Introdução a Programação – IF669
http://www.cin.ufpe.br/~if669

Programação com Interface Gráfica

AULA 16

Ricardo Massa F. Lima
rmfl@cin.ufpe.br

Sérgio C. B. Soares
scbs@cin.ufpe.br

Centro de Informática — Universidade Federal de Pernambuco — CIn.ufpe.br

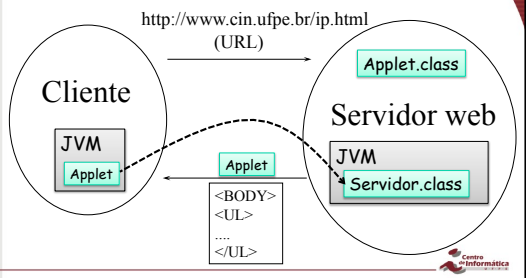
Até aqui...

- Criamos a infra-estrutura para a aplicação bancária
  - classes que representam os tipos de conta
  - classe que implementa regras de negócio
  - interface e classe que armazena contas
- Mas ainda não temos uma interface **decente** com o usuário

Interfaces gráficas em Java

- Por ser em Java as interfaces gráficas criadas estão aptas a serem executadas pela internet
- Um `applet` é um programa Java que executa na internet
  - Subclasse de `java.applet.Applet`

Clientes e Servidores WWW



Applets versus HTML+CGI+Javascript

- Applets eliminam gargalos:
  - cliente fala direto com o servidor da aplicação (usando objetos ao invés de strings)
  - qualquer tipo de processamento pode ser feito no cliente
  - carga do servidor pode ser distribuída
- Java dá suporte a princípios de engenharia de software
- Problemas: eficiência e portabilidade

WWW e Java: Interação

- *Applets* são programas Java disponibilizados via WWW, através de uma página HTML

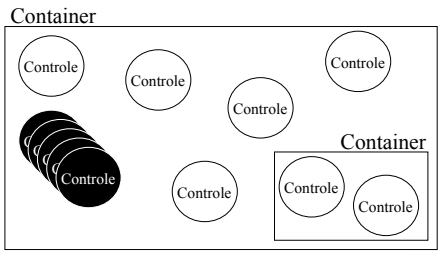
```
<HTML>
...
<applet codebase="http://www.cin.ufpe.br/ip"
        code="AppletBanco.class"
        width=10  height=90>
</applet>
...
</HTML>
```

AppletBanco:
Atributos (como qualquer outra classe Java)

```
public class AppletBanco extends Applet {
  //Ligação com o banco
  private Banco fachada = null;

  //Controles
  Button buttonProcurar = new Button();
  Label labelValor = new Label();
  TextField textFieldVal = new TextField();
  Button buttonDebito = new Button();
  ...
```

Container e Controles

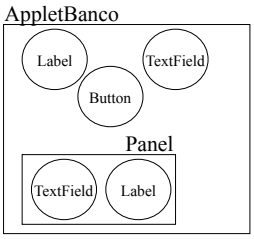


Componentes do AWT

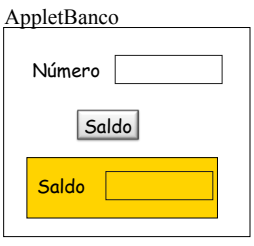
- Containers
  - **Container**
  - **Panel**
  - **Window**
  - **Frame**
  - **Dialog**
  - **Applet**
  - ...
- Controles
  - **Button**
  - **Canvas**
  - **Label**
  - **TextField**
  - **Choice**
  - **List**
  - ...

Mesmos componentes do Swing acrescentando J no inicio do nome

Container e Controles



Container e Controles



Mas como os controles e os componentes se comunicam?

- Comunicação por eventos
  - editor vs assinante (*publisher* vs *subscriber*)
- Editora vs. Assinante do jornal
  - Botão é a editora
    - —Avisa aos assinantes quando for clicado
  - Applet é o assinante
    - —Pede ao botão para ser avisado

Em Java

- Objetos se comunicam através de métodos
- Eventos = Comunicação assíncrona
  - O applet chama um método do botão para assinar o serviço
  - O botão chama um método do applet para avisar quando for clicado

Vamos implementar um botão

```
public class Botao {
  private Object[] assinantes;
  private int indice;
  ...
  public void assinar(Object o) {
    assinantes[indice] = o;
    indice++
  }
  private void avisa() {
    for(int i=0; i < indice; i++) {
      assinantes[i].botaoClicado();
    }
  }
}
```

Tem esse método em Object?

Mas espera...

- O botão tem de chamar sempre um mesmo método dos assinantes
  - Como obrigar que todo assinante tenha esse mesmo método?
- Lembram de interfaces?

Uma interface para assinante

```
public interface Assinante {
    void botaoClicado();
}
```

Vamos mudar o botão para aceitar apenas assinantes...

Vamos implementar um botão

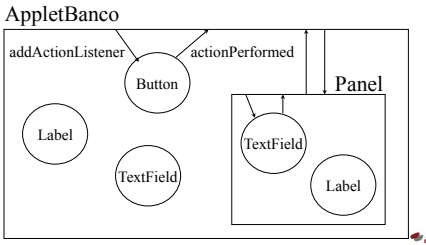
```
public class Botao {
  private Assinante[] assinantes;
  private int indice;
  ...
  public void assinar(Assinante o) {
    assinantes[indice] = o;
    indice++
  }
  private void avisa() {
    for(int i=0; i < indice; i++) {
      assinantes[i].botaoClicado();
    }
  }
}
```

Agora sim!

Mas em Java...

- Os nomes são ligeiramente diferentes
  - **`Assinante` é `ActionListener`**
  - **`botaoClicado` é `actionPerformed`**

Comunicação entre Containers e Controles: Eventos



AppletBanco: Inicialização

```
public void init() {
    ...
  javax.swing.JLabel lb_title = new JLabel();
  jContentPane = new javax.swing.JPanel();
  jContentPane.setLayout(null);
  lb_title.setBounds(63, 14, 178, 23);
  lb_title.setText("Applet Banco");
  jContentPane.add(lb_title, null);
  jContentPane.add(getJTextField(), null);
  jContentPane.add(getJTextField2(), null);
  jContentPane.add(getJButton(), null);
  jContentPane.add(getJButton2(), null);
...
}
```

AppletBanco: Tratamento de Eventos

```
private JButton getJButton2() {
 if (bt_creditar == null) {
   bt_creditar = new JButton();
   bt_creditar.setBounds(197, 79, 86, 25);
   bt_creditar.setText("Creditar");
   bt_creditar.addActionListener(assinante);
 }
 return bt_creditar;
}
```

Na verdade o código gerado é um pouco mais confuso

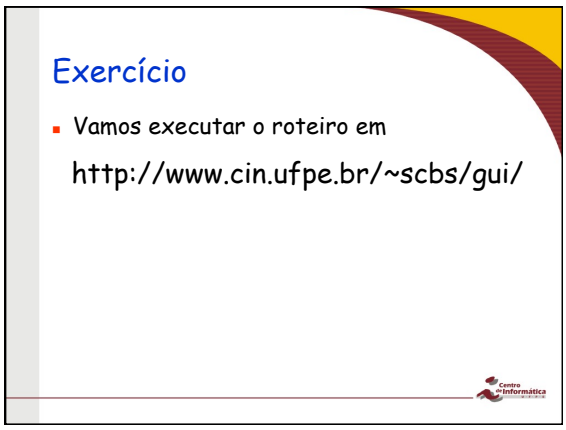
AppletBanco: Tratamento de Eventos

```
new java.awt.event.ActionListener() {
  ...
}
  bt_creditar.setBounds(197, 79, 86, 25);
  bt_creditar.setText("Creditar");
  bt_creditar.addActionListener( ... );
}
public void actionPerformed(ActionEvent e) {
  creditar();
}
```

```
private void creditar() {
 String numero = this.tf_numero.getText();
 String v = tf_valor.getText();
 try {
   double valor = new Double(v).doubleValue();
   banco.creditar(numero, valor);
   JOptionPane.showMessageDialog(this,
          "Crédito executado com sucesso!");
 } catch (ContaNaoEncontradaException e) {
   JOptionPane.showMessageDialog(this,
          "Conta " + numero + " nao existe!");
 } catch (NumberFormatException e) {
   JOptionPane.showMessageDialog(this,
          "Digite um número!");
 } finally {
   ...
 }
}
```

Applets: Aspectos de Segurança

- *Applets* devem satisfazer várias restrições, impostas pelos folheadores:
  - não ter acesso a arquivos do cliente
  - só se conectar com o servidor de origem
  - não usar métodos nativos
- Restrições podem ser eliminadas para *applets* assinados e transmitidos de forma segura!


Exercício
Vamos executar o roteiro em
http://www.cin.ufpe.br/~scbs/gui/