USP - Universidade de São Paulo

IME - Instituto de Matemática e Estatística

Orientação a objetos

Prof. Marco Aurélio Gerosa
gerosa@ime.usp.br

Histórico da Orientação a objetos

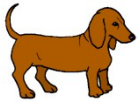
- Linguagem Simula 67 (anos 60 na Noruega)
  - Usada para simulação de navios, cada objeto era responsável por seus dados e comportamento
- Smalltalk (década de 70) no Xerox PARC
  - Linguagem dinâmica, com classes, objetos, métodos, herança
- Extensões ao LISP (década de 70) para incorporar os conceitos da orientação a objetos
- C++ (década de 80)
- Explosão de linguagens orientadas a objetos (décadas de 90 e 00)

Conceitos básicos

- *"Um sistema orientado a objetos é aquele cujos componentes são partes encapsuladas de dados e funções, que podem herdar atributos e comportamento de outros componentes da mesma natureza, e cujos componentes comunicam-se entre si por meio de mensagens."*
  - *Eduard Yourdon*
- Objeto



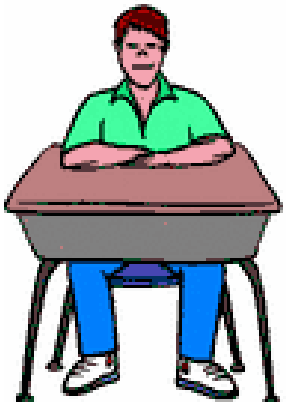


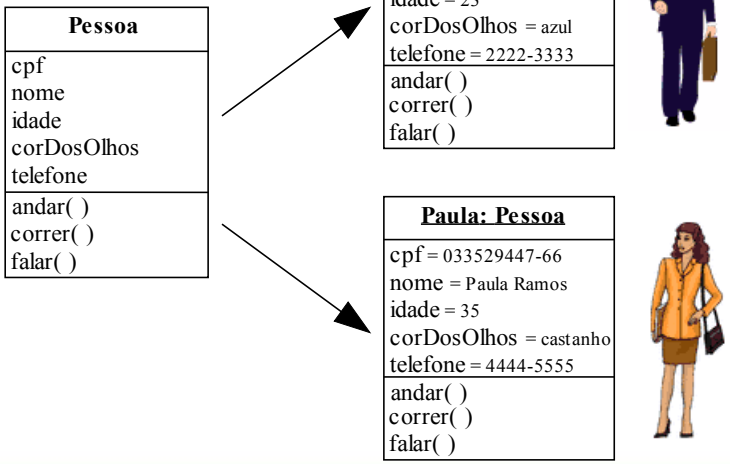





- Classe conceitual
- Classe de software

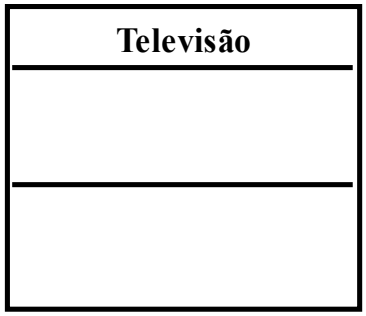
Classe

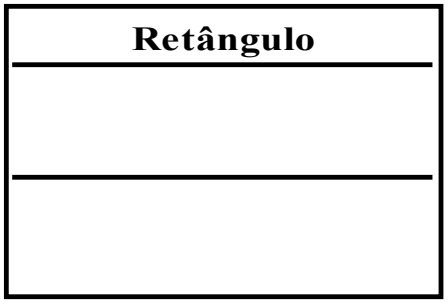
Objeto no mundo real	Classe Conceitual	Classe de Software	Objeto de Software
	**Aluno** mat nome	**Aluno** - mat: int - nome: String + matricular() + trancarMat()	**obj: Aluno** - mat = 1234 - nome = "José" + matricular() + trancarMat()



Outros conceitos

- Instanciar
- Instância
- Atributo
- Valor de atributo
- Operação
- Método

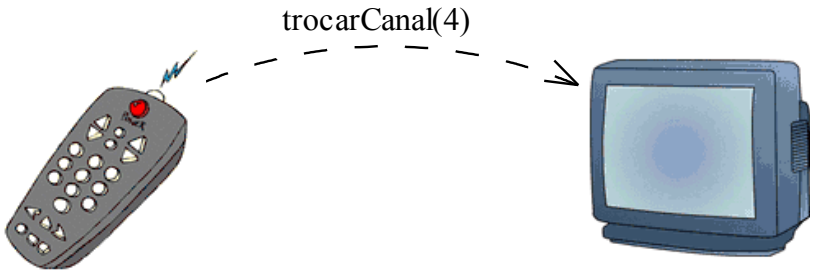




Mensagem

ControleRemoto

botao1( )
botao2( )
botao3( )
botao4( )
botaoAumentarVolume( )
botaoDiminuirVolume( )



Televisão

marca
modelo
polegadas
canal
volume

ligar( )
desligar( )
trocarCanal(c)
aumentarVolume( )
diminuirVolume( )

Implementação

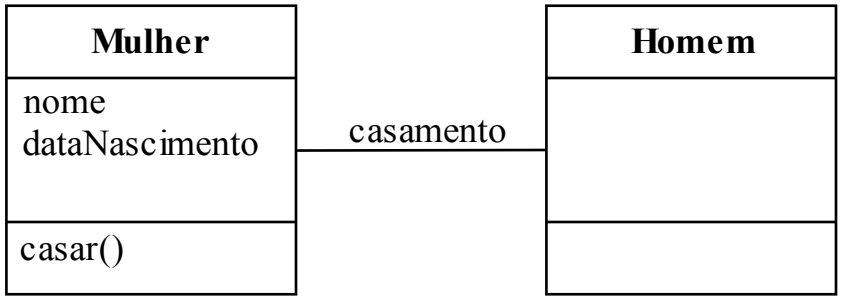
- Operador de acesso
- Construtor
- Variáveis x objetos (identidade de objetos)
- Linguagens puras x híbridas
- Estilo de programação OO:
  - função(dados)
  - objeto.função( )

```
str = "A casa é amarela.";
String.substring(str,posIni,posFinal);

str = "A casa é amarela.";
str.substring(posIni,posFinal);
```

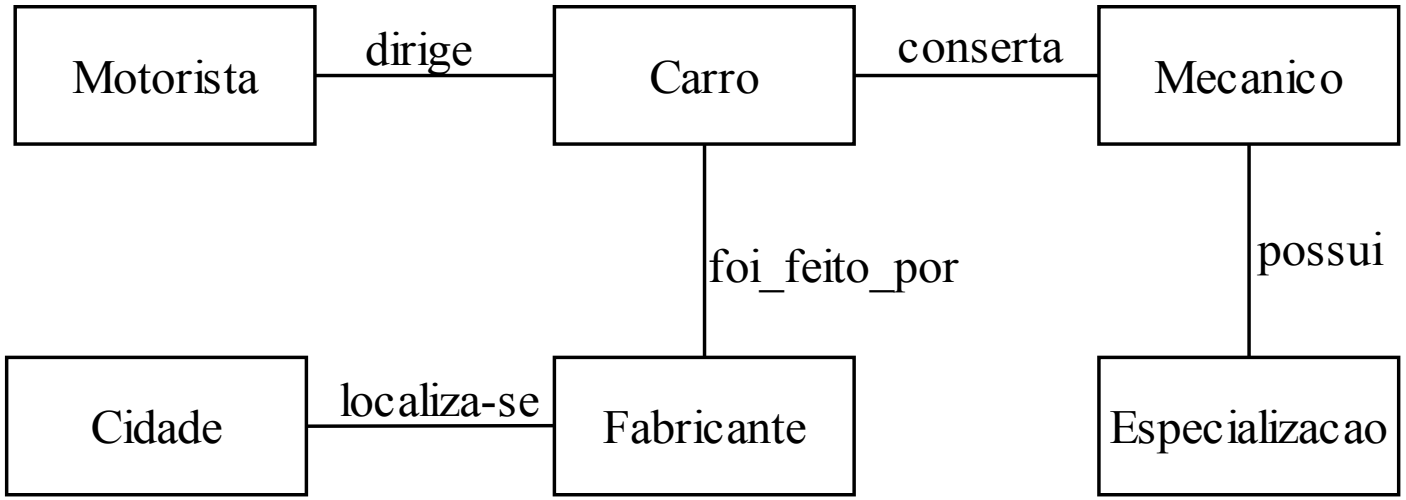
- Assinatura de operações

Relacionamentos

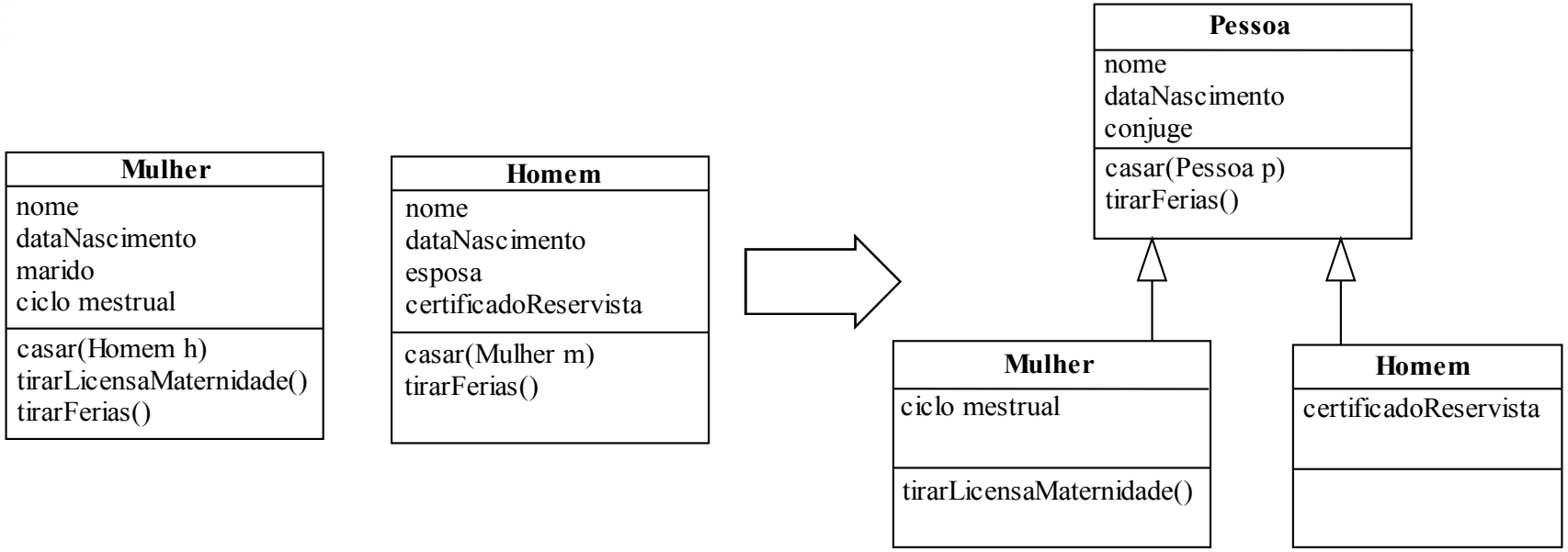
- Ligações entre objetos
- Associações entre classes




```
class Mulher {

    String nome;
    Homem marido;
    Date dataNascimento;

    public void casar(Homem h) {
        //....
    }

}
```



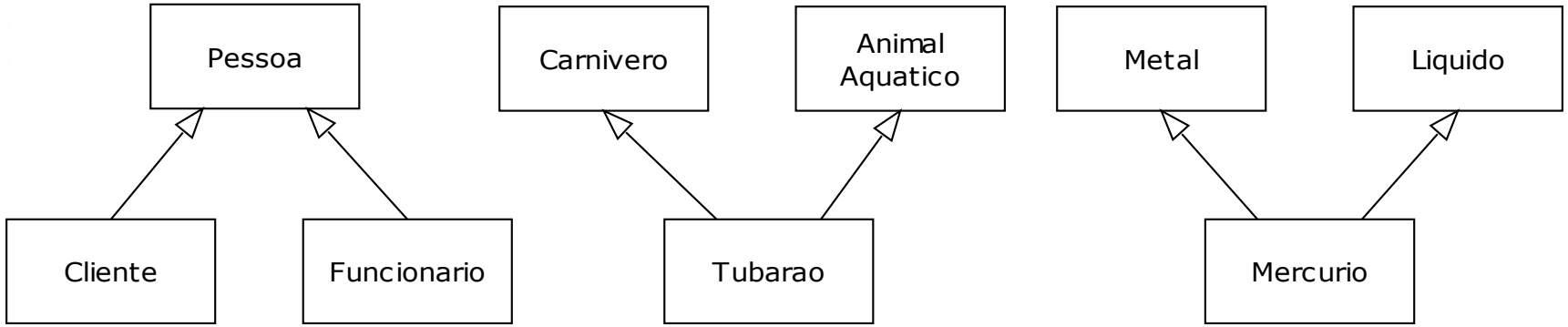
Herança



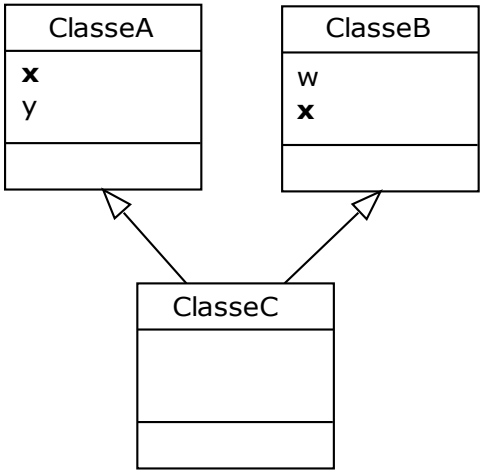

```
class Mulher extends Pessoa {
```

- Superclasse e subclasse
- Processo de generalização ou especialização
- Hierarquia de classes
- Herança simples x herança múltipla

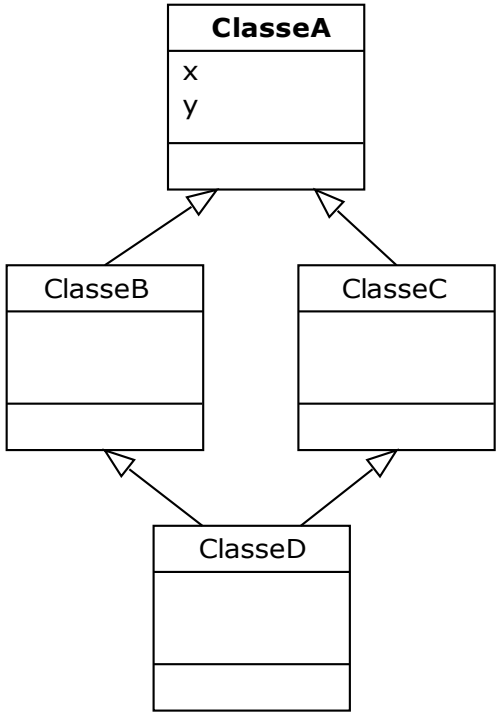
Herança múltipla



- Problemas da herança múltipla



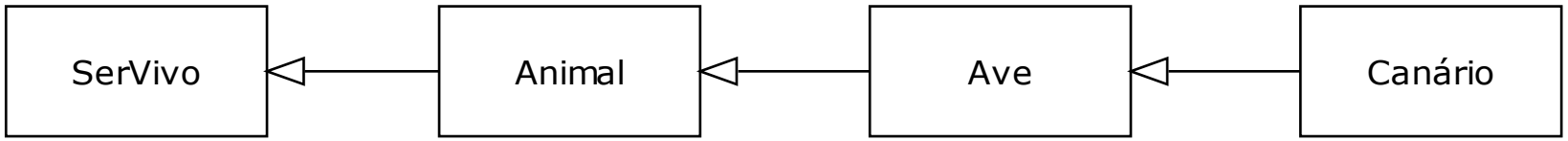
(Colisão de nomes)



(Herança repetida)

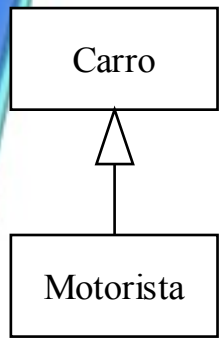
Herança

- Desenvolvedores podem evitar a codificação redundante
- Possibilita expressar a similaridade entre classes, simplificando a definição de classes semelhantes.
- Uma classe derivada (uma especialização) pode incluir novos métodos e atributos ou redefinir os métodos herdados.
- **Todo membro de uma subclasse também pode ser visto como sendo um objeto da superclasse. Desta forma, todos os atributos e associações da superclasse devem valer para os objetos da subclasse.**

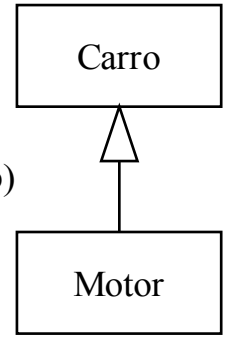


Em quais casos a herança está errada?

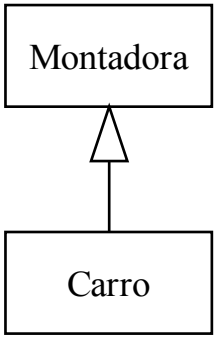
a)



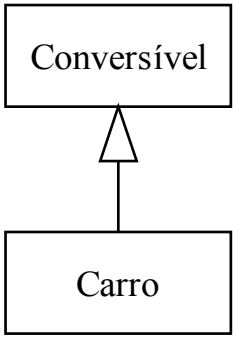
b)



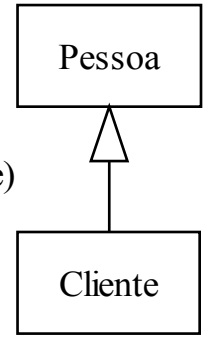
c)



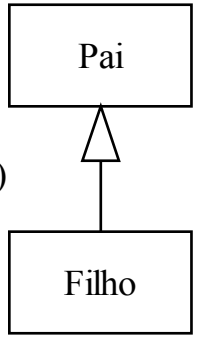
d)



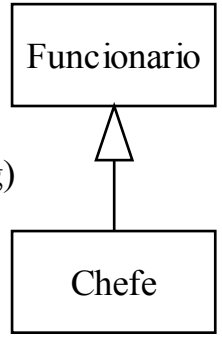
e)



f)

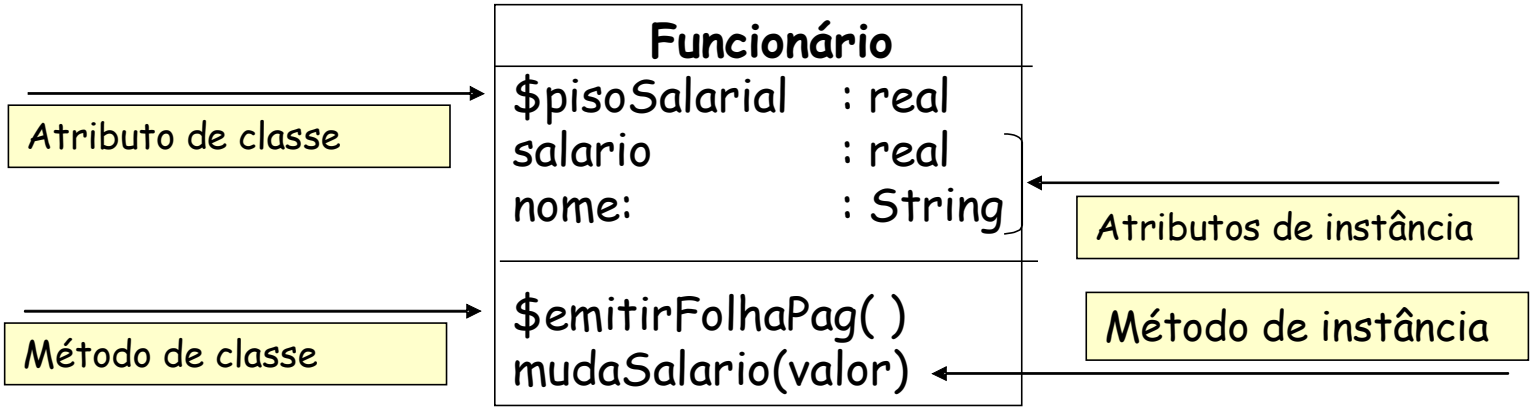


g)

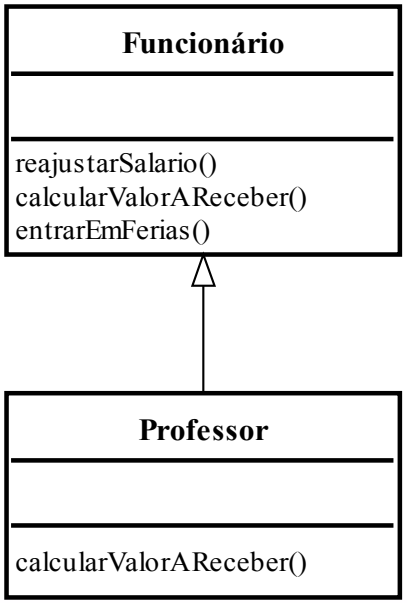


Conceitos adicionais

- Métodos e atributos de classe



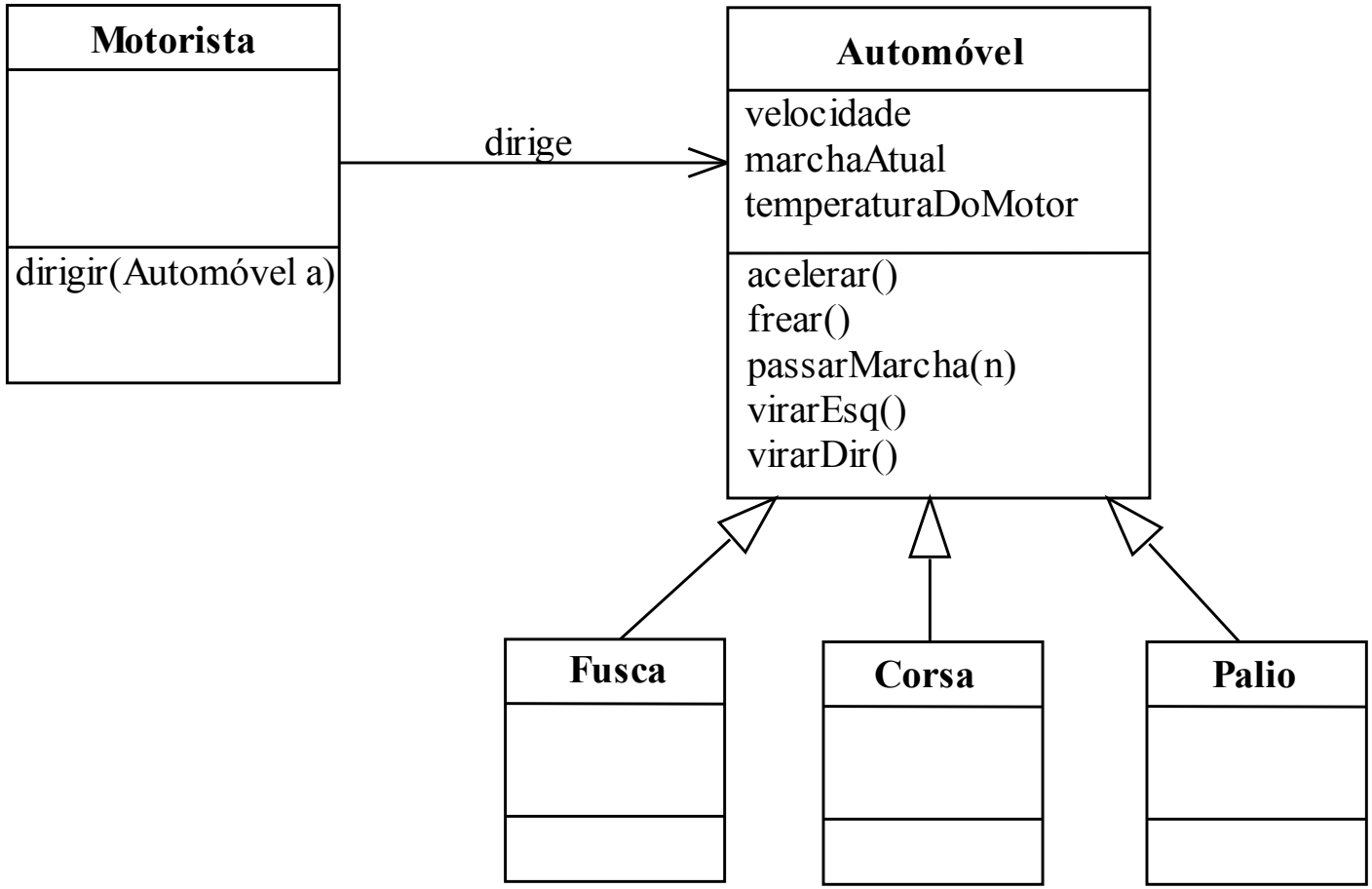
- Sobrecarga de método
- Sobrecarga de nome
- Sobrecarga de operador



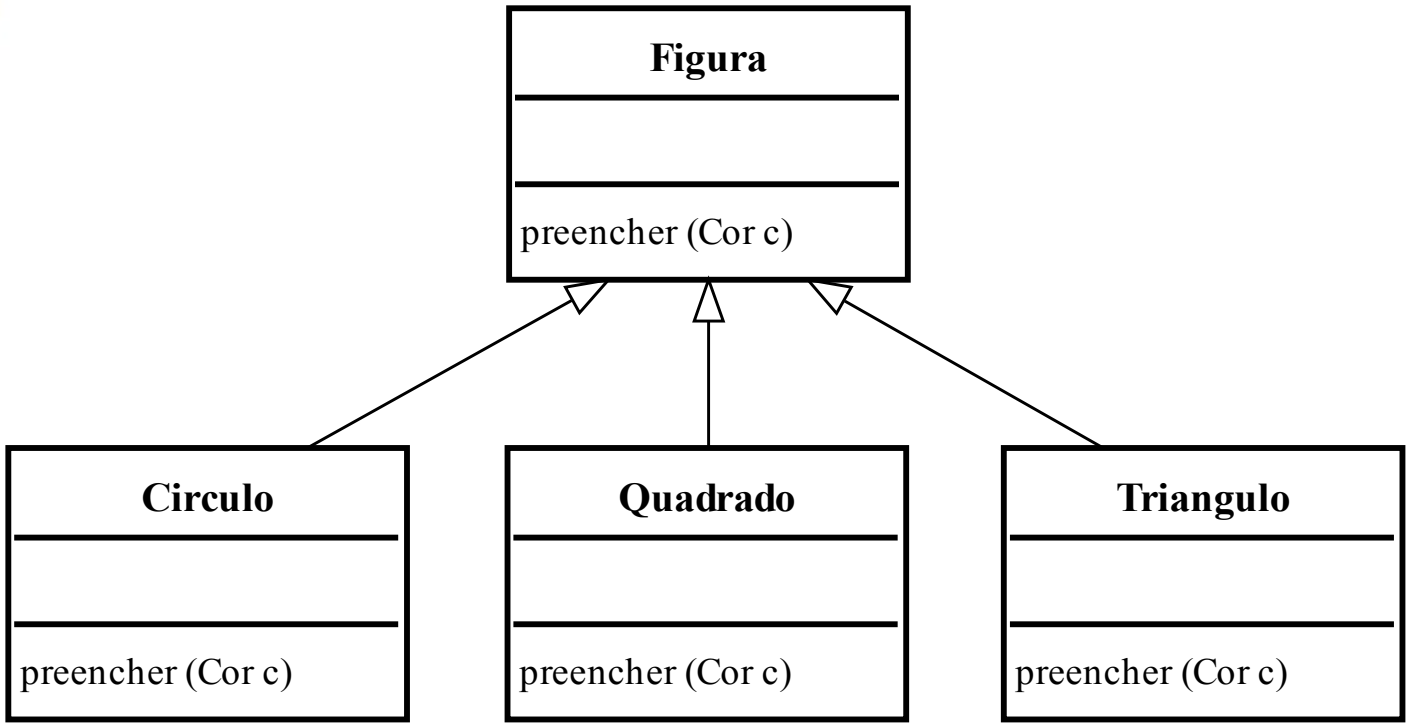
- Eventos
- Reflexão

Polimorfismo

- Habilidade de tomar várias formas



Polimorfismo II




```
public void metodo (Figura f, Cor c) {

    f.preencher(c);

}
```

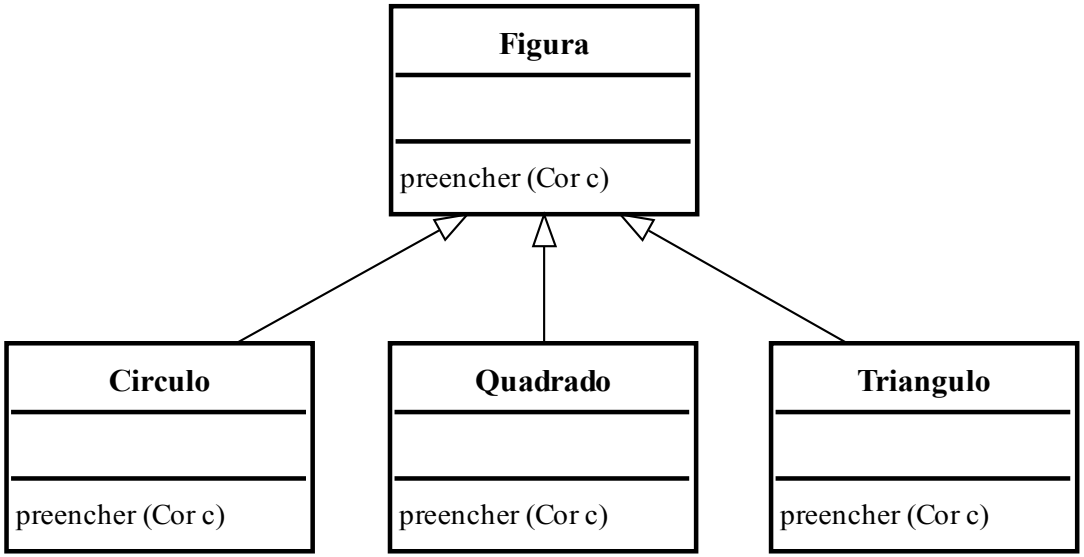
Ligação dinâmica

- Ligação estática – durante a compilação ou link-edição é feita a "amarração" entre uma chamada de método e sua implementação
- Ligação dinâmica – durante a execução que é identificada qual é a implementação que será usada para uma chamada de método.

- A ligação dinâmica torna-se a base para o polimorfismo.

Operações e classes abstratas

- Uma classe abstrata não pode instanciar objetos.
- Tem alguma operação não implementada (que é sobrecarregada pelas subclasses)
- Serve basicamente para o polimorfismo



Interface

- Todas as operações não são implementadas
- Simula a herança múltipla (em termos de polimorfismo) pois não padece dos mesmos problemas.