Aplicações de Linguagem de Programação Orientada a Objeto

Introdução e AWT

Professora Sheila Cáceres

Lembrando...

Introdução a Orientação a Objetos

- No mundo real, tudo é objeto!;
- Os objetos se relacionam entre si de diversas maneiras;
- Um programa orientado a objetos é estruturado como uma comunidade de agentes que interagem entre si, denominados **objetos.;**
- Cada objeto tem um papel a cumprir;
- Cada objeto oferece um serviço ou realiza uma ação que é usada por outros membros da comunidade;

Exemplo real: montagem de um computador


PCB MADE IN CHINA
GIGABYTE

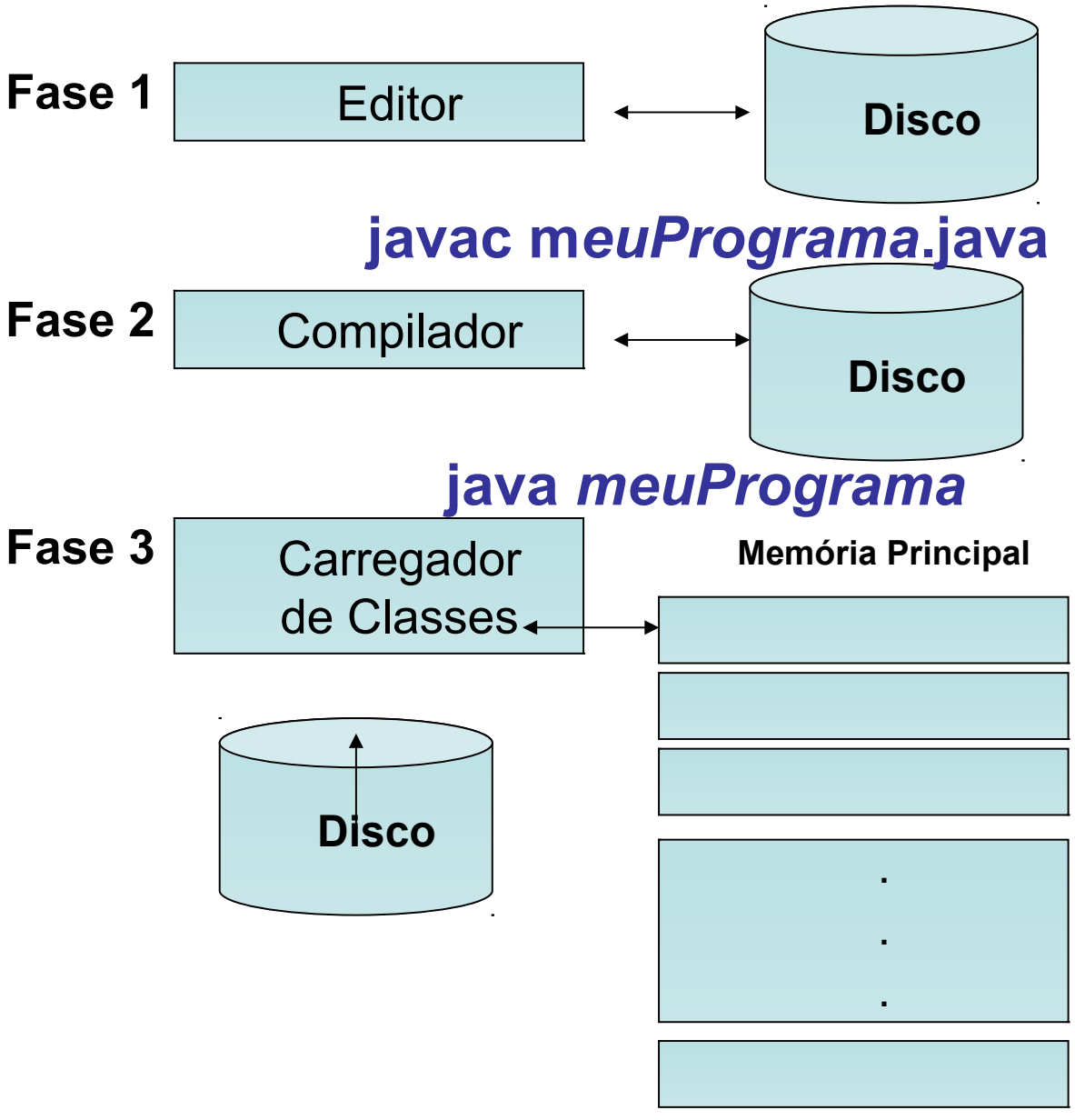

intel
Core 2 Duo


ELITEGROUP

O que você entende por essas figuras???

Se você usar a OO, o que você entende por essas figuras??

Processo



O programa é criado no editor e armazenado em disco.

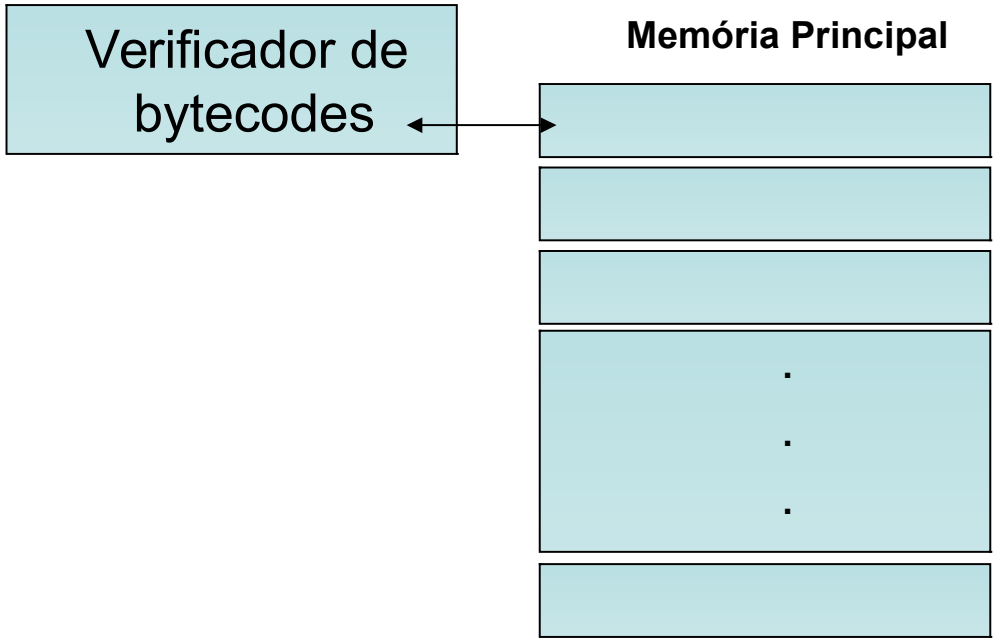
O compilador cria bytecodes (arquivo .class) e os armazena em disco.

O carregador de classe coloca bytecodes na memória.

Fonte: Deitel & Deitel, 2003.

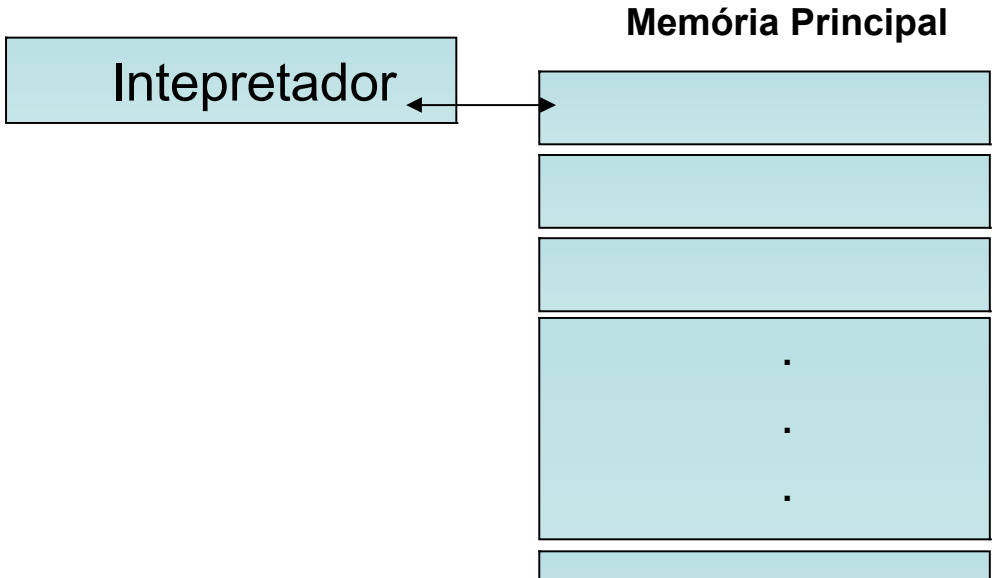
Processo

Fase 4



O verificador de bytecodes confirma que todos os bytecodes são válidos e não violam restrições de segurança do Java

Fase 5



O interpretador lê os bytecodes e os traduz para uma linguagem que o computador pode entender, possivelmente armazenando valores dos dados enquanto executa o programa.

Fonte: Deitel & Deitel, 2003.

Processo

javac *MeuPrimeiroPrograma*.java

- Compila o arquivo .java, gerando o arquivo .class, que contém o bytecode da classe.

java *MeuPrimeiroPrograma*

- Executa o programa Java

start java *MeuPrimeiroPrograma*

- Executa o programa Java em uma nova janela

set CLASSPATH=*C:\dir1;C:\dir2;.*

- Define os locais onde a JVM vai procurar por pacotes e classes Java

Abstract Windowing toolkit
AWT

AWT

- Presente desde a primeira versão de Java para criar interfaces gráficas.
- Baseada nos componentes nativos do sistema.
- Foi posteriormente substituida por Swing.
- Pode ser usada adicionando:

```
import java.awt.*;
```

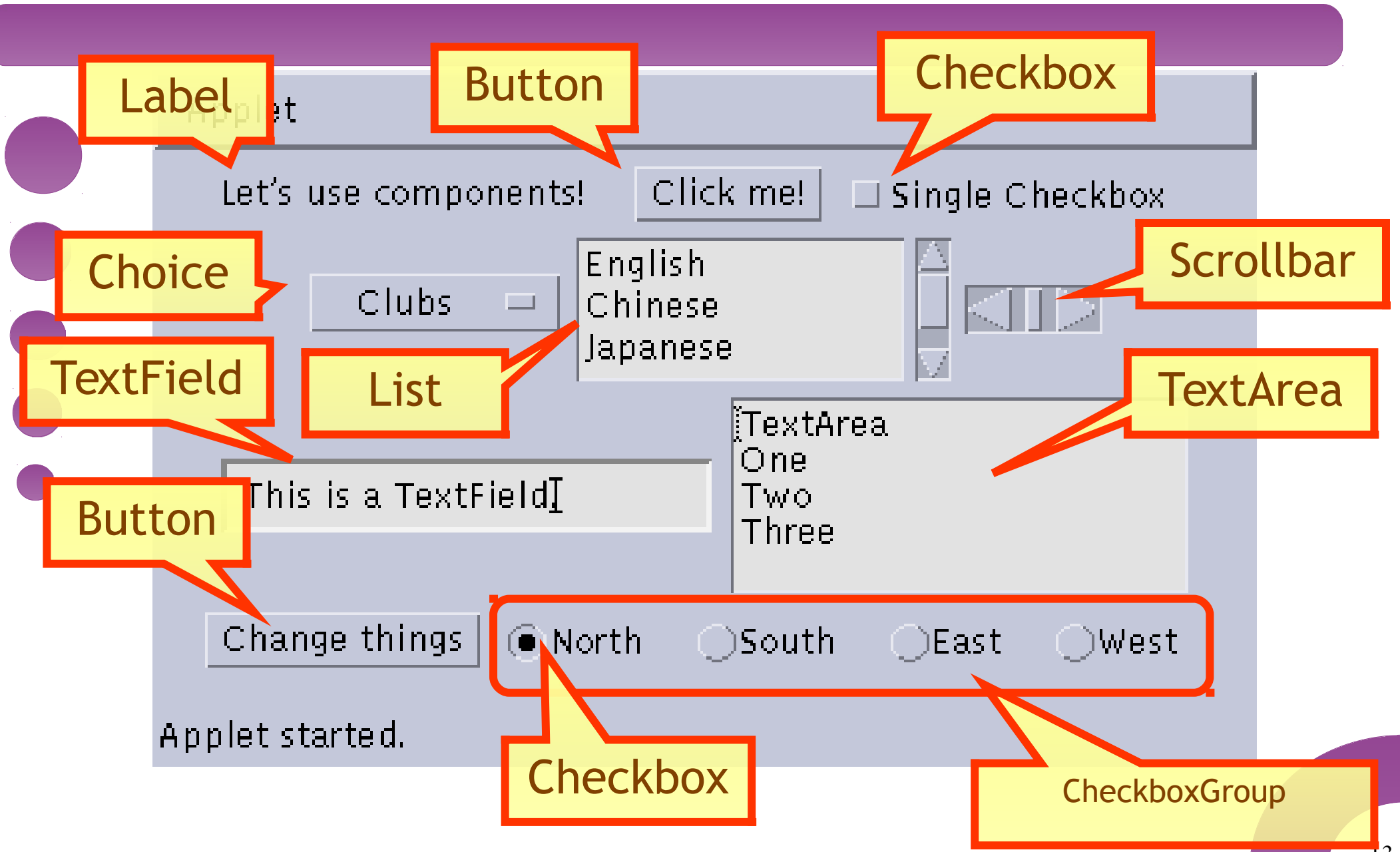
Componentes (classe Component)

- Os componentes são a especie de objetos que podem formar parte da nossa interface.
- Os componentes tem que estar situados obrigatoriamente em um contenedor de componentes (container).
- Algumas subclasses comuns de Component são Button, Checkbox, Label, Scrollbar, TextField, e TextArea

Containers (class Container)

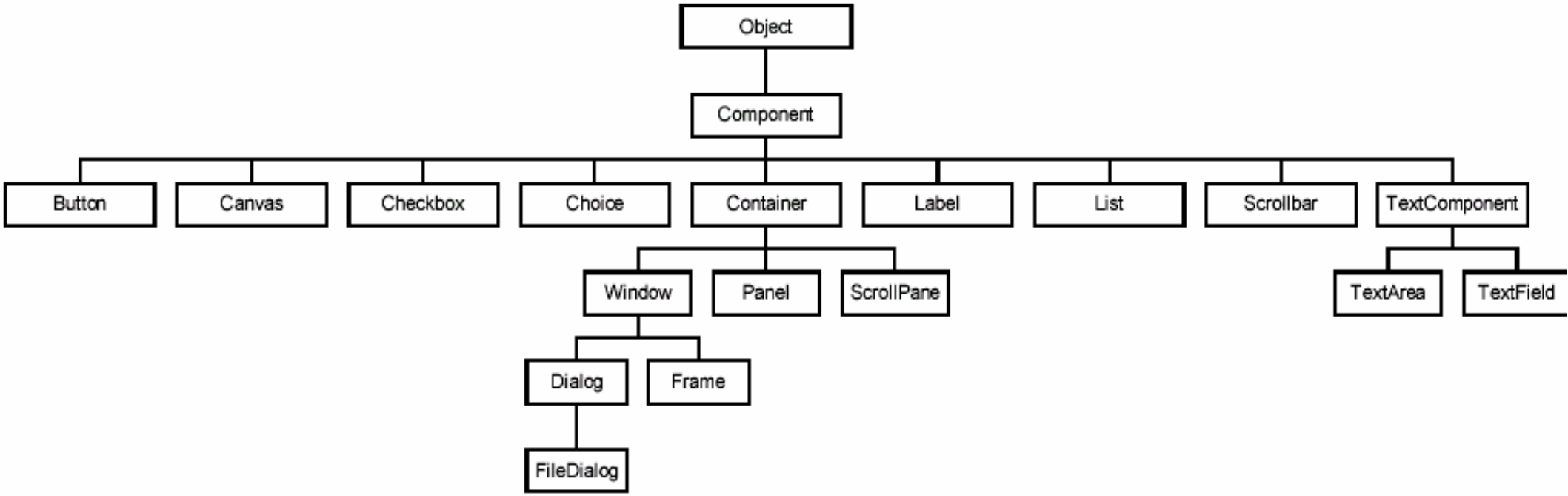
- Um container serve para conter e mostrar componentes.
- Fornece métodos para adicionar e remover componentes e para trabalhar com layout.
- Um container pode conter outros containers porque todo container é um componente.
- Os componentes tem que estar situados obrigatoriamente em um container.
- Todos os componentes exceto os contenedores de mais alto nivel (Windows e descendentes) precisam estar dentro de um contenedor.
- Algumas subclasses de Container são Panel (e Applet), Window, e Frame
- Para adicionar um componente a um contenedor usamos o método add() da classe Container.
  - nomeContainer.add(nomeComponente);

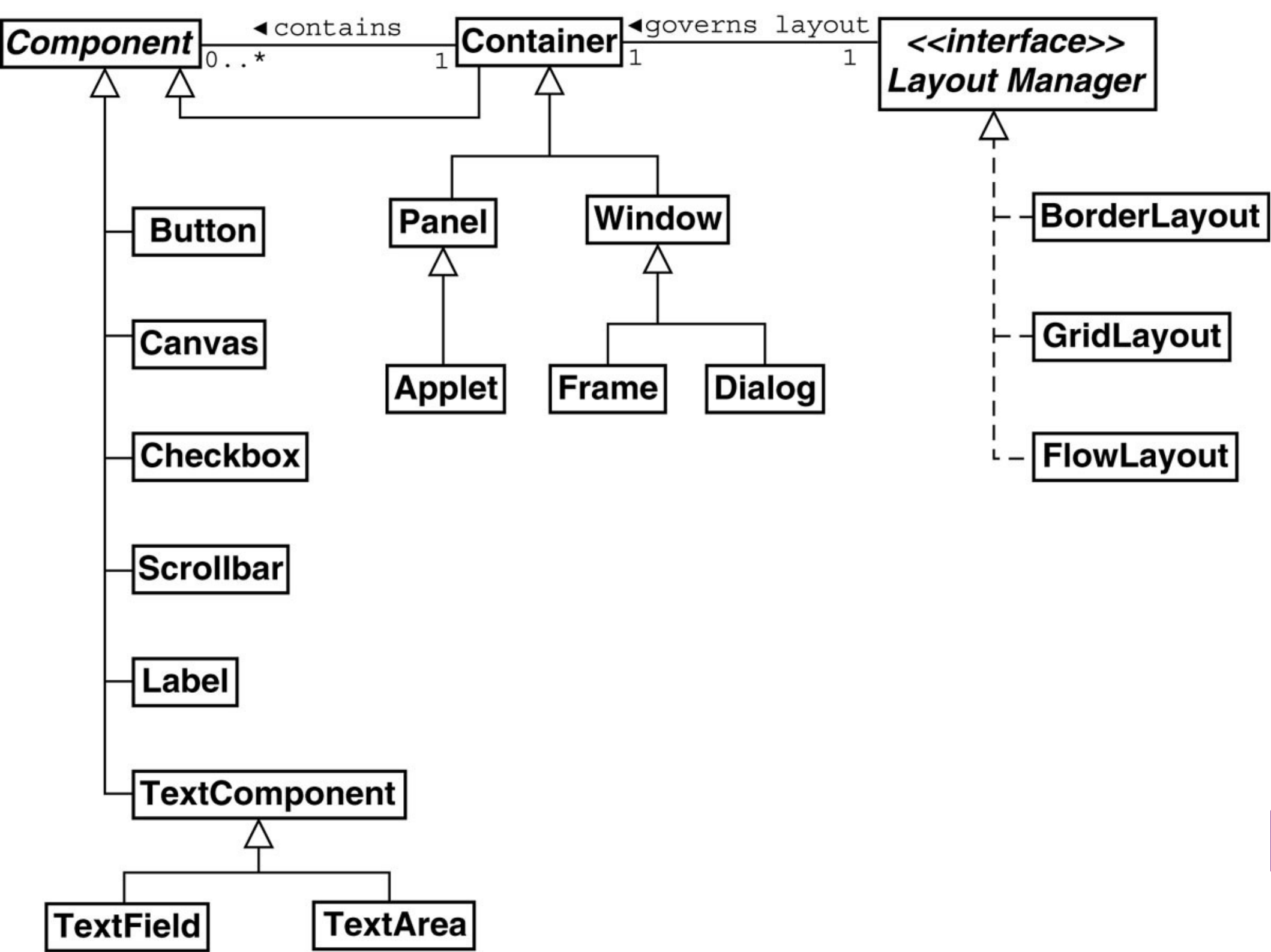
Alguns tipos de Componentes



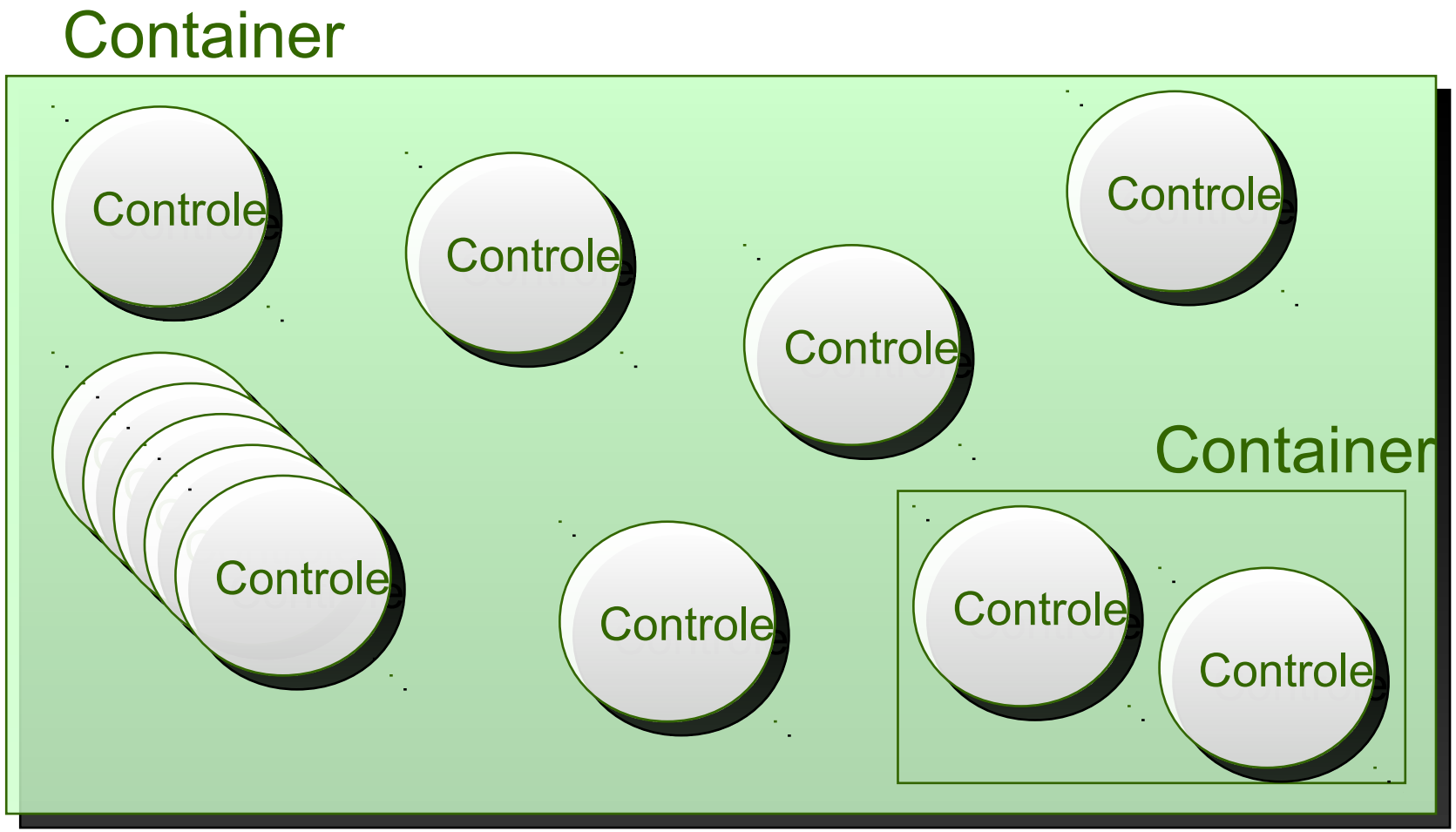
Hierarquia de Classes

- Todos os componentes de AWT são objetos que pertencem à hierarquia de classes a seguir:

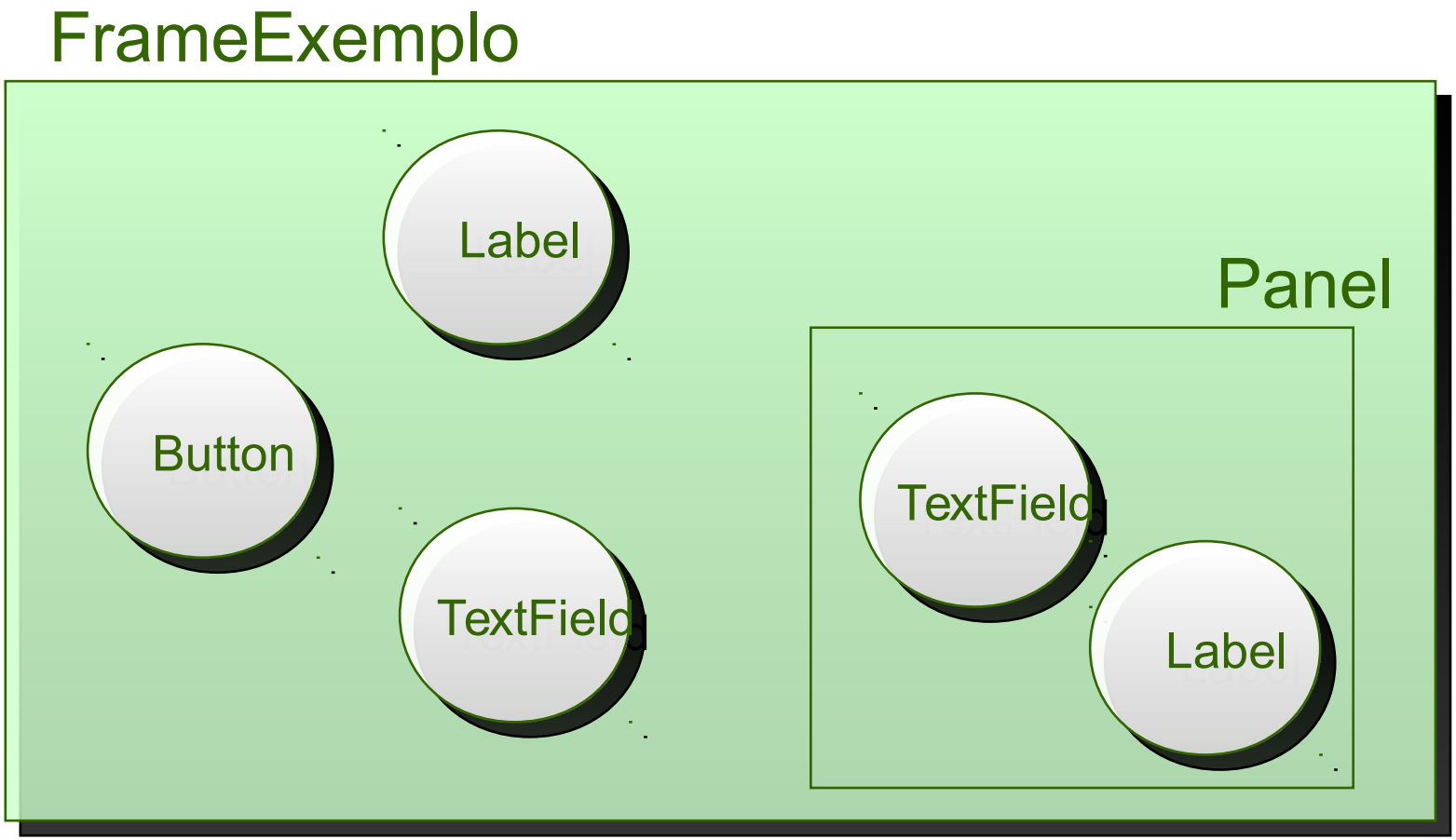



Component
contains
0..*
1
Container
governs layout
1
1
<<interface>>
Layout Manager
Button
Canvas
Checkbox
Scrollbar
Label
TextComponent
TextField
TextArea
Panel
Window
Applet
Frame
Dialog
BorderLayout
GridLayout
FlowLayout

Container e Controles



Container e Controles



Containers Concretos de java.awt

- Panel
  - Representa um grupo de elementos
  - Deve ser incluído em outro *container*
  - Usado para estruturar a interface
- Frame
  - Estende java.awt.Window
  - Representa uma janela
  - Possui título e borda
  - Pode possuir menu
- Dialog

Construindo uma aplicação

- Criar um contenedor para mostrar a interface, usualmente um Frame ou Dialog (aplicação Desktop), ou um Applet
- Cria alguns Components como: buttons, text areas, paneles, etc.
- Adicionar os componentes ao contenedor.
- Distribui, ou "*lay out*", os componentes
- Adicionar Listeners aos componentes
  - Ao iteragir com os componentes se origina um Evento.
  - Um Listener é ativado quando um evento acontece e executa código para lidar com ele.

Construindo uma aplicação

O container básico para criar a nossa interface pode ser criado de diversas maneiras:

- No main do programa (exemplo abaixo).
- Como atributo em uma classe nova (composição).
- Herdando de uma classe container.

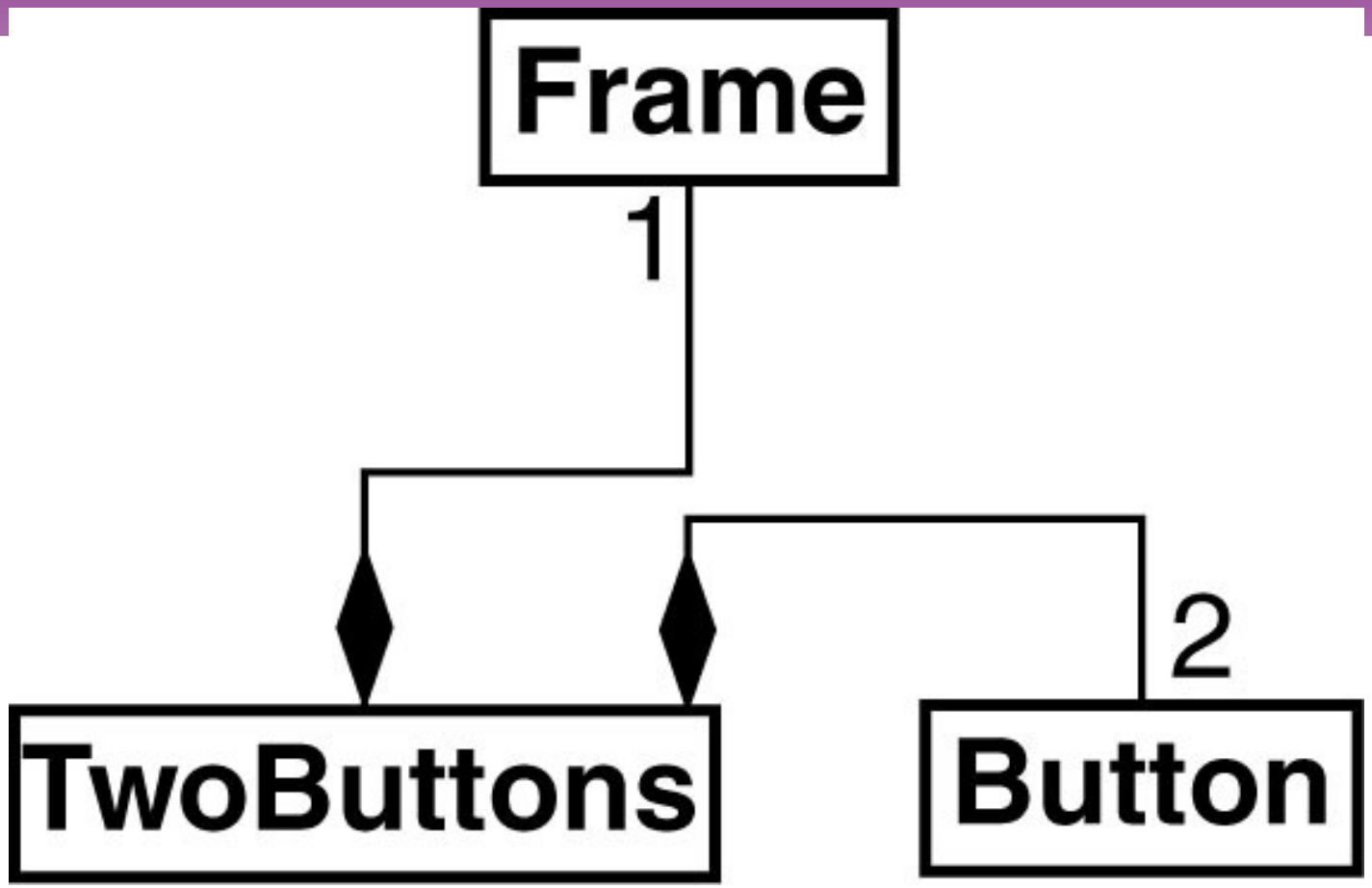
```
import java.awt.*;
public class Mundo {
  public static void main(String[] args) {
    Frame janela = new Frame("Mundo");
    Label mensagem = new Label("Olá Mundo!");
    janela.add(mensagem);
    janela.pack();
    janela.setVisible(true);
  }
}
```

Exemplo de Frame (Composição)

```
public class TwoButtons {
  Frame f;
  Button redButton, blueButton;

  public TwoButtttons() {
    f = new Frame("Two Buttons Frame");
    redButton = new Button("Red");
    blueButton = new Button("Blue");
    f.setLayout(new Flowlayout());
    f.add(redButton);
    f.add(blueButton);
    f.pack();
    f.setVisible(true);
  }
}
```

Exemplo de Frame (Composição)



Notação UML

- The filled diamond represents *composition*
- This shows that the class TwoButtons contains a class Frame which contains a class Buttons
- A good UML diagram only shows the classes and associations that are important to understanding the architecture
- This UML diagram does not include super classes like `Component` or `Container`

Using Inheritance

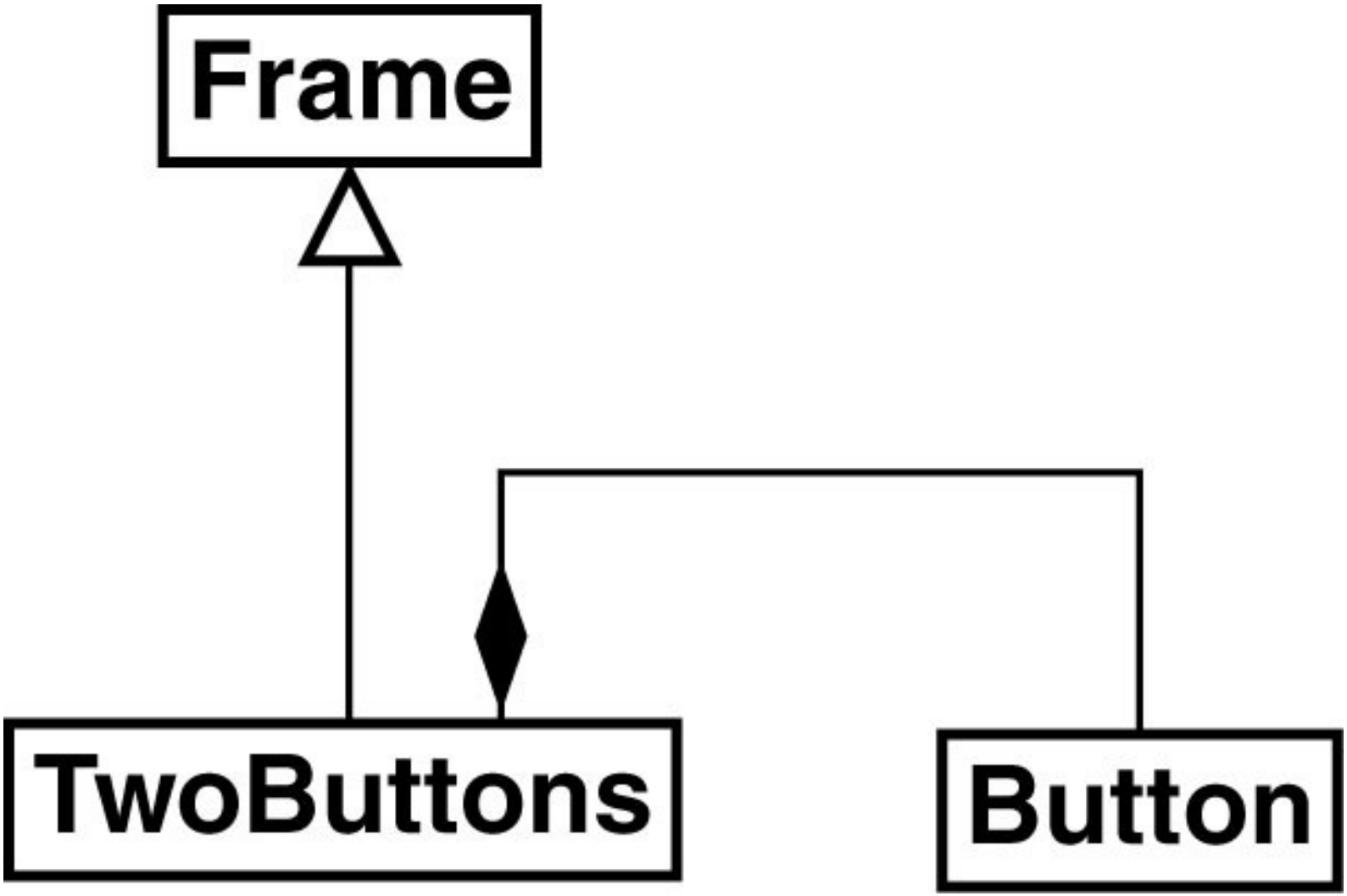
- It is possible to use inheritance to allow `TwoButtons` to become a frame in its own right
- Note that the UML diagram will show `TwoButtons` as a subclass of `Frame`
- You will need to use `super()` to set the frame title

Exemplo de Frame (Herança)

```
public class TwoButtons extends Frame {
  Button redButton, blueButton;

  public TwoButtttons() {
    super("Two Buttons Frame");
    redButton = new Button("Red");
    blueButton = new Button("Blue");
    f.setLayout(new Flowlayout());
    f.add(redButton);
    f.add(blueButton);
    f.pack();
    f.setVisible(true);
  }
}
```

Exemplo de Frame (Herança)



Bibliografia

Algumas Figuras e/ou slides foram extraídos do material apresentado a seguir:

- Slides do prof Paulo Borba, Orientação a Objetos em Java, Ufpe.
- Slides from S.N. Kamin, D. Mickunas, E. Reingold, “An Introduction to Computer Science Using Java (2nd Edition)”.