Programação Orientada a Objetos

1.1 - Perspectiva histórica:

Conceitos

A evolução das linguagens de programação tem-se feito na procura de ferramentas:

-cada vez mais próximas da percepção humana

- e que permitam lidar com problemas de maior dimensão e complexidade

procedimento => módulo => tipo abstracto de dados => orientação a objectos

A procura de mecanismos que permitissem a divisão de um problema complexo em sub-problemas começou por dar origem à noção de procedimento.

(+) Tornou possível raciocinar sobre unidades de menor dimensão

(-) A interdependência entre as estruturas de dados e as operações que as manipulam implica que se alterarmos a implementação de uma estrutura de dados temos também que alterar as suas operações

(-) Não sendo unidades de código independentes, não podiam ser compilados separadamente

O passo seguinte foi a criação de uma abstracção que permitisse definir um programa como um conjunto de entidades ou módulos, relativamente autónomos, que encapsulam dados e funcionalidade. Surgiu o conceito de módulo.

(+) Um módulo oferece uma interface para o exterior através da qual os seus dados podem ser manipulados.
A sua estrutura interna não é conhecida por outras entidades, podendo ser compilado e armazenado em bibliotecas para posterior utilização

(-) Não é possível criar dinamicamente várias instâncias de um mesmo módulo

O conceito de Tipo Abstracto de Dados (TAD) permite a definição de tipos de dados que incluem um conjunto de variáveis e as operações que os manipulam.

(+) A partir de um TAD podem ser instanciadas variáveis tal como para qualquer outro tipo de variável. O compilador poderá também verificar a coerência da utilização desse TAD.

Da evolução dos Tipos Abstractos de Dados surge finalmente o conceito de "Orientação a Objectos"

A ideia foi tornar o modelo anterior mais flexível, permitindo que um tipo seja progressivamente especializado, redefinindo ou incrementando a sua funcionalidade.

Pretende-se poder reutilizar o software já desenvolvido e testado, adicionando-lhe o comportamento que as novas aplicações necessitem.

Um conceito comum a todas as linguagens que se reclamam como "Orientadas a Objetos" é o de encapsulamento.

O encapsulamento é tradicionalmente importante em computação, na medida em que permite decompor grandes sistemas em sub-sistemas autónomos menores que podem ser mais facilmente desenvolvidos e mantidos.

A Programação Orientada a Objetos (POO) formaliza o encapsulamento, permitindo descrever um sistema como um conjunto de entidades ou "objectos" autónomos, no sentido de que o funcionamento de um objeto não depende da estrutura interna de outros objectos.

Um outro conceito importante em POO é o de herança. Através de mecanismos de herança é possível criar novos tipos de objetos partindo dos já existentes através da especificação de como o novo tipo difere do original.

Encapsulamento e herança vão permitir a **reutilização** dos objetos já definidos, sem modificação, para resolver novos problemas.

Linguagens

A noção de "objeto" em programação surgiu pela primeira vez no fim dos anos 60 com a linguagem *Simula (Simula-67):*

- extensão do ALGOL-60:

- desenvolvida por Ole-Johann Dahl e Kristen Nygaard no "Norwegian Computing Center"

- tinha como objectivos a descrição, simulação e modelação de sistemas de acontecimentos discretos

- um objecto em Simula correspondia a um fenómeno em estudo num sistema de referência

Um **objeto em Simula** era uma entidade que possuía:

- identidade (única)

- estrutura (definida pelos seus atributos ou propriedades)

- comportamento (definido pelas acções que podia realizar)

- interacção (forma de relacionamento com as outras entidades)

A terminologia de “Orientação a Objectos” surge nos anos 70 com a linguagem *Smalltalk*

- desenvolvida no ”Xerox Palo Alto Research Center”
pelo “Learning Research Group” liderado por Alan Kay

- pretendia-se uma linguagem para programação de aplicações cuja construção iria evoluindo à medida das necessidades do utilizador

A utilização eficiente dos tipos de objectos já definidos implica a existência de ferramentas adequadas à sua manipulação:

. O Smalltalk-80 é não só uma linguagem de programação,

mas também

um ambiente de desenvolvimento que permite através de um conjunto de interfaces gráficas, consultar e manipular todos os componentes do sistema.

A utilização das técnicas de Orientação a Objectos permaneceu restrita a um pequeno grupo até 1986:

Com a realização, nesse ano das conferências

- "Object-Oriented Programming Workshop" e

- 1ª "Int'l Conference on Object-Oriented Programming Languages Systems and Applications".

- Com o desenvolvimento da tecnologia de construção de workstations que permitem ambientes de programação sofisticados a computação OO teve rápida expansão.

Objective-C, C++, Visual C++ *(extensões da linguagem C)*

CLOS – Common Lisp Object System

ObjectPascal, Delphi (extenções da linguagem Pascal)

VisualBasic

…

Java

- linguagem desenvolvida no início dos anos 90 pela equipa da Sun Microsystems liderada por James Gosling.

C# (Microsoft .NET)

…

1.2 – Características da Programação Orientada a Objectos

Abstracção

"processo de formular conceitos gerais por extracção de propriedades comuns de exemplos específicos"

A abstração é uma ferramenta para manipular a complexidade,

quer na elaboração de modelos conceptuais,
onde os conceitos gerais são identificados por um nome
para abstrair sobre os atributos e comportamentos do conceito,

quer na resolução de problemas,

através da sua divisão em sub-problemas, os detalhes da solução de cada componente podem ser abstraídos pelo seu resultado

Muita da história da computação tem a ver com a criação de técnicas e ferramentas para suportar um processo de abstracção.
No entanto esse processo tem geralmente mais a ver com a arquitectura do hardware do que com o domínio dos problemas.

Por exemplo:
as abstracções fornecidas pelo C, Fortran ou Pascal (linguagens imperativas) estão directamente relacionadas com uma implementação eficiente em arquitecturas de Von Neumann.

Em oposição, as linguagens lógicas e funcionais baseadas em modelos matemáticos bem definidos, fornecem abstracções associadas com a representação dos problemas a resolver.

Têm no entanto a limitação de serem dirigidas a domínios específicos de problemas e são geralmente menos eficientes.

A filosofia da POO é também

fornecer abstracções que têm a ver com a representação dos problemas

mas com o objectivo mais geral de abarcar vários domínios de aplicação.

Enquanto na programação imperativa ou procedimental se considera por um lado as estruturas de dados que vão conter a informação do sistema e por outro os procedimentos que a vão manipular,

em POO cada fenómeno ou conceito relevante vai ser representado por uma entidade autónoma.

- Essas entidades, ou objectos, vão conter os dados e as operações necessárias para a definição do estado e do comportamento do conceito associado.

- Cada computação é expressa em termos de comunicação entre os vários objectos.

Em POO a abstracção é definida pelos seguintes princípios:

. Abstracção de dados (1)

. Partilha de comportamento (2)

. Evolução (3)

1 - Abstracção de dados

- Consiste em abstrair uma entidade, através de um conjunto de operações que definem a sua interface externa ou comportamento

As técnicas de abstracção de dados, às quais é dada a designação genérica de encapsulamento, têm a ver com dois tipos de mecanismos:

- modularização e
- restrições de acesso à informação

Modularização

- consiste na divisão de um sistema em entidades ou módulos, contendo cada um todas as estruturas de dados e algoritmos necessários para implementar essa parte do sistema.

- cada módulo pode ser facilmente reutilizado, e eventuais alterações na funcionalidade de uma entidade não põem em causa a funcionalidade dos restantes componentes.

Restrições de acesso à informação

- através destes mecanismos é possível controlar o acesso aos vários elementos de uma entidade, sejam estruturas de dados, ou procedimentos.

-os “utilizadores” de um módulo não têm autorização para manipular os seus detalhes internos, isto é, a única forma de aceder a uma entidade será invocar uma das suas operações externas.

2 - Partilha de comportamento

A abstracção de dados introduz o conceito de comportamento de uma entidade, como o conjunto de operações externas dessa entidade.

O princípio da partilha de comportamento estende esse conceito, permitindo que várias entidades partilhem um mesmo conjunto de operações.

O mecanismo mais usual é o da classificação.

Através deste um grupo de entidades pertencentes à mesma classe vai ter um comportamento comum.

Um refinamento da classificação consiste em estabelecer uma relação de inclusão entre classes.

Dada uma classe C com um determinado comportamento, pode definir-se uma classe C1, derivada de C, que contém o comportamento de C mais um conjunto de operações exclusivas dos elementos de C1.

O comportamento da classe C é partilhado pelos elementos das classes C e C1 e eventualmente por elementos de outras classes, definidas a partir de C1, formando uma hierarquia de classes.

3 - Evolução

Em computação os requisitos de um sistema evoluem rapidamente. Podemos considerar evolução como

a necessidade de adicionar novas funcionalidades e de modificação das existentes, ou,

para sistemas em que os objectivos finais não estejam bem definidos, a evolução pode consistir num desenvolvimento incremental, em que as várias entidades vão sendo sucessivamente especializadas até à solução completa.

A filosofia da POO é suportar ambos os tipos de evolução

Resumindo,

a diferença importante da orientação a objectos em relação à programação convencional (imperativa ou procedimental)

consiste na capacidade de "estender" um sistema por simples adição de novo código, em situações onde, com técnicas convencionais, seria necessário modificar o código anterior.

O modelo mais simples de linguagem Orientada a Objectos é:

"Uma linguagem é Orientada a Objectos, se suporta Objectos, se esses objectos pertencem a Classes e se hierarquias de classes podem ser definidas, incrementalmente, por um mecanismo de Herança."

O Objecto constitui a unidade fundamental de encapsulamento, enquanto Classes e Herança realizam os princípios de partilha de comportamento e evolução.