PARADIGMAS DE PROGRAMAÇÃO

2023/2024

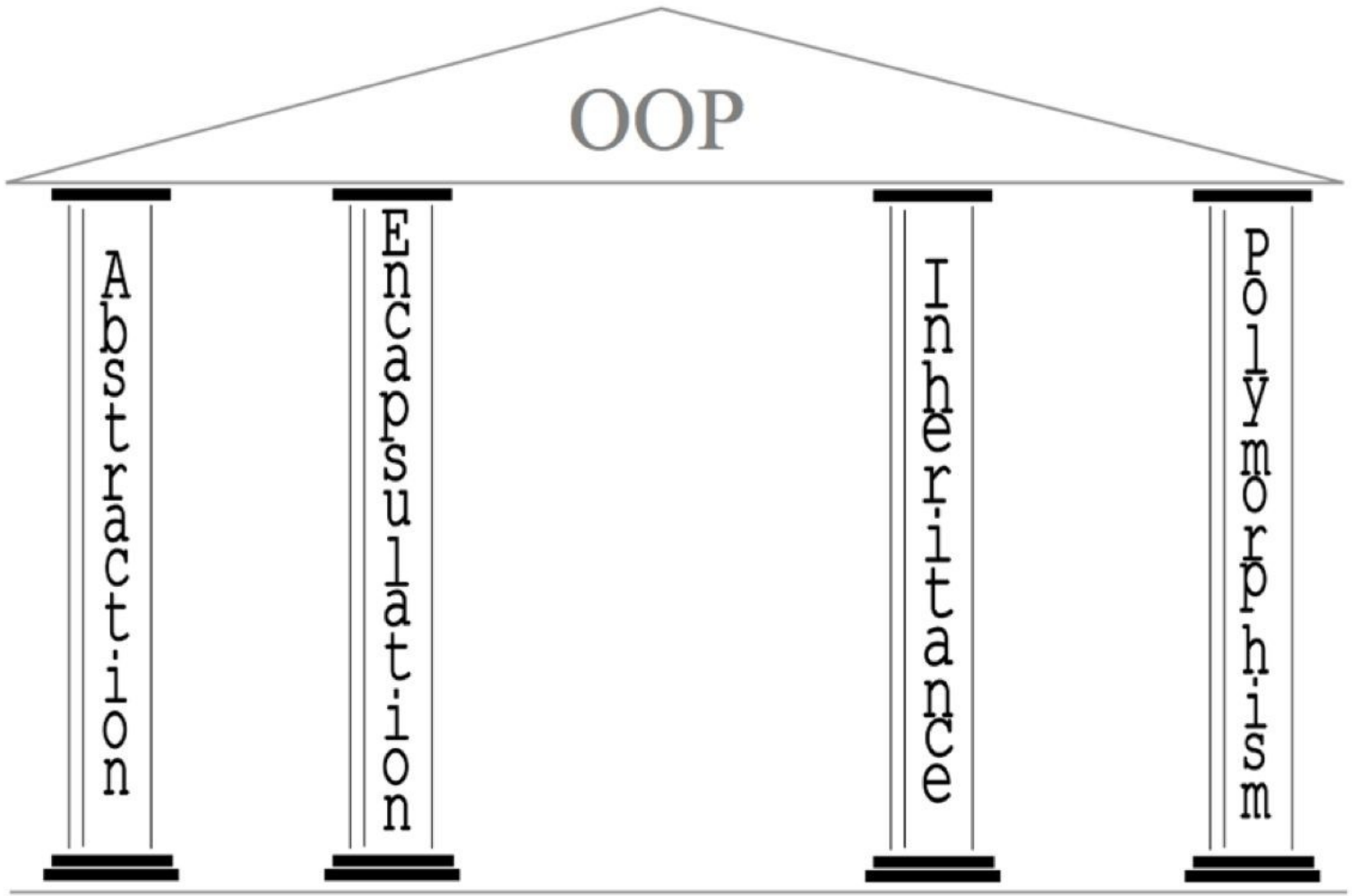
P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Aula 10

1. Palavras Reservadas
2. Abstração
3. Encapsulamento
4. Herança
5. Polimorfismo
6. Palavras Reservadas Usadas
7. Links Úteis

Ricardo Santos | rjs@estg.ipp.pt



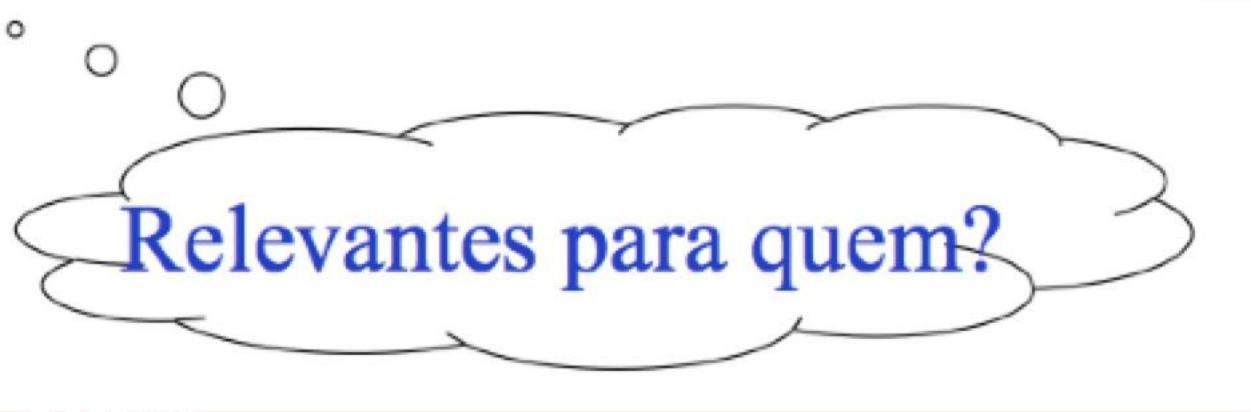
ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

P.PORTO

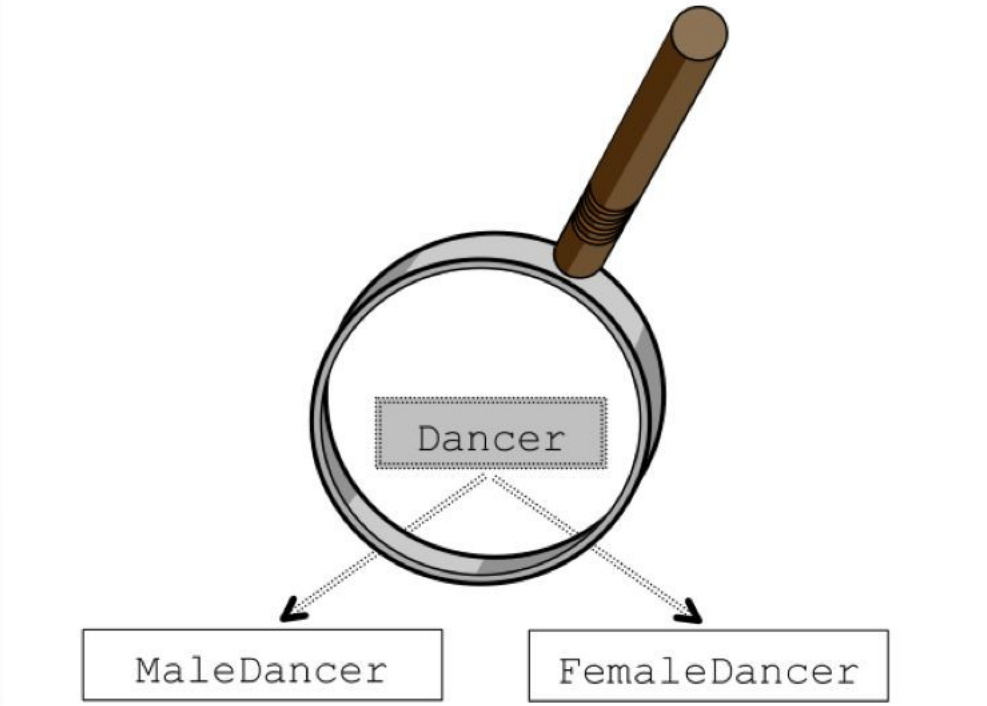
ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Abstracção

- Abstracção diz-nos que devemos ignorar características, propriedades ou comportamentos irrelevantes e enfatizar aqueles que são mais relevantes...



- ... relevantes para o problema que estamos a estudar (com um olho no futuro e na reutilização em problemas similares)



ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Encapsulamento

- A propriedade do encapsulamento diz que os dados da classe que representam o estado do objeto devem ser declarados como privados
- Alguns dos métodos também podem ser declarados como privados (caso se justifique)

- A classe interage com outras classes (denominadas de clientes desta classe) apenas a partir do construtor e métodos públicos que servem de interface para os clientes da classe

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

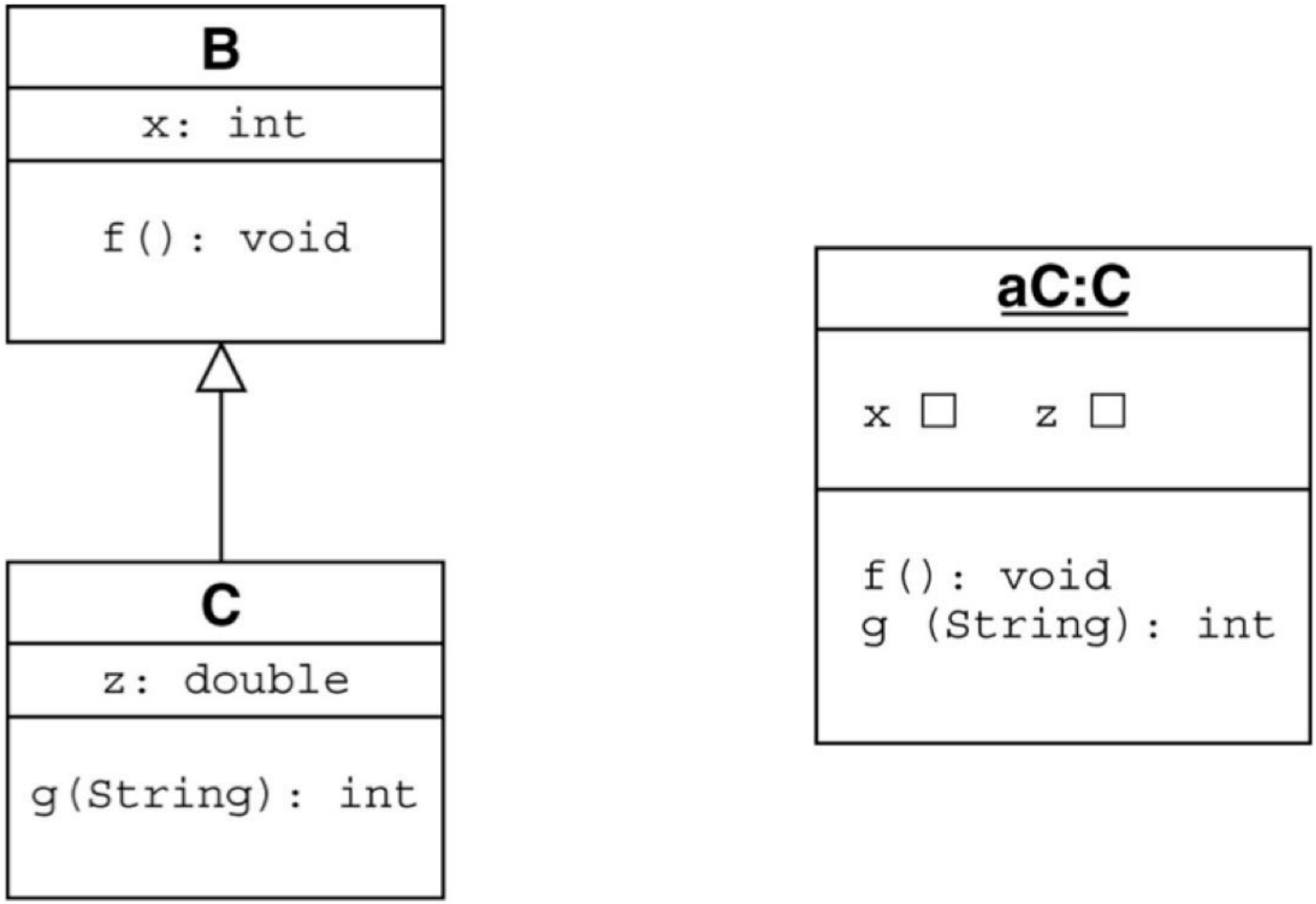
Herança

- Permite aos programadores personalizarem uma classe para um propósito específico, sem modificarem a classe original (superclasse)
- É permitido à classe derivada (subclasse) adicionar métodos ou mesmo redefini-los
- A subclasse pode adicionar variáveis mas não as pode redefinir

- A classe **C** é uma subclasse da classe **B** (a sua superclasse) se a declaração for desta forma

```
class C extends B {
   ...
}
```

- A subclasse é uma **especialização** da superclasse
- A superclasse é uma **generalização** da subclasse


B
x: int
f(): void
C
z: double
g(String): int
aC:C
x
z
f(): void
g (String): int

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Herança e Mensagens

- Quando **C** é uma subclasse de **B**
  - Os objectos de **C** podem responder a todas as mensagens que **B** também pode
  - De uma forma geral os objectos de **C** podem ser usados sempre que poder ser usado um objecto **B**

- É possível que uma subclasse de **B** possa ter métodos e variáveis que não tenham sido definidas em **C**
  - É um caso em que um objecto de **B** não pode ser usado no lugar de um objecto de **C**

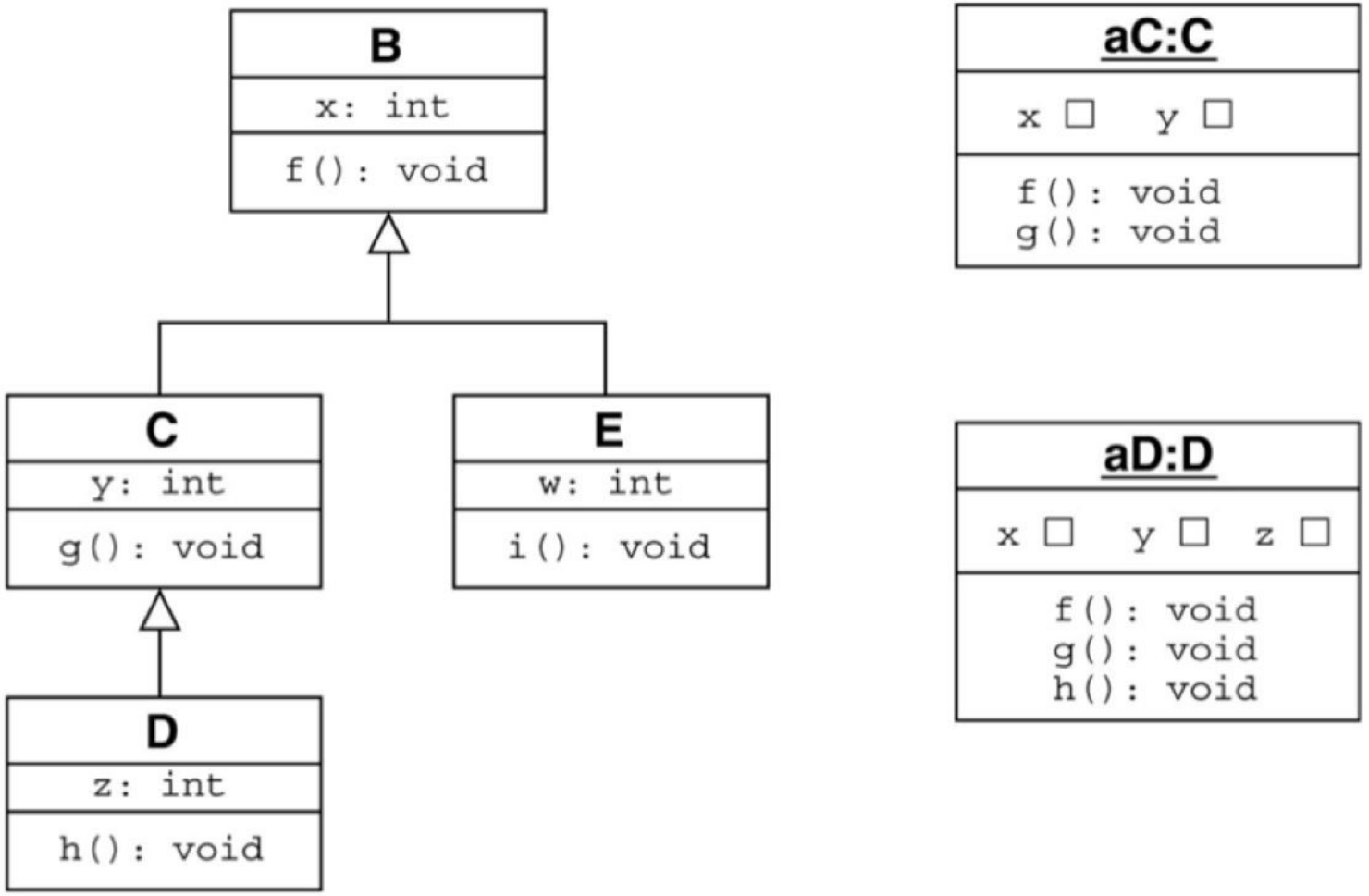
ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Hierarquia de Herança

- Uma classe pode ter várias subclasses e cada uma das subclasses pode ter as suas subclasses próprias
- A colecção de todas as subclasses descendentes de um ancestral comum é denominado de hierarquia de herança

- As classes que aparecem por baixo de um dada classe na hierarquia de herança são os seus **descendentes**

- As classes que aparecem por cima de uma dada classe na hierarquia de herança são os seus **ancestrais**


B
x: int
f(): void
C
y: int
g(): void
E
w: int
i(): void
D
z: int
h(): void
aC:C
x
y
f(): void
g(): void
aD:D
x
y
z
f(): void
g(): void
h(): void

Exemplo: Viagem a um aviário

- Considere uma colecção de aves que possuem diferentes propriedades
  - nome
  - cor (algumas com o mesmo nome são de cores diferentes)
  - comem diferentes alimentos
  - emitem diferentes barulhos
  - alguns fazem múltiplos tipos de sons

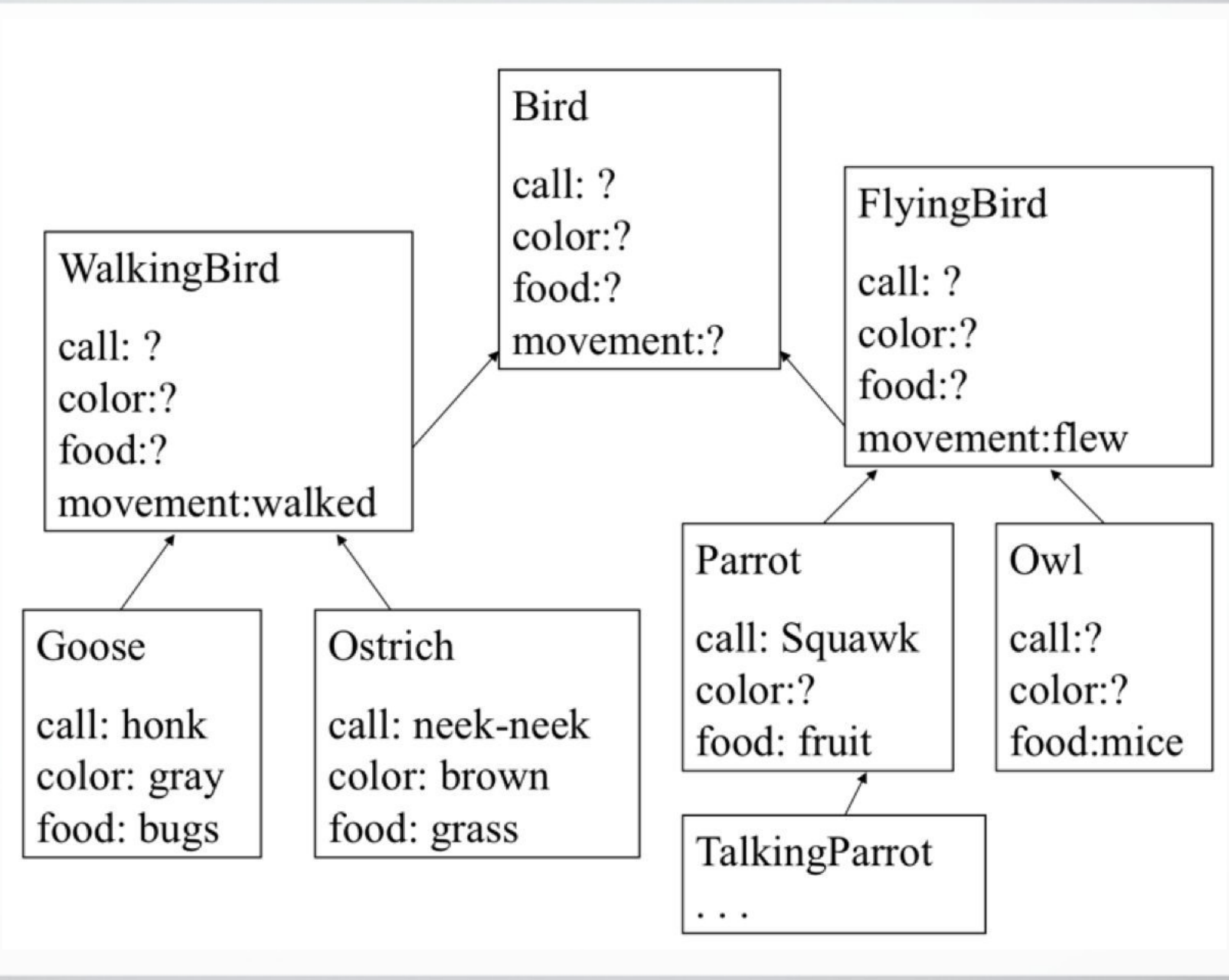
Procuramos realizar um programa que simule esta colecção!

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Esboço

- A chave para a resolução deste problema passa por criar uma hierarquia de herança de **Bird**

- **Estratégia**
  - criar classes para as várias entidades (objectos)
  - identificar as características que as classes têm em comum
  - criar superclasses para armazenar os conjunto de características comuns encontradas


Bird
call: ?
color:?
food:?
movement:?
WalkingBird
call: ?
color:?
food:?
movement:walked
FlyingBird
call: ?
color:?
food:?
movement:flew
Goose
call: honk
color: gray
food: bugs
Ostrich
call: neek-neek
color: brown
food: grass
Parrot
call: Squawk
color:?
food: fruit
Owl
call:?
color:?
food:mice
TalkingParrot
. . .

Questões Pendentes

- Como será representado o **aviário**?
- Será que faz sentido existir o objecto **Bird**?
- e **WalkingBird** faz sentido?
- e **FlyingBird** faz sentido?

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Polimorfismo

- A palavra polimorfismo vem do grego: **muitas formas**
- O polimorfismo é uma das características fundamentais nas linguagens orientadas a objectos, e é a capacidade dos objectos se poderem comportar de forma diferente consoante as suas características ou do ambiente

- Enquanto que a herança se refere às classes (e à sua hierarquia), o polimorfismo diz respeito aos métodos dos objectos

- Existem três tipos de polimorfismo
  - ***Overloading* / Sobrecarga** (ou polimorfismo *ad hoc*)
  - **Polimorfismo paramétrico** (a explorar em Estruturas de Dados)
  - ***Overriding* / Sobreposição** (ou polimorfismo de herança)

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Overloading / Sobrecarga (ou polimorfismo *ad hoc*)

- Com este tipo de polimorfismo podemos ter vários métodos com o mesmo nome e assinaturas diferentes
- Desta forma é possível invocar um determinado comportamento consoante a lista de parâmetros que lhe é passada

Polimorfismo Paramétrico

- Este é o tipo mais complexo de polimorfismo
- Existe apenas um método que foi programado para trabalhar com um conjunto de tipos de dados

- Enquanto que na sobrecarga os métodos têm um tipo de parâmetro na assinatura, neste tipo de polimorfismo o tipo só é passado em tempo de execução

- É também denominado de método ou comportamento genérico

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Overriding / Sobreposição (ou polimorfismo de herança)

- Com este tipo de polimorfismo podemos redefinir um método em classes descendentes de forma a o podermos especificar mais

- O polimorfismo garante que é chamado o método apropriado para um objecto de um tipo específico quando o objecto está disfarçado como um tipo mais geral

- Quando um método é invocado
  - O sistema procura localmente pelo nome do método
  - Caso não encontre procura pelos métodos herdados

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
P.PORTO

Bibliografia

- Barnes, J.D. e Kölling, M. 2008 Objects First with Java - A Practical Introduction using BlueJ. Prentice Hall/Pearson Education.
- http://www.bluej.org/objects-first/#
- Eckel, B. 2002 Thinking in Java (3rd Edition). Pearson Education.
- http://www.mindview.net/Books/TIJ/#