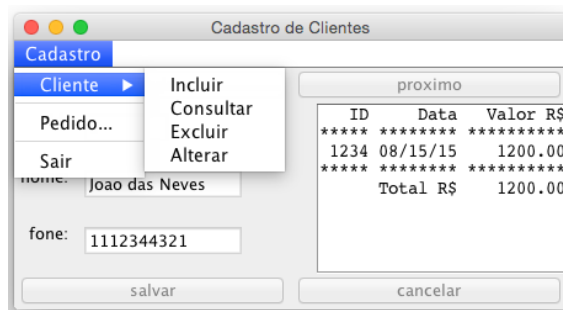


# Interface Gráfica

Programação de Computadores - Aula 10  
Professores: Bonato, Hamilton e Machion



# Imports

- Em geral, usam-se 3 pacotes (*packages*) diferentes
  - **java.awt.\***
    - A biblioteca AWT é uma GUI (*Graphical User Interface*), que delega a criação, estilo e comportamento para a GUI da plataforma nativa (Windows, Linux, Mac, ...).
    - A ideia é que se você fizer uma caixa de texto no Windows ou Linux, o estilo será o mesmo da plataforma, atendendo à promessa de portabilidade do Java, cujo lema é “*write once, run anywhere*”.
  - **javax.swing.\***
    - A biblioteca SWING foi introduzida um pouco depois como uma extensão, no java 1.1, e como parte do pacote padrão no java 1.2, no pacote “javax.swing”.
    - A biblioteca Swing não delega a criação, estilo e comportamento da GUI para a plataforma nativa, mas toma para si este trabalho, conseguindo cumprir o objetivo de portabilidade.
  - **java.awt.event.\***
    - Este pacote define classes e interfaces usadas para manipulação de eventos na AWT. Este pacote e todas as suas classes e interfaces também foram definidas no Java 1.1.

```
1 import javax.swing.JFrame;
2 import javax.swing.JButton;
3 import javax.swing.JTextField;
4 import javax.swing.JLabel;
5 import java.awt.Container;
6 import java.awt.FlowLayout;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
```

# Classe JFrame

- Uma tela com moldura, que pode ser maximizada e minimizada e que é vista pela aplicação, gerando ícone na barra de tarefas.
- Para sua tela ser um JFrame é fácil. Basta usar a herança e estender JFrame.

```
public class TelaCalculadora extends JFrame{  
  
}
```

# Controle de Eventos

- O mecanismo de controle de eventos do Java funciona assim:
  - Os componentes de tela geram eventos quando o usuário faz alguma coisa: quando ele clica o mouse, quando ele digita em um campo, quando ele aperta um botão.
  - O evento gerado é passado para a JVM.
  - Você pode “capturar” esses eventos e mandar o sistema fazer alguma coisa quando eles acontecerem, inscrevendo a classe que irá fazer alguma coisa como ouvinte (*listener*) do evento. Você verá isso mais adiante.
  - Por enquanto, é importante dizer que para poder ser ouvinte de eventos gerados por um botão, a classe precisa implementar a interface ActionListener e, como consequência, implementar o método definido por ela, que é o public void actionPerformed(ActionListener), também visto mais adiante.

```
10 public class TelaCalculadora extends JFrame implements ActionListener{
11
12     public void actionPerformed(ActionEvent e){
13
14     }
15 }
```

## Elementos Gráficos

- Crie suas variáveis de instância, que, neste caso, serão os elementos gráficos da tela, ou *widgets*:

```
12     private JButton botao; // botao
13     private JTextField texto; //campo texto de uma linha
14     private JLabel etiqueta; //etiqueta de nome do campo
```

# Construtor

- No construtor você irá efetivamente montar a tela
- A primeira coisa a fazer é chamar o **super()** passando como parâmetro o título da tela que irá ficar na barra superior da janela.

```
16 public TelaCalculadora(){  
17     //chamar construtor da superclasse e configurar o titulo  
18     super("Calculadora");  
19     |  
20 }
```

## Instanciação dos Elementos de Tela

- Instancie os botões, campos de texto e etiquetas criados; veja que cada construtor é de um jeito.
  - O do botão (JButton) recebe o texto do botão, assim como o da etiqueta (JLabel) recebe o texto da etiqueta.
  - O campo texto (JTextField), por sua vez, recebe o tamanho do campo.
  - Veja a documentação da API do Java para outras opções de construtor.

```
19 //instanciar os widgets
20 botao = new JButton("Soma");
21 texto = new JTextField("0", 10);
22 etiqueta = new JLabel("Valor: ");
```

## Gerenciador de Leiaute

- A tela de um JFrame, além da moldura, tem um painel, que é onde serão colocados os elementos de interface.
- Todo painel tem que ter o seu **gerenciador de layout**.
- Se não tiver, todo elemento de tela colocado neste painel irá assumir o tamanho do painel e, como consequência, somente o último elemento adicionado irá aparecer.
- No caso do JFrame, o painel é o Container, que você pega usando o método getContentPane().

# FlowLayout

- Este é o gerenciador de layout usado neste exemplo.
- Há outros, que serão mencionados mais tarde.
- O comportamento do FlowLayout é o seguinte: ele vai organizando os elemento de tela que vão sendo adicionados, da esquerda para direita, um depois do outro.
- Se acabar o espaço horizontal da tela, então ele vai para a linha de baixo.

```
23 //pega o container (ou painel)
24 Container caixa = getContentPane();
25 //configura o gerenciador de layout
26 caixa.setLayout(new FlowLayout());
```

## Adicionar os elementos de tela

- Use o comando `add(objeto)` para adicionar os elementos no painel (neste caso, `container`).
- O gerenciador de layout é quem irá organizá-los.

```
27 //adiciona na tela na ordem em que quer que apareça
28 caixa.add(etiqueta);
29 caixa.add(texto);
30 caixa.add(botao);
```

## Registrar o listener

- Lembre-se, o listener é quem ouve os eventos e reage a eles;
- Esta tela que estamos fazendo, por implementar a interface `ActionListener`, pode ser registrada como ouvinte de eventos de botão.

```
31 //registra este objeto como listener
32 botao.addActionListener(this);
```



# Ajustes finais da tela

- configure o tamanho (largura x e altura y);
- defina o que fazer quando o usuário clicar no X para fechar - neste caso, encerrar o sistema (mas nem sempre!);
- deixe a tela visível (sempre deve ocorrer por último).

```
33      //configura ajustes finais
34      //configura o tamanho inicial da tela
35      setSize(200,100);
36      //encerra a aplicacao quando clica o xis
37      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
38      //torna a janela visivel
39      setVisible(true);
```

Implementar o método que responderá aos cliques no botão.

- É o actionPerformed.
- No exemplo a seguir, ele verifica quem gerou o evento usando o getSource() e então faz alguma coisa;
- Neste caso, soma 10 ao valor que está o campo texto.
- Note que o método String getText() pega o texto que está digitado em um JTextField, e o setText(String) muda o texto.

```

42     public void actionPerformed(ActionEvent e){
43         if(e.getSource()==botao){
44             int valor = Integer.parseInt(texto.getText());
45             valor+=10;
46             texto.setText(""+valor);
47         }
48     }

```

```

1 import javax.swing.JFrame;
2 import javax.swing.JButton;
3 import javax.swing.JTextField;
4 import javax.swing.JLabel;
5 import java.awt.Container;
6 import java.awt.FlowLayout;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
9
10 public class TelaCalculadora extends JFrame implements ActionListener{
11
12     private JButton botao; // botao
13     private JTextField texto; //campo texto de uma linha
14     private JLabel etiqueta; //etiqueta de nome do campo
15
16     public TelaCalculadora(){
17         //chamar construtor da superclasse e configurar o titulo
18         super("Calculadora");
19         //instanciar os widgets
20         botao = new JButton("Soma");
21         texto = new JTextField("0", 10);
22         etiqueta = new JLabel("Valor: ");
23         //pega o container (ou painel)
24         Container caixa = getContentPane();
25         //configura o gerenciador de layout
26         caixa.setLayout(new FlowLayout());
27         //adiciona na tela na ordem em que quer que apareça
28         caixa.add(etiqueta);
29         caixa.add(texto);
30         caixa.add(botao);
31         //registra este objeto como listener
32         botao.addActionListener(this);
33         //configura ajustes finais
34         //configura o tamanho inicial da tela
35         setSize(200,100);
36         //encerra a aplicacao quando clica o xis
37         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
38         //torna a janela visivel
39         setVisible(true);
40     }
41
42     public void actionPerformed(ActionEvent e){
43         if(e.getSource()==botao){
44             int valor = Integer.parseInt(texto.getText());
45             valor+=10;
46             texto.setText(""+valor);
47         }
48     }
49 }

```

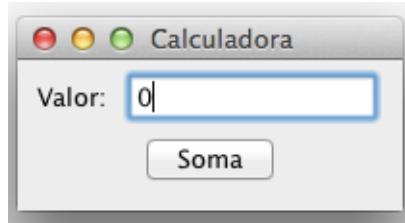
## Código completo

```

1 public class Calculadora{
2
3     public static void main(String[] args){
4         //instancia o JFrame
5         TelaCalculadora tela = new TelaCalculadora();
6     }
7 }

```

# Resultado



Outros gerenciadores de layout

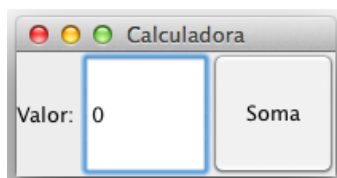
# BorderLayout

- O Border divide a tela em 5 espaços: central, norte (topo), sul (rodapé), leste (direita) e oeste (esquerda).
- Para adicionar coisas no Border você usa o mesmo comando add, só que desta vez com mais um parâmetro, a posição em que quer colocar.
- `container.add(objeto, BorderLayout.SOUTH);`
- `container.add(outroObjeto, BorderLayout.EAST);`

O código fica assim:

```
25 //configura o gerenciador de layout
26 caixa.setLayout(new BorderLayout());
27 caixa.add(etiqueta, BorderLayout.WEST);
28 caixa.add(texto, BorderLayout.CENTER);
29 caixa.add(botao, BorderLayout.EAST);
```

A tela fica assim:



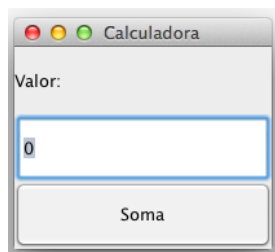
# GridLayout

- Divide a tela em uma matriz de linhas e colunas.
- Por exemplo, ao invocar o construtor `new GridLayout(3,1)` você está criando uma matriz de 3 linhas e 1 coluna.
- Use o `add` para colocar os elementos na tela.
- A cada `add`, o gerenciador irá adicionar o elemento de tela em uma célula da matriz, começando do topo, à esquerda, e preenchendo a linha até o final. Aí pula linha e volta para a esquerda.

O código fica assim:

```
26 caixa.setLayout(new GridLayout(3,1));  
27 //adiciona na tela na ordem em que quer que apareça  
28 caixa.add(etiqueta);  
29 caixa.add(texto);  
30 caixa.add(botao);
```

A tela fica assim:



# Combinação de Layouts

- Para usar mais de um gerenciador em um mesmo JFrame e construir layouts mais elaborados, faça o seguinte:
- Crie um JPanel e dê a ele o gerenciador de layout desejado.
- Adicione os elementos gráficos neste JPanel.
- Adicione o JPanel ao container do JFrame.

O código fica assim:

```
29 caixa.setLayout(new BorderLayout());
30 //cria os painéis secundários
31 JPanel painelSul = new JPanel(new FlowLayout());
32 JPanel painelCentro = new JPanel(new GridLayout(1,2));
33 //adiciona os widgets nos painéis secundários
34 painelSul.add(botao);
35 painelCentro.add(etiqueta);
36 painelCentro.add(texto);
37 //adiciona os painéis secundários no principal
38 caixa.add(painelSul, BorderLayout.SOUTH);
39 caixa.add(painelCentro, BorderLayout.CENTER);
```

A tela fica assim:

