Programação por Objectos
Introdução

História (1)

- [60s] **Simula-67**, Dahl e Nygaard da Univ. de Oslo
  - Primeira linguagem com conceitos OO.
- [70s] **Smalltalk**, da Xerox
  - Primeira implementação prática de uma linguagem OO.
- [80s] **C++**, Stroustrup do Bell Labs
  - Extensão do C com conceitos OO.
  - Popularizou a metodologia de programação OO.

História (2)

- [90s] **Java**, James Gosling da Sun:
  - Sintaxe próxima do C.
  - Programas executados numa máquina virtual. Este facto torna o Java muito lento, em cerca de 50 vezes pior que programas equivalentes em C.
  - Interfaces,apenas com os protótipos dos métodos.
  - Muitas bibliotecas disponibilizadas. A mais famosa é o JNI (*Java Native Interface*), que permite aceder a programas desenvolvidos noutras linguagens.
  - Código muito compacto (ambiente executado no cliente) atraiu projectistas do WWW.

História (3)

- [00s] **C#** (*C-sharp*), da Microsoft
  - Derivado do Java para a plataforma **.NET**.

Estado actual – Fev 2010

Popularidade das principais linguagens de programação, nas interrogações normalizadas de pesquisas no Google e Yahoo!

1. Java: 17.35%
2. C: 16.60%
3. PHP: 10.00%
4. C++: 9.45%
5. Visual Basic: 7.05%

Indice TIOBE Programming Community:
http://www.tiobe.com/tpci.htm

Motivação (1)

- Distribuição dos recursos humanos na empresa típica:
  - 80% dedicados à manutenção dos sistemas existentes.
  - 20% estão livres para responder às necessidades de novas aplicações.

Motivação (2)

- A nível da manutenção:
  - O crescimento dos seus custos deve-se, em boa parte, à dificuldade crescente de fazer alterações sobre alterações.
  - As sucessivas alterações são tipicamente realizadas sobre código com o consequente afastamento das especificações iniciais que não vão sendo actualizadas.
  - Alterações ulteriores requerem trabalhar em exclusivo com o código sem o apoio importante das descrições de mais alto nível (produzidas na fase de desenho e análise).

Motivação (3)

- O aumento da produção de *software* pode-se conseguir de duas formas distintas:
  - Aumentando o número de analistas e programadores.
  - **Aumentando a sua produtividade e/ou promovendo a re-utilização de componentes de *software* já existente.**

Motivação (4)

- A redução das tarefas de manutenção aponta para uma automação cada vez maior na transformação das especificações em código.
- Neste sentido, surgiram as **ferramantas CASE** (*Computer-Aided Software Engineering*) que permitem integrar actividades que abrangem o desenho, análise, programação e testes.

Conceitos primitivos da abordagem OO (1)

- **Objecto**:
  - Algo que existe (no mundo real ou sistema informático), que é criado e eventualmente destruído.
  - Entretanto, enquanto existe, pode sofrer alterações no seu **estado** por interação com outras entidades.
  - O seu estado reflecte-se no valor dos **atributos**.
  - Muda de estado quando algum **método** é chamado.
- Por exemplo:
  - Objecto: conta bancária.
  - Atributos: saldo.
  - Método: débito, crédito, juro.
- **Especificar um sistema consiste em definir um conjunto de objectos.**

Conceitos primitivos da abordagem OO (2)

- **Classe**:
  - Em geral surgem muitos objectos similares, do mesmo tipo, que são agrupados em classes.
  - Na realidade temos de especificar as classes e não os objectos, os objectos surgem como **instâncias** das classes.
- Por exemplo:
  - Especificamos a classe conta bancária.
  - Admitimos que essa conta possa ter muitas instâncias.
- **O sistema assim construído consiste numa comunidade de objectos (instâncias de classes) que interactuam uns com os outros (através de chamadas a métodos).**
- **Os objectos da comunidade evoluem independentemente uns dos outros excepto nos momentos de interacção.**

Ingredientes da abordagem OO (1)

- **Encapsulamento**:
  - Agrupamento de ideias relacionadas numa unidade, que pode então ser referênciada apenas por um nome.
  - Em OO, refere-se ao agrupamento de atributos e métodos. Na realidade trata-se do encapsulamento do estado conjuntamente com os mecanismos para aceder e modificar o estado.

Ingredientes da abordagem OO (2)

- **Especialização/herança**:
  - Mecanismo que promove a re-utilização.
  - A herança permite que um objecto seja simultâneamente uma instância de mais do que uma classe.
  - O mecanismos de herança da **subclasse** B sobre a **superclasse** A permite à classe B reaproveitar tudo o que se queira da classe A.
  - Herança **simples**, onde uma classe tem apenas uma superclasse directa, versus herança **múltipla**, onde uma classe pode ter várias superclasses directas.

Ingredientes da abordagem OO (3)

- **Especialização/herança** (cont):
  - Por exemplo:
    - Quer ainda na fase de especificação, quer na fase de manutenção, pode surgir a necessidade de introduzir contas poupança que são contas, mas que têm algumas pecularidades (mais atributos, mais métodos, comportamento mais constrangido).
    - A especialização permite especificar a classe conta poupança reaproveitando tudo o que se queira da especificação de conta (saldo, depósito).
    - Outros objectos que interactuem com contas também poderão interactuar com contas poupança, encarando-as como contas (sem dar pelas referidas peculiariedades).

Ingredientes da abordagem OO (4)

- **Polimorfismo**:
  - **Redefinição** de um método com o mesmo identificador em classes distintas, podendo este ter diferentes implementações em cada uma destas classes.
  - Em OO, o polimorfismo é normalmente implementado através de **ligação dinâmica**, i.e., o método a ser executado é determinado apenas em tempo de execução (e não em tempo de compilação).

Ingredientes da abordagem OO (5)

- **Polimorfismo** (cont):
  - Por exemplo:
    - A conta poupança redefine o método de juro, pois o juro desta conta não é calculado da mesma forma que o juro da conta.
    - Em tempo de compilação, o sistema oferece uma forma de actualizar o juro das contas (sem distinguir conta de conta poupança), percorrendo de alguma forma todas as contas do banco e chamando a estas o método juro.
    - Em tempo de execução, o método juro a ser executado é determinado pelo tipo de conta sobre o qual o método foi chamado.

Pros/cons da abordagem OO

- Vantagens:
  - Aproximação do mundo real.
  - Encapsulamento da informação.
  - Extensível, sendo mais fácil alterar e/ou acomodar novos requisitos.
  - Reutilização, por herança de classes mais gerais.
- Desvantagens:
  - Abordagem nova, com conceitos mais complexos.
  - Desempenho inferior.

Aplicações da abordagem OO

- Banca e seguros
- Robótica
- Telecomunicações
- Desenho VLSI
- Simulação
- Bases de dados
- Modelação matemática
- Controlo de tráfego aéreo
- Aplicações gráficas
- …