

Programação II

Cassio Diego

Aula de hoje

UNIDADE 1

1. Criação de interfaces gráficas usando as JFC/Swing
 - 1.1. Introdução (revisão de orientação a objetos)
 - 1.2. Hierarquia de classes
 - 1.3. Modelos de desenvolvimento de interfaces gráficas
 - 1.3.1. Desenvolvimento do SWING para GUI
 - 1.3.2. Gerenciadores de layout
 - 1.3.3. Layouts compostos
 - 1.3.4. Manipulação de aspectos visuais
 - 1.3.5. Variações de componentes visuais

Antes da orientação a objetos

ANTES DA ORIENTAÇÃO A OBJETOS

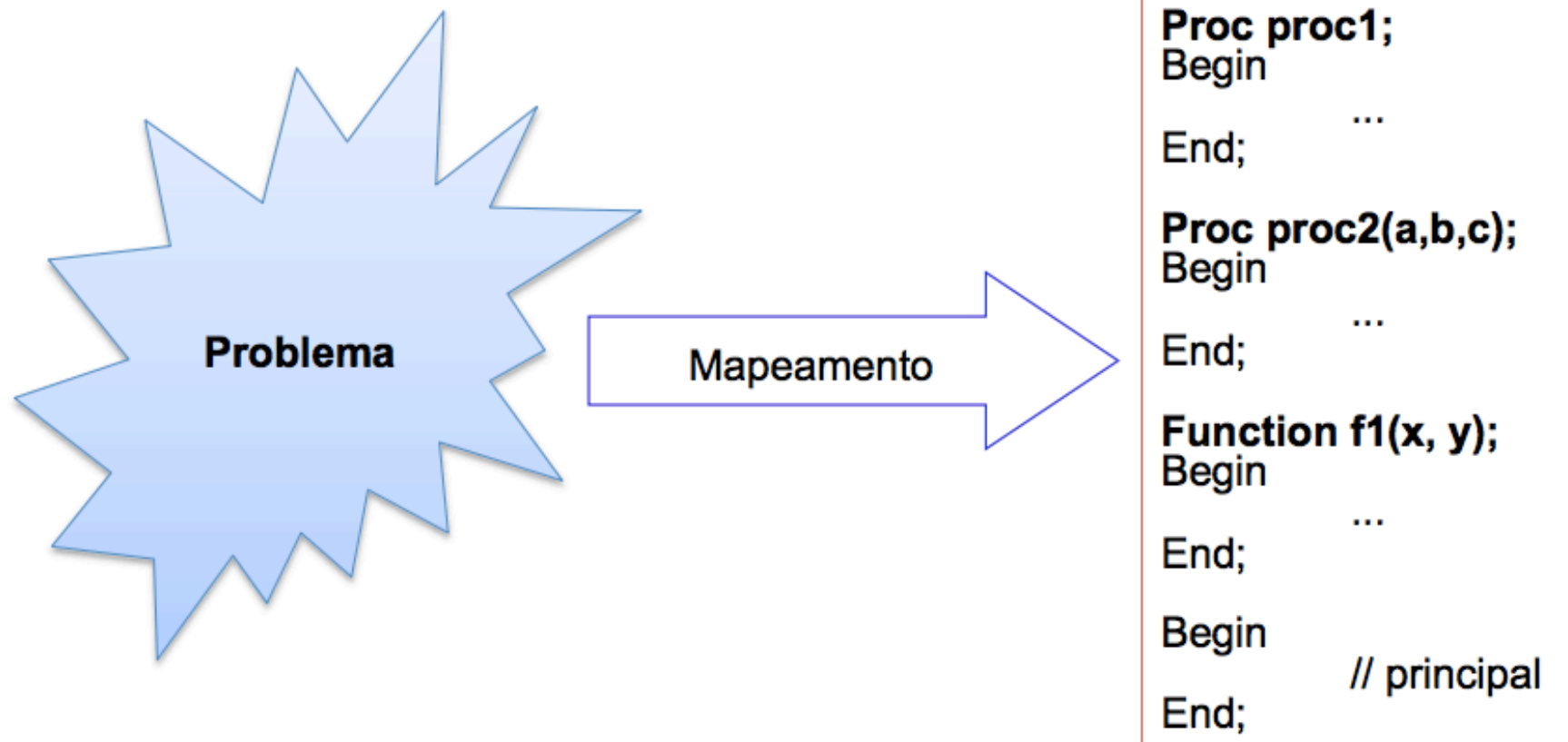
- Estrutura de sequência
- Estrutura com funções
- Estrutura de controle
- Estrutura de repetição

Antes da orientação a objetos

- Programação estruturada
 - Muito código desenvolvido;
 - Repetição de código em grande escala;
 - Dificuldades de manter e reutilizar código.

Antes da orientação a objetos

- Abstrações procedimentais



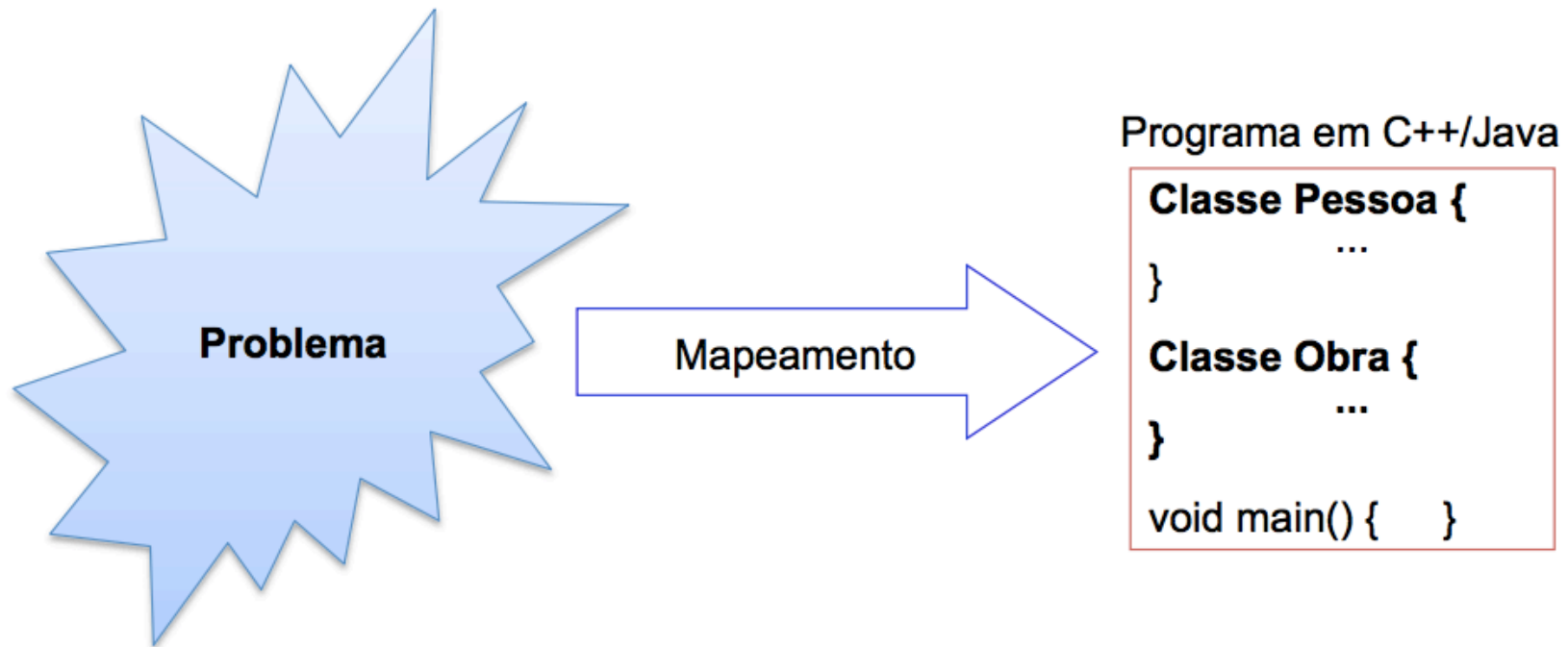
Programação Orientada a Objetos

VANTAGENS

- Facilita o mapeamento em código executável;
- Facilita manutenção e reuso de código;

Introdução

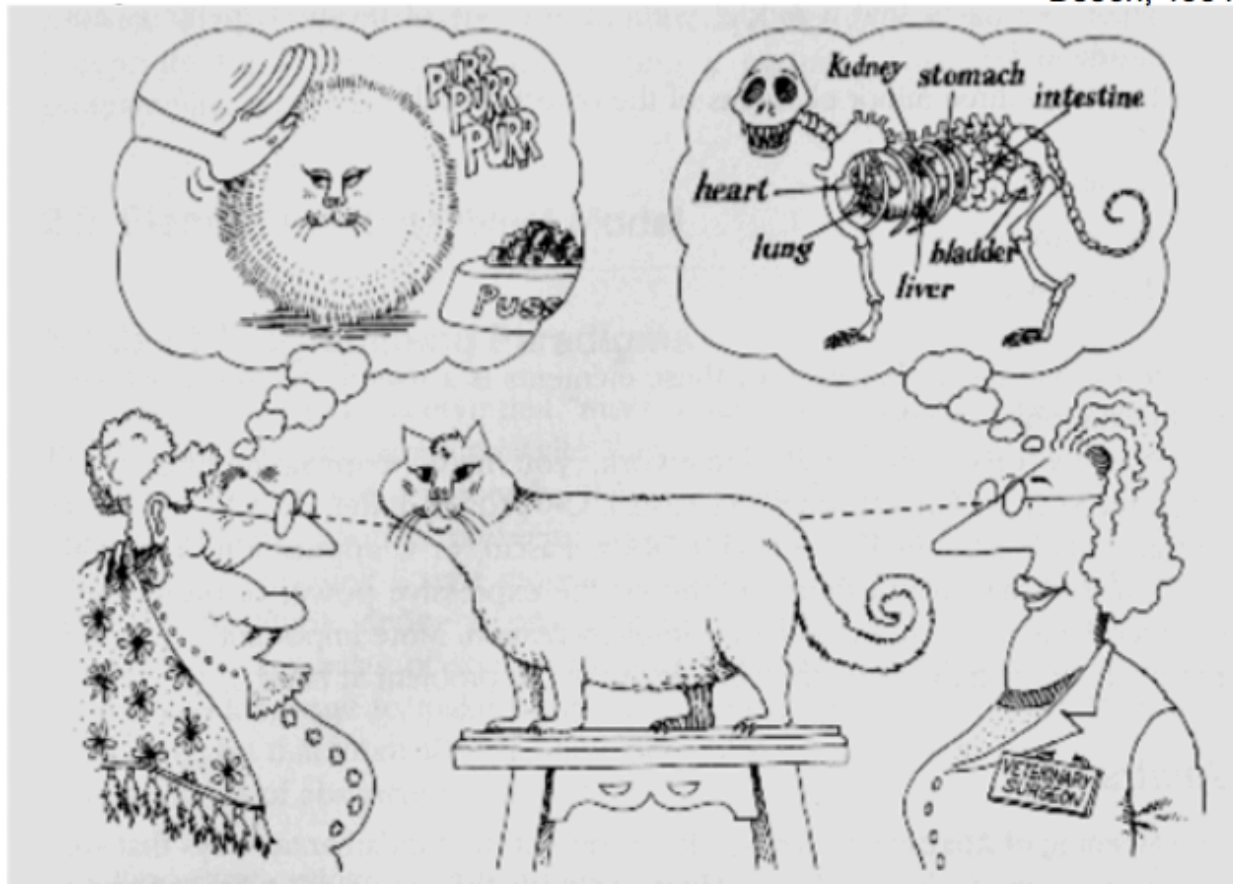
ABSTRAÇÃO



Introdução

ABSTRAÇÕES

Booch, 1991



Introdução

MECANISMOS DE ABSTRAÇÃO

- Classes;
- Métodos;
- Herança;
- Composição...

Introdução

CLASSE

- Definição de um conjunto de objetos que compartilham estrutura e comportamentos em comum;
- Objetos são criados por meio de classes;
- A abstração mais importante diz respeito aos dados;
- Teoria dos conjuntos é adotada como modelo semântico para definição dos dados.

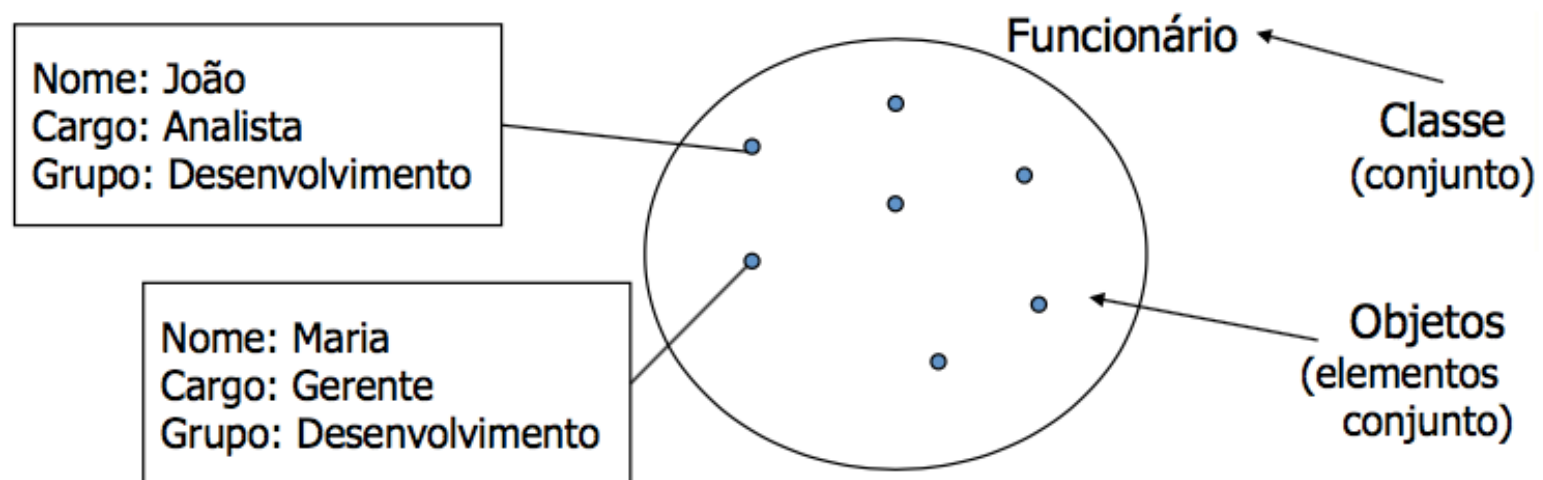
Introdução

CLASSE

- Uma classe implementa um tipo abstrato de dados
- Tipo abstrato de dados: definição de um tipo onde somente as operações de manipulação estão visíveis externamente.

Introdução

OBJETO



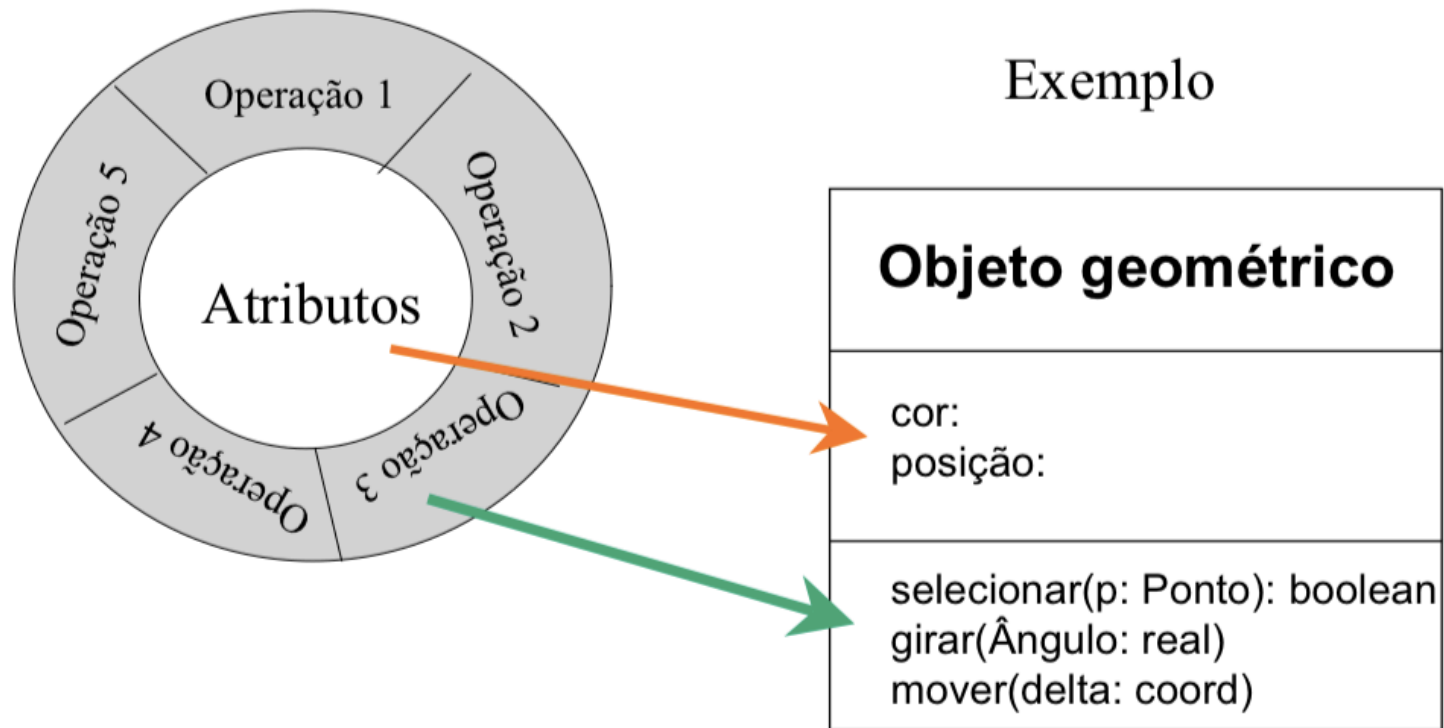
Introdução

OBJETO

- Um objeto é um elemento de uma classe;
 - Um objeto deve pertencer a uma classe.
 - O objeto é o elemento que efetivamente armazena as informações de um programa.
- Objetos trocam mensagens entre si;
 - O funcionamento de um programa OO é caracterizado pela troca de mensagens entre os objetos criados.

Introdução

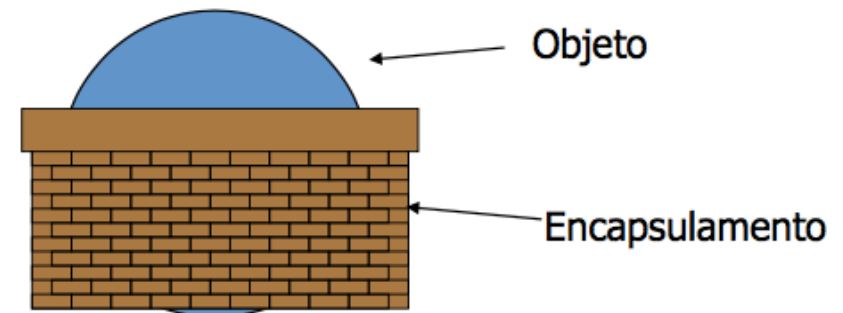
OBJETO



Introdução

ENCAPSULAMENTO

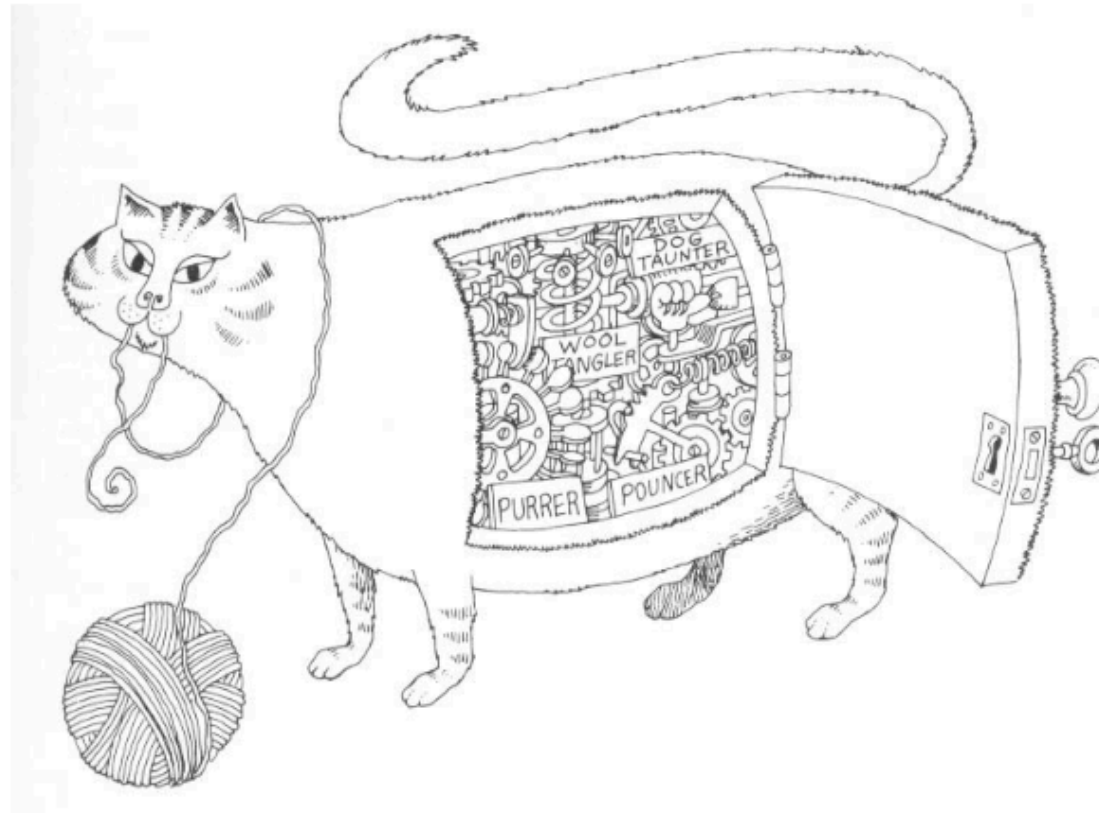
- Influência dos circuitos integrados;
- Podem ser livremente combinados;
- Não podem ser modificados.



Introdução

ENCAPSULAMENTO

Booch, 1991



Introdução

POLIMORFISMO

- É a habilidade de variáveis terem “mais de um tipo”. Funções são ditas polimórficas quando seus parâmetros podem ter mais de um tipo.

Hierarquia de classes

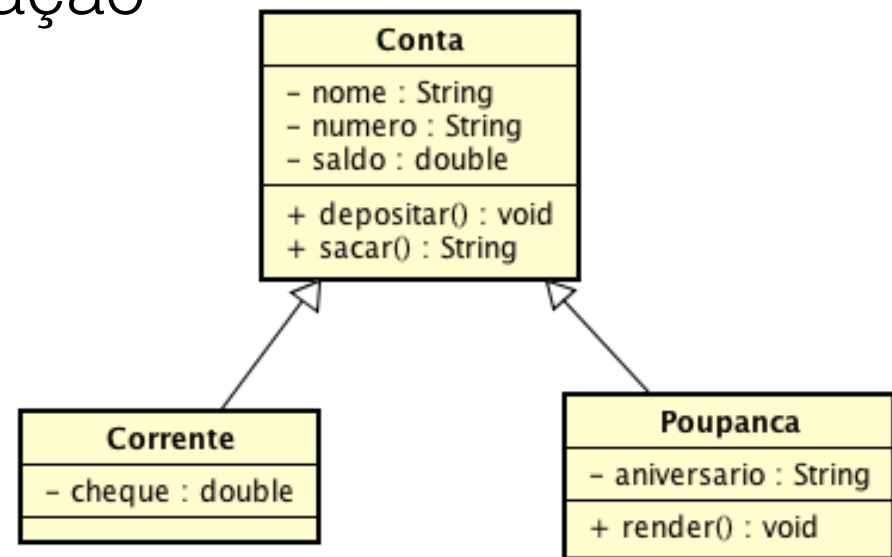
VISIBILIDADE

Notação	Modificador de acesso	A parte visível
+	public	Dentro da própria classe e para qualquer outra classe
–	private	Somente dentro da própria classe
#	protected	Somente dentro do próprio pacote e das subclasses em outros pacotes
~	package	Somente dentro da própria classe e das classes dentro do mesmo pacote

Hierarquia de classes

HERANÇA

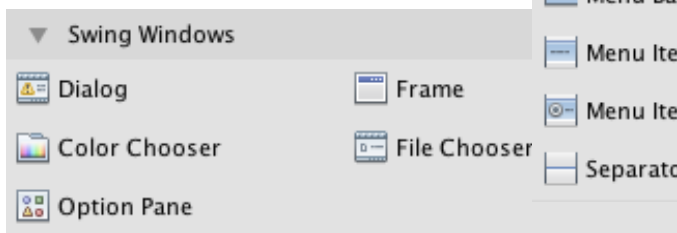
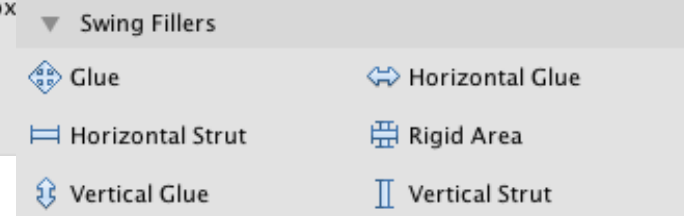
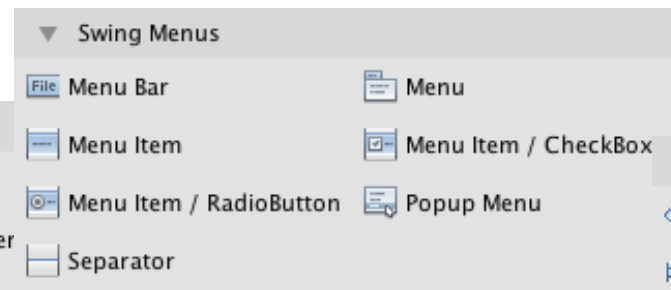
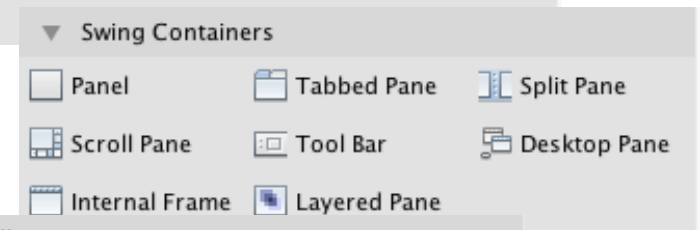
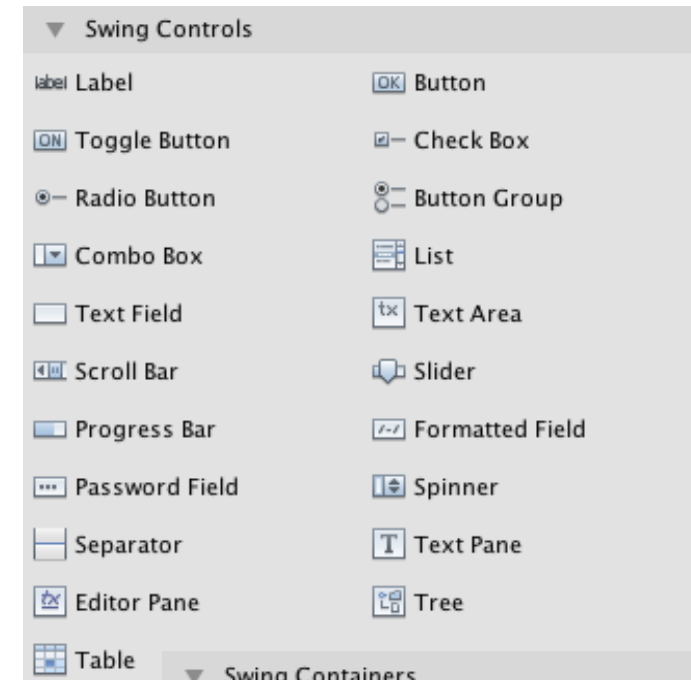
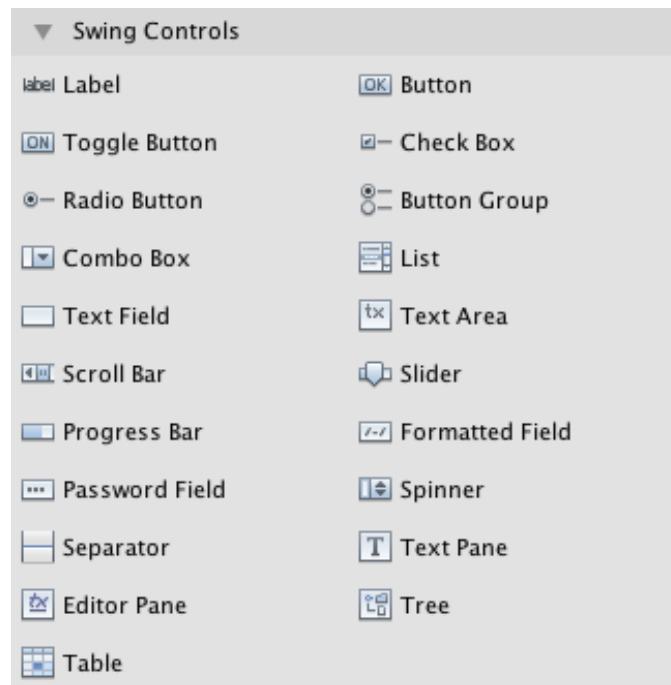
- Super classes / subclasses
- Generalização / especialização



Modelos de interfaces gráficas

DESENVOLVIMENTO DO SWING PARA GUI

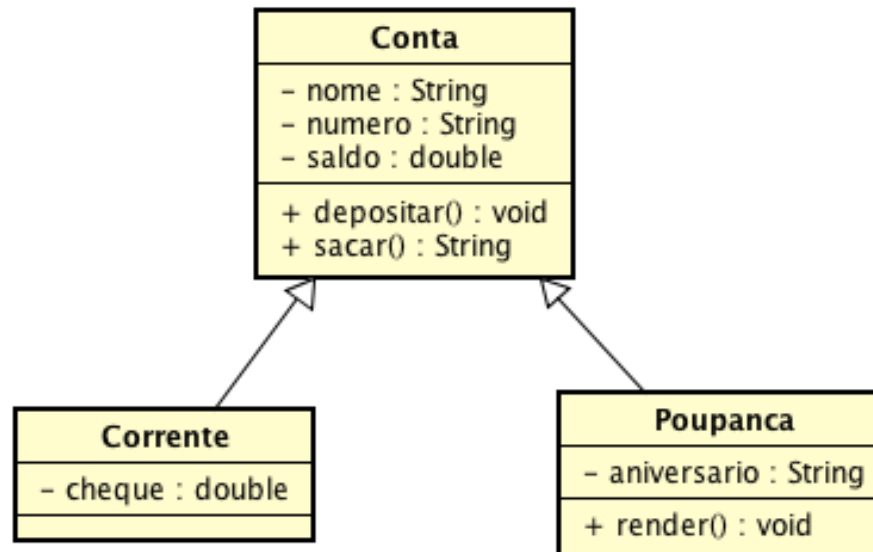
- JOptionPane
- JLabel
- JButton
- JComboBox
- List
- JPanel



Prática no laboratório

Exercício 1

- Utilizando recursos de interface em Java, desenvolva um sistema simples para controle de contas correntes e poupanças de acordo com o diagrama de classes abaixo:



Próxima aula

UNIDADE 1

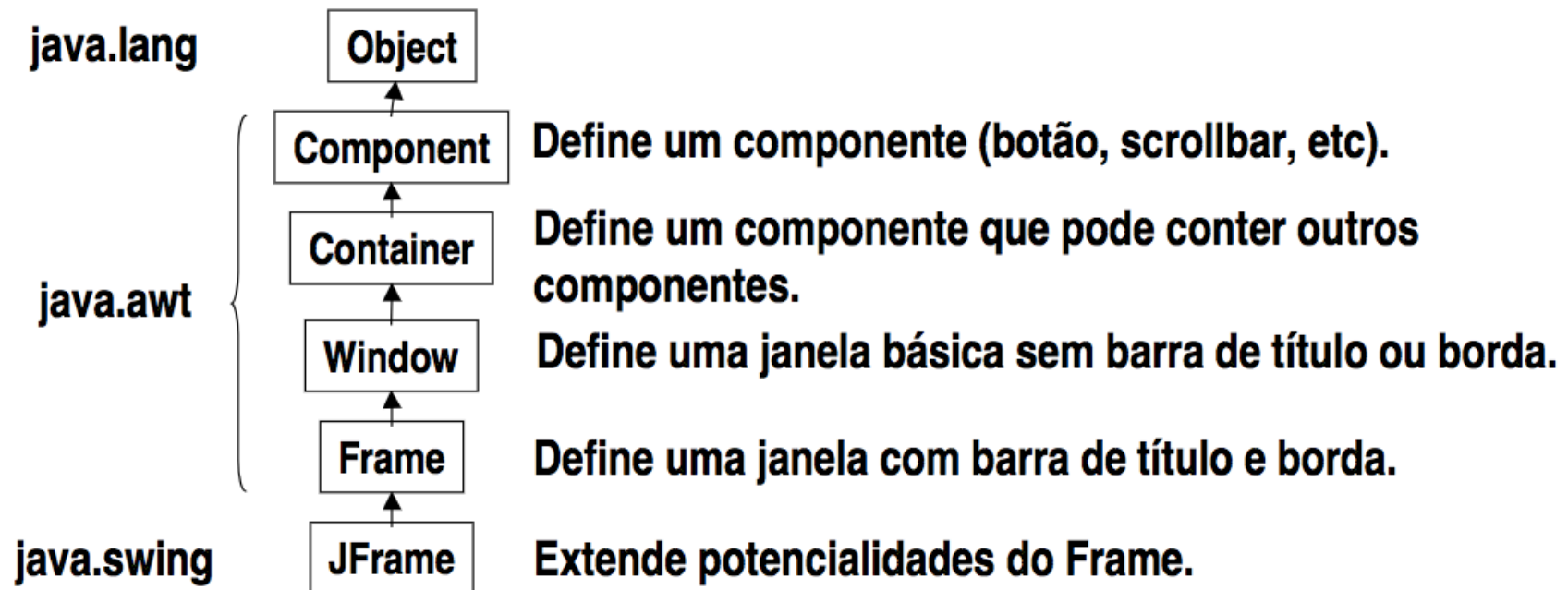
1. Criação de interfaces gráficas usando as JFC/Swing
 - 1.1. Introdução
 - 1.2. Hierarquia de classes
 - 1.3. Modelos de desenvolvimento de interfaces gráficas
 - 1.3.1.Desenvolvimento do SWING para GUI (cont.)
 - 1.3.2.Gerenciadores de layout
 - 1.3.3.Layouts compostos
 - 1.3.4.Manipulação de aspectos visuais
 - 1.3.5.Variações de componentes visuais

Exercício 2

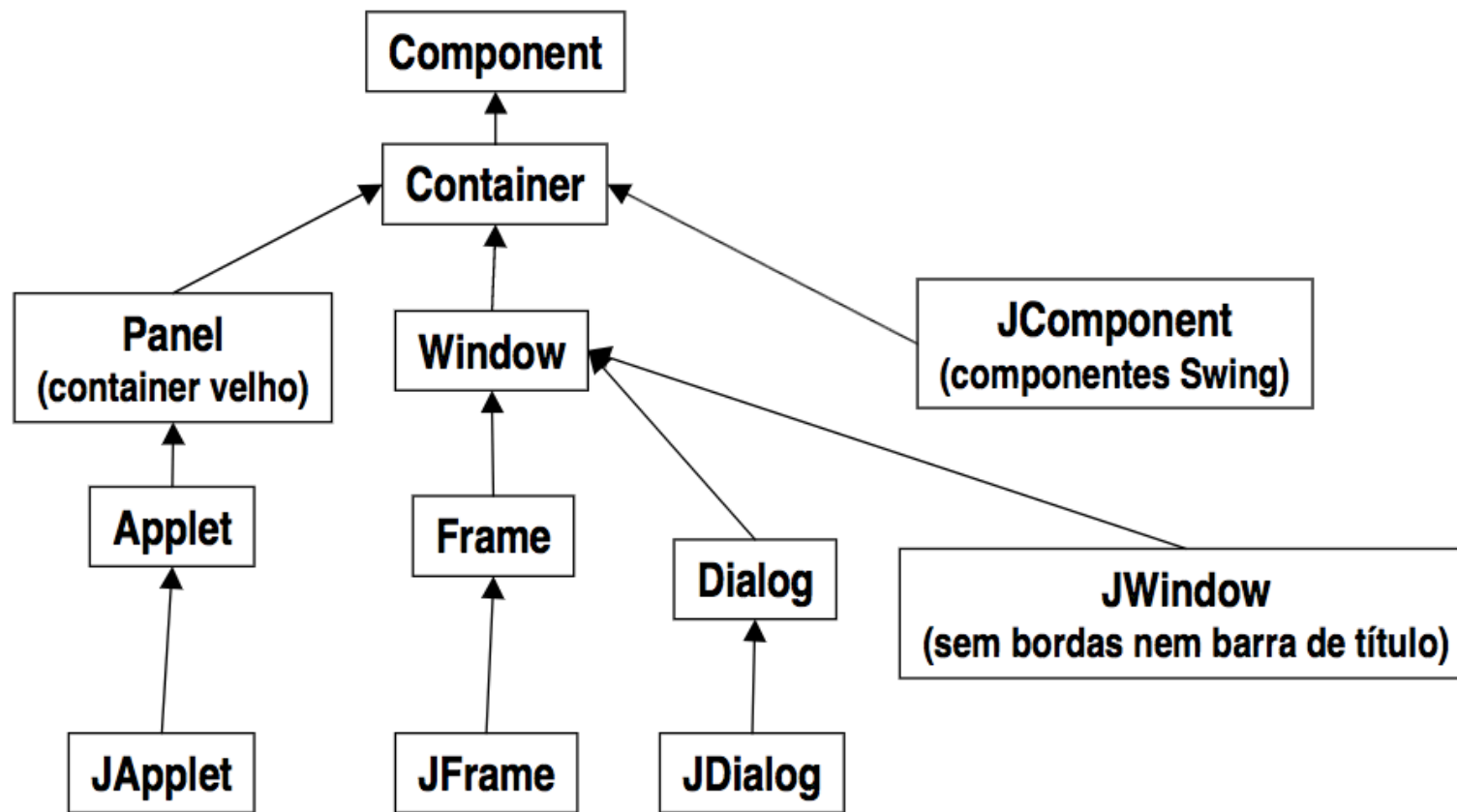
- Escreva um programa em java com interface que armazene os dados de cadastro de usuário, respeitando a modelagem abaixo.

Usuario
- nome : String - nascimento : String - email : String - telefone : String
+ add() : void + edit() : void + remove() : void + view() : String

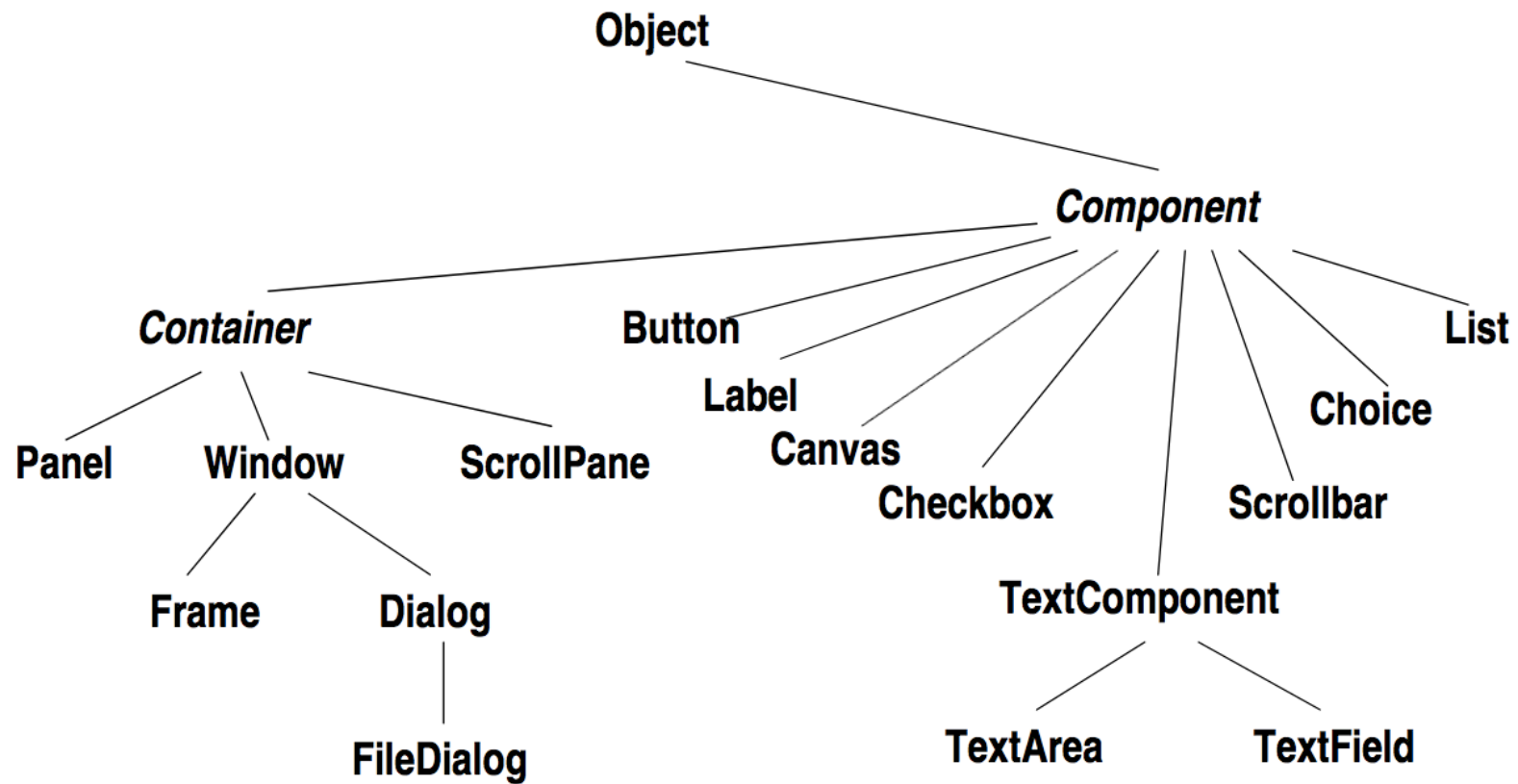
Criando uma janela



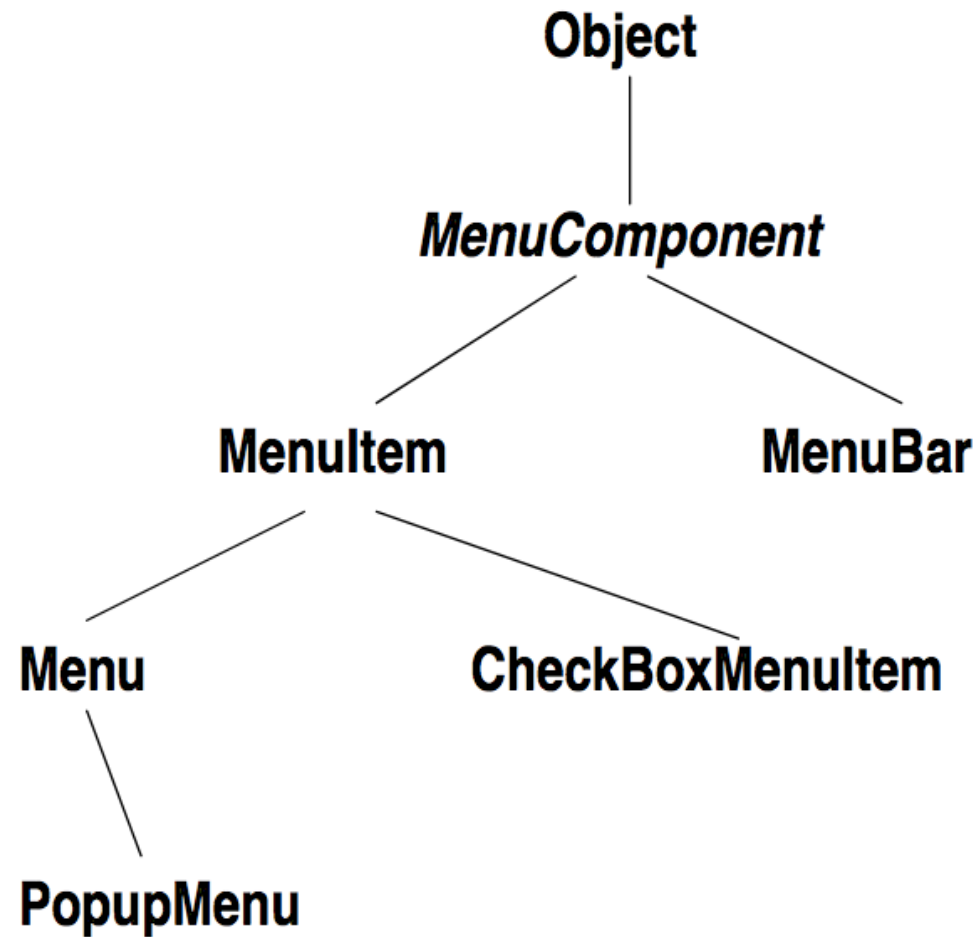
Criando uma janela



Classes AWT



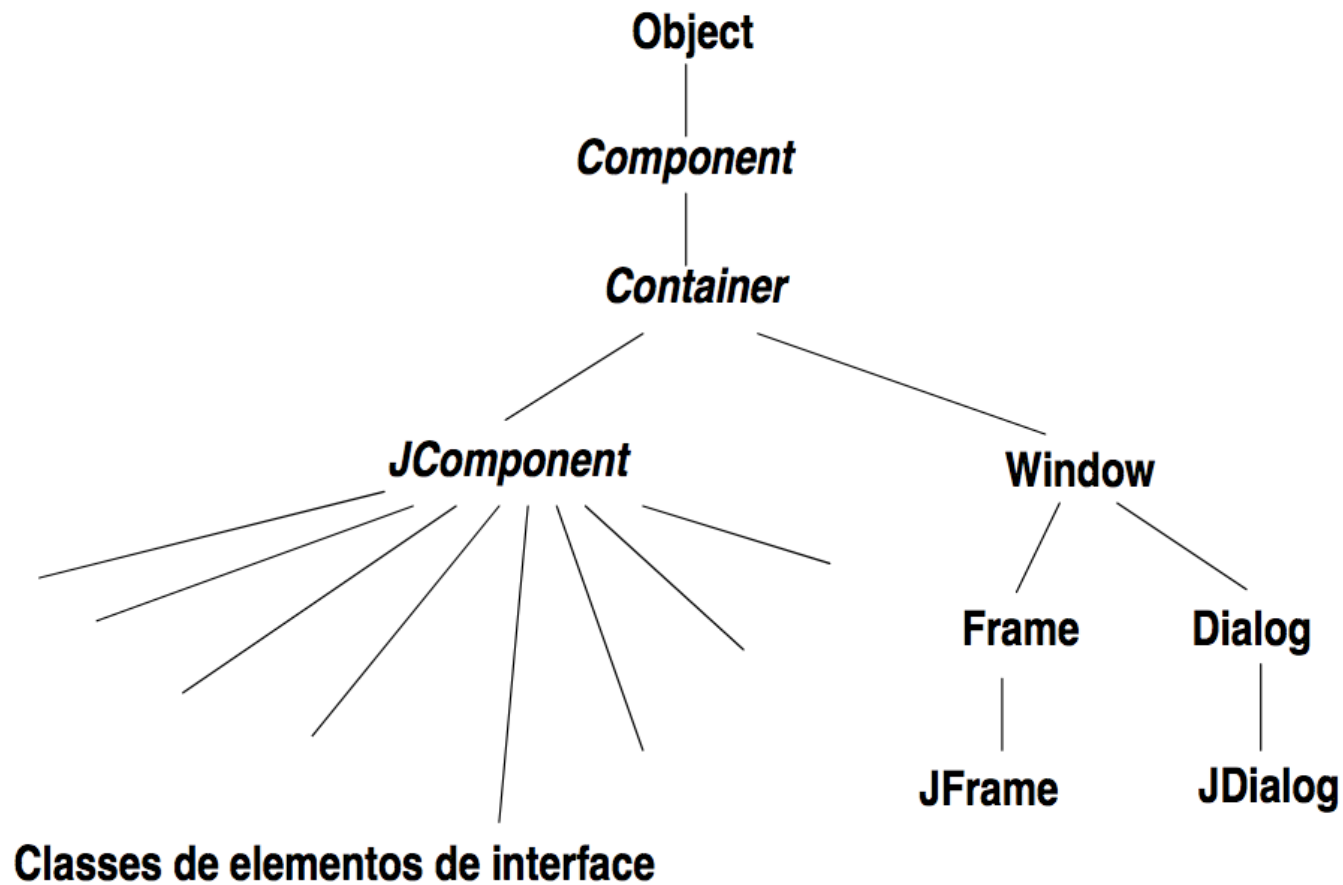
Classes AWT



Pacote Swing

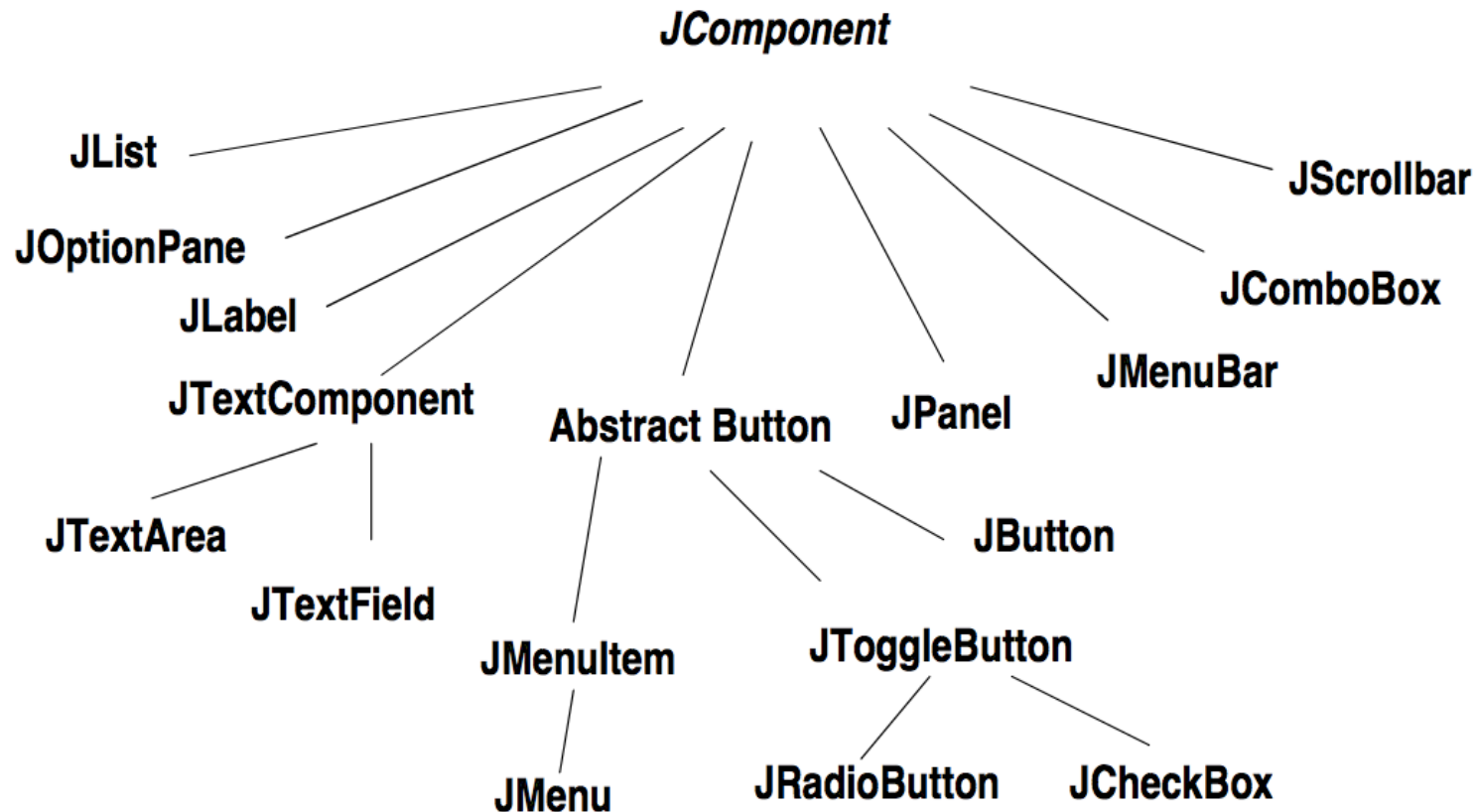
- Criado em 1997;
- Extensão da AWT (*Abstract Window Toolkit*);
- Classes implementadas inteiramente em java
- Mesma estrutura de componentes AWT
- Componentes fornecem melhores alternativas para implementação de interface gráfica:
 - JButton no lugar de Button
 - JFrame no lugar de Frame
- As classes de Swing fazem parte de um conjunto mais genérico de classes com capacidades gráficas: JFC (*JavaFoundation Classes*).

Classes Swing



Classes Swing

HIERARQUIA DE COMPONENTES



Interfaces

MANEIRAS DE PROGRAMAR

- Posicionamento absoluto;
- Gerenciadores de layout;
- Programação visual com IDE;

Gerenciadores de layout

- **FlowLayout**

- Os componentes são organizados de acordo com a ordem que são adicionados na caixa, sempre da esquerda para direita.

Ex.:

```
Panel c = new Panel( );  
c.add(new Button("1"));  
c.add(new TextField(9));  
c.add(new Button("dois"));  
c.add(new Button("três"));
```



```
layout = new FlowLayout();  
layout.setAlignment(FlowLayout.CENTER_ALIGNMENT);  
layout.setAlignment(FlowLayout.RIGHT_ALIGNMENT);  
layout.setAlignment(FlowLayout.LEFT_ALIGNMENT);  
layout.setAlignment(FlowLayout.TOP_ALIGNMENT);  
layout.setAlignment(FlowLayout.BOTTOM_ALIGNMENT);
```

Gerenciadores de layout

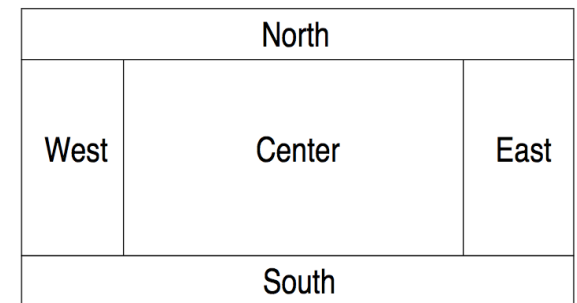
- BorderLayout

- Organiza os componentes na caixa dividida em cinco regiões:

- NORTH, SOUTH, EAST, WEST, CENTER

- Ex.:

```
layout = new BorderLayout();  
add(botoes[0], BorderLayout.NORTH);  
add(botoes[1], BorderLayout.SOUTH);  
add(botoes[2], BorderLayout.EAST);  
add(botoes[3], BorderLayout.WEST);  
add(botoes[4], BorderLayout.CENTER);
```

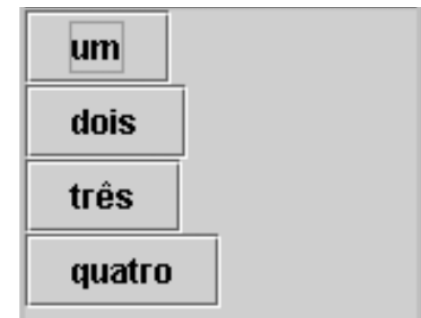


Gerenciadores de layout

- BorderLayout

- Respeita o tamanho dos componentes organizando em linha e coluna
- EX.:

```
Panel c =new JPanel();  
// TROCANDO O LAYOUT DO JPANEL  
c.setLayout(new BorderLayout(c, BorderLayout.Y_AXIS));  
// ADICIONANDO NAS POSIÇÕES DESEJADAS  
c.add(new JButton("um"));  
c.add(new JButton("dois"));  
c.add(new JButton("três"));  
c.add(new JButton("quatro"));
```



Gerenciadores de layout

- GridLayout

- Componentes alocados em grade (linhas e colunas)

- EX.:

```
c.setLayout(new GridLayout(2,2));  
c.add(new Button("um"));  
c.add(new TextField(5));  
c.add(new Button("dois"));  
c.add(new Button("três"));
```



```
// 2 POR 3; LACUNAS DE 5
```

```
gridLayout1 = new GridLayout(2, 3, 5, 5);
```

```
// 3 POR 2; NENHUMA LACUNA
```

```
gridLayout2 = new GridLayout(3,2);
```

```
// OBTÉM O PAINEL DE CONTEÚDO
```

```
container = getContentPane();
```

```
setLayout(gridLayout1);
```

Layouts compostos

COMPONDO LAYOUTS USANDO PANELS

- A classe *Panel* é derivada de *Container* e pode ser usada para agrupar *Components* de maneira a criar *Layouts* compostos.
- Por ser também um *Container*, um *Panel* pode ter seu *Layout* próprio.

Manipulação de aspectos visuais

PRÁTICA NO LABORATÓRIO

Variações de componentes visuais

PRÁTICA NO LABORATÓRIO

Códigos

<https://github.com/cassiodiego/CCT0418>

Próxima aula

UNIDADE 2

2. Tratamento de eventos para interfaces gráficas

2.1. Uma forma de deletar tratamentos de eventos

2.2. Manipulação de eventos

2.2.1.Eventos comuns

2.2.2.Eventos de janelas

2.2.3.Eventos de botões e menus

2.2.4.Eventos de textos

2.2.5.Eventos de listas

2.2.6.Eventos de combos

2.2.7.Eventos de tabelas

2.2.8.Inserção de teclas de atalho

Referências

- DEITEL, H. M.; DEITEL, P. J.. Java: Como programar. 8a. ed. Rio de Janeiro: Pearson, 2010.
- FLANAGAN, David . Java: O guia essencial. 5a. ed. Rio de Janeiro: Bookman, 2006.
- SANTOS, Fabiano. Linguagens de programação. Rio de janeiro: SESES, 2015.