



# CCM310

# Arquitetura de Software e

# Programação Orientada a Objetos

Profa. Dra. Gabriela Biondi

Prof. Dr. Isaac Jesus

Prof. Dr. Luciano Rossi

GUI  
*Graphical User Interface*

# Por que utilizar interface gráfica?

- Mais amigável para o usuário final
- Interação mais rápida e mais produtiva para o usuário final

# GUI e Orientação a Objetos

- **Um componente GUI é um objeto!**

Exemplos de Classes:

- Janelas
- Botões (*Buttons*)
- Caixas de Texto (*Text Panes*)
- Caixas de Combinação (*Combo Box*)
- etc

# GUI Toolkits

- Swing (AWT - *Abstract Window Toolkit*)
- JavaFX

# Algumas IDEs para trabalhar com GUI

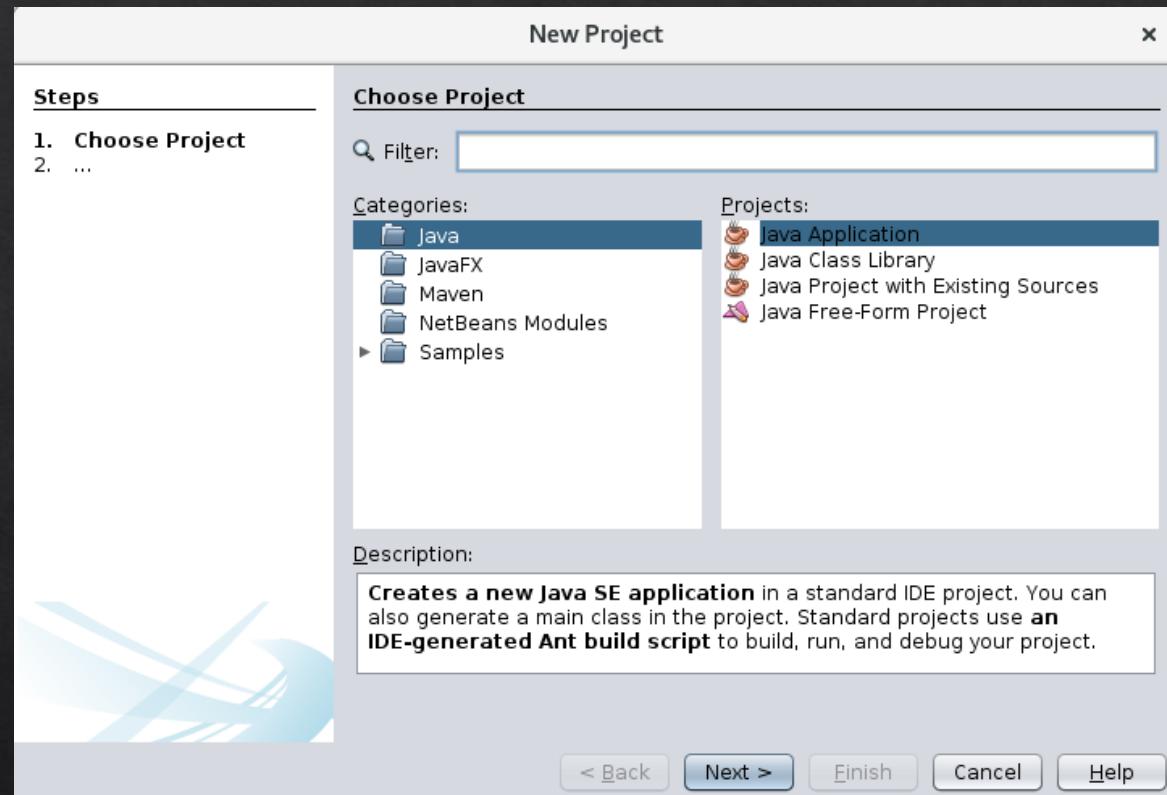


Exemplo:  
GUI para somar 2 números em Java

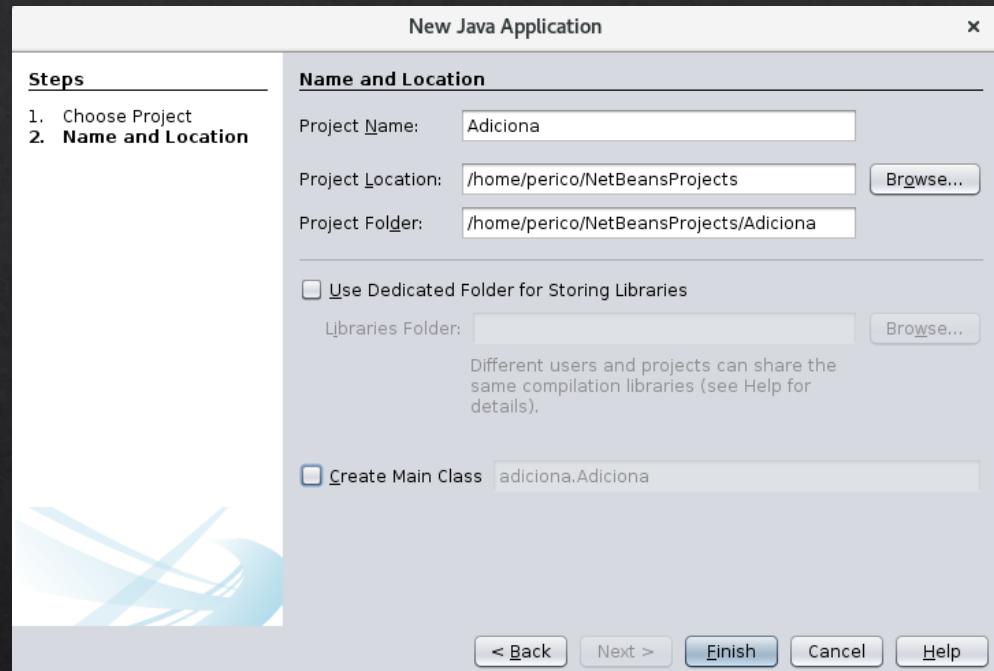
# Criando a Interface Gráfica no NetBeans

1. Abra o NetBeans

2. Clique em Arquivo - Novo Projeto - Categorias (*Java with Ant*) / Projetos (*Aplicação Java*) - Next

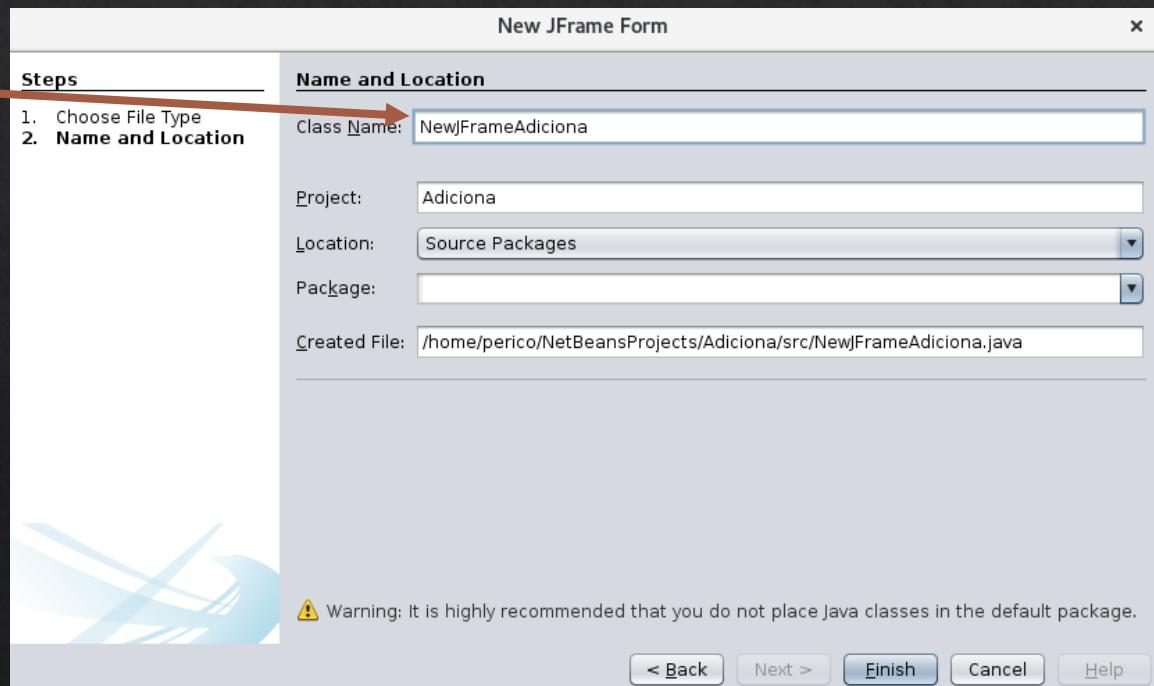


### 3. Dê um nome para seu projeto



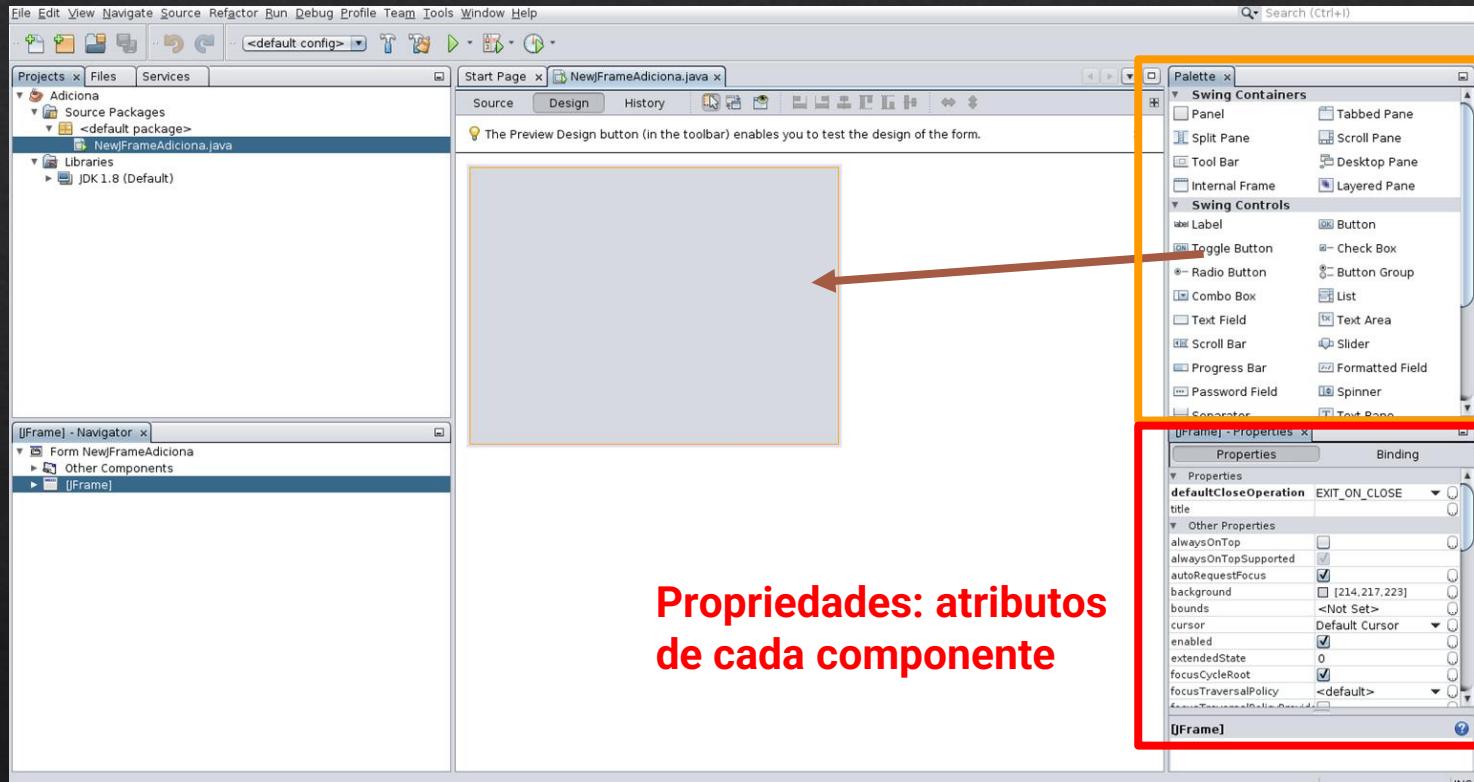
### 3. Clique em Terminar ou Finish

6. Clique com o botão direito do mouse em cima do nome do projeto
7. Selecione - Novo - **JFrame Form**
8. Dê um nome para a classe do formulário
9. Clique em Finish



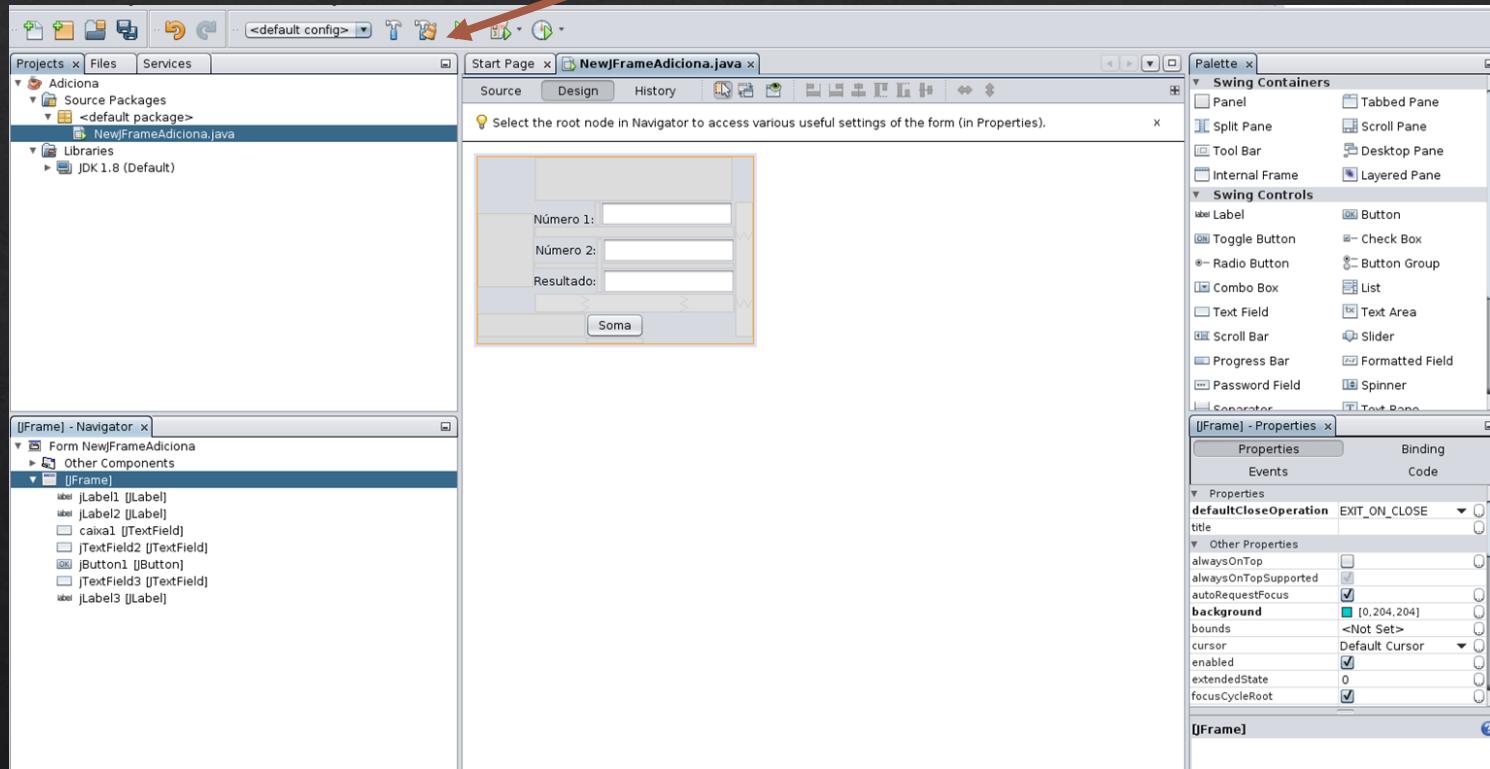
10. Selecione os componentes na tela da direita e arraste para o *JFrame*, que está no meio da tela

Paleta: componentes GUI

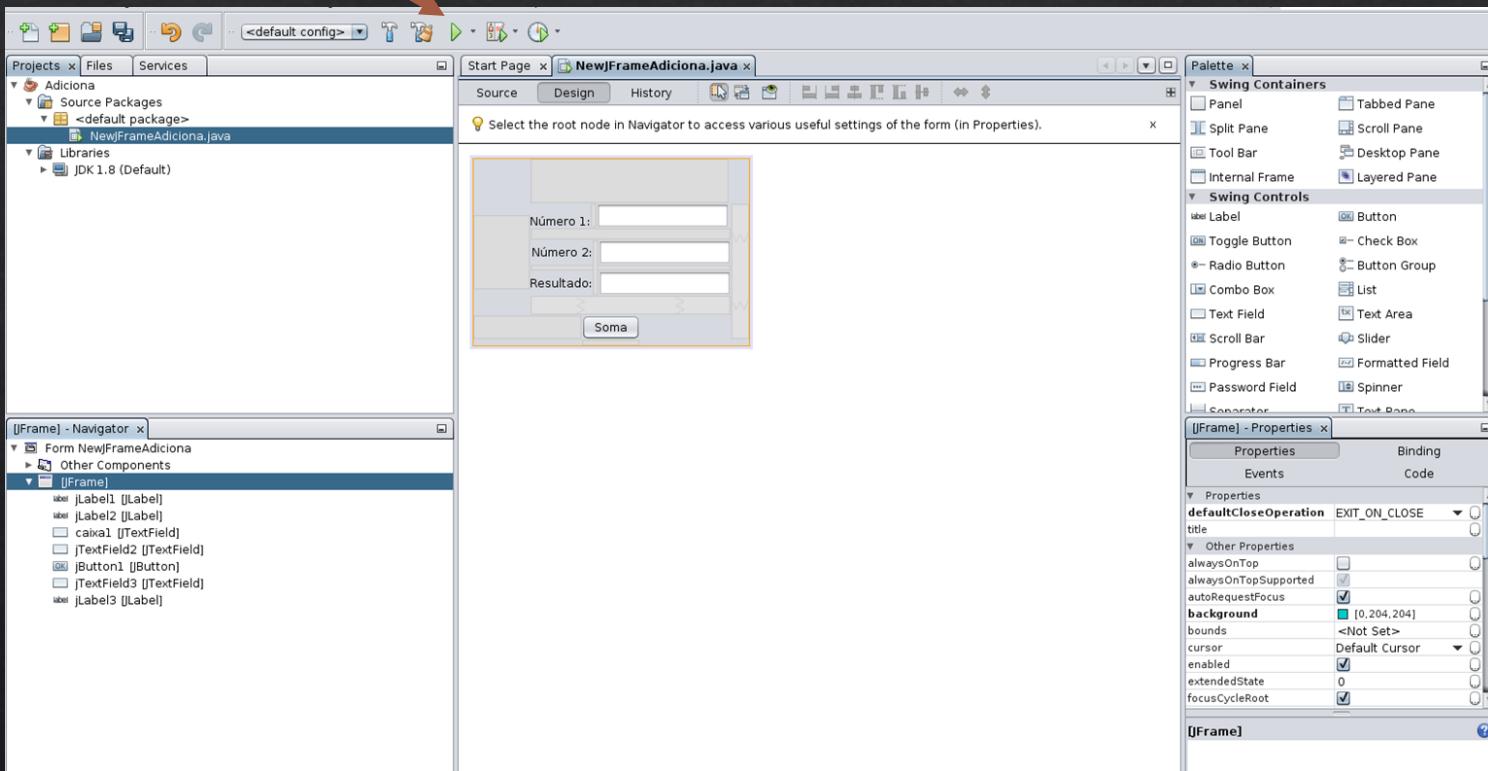


11. Depois de criar a interface gráfica, instancie um objeto da interface gráfica criada na sua classe principal
12. Depois, chame o método *setVisible()*, com o argumento *true*

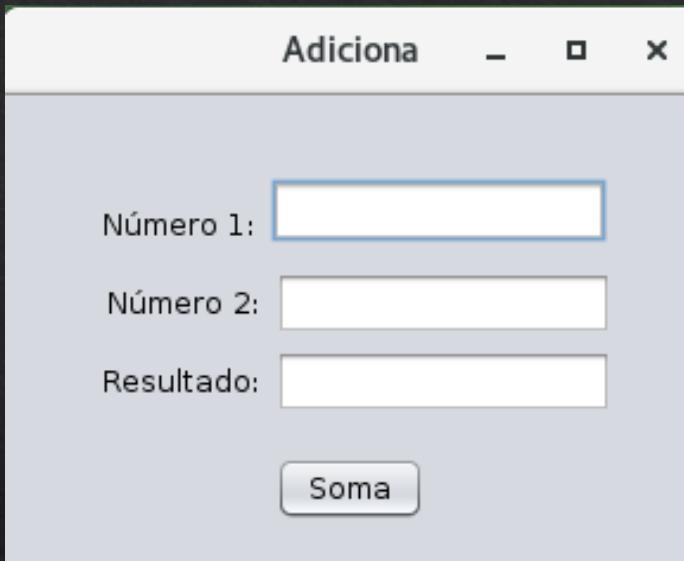
# 13. Clique em limpar e construir o projeto



# 14. Clique em rodar / executar



A janela criada deve aparecer na tela:



# Adicionando lógica

1. Clique duas vezes no botão “*Soma*”
2. O NetBeans vai sair da área de *Design* e vai para a área *Source*
3. Você deve programar o que você quer que aconteça quando o botão for clicado no método  
*jButtonActionPerformed*
4. Você deve utilizar os métodos já existentes em cada objeto adicionado à interface para manipular dados e eventos

Será que existe outra maneira de  
organizar o código?

Talvez uma arquitetura de software?

# MVC - Model-View-Controller

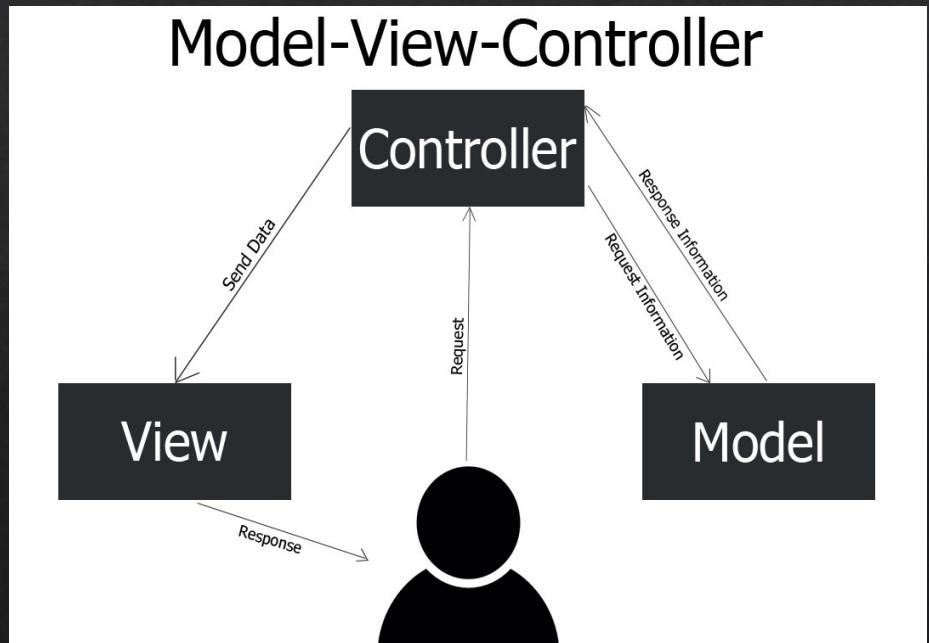
- A arquitetura MVC é utilizada para separar as funcionalidades e a apresentação de uma aplicação
- Divide um dado aplicativo em três partes interconectadas:
  - **Model** (Modelo)
  - **View** (Visualização)
  - **Controller** (Controle)

# MVC - Model-View-Controller

- MVC permite a reutilização eficiente do código e o desenvolvimento paralelo
- Tradicionalmente usado para interfaces gráficas de usuário (GUIs)
- Também para:
  - Web
  - Mobile

# MVC - Model-View-Controller

- **Model** (Modelo): lógica, regra do negócio
- **View** (Visualização): interface com o usuário
- **Controller** (Controle): intermediação entre View e Model

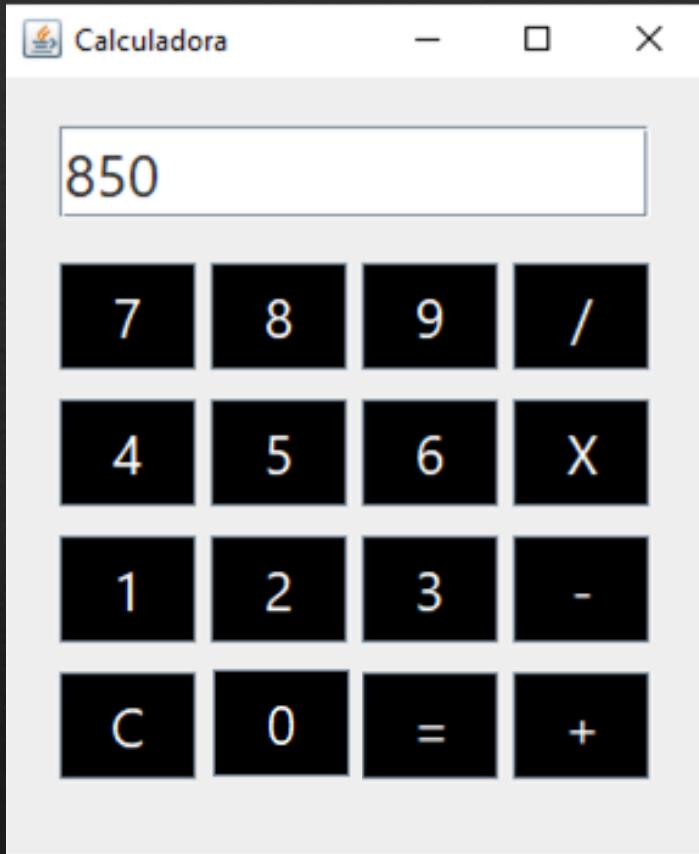


# Exemplo

- Faça novamente o exemplo de adicionar dois números, separando em M (*Model*), V (*View*) e C (*Controller*).
- Faça cada um em um pacote diferente.

# Exercício 1

- Crie um novo projeto chamado Calculadora:
- Crie um *JFrame* e adicione todos os componentes GUI necessários
- Faça a calculadora com as 4 operações básicas ( + , - , \* , / )
- Utilize Herança, Polimorfismo e Classe Abstrata!
- Utilize a arquitetura MVC



# JAR

*(Java ARchive)*

# JAR

- Depois que um aplicativo fica pronto, é esperada a existência de muitas classes
- Para simplificar o uso é mais fácil compactar todas as classes em um arquivo só
- Então cria-se o *jar* (Java ARchive)
- O arquivo *.jar* possui um conjunto de classes (e arquivos de configurações) compactados, no estilo de um arquivo zip.
- Podemos executar o aplicativo diretamente com o seu *.jar*
- Assim, o arquivo *.jar* serve tanto como executável quanto como biblioteca

*jTabbedPane*

# *jTabbedPane*

- Painel com Guias
- container de painéis:
  - modulariza um formulário em diversas seções



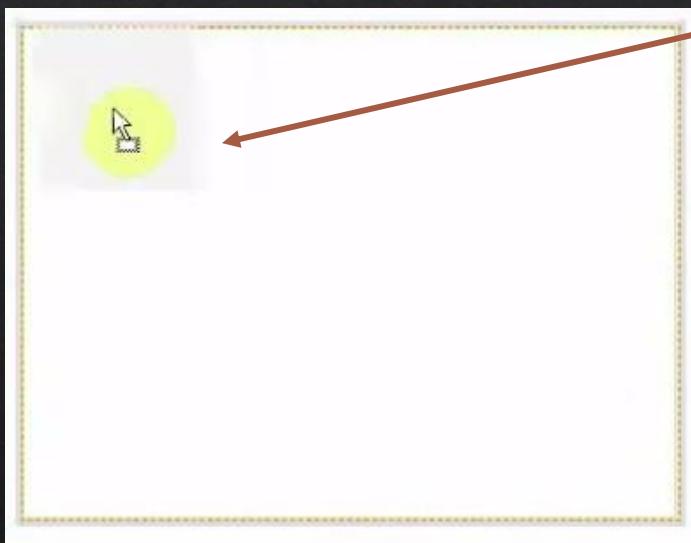
# *jTabbedPane*

- Arraste para a janela principal o componente “Painel com Guias”:



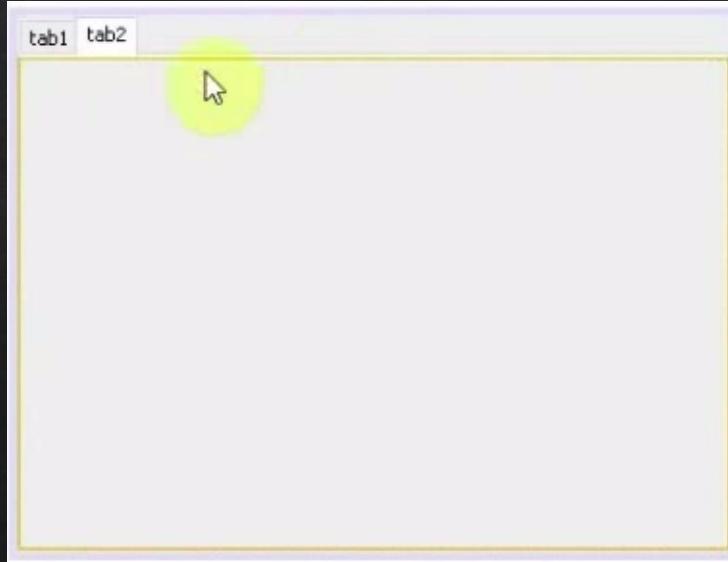
# *jTabbedPane*

- Arraste o componente “Painel” para dentro do “Painel com Guias”:



## *jTabbedPane*

- Cada “Painel” arrastado deve gerar uma nova guia!



*Combobox*

# *jComboBox*

- Como colocar os itens no *jComboBox*:
  - *Propriedades -> Model*
- Para ler o que foi selecionado no *jComboBox*:

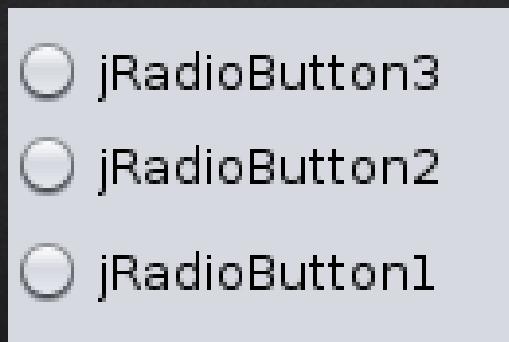


```
String tam = jComboBox1.getSelectedItem().toString();
```

# Botões de Opção *(RadioButton)*

# RadioButton

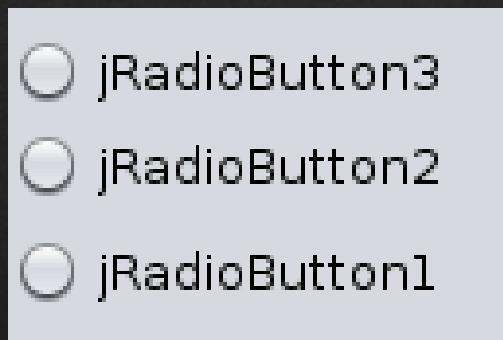
*jRadioButton*



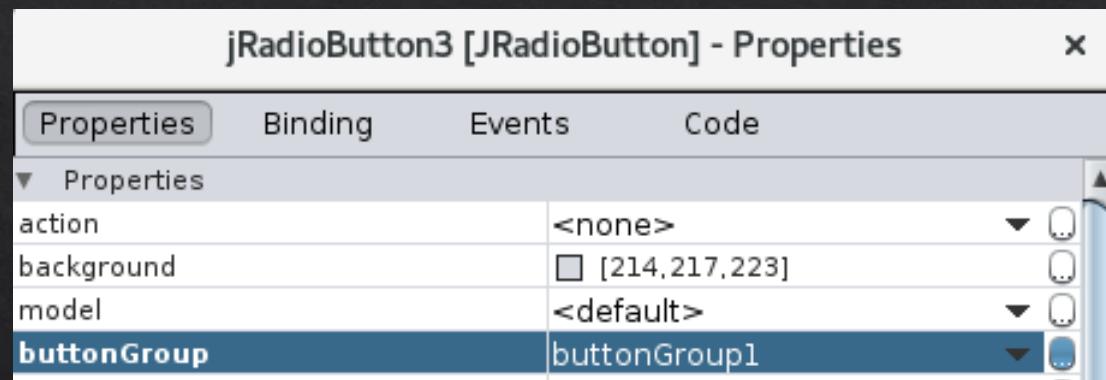
- Serve para permitir que o usuário selecione somente uma opção dentre todas opções dadas
- Para funcionar em grupo, todos os RadioButtons precisam pertencer ao mesmo Grupo! (*ButtonGroup*)

# RadioButton

*jRadioButton*



- Conseguimos associar cada *jRadioButton* a um grupo nas propriedades dos botões:



- O *ButtonGroup* deve existir no *JFrame* (é necessário arrastar um *ButtonGroup* para o frame)

# RadioButton

*jRadioButton*

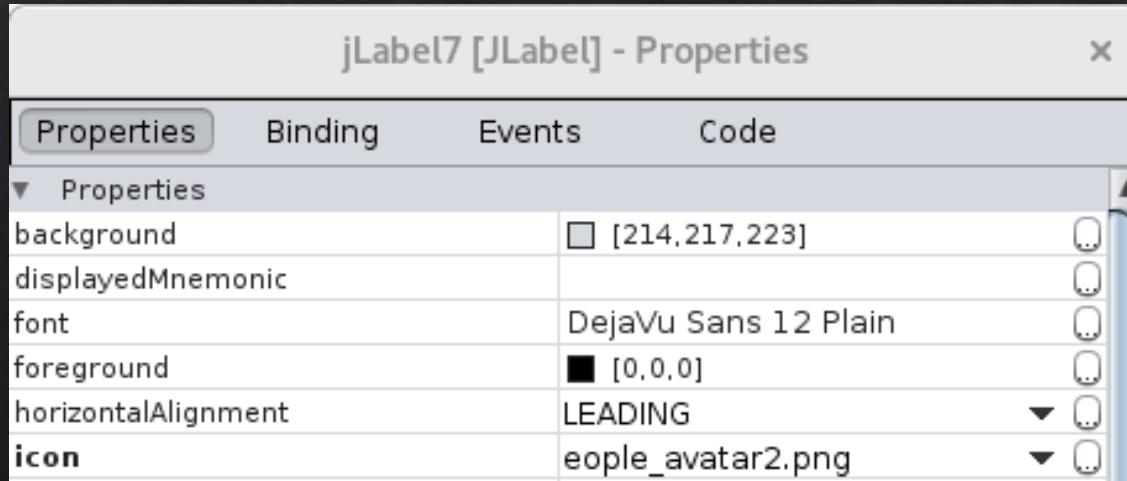
- jRadioButton3
- jRadioButton2
- jRadioButton1

- No código podemos usar o método *isSelected()* do *jRadioButton* para verificar qual está selecionado!

# Inserindo Imagens

# Imagens

- Normalmente, imagens podem ser inseridas como *jLabels*
- Nas propriedades do *jLabel*, podemos atribuir uma imagem pela propriedade *icon*:



# Menus

# Menus

- Para inserir menus (*Swing Menus*), arrastamos o componente Menu Bar (*jMenuBar*) para o JFrame.
- Então podemos adicionar *jMenus* e em cada *jMenu* podemos adicionar *jMenuItem*:
  - *jMenu* são os nomes que aparecerão na barra de menu
  - *jMenuItem* são os itens de cada *jMenu*
  - Cada *jMenuItem* pode ser programado como se fosse um botão tradicional (dois cliques no componente para programar o comportamento de clique)

# Adicionando outras janelas ao Projeto

# Outras janelas

- Para inserir novas janelas, podemos criar outros *jFrames*
- Na janela principal, podemos instanciar o *jFrame* novo na hora que quisermos visulizá-lo e devemos também alterar a sua visibilidade para *true*.
- Exemplo: você pode incluir o código no método de clique de um botão:

```
NovoForm novo;  
novo = new NovoForm();  
novo.setVisible(true);
```

# Exercício 1: cadastro permanente de usuários e tela de exibição por CPF.

Cadastro

Usuários

Nome:

Sobrenome:

Idade:

CPF:

Sexo:

Masculino

Feminino



Cadastra

Procura usuário

Procurar usuário por CPF

Exibir



# Exercício 1

- São duas janelas!
- A segunda janela é acessada pelo menu da primeira janela: *Usuários -> Exibir*
- O cadastro deve ser feito em ArrayList

Obrigada pela sua  
participação, nos vemos na  
próxima aula! :)