

# Programação Orientada a Objetos

Marcos Lapa dos Santos  
marcoslapa@gmail.com



**POO**

---

**Interface Java Swing**

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Java Swing

- Nas primeiras versões do Java a única forma de fazer programas gráficos era através do AWT (Abstract Window Toolkit).
  - Biblioteca de baixo-nível
  - Oferece uma infra-estrutura mínima de interface gráfica
  - Depende de código nativo da plataforma onde roda
  - Traz alguns problemas de compatibilidade entre as plataformas
  - Nem sempre o programa fica com a aparência desejada em todos os sistemas operacionais.
- O Swing foi criado como uma extensão do Java a partir da versão 1.2

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Java Swing

- JFC (Java Foundation Classes) oferece uma interface muito mais rica em comparação ao AWT:
  - Swing é o nome dado à coleção de **componentes visuais** (Escritos completamente em Java)
  - É preciso importar **java.awt** e **javax.swing** para usar a JFC
  - JFC/Swing **substitui** os componentes AWT
  - Mantém e estende a interface de **eventos** e **layout**
  - “Look & Feel”, Drag & drop, cut & paste
  - Baseada em **JavaBeans**: ferramentas GUI conseguem gerar código legível e reutilizável

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### IDE

- Em nosso curso utilizaremos a IDE JCreator e a NetBeans disponibilizada pela Oracle em seu site.
- Ambas são free.
- NetBeans possui alta integração com os componentes Swing, e será usada na segunda parte da disciplina.
- Os trabalhos poderão ser entregues em NetBeans ou JCreator

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Componentes

- Todo componente Swing contém um "J" na frente, como em JButton por exemplo. Componente AWT não contém inicial alguma ( "Button" no caso ). Outros exemplos:
- Frame (AWT) e JFrame (Swing)
  - Servem de base para qualquer aplicação gráfica
- Panel e JPanel
  - Container de propósito geral
  - Serve para agrupar outros componentes e permitir layout em camadas

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### JFrame

- Componente Top-Level (Nível mais alto)
- Não está contida dentro de outra window
- A Classe deve ter um método **main** instanciando o novo frame e chamando o método **setVisible(true)**:

```
public static void main(String[] args) {  
    JFrame appframe = new JFrame();  
    appframe.setSize( 420,250 );  
    appframe.setVisible(true);  
}
```



Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Principais Métodos - JFrame

- **Construtor** – Declarando um novo JFrame
  - `JFrame jFrame1 = new JFrame();`
  - `JFrame jFrame2 = new JFrame("Título do Frame");`
- **setSize(int x,int y)** – Define o tamanho que o Frame terá, respectivamente Largura e Altura;
- **setVisible(true)** - exibe o Frame na tela;
- **setLayout(Tipo do Layout)** – modifica o Layout padrão;

Prof. Marcos Lapa – marcoslapa@gmail.com

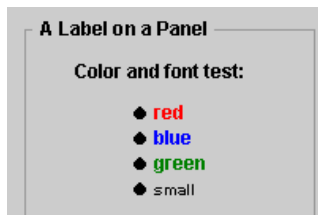


## POO

---

### JPanel

- Container de propósito geral
- Serve para agrupar outros componentes e permitir layout em camadas



Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Principais Métodos - JPanel

- **Construtor** – Declarando um novo JPanel
  - `JPanel jPanel = new JPanel();`
- **`setBorder(BorderFactory.createEtchedBorder());`** - Modifica a borda para o painel;
- **`setBounds(new Rectangle(5, 126, 395, 61));`** - Diz a posição x,y, largura e altura do painel;
- **`add(componente);`** - adiciona um componente em um painel;
- **`setToolTipText("Sou o Painel dos botões!");`** - Coloca um texto de ajuda (exibido quando o mouse fica sobre o painel);

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Componente - JButton

- Componente swing que implementa um botão na aplicação;
- Construtor sem parâmetros
  - `JButton jButtonNovo2 = new JButton();`
- Construtor com parâmetros
  - `JButton jButtonNovo2 = new JButton("texto que ira aparecer no botão",imagem1);`



Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Principais Métodos - JButton

- **`setToolTipText("insere novo item");`** - coloca um texto de ajuda (hint) (quando o mouse ficar sobre o botão, este texto é exibido);
- **`setText("Novo");`** - Seta um texto para o botão;
- **`setBounds(new Rectangle(160, 3, 190, 40));`** - seta a posição x, y, largura e altura do botão;
- **`setMnemonic('C');`** - define a letra de atalho para o clique do botão.

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Componente - JLabel

- Componente swing usado para rotular Campos ou colocar algum texto na aplicação
- Muito usado antes de campos de texto (textfields) para indicar o campo a ser preenchido:



Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Principais Métodos JLabel

- **setBounds(190, 40, 34, 15);** - determina a posição x, y, largura e altura do JLabel;
- **setText("Texto");** - Coloca um texto para o jlabel
- **setToolTipText("Texto do Cpf");** - coloca um texto de ajuda ao JLabel
- **setFont(new java.awt.Font("Arial Black", 2, 11));** - Determina a fonte do texto (o 11 significa tamanho da fonte e o 2 a formatação)(1 negrito, 2 itálico)

Prof. Marcos Lapa – marcoslapa@gmail.com

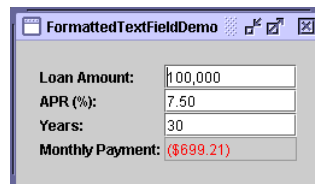


## POO

### JTextField

- Componente swing para colocar Campos de texto na aplicação;
- Muito usado para Entrada de dados;
- Exemplo: um campo cpf será preenchido via swing e depois salvo no BD;

Years:



FormattedTextFieldDemo

Loan Amount: 100,000

APR (%): 7.50

Years: 30

Monthly Payment: (\$699.21)

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Principais Métodos JTextField

- **setBounds(190, 10, 100, 25);** - determina a posição x, y, largura e altura do JTextField;
- **setText("Texto");** - Determina um texto para o JTextField
- **setToolTipText("Campo onde vc digita o cpf");** - Determina um texto de ajuda para o textfield
- **setEditable(false);** - Determina se o textField estará em edição ou não

Prof. Marcos Lapa – marcoslapa@gmail.com

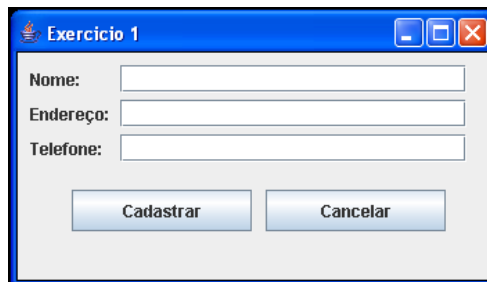




## POO

### Exercício 1

- Crie a seguinte tela usando os componentes swing já vistos:
  - Utilize um **JPanel** para adicionar os componentes visuais
  - Use o **JCreator** para fazer o exercício



Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### JMenu

- Componente que define um menu para sua aplicação
- Necessário criar um Componente **JMenuBar** (componente que implementa uma barra de menus)
- Construtor
  - `JMenu jMenu = new JMenu();`



Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Principais Métodos - JMenu

- **add(JmenuItem)** – Adiciona um submenu ao menu;
- **setText("Texto")** – Define um texto ao menu;
- **setBorder(new javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.LOWERED))** – define uma borda para o JMenu;
- **setIcon(new javax.swing.ImageIcon("C:\\imagem2.jpg"))**; - Define um ícone (imagem) para ser exibida junto com o texto do menu;
- **setMnemonic('s')**; - Define uma letra de atalho para o menu;

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Principais Métodos - JMenuBar

- Construtor
  - JMenuBar jMenuBar = new JMenuBar();
- **add(JMenu)**; - adiciona à barra de menus um componente JMenu

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Componente - JMenuItem

- Componente utilizado para definir Itens de menu;
  - JMenuItem jMenuItem = new JMenuItem();
- Métodos
  - setText("texto"); - define um texto para o item de menu;

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Exercício 1.1

- Adicione uma barra de menu à tela do Exercício 1 contendo o Menu **Cliente** e os itens de Menu **Novo** e **Editar**.
- Ao final, modifique o título do Frame principal para **Exercício 1.1**

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### JTextArea

- Componente que coleta dados de entrada com mais de uma linha de comprimento;
- Construtores
  - `JTextArea jTextArea = new JTextArea(4,20)` // 4 linhas e 20 colunas;
  - `JTextArea jTextArea = new JTextArea();`

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Principais Métodos - JTextArea

- **`setText("text")`**; - Define um texto para o componente;
- **`append("texto")`**; - acrescenta um texto ao final do texto que já se encontra no componente;

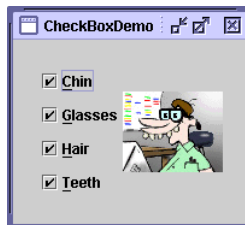
Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### JCheckBox

- Componente de caixa de seleção;
- Pode-se selecionar clicando em um item e desmarcar clicando neste item novamente;
- Necessita de um rótulo ao lado para identificar sua finalidade;
- Pode ser selecionado uma, várias ou nenhuma das caixas de seleção;



Prof. Marcos Lapa – marcoslapa@gmail.com

## POO

### Principais Métodos - JCheckBox

- **boolean isSelected();** - retorna o estado da caixa de seleção;
- **void setSelected(boolean estado)** - coloca a caixa de seleção em um novo estado;

Prof. Marcos Lapa – marcoslapa@gmail.com

## POO

### JRadioButton

- Usuário só pode fazer uma escolha dentre as várias opções;
- Quando outra caixa de seleção é marcada a caixa anterior é automaticamente desmarcada;
- Utiliza-se o objeto ButtonGroup para implementar um grupo de botões;
- O objeto ButtonGroup controla a desativação do botão selecionado anterior quando um novo é selecionado;
- Para adicionar os JRadioButton ao grupo de botões utiliza-se um método add;

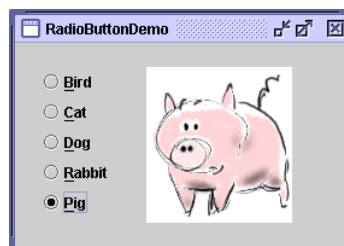
Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Principais Métodos - JRadioButton

- Construtor
  - `JRadioButton jRadioButton = new JRadioButton();`
- Métodos
  - `void setSelected(boolean estado)` - coloca a caixa de seleção em um novo estado;



Prof. Marcos Lapa – marcoslapa@gmail.com

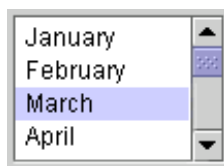


## POO

---

### JList

- Componente de lista que permite colocar objetos dentro de uma só caixa;
- Para selecionar um objeto da lista não se usa botões clica-se nos próprios itens;



Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Principais Métodos JList

- **Construtor**
  - `String[] palavras = {"item", "item2"};`
  - `JList listapalavras = new JList(palavras);`
- **setSelectedIndex(1);** - Seleciona um determinado item da lista
- **getSelectedValue();** - retorna um determinado valor selecionado

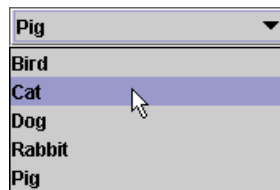
Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### JComboBox

- É uma caixa de lista suspensa similar a uma caixa de lista só que ocupa menos espaço;
- Quando o usuário clica no campo aparece uma lista de opções, permitindo o usuário escolher uma delas;



Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Principais Métodos - JComboBox

- **Construtor:**
  - O construtor do JComboBox segue o mesmo raciocínio do construtor do JList (podemos passar os valores em um array de strings)
- **void setEditable(boolean b)** – permite mudar o estado de edição do componente;
- **void addItem(Object item);** - adiciona um item à lista de itens
- **void removeItem(Object item);** - remove um item da lista
- **Object getSelectedItem();** - retorna o item atualmente selecionado

Prof. Marcos Lapa – marcoslapa@gmail.com





## POO

---

### JTable

- Apresenta uma grade bidimensional de objetos;
- São comuns em Interface com o usuário
- Um componente JTable não armazena seus próprios dados, mas os obtém de um modelo de tabela;
- Pode ser preenchido a partir de um array com duas dimensões.

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

---

### Caixas de diálogo

- São caixas de diálogo separadas que dão informações ou que podem obter informações dos usuários;
- A classe JOptionPane possui quatro métodos estáticos:
  - showMessageDialog
  - showConfirmDialog
  - showOptionDialog
  - showInputDialog

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Layouts Managers

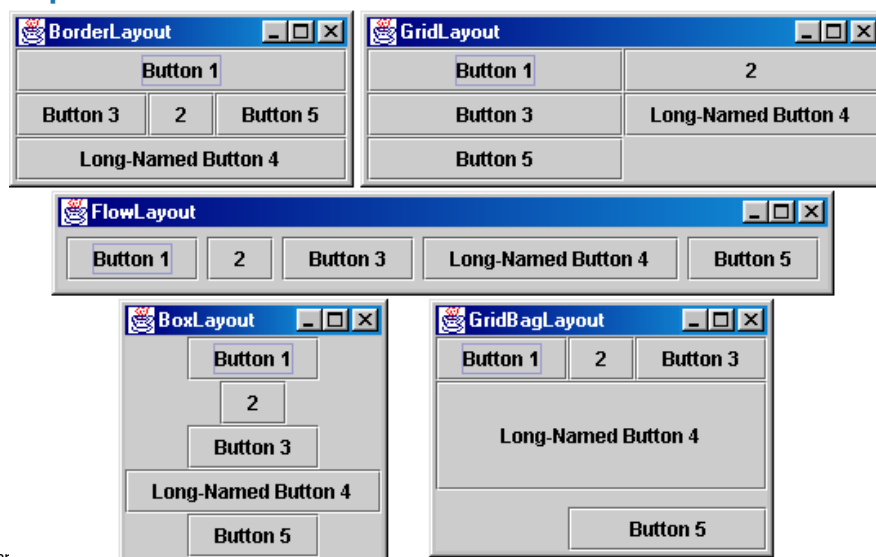
- São objetos que determinam como os componentes visuais de um container serão exibidos;
- Objetivo - permitir o ajuste automático da interface de acordo com a plataforma;
- A API padrão Java possui seis layout managers, sendo eles: FlowLayout, GridLayout, BorderLayout, CardLayout, GridLayout e BoxLayout.

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Exemplos



Prof. Marcos Lapa – marcoslapa@gmail.com

## POO

### Layout Managers

- **FlowLayout** - Armazena os componentes da esquerda para direita;
- **BoxLayout** - Armazena os componentes numa simples coluna, respeitando os seus tamanhos;
- **BorderLayout** - Possui cinco áreas para colocação de componentes (NORTH, SOUTH, WEST, EAST e CENTER).
- **CardLayout** - Organiza os componentes na forma de um “baralho”, de maneira que um só dos componentes está visível num determinado momento, os outros ficam escondidos por sua vez.
- **GridLayout** - Redimensiona os componentes para o mesmo tamanho e exibe-os numa tabela com número de linhas e colunas especificadas.

Prof. Marcos Lapa – marcoslapa@gmail.com



## POO

### Exercício 2

- Crie a seguinte tela usando componentes swing no **JCreator**:

Prof. Marcos Lapa – marcoslapa@gmail.com

