1. Interfaces

Linguagem de Programação II

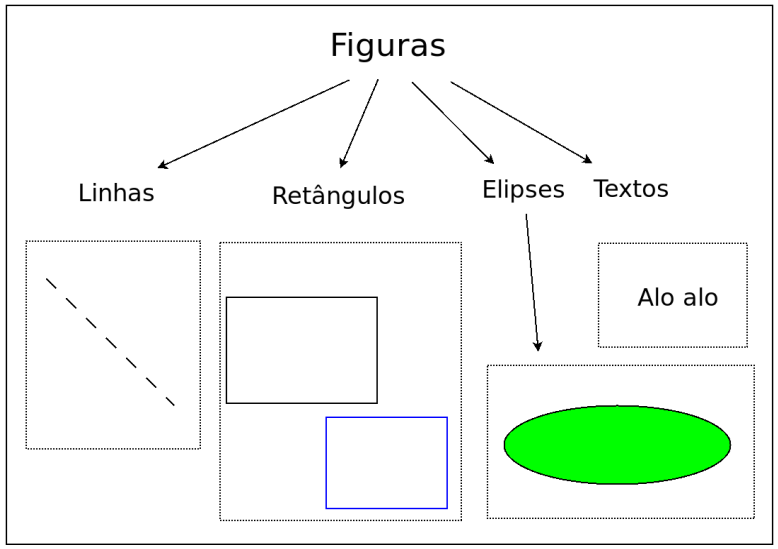
`https://github.com/fsantanna-uerj/LP2/`

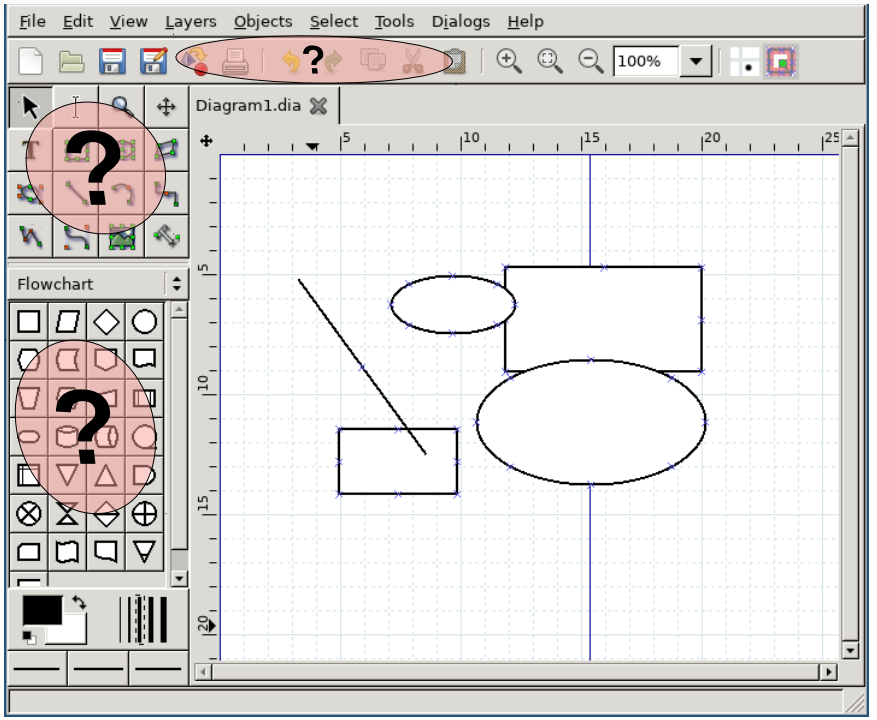
Francisco Sant'Anna

`francisco@ime.uerj.br`

Hierarquia de Classes

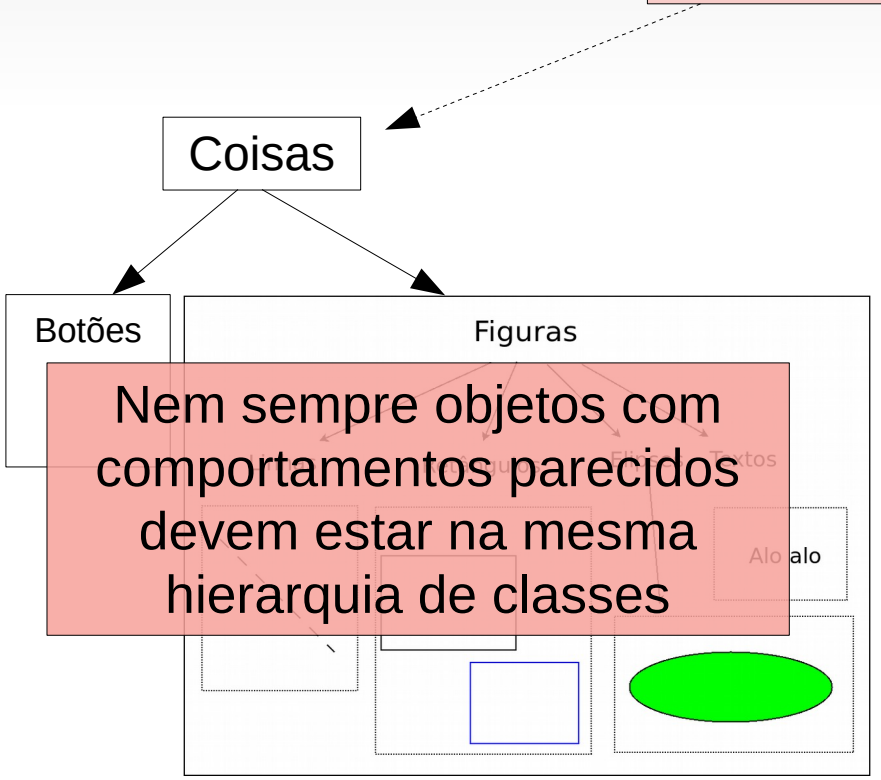
- Figura
  - Retângulo, Elipse, Texto, etc.
  - x, y, w, h, color, paint, click, move, resize, etc

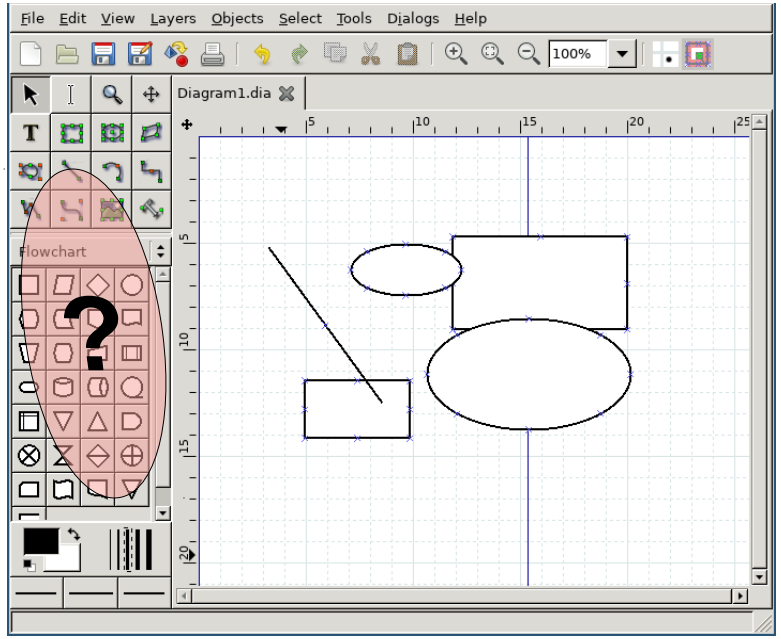




Hierarquia de Classes

- 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`

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);
}
```

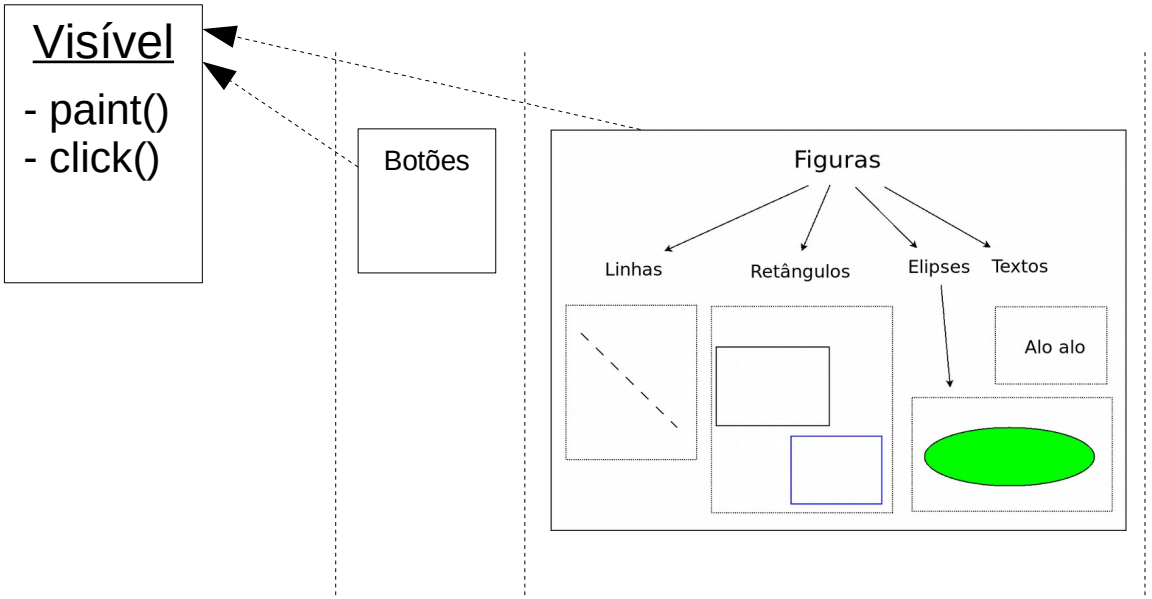
Interfaces

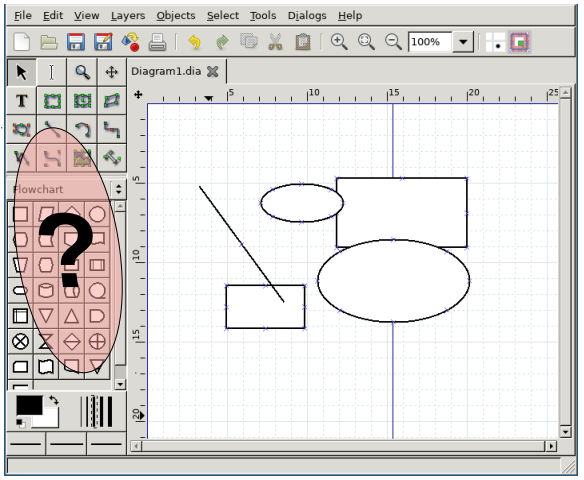
```
interface Kickable {
  void kick (int vel);
  int  grab (void);
}

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

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

Interfaces são sempre abstratas.
Não há implementação.

Classes e Interfaces

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



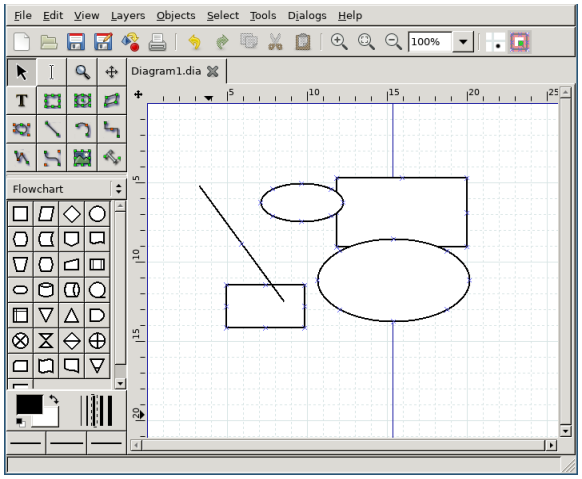


Interfaces

- 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`

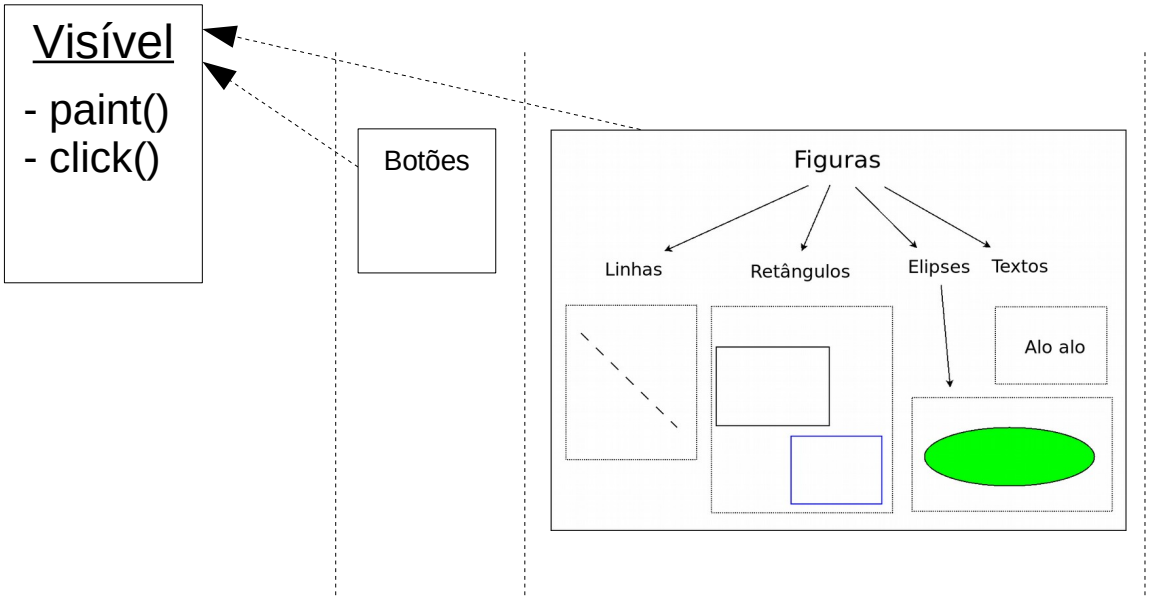
3. Adaptando o Projeto

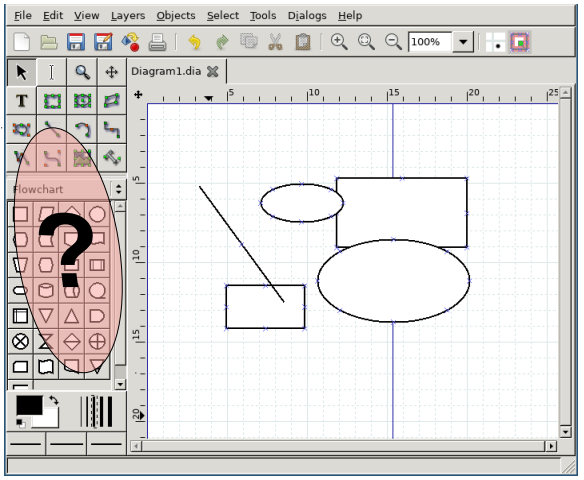
Linguagem de Programação II
`https://github.com/fsantanna-uerj/LP2/`

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



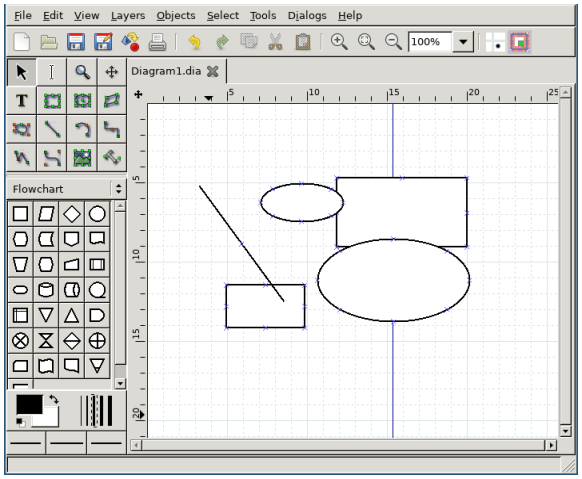


Interfaces

- 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`
- Interface `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.

09-Interface/

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`
- Interface `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`

4. Classe Abstrata vs Interface

Linguagem de Programação II
`https://github.com/fsantanna-uerj/LP2/`

Francisco Sant'Anna
`francisco@ime.uerj.br`

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

Classe Abstrata vs Interface

- **Classe:**
  - arcabouço para grupo de classes com propriedades similares
- **Interface:**
  - funcionalidades comuns para classes diversas

Classe Abstrata vs Interface

- `extends` vs `implements`
- ***is a*** vs ***behave as***
  - Rect ***is a*** Figure
  - Button ***behave as*** Visible
- Subtipagem nominal
  - posso usar o subtipo no lugar do supertipo
  - `void f (Figure fig) { … fig.x … }`
  - `void g (IVisible vis) { … vis.paint … }`

Classe Abstrata

- Não pode ser instanciada
- Implementação padrão de métodos e propriedades
- Hierarquia em árvore
  - não permite herança múltipla
- Acesso a propriedades dentro da hierarquia

Interface

- Não pode ser instanciada
- Métodos são públicos e abstratos
- Hierarquia em diamante
  - permite herança múltipla

Exercícios

1. Por quê muitos autores consideram herança múltipla como uma prática ruim?
   - Considere o “problema do diamante” na sua resposta.
2. Descreva um caso de uso para classes abstratas e outro para interfaces. Justifique a escolha em cada caso.

4. Classe Abstrata vs Interface

Linguagem de Programação II
`https://github.com/fsantanna-uerj/LP2/`

Francisco Sant'Anna
`francisco@ime.uerj.br`

5. Comparação com C

Linguagem de Programação II

`https://github.com/fsantanna-uerj/LP2/`

Francisco Sant'Anna

`francisco@ime.uerj.br`

Herança e Polimorfismo em C

- Structs aninhadas (*extends*)
- Ponteiros para funções (*overrides*)

```
abstract class Figure {
  int x, y;
  int r, g, b;
  abstract void print (void);
}
```

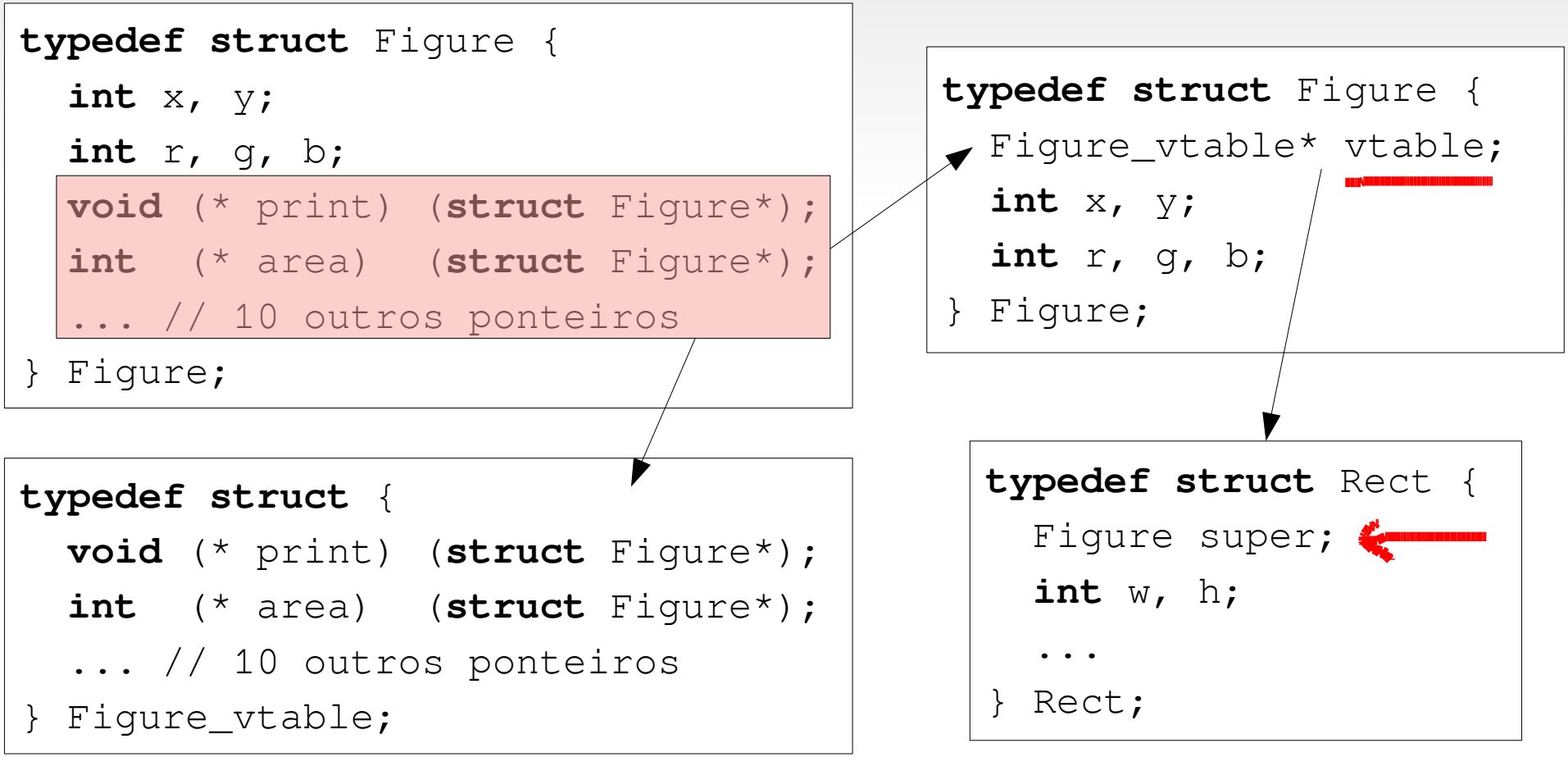
```
typedef struct Figure {
  int x, y;
  int r, g, b;
  void (* print) (struct Figure*);
} Figure;
```

```
class Rect extends Figure {
  int w, h;
  ...
  void print (void) {...}
}
```

```
typedef struct Rect {
  Figure super;
  int w, h;
  ...
} Rect;
```

Virtual Method Table (vtable)

- … usar ponteiro único para tabela com 12 ponteiros.



Virtual Method Table (vtable)

- … usar ponteiro único para tabela com 12 ponteiros.

```
typedef struct Figure {
  Figure_vtable* vtable;
  int x, y;
  int r, g, b;
} Figure;
```

```
Figure_vtable rect_vtable = {
  rect_print,
  rect_area,
  ...
};
```

- 8 bytes por objeto
- acesso indireto

```
Rect* rect_new (int x, int y, ...) {
  Rect* this = malloc(sizeof(Rect));
  ...
  super->vtable = &rect_vtable;
  ...
  return this;
}
```

```
void main (void) {
  ...
  for (int i=0; i<N; i++) {
    figs[i]->vtable->print(figs[i]);
  }
  ...
}
```

O Problema

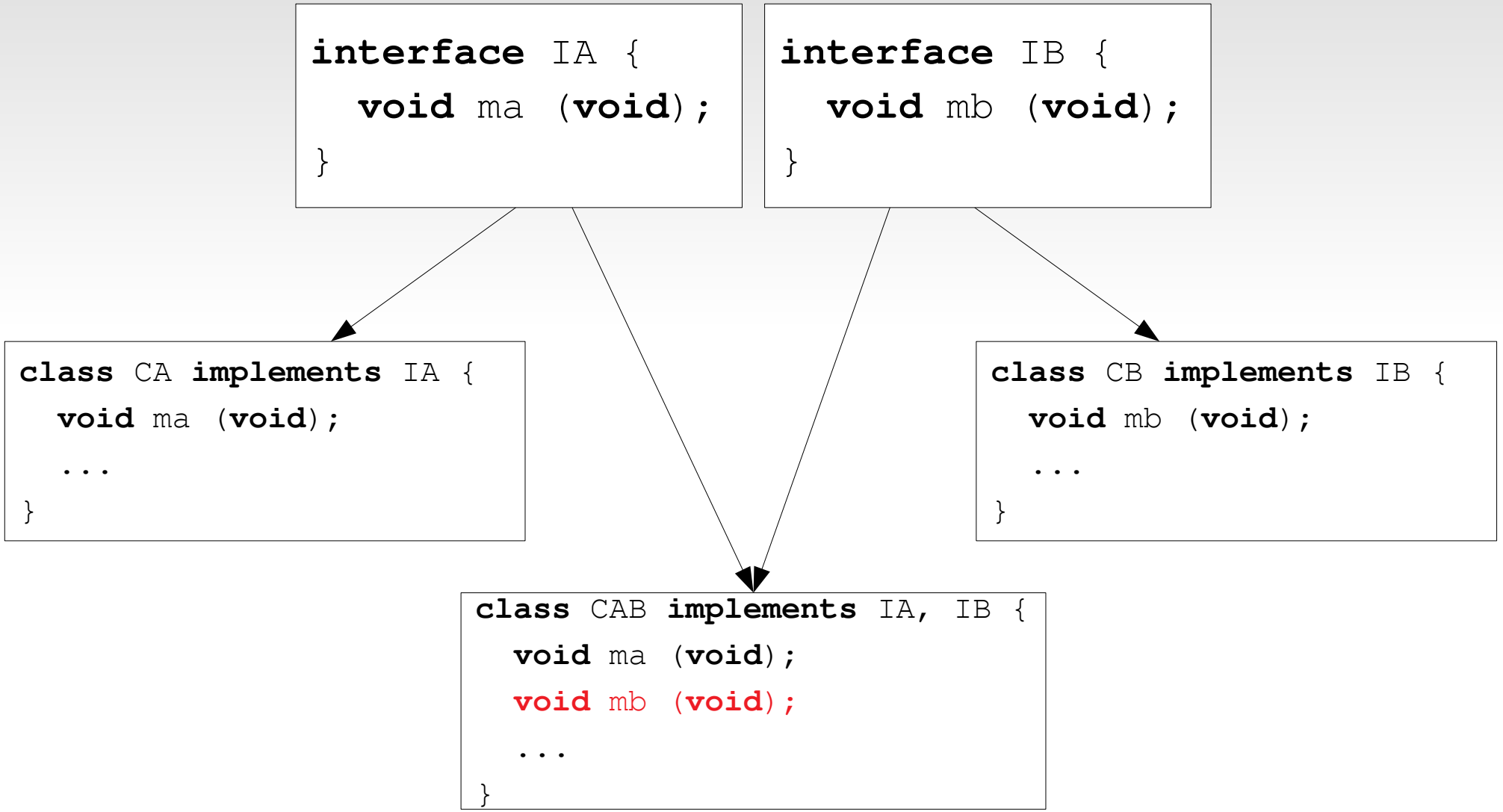
```
interface IVisible {
  void paint (Graphics g);
  boolean clicked (int x, int y);
}
```

```
class Figure implements IVisible {
  int x, y;
  int w, h;
  Color bg, fg;
  void paint (Graphics g);
  boolean clicked (int x, int y) { … }
}
```

```
class Button implements IVisible {
  int idx;
  Figure fig;
  void paint (Graphics g);
  boolean clicked (int x, int y) { … }
}
```

Métodos em *offsets* diferentes impossibilitam despacho dinâmico.

O Problema


interface IA {
void ma (void);
}
interface IB {
void mb (void);
}
class CA implements IA {
void ma (void);
...
}
class CB implements IB {
void mb (void);
...
}
class CAB implements IA, IB {
void ma (void);
void mb (void);
...
}

Interface Method Tables (itables)

- Uma *itable* para cada interface implementada
- Busca em uma tabela hash a *itable* correspondente ao método sendo chamado.

```
class Figure implements IVisible {
   ...
}

public static main (...) {
   IVisible vis = ...
   vis.paint();
}
```

```
typedef struct Figure {
   Figure_vtable* vtable;
   HashTable itables;
   ...
} Figure;

void main (void) {
   IVisible* vis = ...
   vis->itables[“IVisible”]->paint(vis);
}
```

5. Comparação com C

Linguagem de Programação II
`https://github.com/fsantanna-uerj/LP2/`

Francisco Sant'Anna
`francisco@ime.uerj.br`