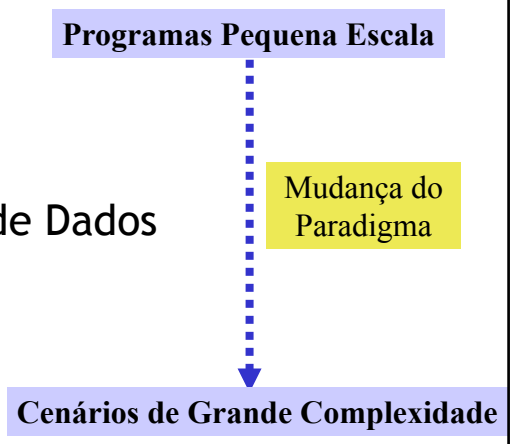
Programação Orientada a Objectos
Conceitos Gerais

UA, DETI, Programação III
José Luis Oliveira, Carlos Costa
2017/18

Metodologias de Programação

1. Procedimentos
2. Módulos
3. Tipos Abstractos de Dados
4. Objectos

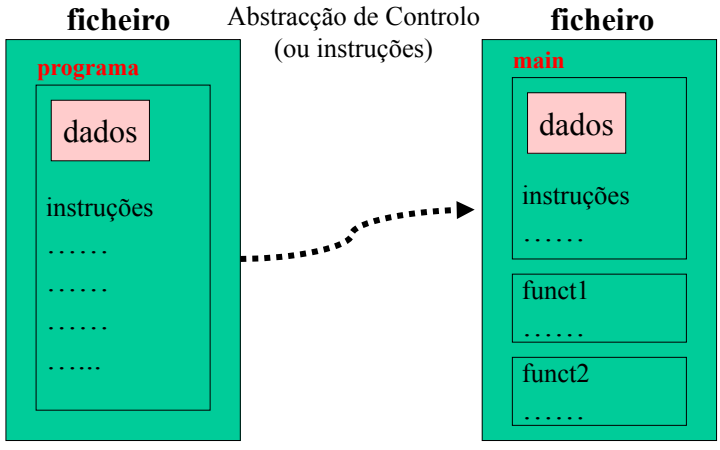


1. Procedimentos

- Programação tradicional.
- As funções/procedimentos assumem o papel estruturante dos programas.
  - Dados desempenham um papel secundário
- Polarizada para a eficiência de processamento.
- Baseia-se na filosofia:
  - **Identificar as funções a utilizar e desenvolver para cada uma o melhor algoritmo**
- Linguagens
  - Fortran, Algol, Pascal, C.

3

1. Procedimentos



Problema:
o facto de serem unidades de código que são compilados conjuntamente com o programa principal limita a sua reutilização (só utilizando "copy & paste").

4

2. Módulos

Conceito:

- consiste em separar um programa em pequenos módulos independentes:
  - cada um deles "auto suficiente" e com relações bem definidas com outros módulos relevantes.
  - podem ser compilados separadamente;
- garantindo que:
  - diferentes pessoas podem trabalhar de forma independente em diferentes módulos;
  - a "fusão" dos vários módulos num único programa é consistente;
  - alterações num módulo terão um impacto mínimo ou nulo nos restantes;

5

2. Módulos

- Surgiu então um novo conceito:

  Tipo de Dados

  \+

  Conjunto de Operações Associadas

- O paradigma é:
  - **Decidir quais os módulos necessários.**
  - **Dividir o programa segundo esses módulos de forma a esconder os dados e a funções respectivas**
- Linguagens:
  - Modula-2, Ada.

6

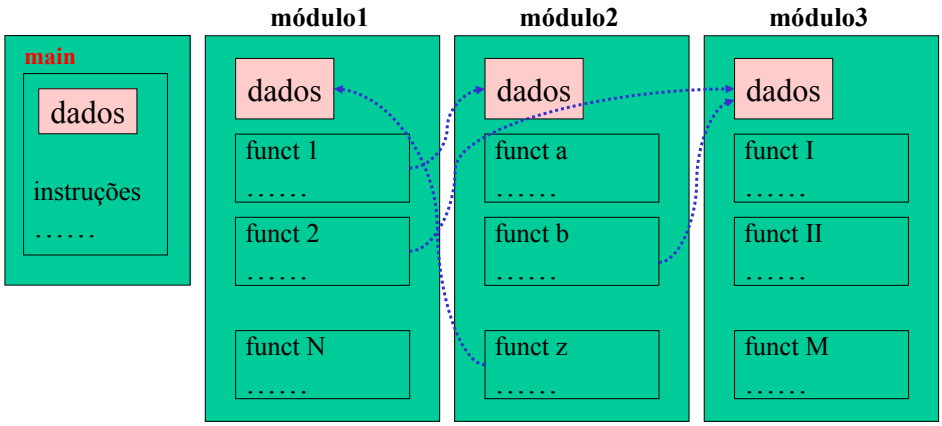
2. Módulos

Vantagens

- Cada módulo pode ser desenvolvido, analisado e testado de forma independente;
- Reduz a complexidade do programa global através da implementação de mecanismos de abstracção para facilitar tarefas;
- Isolamento de domínio de responsabilidades:
  - cada módulo pode ser responsabilidade de entidades (pessoas/empresas) distintas;
- Reutilização de código:
  - ao desenvolvermos um módulo especializado numa tarefa/funcionalidade, podemos facilmente reutilizá-lo noutro programa com as mesmas necessidades;

7

2. Módulos



Problema:
As funções de um módulo continuam a aceder a dados de outros módulos, comprometendo assim a desejada independência e reutilização em qualquer contexto.

8

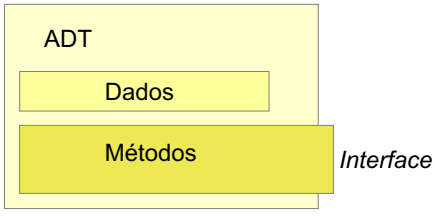
3. Tipos abstractos de dados

- ADT (Abstract Data Types)
- Consiste em definir tipos que especificam propriedades e funcionalidades comuns a diversas entidades (objectos).
- Evolução da organização em Módulos
- O paradigma de programação é:
  - **Decidir que tipos são necessários.**
  - **Fornecer um conjunto completo de operações para cada tipo**
- Linguagens:
  - Ada, C++, Java.

9

3. Tipos abstractos de dados

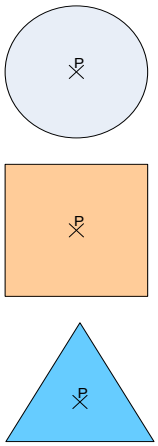
- Porquê "abstracto"?
  - Porque o Tipo de Dados fornece um conjunto de serviços (interface) bem definido (métodos, parâmetros, retorno) sem necessidade de conhecer a sua implementação
- Comparação Modelação Procedimental # ADTs
  - Na primeira dá-se relevo às estruturas de dados omitindo o seu comportamento
  - Cada Tipo Abstracto de Dados define um contracto semântico. Externamente não são conhecidos os dados nem os detalhes de construção



10

3. Implementação de tipos abstractos de dados - Problema

- Considerando três elementos gráficos distintos - círculos, triângulos, quadrados - é possível encontrar propriedades comuns que permitam a definição de um tipo genérico
  - O centro P e a cor C.
- Algumas funções tem implementações diferentes consoante o elemento gráfico
  - draw()
  - area()
- Problema!
  - Não distinção entre propriedades genéricas e propriedades específicas (sub-tipos de objectos).



11

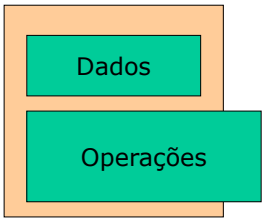
4. Objectos

- Ênfase nos objectos e no seu comportamento
  - "Ask not first what the system does; Ask what it does it to!" (B. Meyer)
- Paradigma :
  - **Decidir quais as classes necessárias**
  - **Fornecer um conjunto completo de operações para cada classe**
  - **Usar herança para expressar aspectos comuns**
- Características principais da POO
  - Encapsulamento
  - Herança
  - Polimorfismo

12

Encapsulamento

- Permite agrupar sob uma forma única de abstracção de dados um conjunto de elementos (ADT).
- Inclui os dados e métodos que manipulam esse dados (comportamento e estado).
- O utilizador de uma classe tem apenas acesso à sua interface (*information hiding*)



Interface (pública)

13

Encapsulamento - exemplo

1

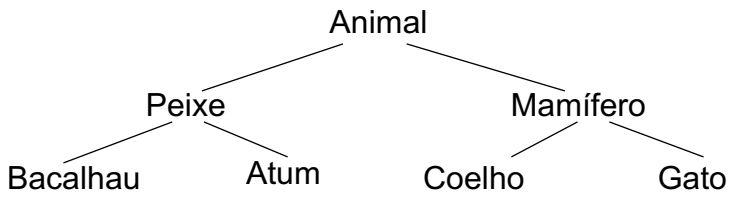
```
public class Figure {
    private int px;
    private int py;
    private int cor;

    public  void draw();
}
```

2 **Melhor**

```
public class Point {
    private int x;
    private int y;
   //...
}

public class Figure {
    private Point p;
    private int cor;

    public  void draw();
}
```

Herança

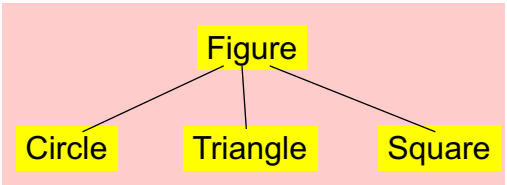
- Mecanismo através do qual várias classes podem ser relacionadas hierarquicamente.
- Explora similaridades (generalização e especialização)
- Promove reutilização e extensibilidade
- Explicita características e operações comuns



15

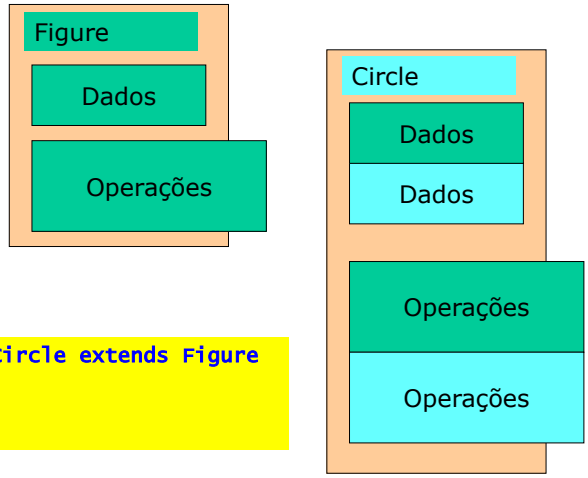
Herança

```
public class Figure {
   private   Point p;
   private   int cor;

   public void draw();
}

public class Circle extends Figure {
   //..
}
```



16

Herança




```
public class Circle extends Figure
{
   //..
}
```

17

Polimorfismo

- Capacidade de um objecto tomar várias formas
  - Uma figura pode ser um círculo, um quadrado, um triângulo, …
- Ligação Estática *vs* Ligação Dinâmica
  (*early binding* vs *late/dynamic binding)*
- Promove extensibilidade
- Depende da existência de herança

18

Garbage Collection (conceito opcional)

- Indispensável para alguns puristas da POO
  - Para outros .. nem tanto.
- Permite ao utilizador alguns mecanismos de controlo sobre a reserva e libertação de memória
- O utilizador pode implementar *garbage collection* automático através das classes (construtores e destrutores)
- Liberta o programador
  - … mas à custa de perda de desempenho.
- Linguagens
  - Java, Smalltalk, EIFFEL.

19

POO - Terminologia

- **Classes**
  - abstracção que caracteriza múltiplas entidades com comportamentos semelhantes.
  - Outros termos: Tipo, tipo de objecto
- **Métodos**
  - funções dentro de uma classe que actuam sobre as suas instâncias.
  - Outros termos: Mensagens, Funções, Operações
- **Atributos**
  - características (dados) numa classe que diferem de objecto para objecto
  - Outros termos: Propriedades, Dados

20

POO - Terminologia

- **Objectos**
  - concretização de uma classe numa entidade particular;
  - as instâncias de uma classe tem todas as mesmas operações, mesmas estruturas de dados, mas valores próprios.
  - Outros termos: Variável, Instância (de uma classe)
- **Encapsulamento e visibilidade**
  - Agregação de métodos e atributos numa entidade
  - interface privada - constituída por dados e métodos que não são visíveis pelo exterior de cada objecto;
  - interface pública - composta pelo conjunto de operações que o objecto disponibiliza.
  - Outros termos: *information hiding*

21