] botaoDeSelecaoString={"primeiro","segundo","terceiro"};
   JRadioButton opcao1, opcao2, opcao3, opcao4;
   ButtonGroup grupoBotao;
   JPanel grupoBotaoPanel;
   JLabel titulo;
   JTextField campoDeTexto;
   JTextArea areaDeTexto;
   JPanel grupoTexto;
   Panel p;
   Container cp;
```

```
public void init() {
    //botoes
    botao1 = new JButton("botao 1");
    botao1.addActionListener(this);
    botao2 = new JButton("botao 2");
    botao2.addActionListener(this);
    grupoBotao12 = new JPanel();
    grupoBotao12.add(botao1);
    grupoBotao12.add(botao2);
    //botao de selecao
    botaoDeSelecao = new
JComboBox(botaoDeSelecaoString);
    botaoDeSelecao.addActionListener(this);
```

```
//botao de opcao
    grupoBotao = new ButtonGroup();
    grupoBotaoPanel = new JPanel();
    opcao1 = new JRadioButton("Opcao 1");
    opcao2 = new JRadioButton("Opcao 2");
    opcao3 = new JRadioButton("Opcao 3");
    opcao4 = new JRadioButton("Opcao 4");
    grupoBotao.add(opcao1);
    grupoBotao.add(opcao2);
    grupoBotao.add(opcao3);
    grupoBotao.add(opcao4);
    opcao1.addActionListener(this);
    opcao2.addActionListener(this);
    opcao3.addActionListener(this);
    opcao4.addActionListener(this);
    grupoBotaoPanel.add(opcao1);
    grupoBotaoPanel.add(opcao2);
    grupoBotaoPanel.add(opcao3);
    grupoBotaoPanel.add(opcao4);
```

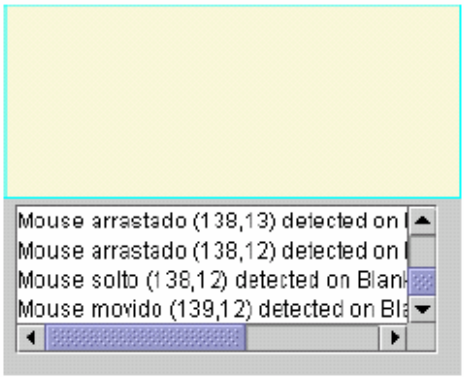
```
//Cria um painel agrupando um Label e um Campo de Texto
    titulo = new JLabel("    Entre com um texto    ");
    campoDeTexto = new JTextField(20);
    grupoTexto = new JPanel();
    campoDeTexto.addActionListener(this);
    grupoTexto.add(titulo);
    grupoTexto.add(campoDeTexto);
    //Content Pane
    cp = getContentPane();
    p = new Panel ();
    areaDeTexto = new JTextArea();
    areaDeTexto.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(areaDeTexto,
JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,     JScrollPane.HORIZONTAL_S
CROLLBAR_AS_NEEDED);
    scrollPane.setPreferredSize(new Dimension(300, 150));
    p.add(grupoBotao12);
    p.add(botaoDeSelecao);
    p.add(grupoBotaoPanel);
    p.add(grupoTexto);
    p.add(scrollPane);
    cp.add(p);
  }
```

```
//Este metodo responde as acoes dos componentes
  public void actionPerformed(ActionEvent e) {

    if(e.getSource() instanceof JButton) {
      areaDeTexto.append("Um botao foi apertado \n");
            if(e.getSource()==botao1) {
              areaDeTexto.append("O botao 1 foi apertado \n");
            }else {
              areaDeTexto.append("O botao 2 foi apertado \n");
            }
      }
    if(e.getSource() instanceof JTextField) {
      areaDeTexto.append("O texto digitado foi " + campoDeTexto.getText() + " \n");
      }
  }

  class Panel extends JPanel {

    public void paintComponent (java.awt.Graphics g) {
          super.paintComponent(g);
    }
  }

}
```

```
if(e.getSource() instanceof JComboBox) {
        areaDeTexto.append("O botao de selecao escolhido foi o " +
botaoDeSelecao.getSelectedItem() + " \n");
      }
       if(e.getSource() instanceof JRadioButton) {
      if(opcao1.isSelected()) {
              areaDeTexto.append("O botao de opcao foi acionado \n");
              areaDeTexto.append("A opcao escolhida foi a opcao1 \n");
            }
            if(opcao2.isSelected()) {
        areaDeTexto.append("O botao de opcao foi acionado \n");
                areaDeTexto.append("A opcao escolhida foi a opcao2 \n");
      }
      if(opcao3.isSelected()) {
        areaDeTexto.append("O botao de opcao foi acionado \n");
                areaDeTexto.append("A opcao escolhida foi a opcao3 \n");
      }
            if(opcao4.isSelected()) {
        areaDeTexto.append("O botao de opcao foi acionado \n");
                areaDeTexto.append("A opcao escolhida foi a opcao4 \n");
      }
    }
```

Exemplo 2 (prof. Aruquia)

- **Prof. Aruquia**
- **Ações de mouse**
  - **Clic**
  - **Arrasto**
  - **Movimento**



```
import javax.swing.*;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Insets;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;

import java.awt.event.MouseMotionListener;
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;

public class MouseMotionEventDemoV2 extends JApplet
                        implements MouseMotionListener, MouseListener {
   BlankArea areaMouse;
   JTextArea areaTexto;

   public void init() {
      JPanel contentPane = new JPanel();

      GridBagLayout gridbag = new GridBagLayout();
      GridBagConstraints c = new GridBagConstraints();
      contentPane.setLayout(gridbag);
```

```
      //Define como redimensionar
            c.fill = GridBagConstraints.BOTH;
            c.gridwidth = GridBagConstraints.REMAINDER;
            c.weightx = 1.0;
            c.weighty = 1.0;

            //Espaco entre a areaMouse e as bordas do applet e da areaTexto
            c.insets = new Insets(0, 0, 0, 0);
            areaMouse = new BlankArea(new Color(0.98f, 0.97f, 0.85f));
            gridbag.setConstraints(areaMouse, c);
            contentPane.add(areaMouse);

            //Espaco entre a areaTexto e as bordas do applet e da areaMouse
            c.insets = new Insets(3, 6, 9, 12);
            areaTexto = new JTextArea();
            areaTexto.setEditable(false);
            JScrollPane scrollPane = new JScrollPane(areaTexto,
                  JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
                  JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
            scrollPane.setPreferredSize(new Dimension(200, 75));
            gridbag.setConstraints(scrollPane, c);
            contentPane.add(scrollPane);
```

```
        //Registro para eventos do mouse na areaMouse.
        areaMouse.addMouseMotionListener(this);
        areaMouse.addMouseListener(this);
        addMouseMotionListener(this);
        addMouseListener(this);

        setContentPane(contentPane);
    }

    //interface MouseListener
    public void mouseEntered(MouseEvent e) {
      escreve("Mouse entrou", e);
    }

    //interface MouseListener
    public void mouseExited(MouseEvent e) {
      escreve("Mouse saiu", e);
    }

    //interface MouseListener
    public void mouseClicked(MouseEvent e) {
      escreve("Mouse clicado", e);
    }
```

```
//interface MouseListener
  public void mousePressed(MouseEvent e) {
    escreve("Mouse pressinado", e);
  }
  //interface MouseListener
  public void mouseReleased(MouseEvent e) {
    escreve("Mouse solto", e);
  }
 //interface MouseMotionListener
  public void mouseDragged(MouseEvent e) {
    escreve("Mouse arrastado", e);
  }
  //interface MouseMotionListener
  public void mouseMoved(MouseEvent e) {
    escreve("Mouse movido", e);
  }

  void escreve(String eventDescription, MouseEvent e) {
     areaTexto.append(eventDescription
                + " (" + e.getX() + "," + e.getY() + ")"
                + " detected on "
                + e.getComponent().getClass().getName()
                + "\n");
  }
}
```

What happens next?

- After a value has been input from a `JTextField`, the application waits for another value to be input and waits for another value to be input …
- Creating an event loop
- The application is halted
  - by closing the window
  - adding another button to signal the end

Review of GUI Components

`JFrame` A kind of window in which components can be placed

`Container` A pane in the frame into which objects can be placed

`JLabel` A component where text can be displayed

`JButton` A component that generates an event when the user clicks on it with the mouse

`JTextField` A component in which the user can type a value; the user must first place the cursor in the field by clicking inside the field

Graphical User Interfaces

- GUIs are built from GUI components (also called widgets for window gadgets)
- GUI component classes are part of java.awt (Abstract Windowing Toolkit package) and javax.swing
- GUIs are event-driven; they generate events when the user interacts with the GUI
- An event is an action such as clicking the mouse, clicking a button, that takes place asynchronously (not at a particular time) with respect to the execution of the program

2 Steps for processing an event

- Register an event listener object to "listen" for specific types of events
- Implement event handler method(s) within listener to be called automatically in response to a particular type of event
- A class that implements an event listener interface must provide a definition for every method of that interface