

# *1. Interfaces*

Linguagem de Programação II

<https://github.com/fsantanna-uerj/LP2/>

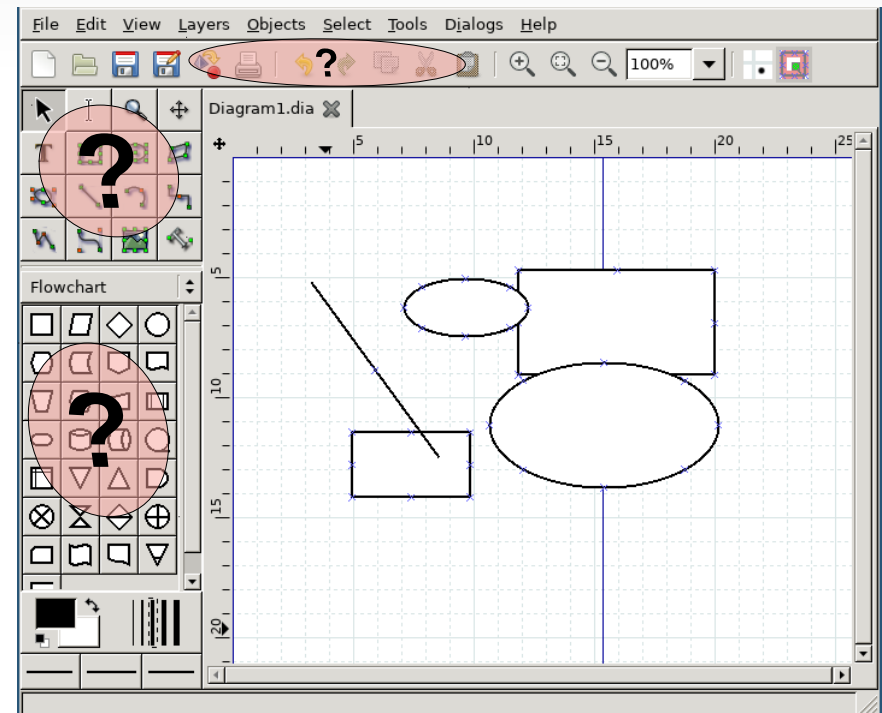
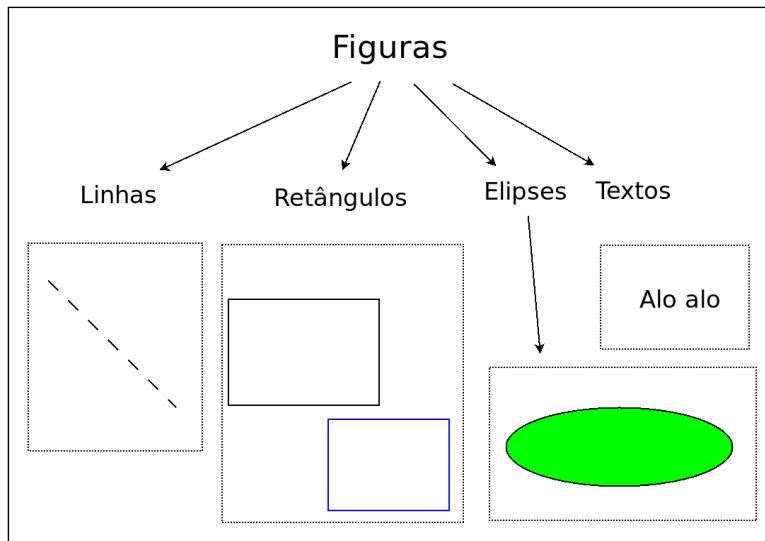
Francisco Sant'Anna

francisco@ime.uerj.br



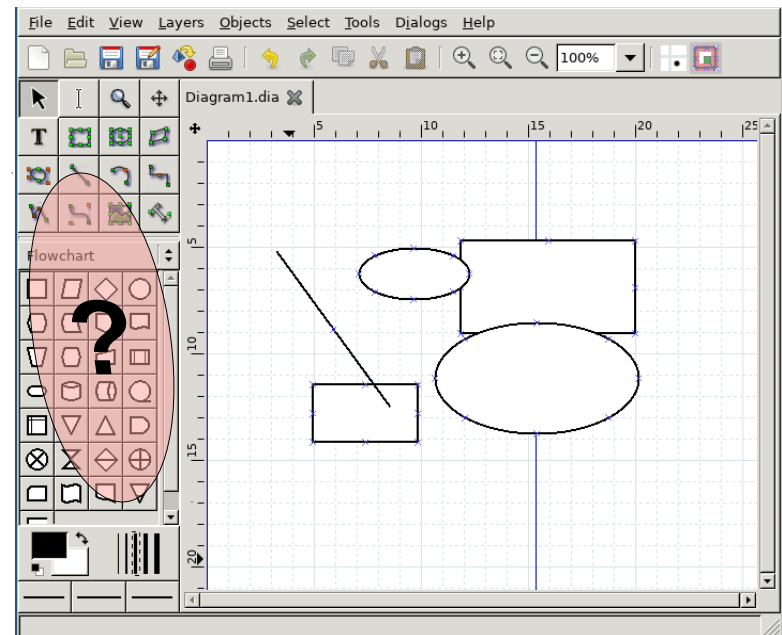
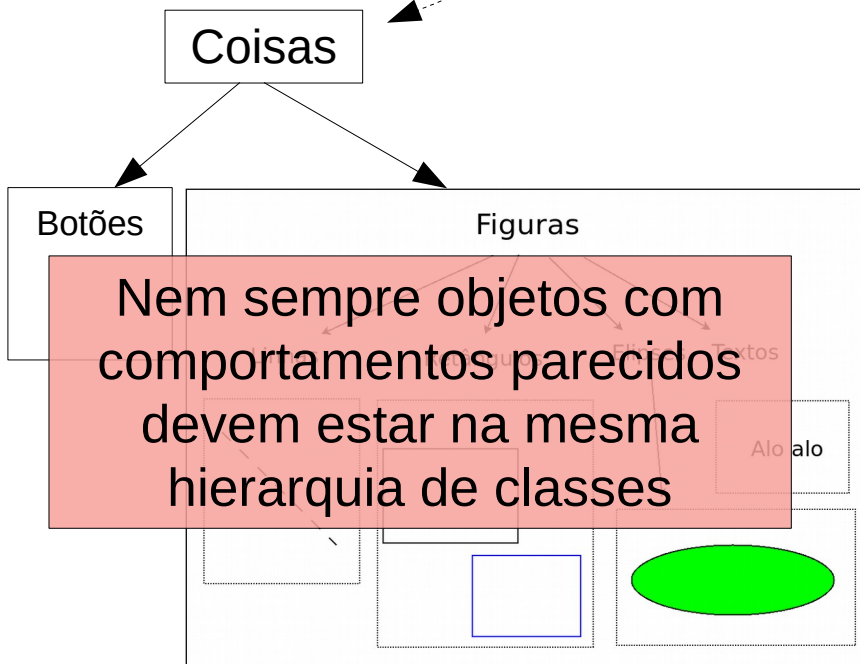
# Hierarquia de Classes

- Figura
  - Retângulo, Elipse, Texto, etc.
  - x, y, w, h, color, paint, click, move, resize, etc



# Hierarquia de Classes

- Figura
  - Retângulo, Elipse, Texto, etc.
  - x, y, w, h, color, paint, click, move, resize, etc



# Interfaces vs Classes

- Foco na interação e não nas propriedades:
  - “*Objetos que posso desenhar e clicar.*”
    - Interações: paint, click (*interface de métodos*)
- Classes – substantivos – *is a* – (x **is a** Figure)
  - Figura, Frame, Main
- Interfaces – adjetivos – *behave as* – (x **behave as** Visible)
  - Visível, Comparável, Executável

# *1. Interfaces*

Linguagem de Programação II

<https://github.com/fsantanna-uerj/LP2/>

Francisco Sant'Anna

francisco@ime.uerj.br



## *2. Exemplos*

Linguagem de Programação II

<https://github.com/fsantanna-uerj/LP2/>

Francisco Sant'Anna

[francisco@ime.uerj.br](mailto:francisco@ime.uerj.br)



# Interfaces



`chutar()`  
`segurar()`



`plainar()`  
`pousar()`



`encher()`  
`esvaziar()`



# Interfaces



“Chutável”:  
chutar()  
agarrar()

```
interface Kickable {  
    void kick (int vel);  
    int grab (void);  
}
```



“Voável”:  
plainar()  
pousar()

```
interface Flyable {  
    void plane (int t);  
    void land (void);  
}
```



“Enchível”:  
encher()  
esvaziar()

```
interface Fillable {  
    void fill (int ml);  
    void drain (int ml);  
}
```



# Interfaces



```
interface Kickable {  
    void kick (int vel);  
    int grab (void);  
}
```

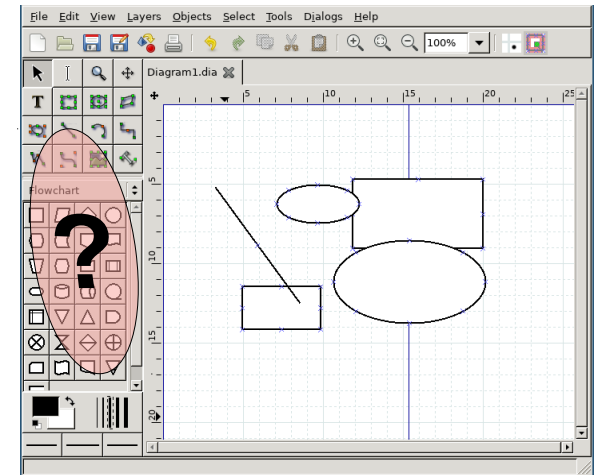
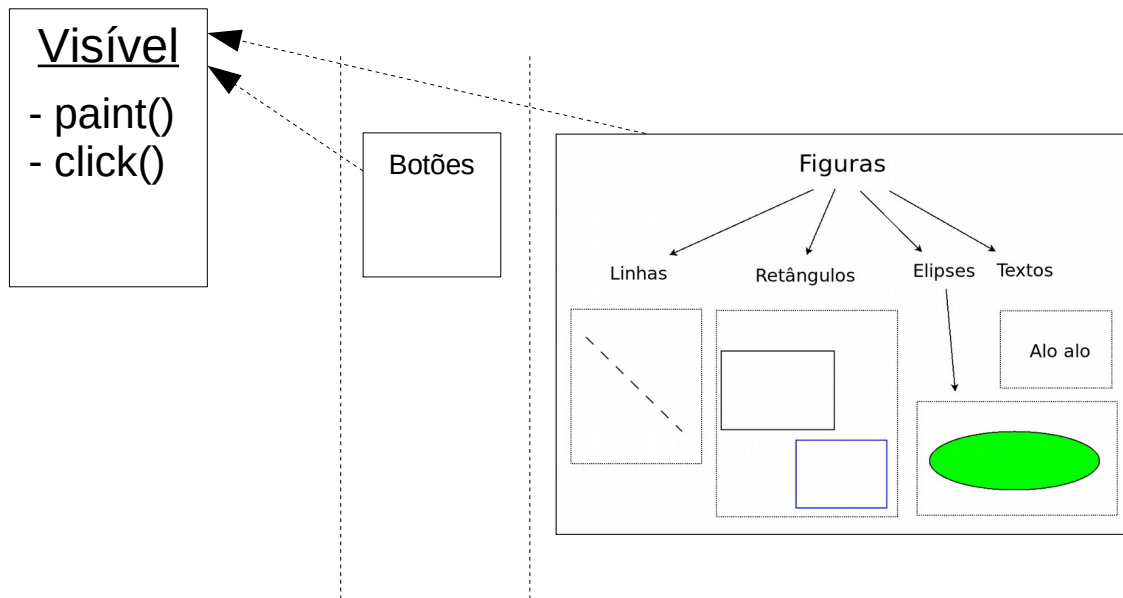
Interfaces são sempre abstratas.  
Não há implementação.

```
class PunchBag implements Kickable {  
    void kick (int vel) {  
        ...  
    }  
    int grab (void) {  
        ...  
    }  
}
```

```
class Ball implements Kickable {  
    void kick (int vel) {  
        ...  
    }  
    int grab (void) {  
        ...  
    }  
}
```

# Classes e Interfaces

- Figura
  - Retângulo, Elipse, Texto, etc.
  - x, y, w, h, color, **paint, click**, move, resize, etc

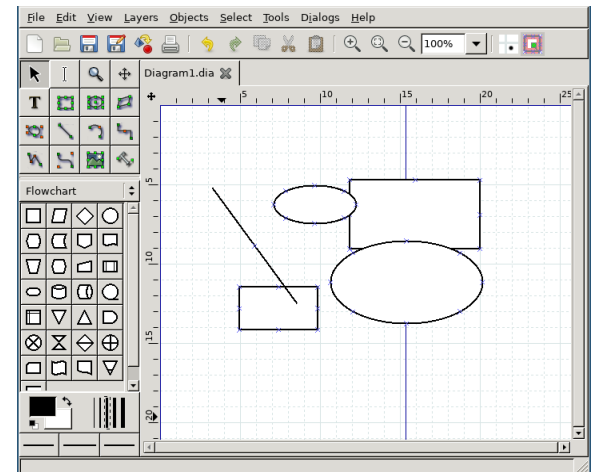


# Interfaces

- Visível
  - **Botões**, Retângulo, Elipse, Texto, etc.
  - paint, click
- **Listas ainda mais heterogêneas**

```
ArrayList<Visible> vis = new ArrayList<Visible>();  
vis.add(new Rect(x1,y1, w1,h1));  
vis.add(new Button(...));  
for (Visible v: vis) {  
    v.paint(g);  
}
```

Polimorfismo e Dynamic Dispatching



# Exercícios

1. Dê mais 2 exemplos de interfaces conforme o Slide 6:
  - Cada interface deve ter pelo menos 3 objetos representativos
  - Cada interface deve ser identificada por um adjetivo
  - Cada interface deve ter pelo menos 2 métodos
    - Os métodos devem ter parâmetros e/ou retornos.
  - Use a sintaxe de Java para descrever a interface e os métodos
2. Como o conceito de “interfaces” se relaciona com “duck typing”?
  - Vide Questão 4.1 do Módulo 2

## *2. Exemplos*

Linguagem de Programação II

<https://github.com/fsantanna-uerj/LP2/>

Francisco Sant'Anna

francisco@ime.uerj.br



## 3. *Adaptando o Projeto*

Linguagem de Programação II

<https://github.com/fsantanna-uerj/LP2/>

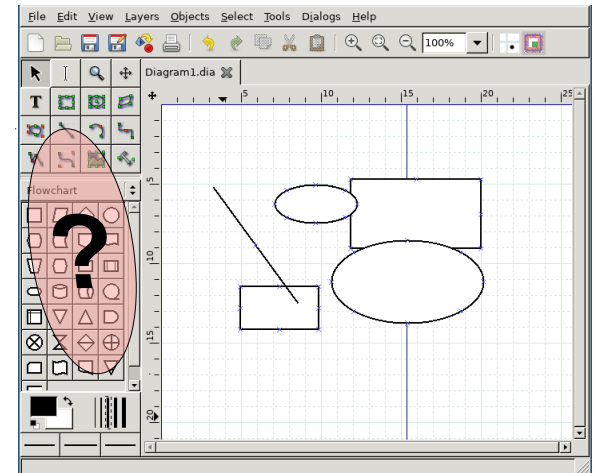
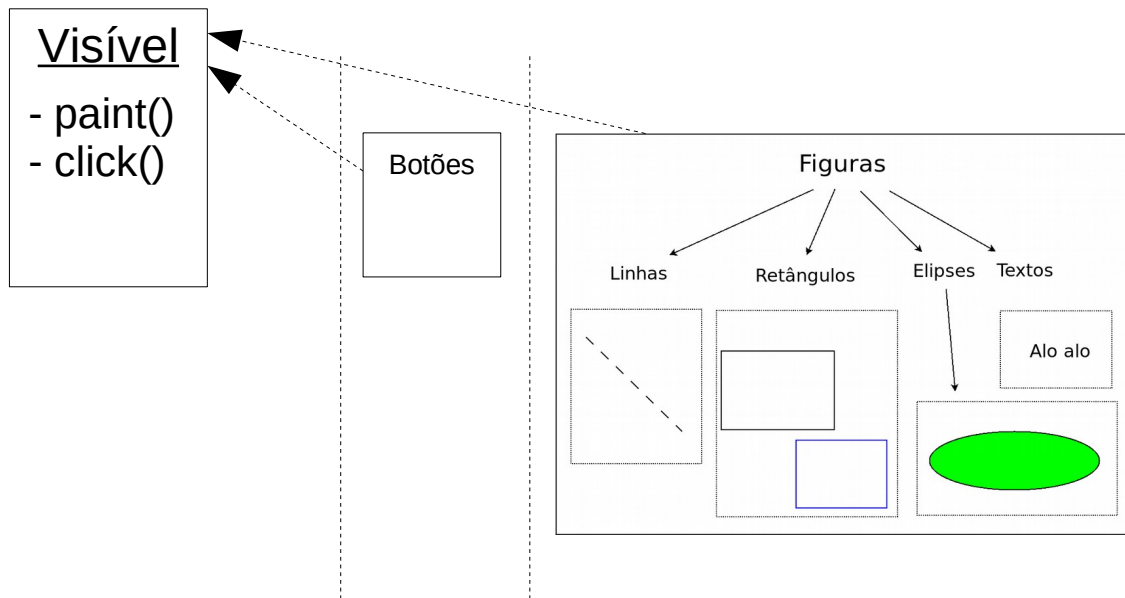
Francisco Sant'Anna

francisco@ime.uerj.br



# Classes e Interfaces

- Figura
  - Retângulo, Elipse, Texto, etc.
  - x, y, w, h, color, **paint, click**, move, resize, etc

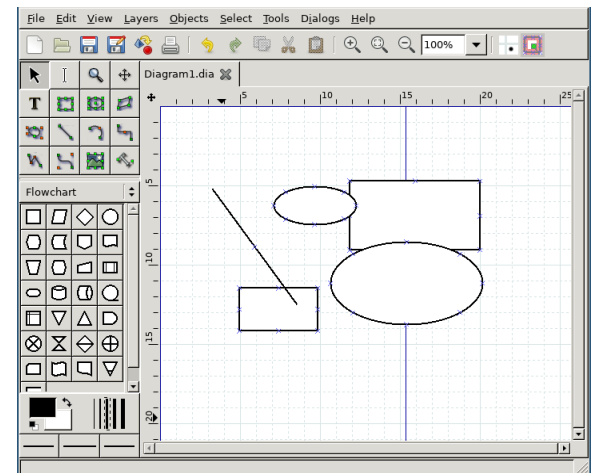


# Interfaces

- Visível
  - **Botões**, Retângulo, Elipse, Texto, etc.
  - paint, click
- **Listas ainda mais heterogêneas**

```
ArrayList<Visible> vis = new ArrayList<Visible>();  
vis.add(new Rect(x1,y1, w1,h1));  
vis.add(new Button(...));  
for (Visible v: vis) {  
    v.paint(g);  
}
```

Polimorfismo e Dynamic Dispatching





# Adaptando o Projeto

- Criar uma interface `IVisible`
  - `void paint (Graphics g);`
  - `boolean clicked (int x, int y);`
- Classe `Figure` deve implementar `IVisible`
- Testes do mouse devem usar o método `clicked`
- Classe `IVisible` precisa estar em diretório separado

# Adaptando o Projeto

```
interface IVisible {  
    void paint (Graphics g);  
    boolean clicked (int x, int y);  
}
```

Interfaces são sempre abstratas.  
Não há implementação.

```
abstract class Figure implements IVisible {  
    int x, y, w, h;  
    Color bg, fg;  
    boolean clicked (int x, int y) { ... }  
}
```

Todas as figuras usam o mesmo  
método para detecção de cliques.

```
class Rect extends Figure {  
    ...  
    void paint (Graphics g) { ... }  
}
```

Cada subfigura usa o seu próprio  
método para desenho na tela.

# Exercício

- Criar uma interface `IVisible`
  - `void paint (Graphics g);`
  - `boolean clicked (int x, int y);`
- Classe `Figure` deve implementar `IVisible`
- Testes do mouse devem usar o método `clicked`
- Classe `IVisible` precisa estar em diretório separado

# *3. Adaptando o Projeto*

Linguagem de Programação II

<https://github.com/fsantanna-uerj/LP2/>

Francisco Sant'Anna

francisco@ime.uerj.br

