Programação em Comunicações

"Programação Orientada por Objectos"

Ademar Aguiar
www.fe.up.pt/~aaguiar
ademar.aguiar@fe.up.pt

Universidade do Porto
Faculdade de Engenharia
FEUP

Objectivos

- Apresentar os princípios e conceitos base sobre orientação por objectos (objectos, classes, encapsulamento, abstracção, herança, polimorfismo) com vista a facilitar a aprendizagem e prática de programação orientada por objectos.
- Apresentar exemplos de classes escritas em Java.

Bibliografia

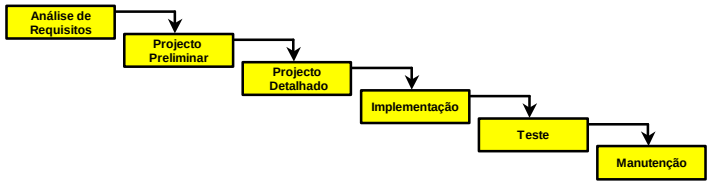
- Grady Booch, *Object-oriented Design With Applications 2nd Ed.*, The Benjamin/cummings Publishing Company Inc., Redwood City, California, 1995.
- Martin Fowler. *Uml Distilled - Applying the standard object modeling language*. Addison-Wesley, 1997.

Introdução

- Questões
  - O que é orientação por objectos?
  - O que é um objecto?
  - O que é uma classe?
  - Exemplo: "Editor Gráfico"

Engenharia de Software

- A engenharia de software é o ramo de engenharia que visa o desenvolvimento de sistemas de **software de qualidade**
- Os **modelos** do ciclo de vida do desenvolvimento de software compreendem várias fases bem definidas. Exemplo: "modelo clássico"



Qualidade de Software

- Qualidade é uma noção multifacetada descrita por um conjunto de **factores externos** e **internos**
- O software de qualidade deve ser:
  - **correcto**,
  - fácil de usar,
  - **robusto**,
  - eficiente,
  - expansível,
  - portável,
  - **reutilizável**,
  - verificável,
  - compatível,
  - íntegro,
  - **modular** (interno),
  - compreensível (interno)

Complexidade do Software

- Sistemas de software simples e complexos
  - Os sistemas especificados, construídos, mantidos e usados por uma mesma pessoa, são normalmente **sistemas simples**
  - Os sistemas com que nos deparamos na indústria de software são normalmente **sistemas complexos**
  - A compreensão de todas as subtilezas de um sistema complexo é muito difícil de conseguir por uma só pessoa
  - Os sistemas complexos quase sempre excedem a capacidade intelectual humana
  - A complexidade diz-se ser uma propriedade inerente ao software porque, apesar de contornável, não é eliminável

Complexidade do Software...

- Porque razão é o software inerentemente complexo?
  - A complexidade do **domínio do problema**
  - A dificuldade na **gestão do processo de desenvolvimento**
  - A **flexibilidade possível** no software
  - A dificuldade em caracterizar o comportamento de sistemas **discretos**
- Consequências de não restringir a complexidade
  - Quanto mais complexo é um sistema mais sujeito está a ruptura
  - Projectos terminados para além dos prazos e custos previstos
  - Projectos que não cumprem na íntegra os requisitos iniciais

Sistemas Complexos

- Cinco Atributos
  - Hierarquia de vários subsistemas interrelacionados, até aos componentes primitivos
  - A escolha dos componentes primitivos dum sistema é arbitrária, e é subjectiva ao observador
  - Os sistemas hierárquicos são compostos por poucos tipos de subsistemas diferentes, mas dispostos em combinações várias
  - É possível separar as relações que envolvem a estrutura interna dos componentes, das relações entre componentes
  - **Um sistema complexo que funcione é fruto da evolução dum sistema mais simples que funcionava**

Técnicas para Organizar o CAOS

- **Decomposição** ("dividir para reinar")
  - Decomposição algorítmica: ilustra a ordem dos eventos
  - Decomposição orientada por objectos: mostra quais os agentes activos ou passivos nos eventos
- **Abstracção**
  - Organização dos estímulos em várias dimensões e sucessivamente em sequências de blocos de informação
  - Consiste em ignorar detalhes não essenciais para o problema
- **Hierarquia**
  - A hiearquia de objectos (agregações) ilustra os padrões de interacção entre objectos diferentes (mecanismos)
  - A hierarquia de classes (generalizações) evidencia a redundância do sistema

Orientação por Objectos

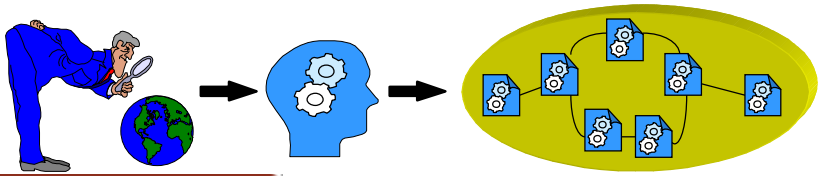
Orientação por Objectos

- Orientação por objectos (OO) é bem mais do que um paradigma de programação
- Aplica-se a várias actividades relacionadas com tecnologias de informação: análise de requisitos, processos, desenho de software, construção e teste de sistemas de software
- A ideia central consiste na construção de modelos de sistemas em torno de entidades que unificam dados e procedimentos: **objectos**

Objectos

"Um **objecto** é algo que possui **estado**, **comportamento** e **identidade**, e pode ser definido de uma forma bastante precisa"

- A **estrutura** e o **comportamento** de objectos idênticos são definidos na sua **classe** comum
- Cada objecto é uma **instância** de uma classe
- Na abordagem OO, as entidades do mundo real são vistas como uma colecção de objectos cooperantes



FEUP Universidade do Porto Faculdade de Engenharia — MRSC, Programação em Comunicações 13

Objectos ...

- Os objectos correspondem directamente a coisas, pessoas, ou processos existentes no mundo real

- Num modelo orientado por objectos, as características do mundo real são representadas por **atributos** e os comportamentos por **operações**

FEUP Universidade do Porto Faculdade de Engenharia — MRSC, Programação em Comunicações 14

Princípios OO Fundamentais

- Abstracção...
  - Ilustra as características essenciais de um objecto
- Encapsulamento...
  - Oculta os detalhes de um objecto
- Modularidade...
  - É a propriedade de um sistema que foi decomposto num conjunto de módulos coesos e fracamente ligados
- Hierarquia...
  - Ordena as abstracções de um sistema
- Tipos...
  - Associam os objectos às suas classes, restringindo sua troca
- Concorrência...
  - É a propriedade que distingue objecto activos de passivos
- Persistência...
  - É a propriedade de um objecto através da qual a sua existência transcende o tempo e/ou espaço

Abstracção

"A abstracção salienta as características dum objecto consideradas essenciais, e que o distinguem de todos os outros tipos de objectos, fornecendo portanto contornos muito bem definidos relativamente à perspectiva do utilizador"

- A abstracção encerra todo o comportamento importante dum objecto
- Exemplos
  - listas, pilhas, conjuntos, vectores, ...
  - quadrados, círculos, rectângulos, triângulos, ...

Encapsulamento

"Encapsulamento é o processo de esconder todos os detalhes dum objecto que não contribuem para a definição das suas características essenciais"

- O encapsulamento facilita a alteração e evolução
- A abstracção e o encapsulamento são complementares
  - A abstracção incide sobre a visão externa dum objecto (**interface**)
  - O encapsulamento oculta o seu interior, colocando uma barreira entre os níveis lógico e físico da representação (**implementação**)

Modularidade

"Modularidade é a propriedade dum sistema que foi decomposto num conjunto de módulos coesos e fracamente interligados"

- Uma divisão em módulos feita de forma arbitrária pode ser pior do que se nenhuma divisão for feita
- Os módulos podem de ser vistos como recipientes físicos nos quais declaramos as classes e objectos
- No projecto estruturado, a modularização consiste em agrupar subprogramas
- No projecto OO, o problema consiste em agrupar classes e objectos

Hierarquia

"Hierarquia é um conjunto ordenado de abstracções"

- O nosso "brainware" é limitado!
  - O número de abstracções diferentes que temos que identificar, é quase sempre superior às que conseguimos entender em simultâneo
- O encapsulamento oculta detalhes
  - Simplifica a visão das abstracções, ao ocultar o seu interior
- Os módulos ajudam a agrupar conceitos
  - Ajudam a agrupar as abstracções logicamente relacionadas
- A identificação de hierarquias ajuda imenso
  - A identificação das hierarquias dum sistema simplifica a nossa compreensão do problema, ao dar uma visão ordenada das abstracções

Herança

- O mecanismo de **herança** explora a hierarquia "kind of", e é um elemento essencial dos sistemas OO.
- Herança define uma relação entre classes, em que uma classe partilha a estrutura ou comportamento definido numa ou mais classes (herança simples ou múltipla)
- Superclasses representam abstracções generalizadas, enquanto que as subclasses representam especializações das superclasses

Tipos e Polimorfismo

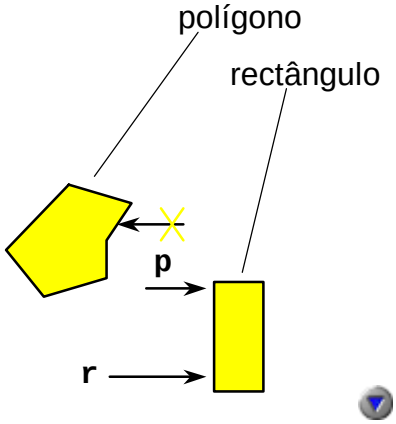
- O polimorfismo é um conceito da teoria de tipos no qual um único nome pode referir objectos de classes diferentes, se relacionadas por superclasses comuns
- O **polimorfismo** surge quando a linguagem de programação suporta herança e a **associação dinâmica** de tipos
- O polimorfismo é talvez a capacidade mais poderosa das linguagens OO, a seguir à capacidade de abstracção

Exemplo: Polimorfismo

```
class Polygon {void draw();};

class Rectangle:Polygon {void draw();};

main() {
   Polygon p;
   Rectangle r;
   ...
   p=r; //correcto !
   p.draw();
   ...
   r=p; //incorrecto !!
}
```



Concorrência

"Concorrência é a propriedade que distingue um objecto activo dum outro que não é"

- A concorrência incide sobre a abstracção e sincronização de processos
- Aos objectos que representam uma abstracção dum processo, chamam-se objectos **activos**
- Sistema OO = um conjunto de objectos que cooperam entre si, alguns dos quais são activos e servem como centros de actividade independente (agentes?)

Persistência

"Persistência é a propriedade dum objecto através da qual a sua existência transcende o **tempo** (isto é, continua a existir após o seu criador ter deixado de existir) e/ou **espaço** (isto é, a localização do objecto muda-se do espaço no qual foi criado)

- Existem objectos com tempos de vida desde o **transitório** até ao **persistente**
- A unificação dos conceitos de concorrência e de objectos dá origem a **linguagens de programação concorrente orientadas por objectos**
- A introdução do conceito de persistência no modelo de objectos, dá origem às **bases de dados orientadas por objectos**

Vantagens da OO

- O modelo de objectos é diferente dos modelos seguidos pelos métodos mais tradicionais de análise, projecto e programação
- Introduz elementos novos que assentam nos antigos
- Orienta na construção de sistemas que incorporam os cinco atributos dum sistema complexo
- Auxilia na exploração do poder expressivo das linguagens de programação orientada por objectos
- O uso do modelo de objectos incentiva a reutilização, não apenas de software (bibliotecas de classes), mas também de projectos inteiros (frameworks)
- O uso do modelo de objectos produz sistemas construídos sobre formas intermédias estáveis, por conseguinte mais flexíveis a alterações e mais capazes de resistir ao tempo
- O modelo de objectos reduz o risco de desenvolvimento de sistemas complexos

Resumo

- O modelo de objectos compreende os princípios da abstracção, encapsulamento, modularidade, hierarquia, tipos, concorrência e persistência
- Uma abstracção ilustra as características essenciais de um objecto
- Encapsulamento é o processo de ocultar os detalhes de um objecto
- Modularidade é a propriedade de um sistema que foi decomposto num conjunto de módulos coesos e fracamente ligados
- Hierarquia é uma ordenação das abstracções de um sistema
- Tipos associam os objectos às suas classes, restringindo a troca de objectos diferentes
- Concorrência é a propriedade que distingue um objecto activo de um passivo
- Persistência é a propriedade de um objecto através da qual a sua existência transcende o tempo e/ou espaço


Fim
FEUP Universidade do Porto Faculdade de Engenharia
MRSC, Programação em Comunicações
27