Programação orientada a objetos

Linguagens de Programação **2016.2017**

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Sumário

Conceitos básicos

Conceitos básicos

Objecto

- Conjunto de operações sobre dados “escondidos”
- Maneira uniforme de encapsular uma combinação de dados e funcionalidades
  - pequeno como simples inteiro
  - grande como um sistema de ficheiros ou base de dados

Variável de instância

- Parte de dados do objecto
- Também designada por campo ou dados-membro

Conceitos básicos

Método

- Parte funcional do objecto
- Também designado por função-membro

Mensagem

- Interacção com o objecto através de operações simples
- Também designada por chamada de função membro

Classe

- Determina a implementação de um objecto

Instância

- Criação de um objecto de uma determinada classe

Objeto

Consiste em

- Dados escondidos
  - Também é possível ter funções escondidas
- Operações públicas
  - Algumas linguagens permitem ter dados públicos

Programa OO

- Não é mais que o envio de mensagens a objectos!

Orientação a objetos

Metodologia de programação

- Organizar conceitos em objectos e classes
- Construir sistemas extensíveis

Características

- Pesquisa dinâmica
- Abstracção
- Sub-tipos
- Herança

Características

Pesquisa dinâmica

Programação OO

objecto → mensagem(argumentos)

O código excutado depende do objecto e da mensagem

Programação convencional

operação(argumentos)

O significado da operação é sempre o mesmo

Esta é a diferença fundamental entre tipo de dados abstratos e objetos!

Exemplo – adição de 2 números

Programação OO

x → soma(y)

A função soma é diferente consoante x for inteiro, real, complexo, …

Programação convencional

soma(x, y)

A função soma tem um único significado

Polimorfismo vs pesquisa dinâmica

Polimorfismo resolvido em tempo de compilação

Pesquisa dinâmica em tempo de execução

Encapsulamento

Construtor do conceito

- visão detalhada

Utilizador

- visão “abstrata”

O encapsulamento separa estas visões

- Implementação
  - Trabalha sobre a representação do objeto
- Interface
  - Conjunto de operações fornecidas pelo construtor da abstração

Herança e sub-tipo

Implementação

Representação interna do objeto

Interface

Visão externa do objeto

Herança

Relação entre implementações

Subtipo

Relação entre interfaces

Interface

Mensagens “percebidas” pelo objeto

Exemplo: ponto

- coord-x
  - devolve a coordenada x do ponto
- coord-y
  - devolve a coordenada y do ponto
- move
  - altera a posição

A interface de um objeto constitui o seu tipo

Subtipo

Se a interface A contém toda a interface B, então os objetos A podem ser utilizados como objetos B

A é subtipo de B

Exemplo

```
Ponto                Ponto_colorido
  coord_x              coord_x
  coord_y              coord_y
  move                 move
                        muda_cor
```

Interface de Ponto_colorido contém Ponto

Ponto_colorido é **subtipo** de Ponto

Herança

Mecanismo de implementação

Novos objetos podem ser definidos re-utilizando a implementação de outros objetos

Exemplo

```
class Ponto
private
    float x, y
public
    Ponto move (float dx, float dy);
class Ponto_colorido
private
    float x, y; cor c
public
    Ponto move(float dx, float dy);
    Ponto muda_cor(cor novac);
```

Subtipo vs. herança

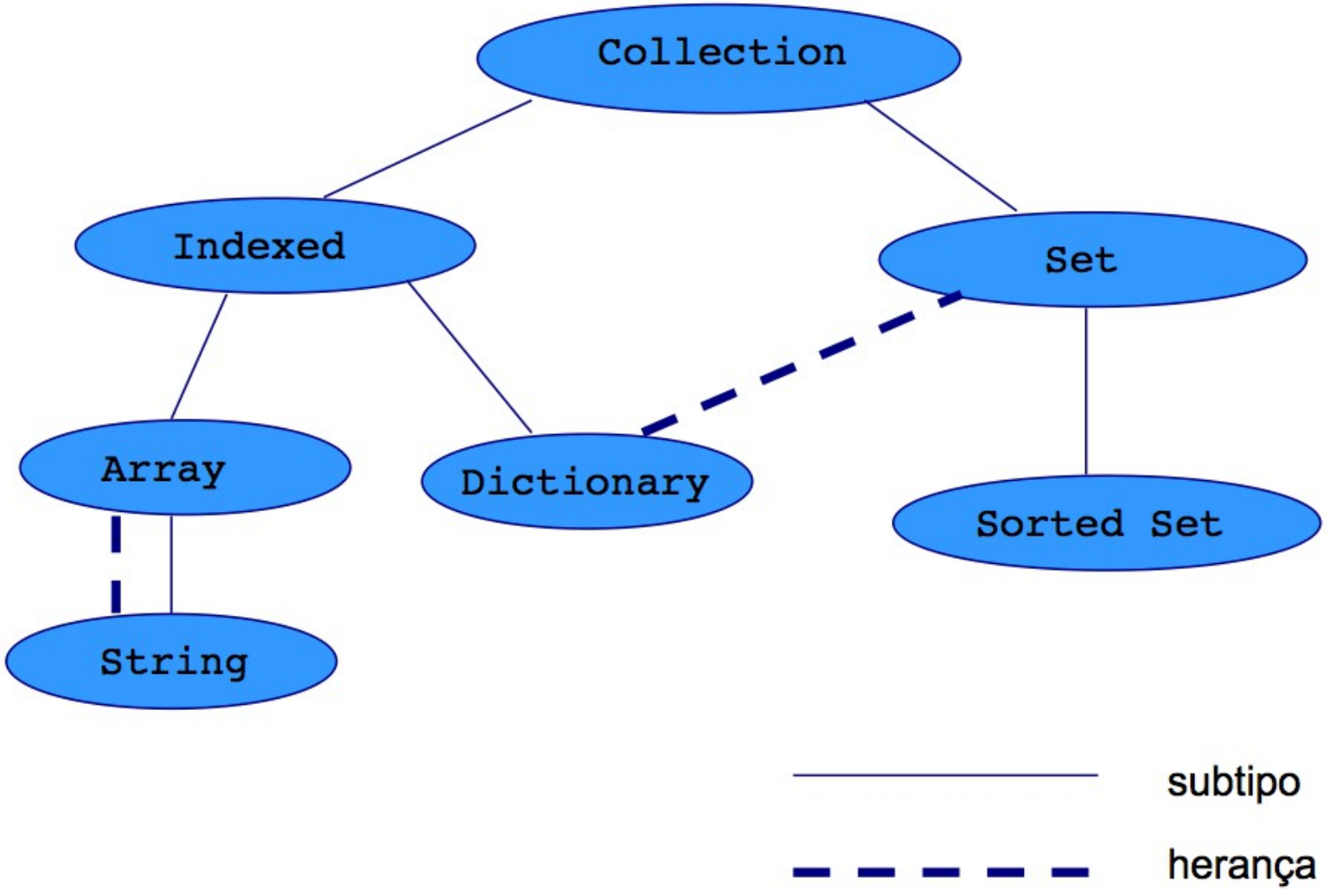
Subtipo

- Ponto_colorido pode ser utilizado em vez de Ponto
- Propriedade utilizada pelo programa cliente

Herança

- Ponto_colorido pode ser implementado reutilizando a implementação de ponto
- Técnica utilizada na implementação das classes

Exemplo


Collection
Indexed
Set
Array
Dictionary
Sorted Set
String
subtipo
herança

Estrutura do programa

Exemplo: biblioteca de geometria

Definição do conceito geral

- `Forma`

Implementação de duas formas

- `Círculo`
- `Rectângulo`

Funções implementadas

- `centra`
- `move`
- `roda`
- `imprime`

Anticipa adições à biblioteca

Forma

Interface de qualquer Forma tem de incluir

- `centra`
- `move`
- `roda`
- `imprime`

Diferentes Forma são implementadas de maneira distinta

- `Rectângulo` → quatro pontos
- `Círculo` → centro e raio

Hierarquia de tipos

Interface

definida em Forma

Implementação

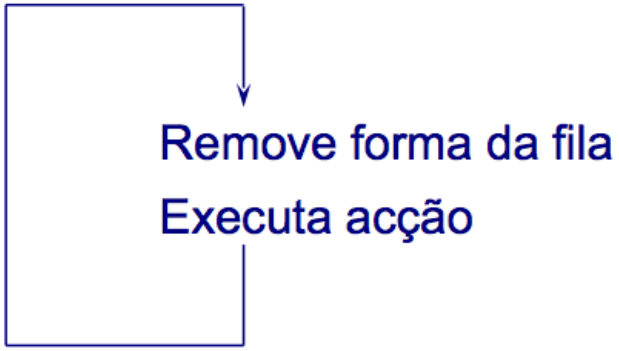
definida em Rectângulo e Círculo

É possível estender a hierarquia com outras formas

Exemplo de utilização

Fila de Forma

Ciclo de processamento



O ciclo de controlo não precisa de saber qual o tipo de cada Forma

Código

	centra	move	roda	imprime
Círculo	c_centra	c_move	c_roda	c_imprime
Rectângulo	r_centra	r_move	r_roda	r_imprime

Programação OO

`circulo → move( x, y)`

Chama função `c_move`

`rectangulo → move( x, y)`

Chama função `r_move`

Programação convencional

`c_move` e `r_move` são colocados na função `move`

Linguagens baseadas em classes

Simula **1960's**

Conceito de objecto utilizado em simulação

Smalltalk **1970's**

Desenho orientado a objectos, sistemas

C++ **1980's**

Adaptação de ideias do Simula ao C

Java **1990's**

Programação distribuida, internet

Variedade de linguagens OO

Linguagens baseadas em classes

- Comportamento do objeto é determinado pela classe
- C++, Java, ...

Baseadas em objectos

- Os objetos são definidos directamente
- Self, JavaScript

Multi-métodos

- Operações dependem dos operandos
- CLOS