Orientação a Objetos
Conceitos

Engenharia de Software I - Prof. Antonio Guardado

Centro Paula Souza

GOVERNO DO ESTADO SÃO PAULO

1 - Antes de Começar - Histórico

- 1967: Simula - introduz os primeiros conceitos de O O
- 1972: Smalltalk
- 1980: C++ linguagem híbrida, derivada da linguagem C
- 1983: Ada criada para uso militar nos EUA
- 1984: Eilffel primeiras características formais de O O
- 1986: O bject pascal
- 1995: JAVA - Linguagem puramente orientada a objetos
- 1995: Várias linguagens agregando conceitos de O O

1.1 - Simula

- A primeira linguagem a incorporar facilidades para definir classes de objetos genéricos na forma de uma hierarquia de classes e subclasses
- Foi idealizada em 1966, na Noruega, como uma extensão da linguagem ALGOL 60.
- Uma classe em Simula é um módulo englobando a definição da estrutura e do comportamento comuns a todas as suas instâncias (objetos).

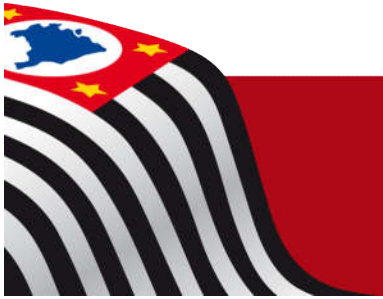
1.2 - Smalltalk

- Smalltalk foi desenvolvida no Centro de Pesquisas da Xerox durante a década de 70
- Incorporou idéias de Simula
- Criou o princípio de objetos ativos, prontos a "reagir" a "mensagens" que ativam "comportamentos" específicos do objeto

1.3 - C++

- Questões no projeto de C++
  - ❑ Ser melhor do que C
  - ❑ Suportar abstração de dados
  - ❑ Suportar programação orientada a objetos
- C++ foi projetada para dar suporte a abstração de dados e programação orientada a objetos

- C++ não impõe um paradigma



1.4 - Ada

- Ada é uma linguagem de programação criada através de um concurso realizado pelo U.S. Departament of Defense (DoD)O principal projetista da equipe foi o francês Jean Ichbiah.
- Esse concurso foi feito para por ordem na situação, o DoD em 1974 usava cerca de 450 linguagens ou dialetos de programação.
- A linguagem foi primeiramente padronizada em 1983 pelo ANSI e em 1985 a Organização Internacional de Padronozação (ISO).

1.5 - Eiffel

- Eiffel é uma Linguagem de Programação avançada, puramente orientada a objeto que enfatiza o projeto e construção de software reusável e de alta qualidade.
- Eiffel foi criada por Bertrand Meyer que tinha uma extensa experiência com programação orientada a objeto, particularmente com SIMULA.

1.6 - Object Pascal

- O Object Pascal é uma linguagem orientada a objetos, isto é, todas as informações são tratadas como objetos
- Todos estes objetos pertencem a uma classe, que são categorias de objetos
- Delphi / Kylix / Lazarus são exemplos de ferramentas que utilizam esta linguagem.

1.7 - Java

- O Java é ao mesmo tempo um ambiente e uma linguagem de programação desenvolvida pela Sun Microsystems, Inc.
- Trata-se de mais um representante da nova geração de linguagens orientadas a objetos e foi projetado para resolver os problemas da área de programação cliente/servidor.
- Os aplicativos em Java são compilados em um código de bytes independente de arquitetura.

1.7 - Java (2)

- Esse código de bytes pode então ser executado em qualquer plataforma que suporte um interpretador Java.
- O Java requer somente uma fonte e um binário e, mesmo assim, é capaz de funcionar em diversas plataformas, o que faz dele um sonho de todos os que realizam manutenção em programas.

2 - Evolução da Orientação a Objetos

- Não aconteceu de repente. Evoluiu de idéias
  - Larry Constantine - (Década de 1960) Foi quem primeiro lançou a idéia de que softwares poderiam ser projetados antes que fossem programados
  - O. J. Dahl e K. Nygaard - (1966) - Foi quem primeiro lançou a idéia de Classes introduzida na linguagem Simula
  - Alan Klay, Adele Goldberg e outros - (1970) Iniciaram o conceito de Mensagem e Herança, usados na linguagem SmallTalk.

2 - Evolução da Orientação a Objetos

- Na programação estruturada temos funções (procedures ou rotinas) e dados (normalmente globais) que podem ser acessados por qualquer função;

- Na programação orientada a objetos, temos funções agregadas aos dados em uma unidade chamada objeto, ou seja, os dados não estão separados das funções, mas sim unidos as mesmas

3 – Conceitos OO

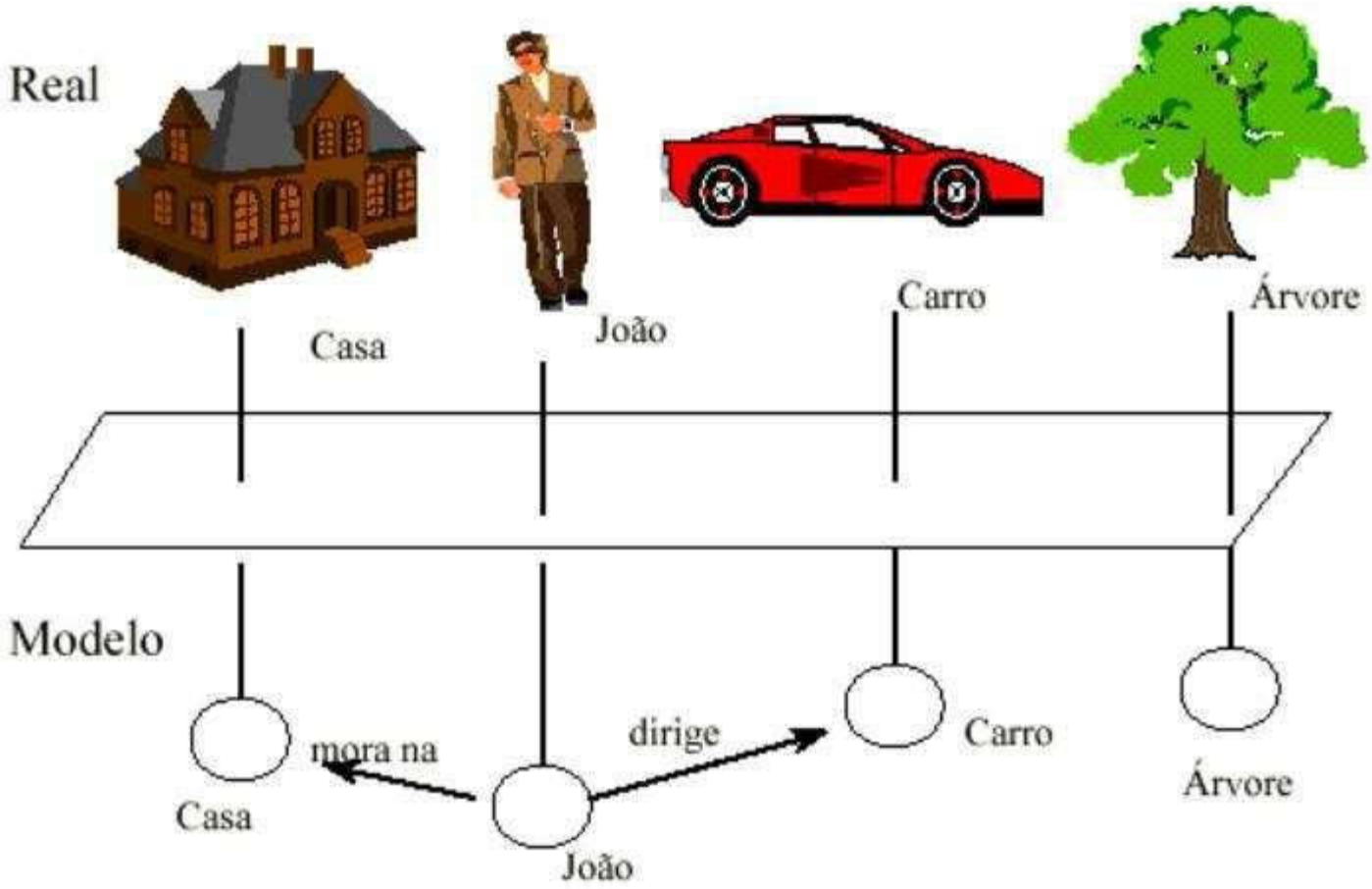
3.1 - Objetivos

- Entender os termos e tecnologias ligadas a Orientações a Objetos
- Relacionar no futuro com a linguagem UML
- Rever métodos de programação na forma de objetos

3.2 - Base

- Na compreensão do mundo, os seres humanos utilizam-se de três métodos de organização dos pensamentos:
  - Diferenciação
  - Distinção entre todo e parte
  - Classificação
- A orientação a objetos, como técnica para modelagem de sistemas, utiliza estes métodos para diminuir a diferença semântica entre a realidade e o modelo

Exemplo


Real
Casa
João
Carro
Árvore
Modelo
mora na
dirige
Casa
Carro
Árvore
João

3.3 - Principais Conceitos

- Objetos e Instâncias
- Classes
- Atributos
- Métodos
- Visibilidade de atributos e operações
- Mensagens

3.3.1 - Objetos

- Um objeto denota uma entidade, seja ela de natureza física, conceitual ou de software.
  - ❑ Entidades físicas: um carro, uma pessoa, uma casa
  - ❑ Entidade conceitual: um organograma de uma empresa
  - ❑ Entidade de software: um botão em uma GUI

3.3.1 - Objetos

- É uma entidade capaz de reter um estado (atributos) e que oferece uma série de operações (métodos) ou para examinar ou para afetar este estado
- Um objeto é um conceito, uma abstração, algo com limites e significados nítidos em relação ao domínio de uma aplicação
- Objetos facilitam a compreensão do mundo real e oferecem uma base real para implementação em computador

3.3.1 - Objeto

- Um objeto é algo que tem:
  - ❑ Estado
  - ❑ Comportamento
  - ❑ Identidade

3.3.1.1 – Estado do Objeto

- O estado de um objeto representa uma das possíveis condições em que um objeto pode existir
- O estado é representado pelos valores das propriedades de um objeto em um determinado instante
- O estado do objeto usualmente muda ao longo do tempo:
- Exemplo (Aluno Maurício):
  - Nome: Maurício
  - Matrícula: 105018
  - Semestre de Ingresso: 2000A

3.3.1.1 – Estado do Objeto - Exemplo

- Exemplo (Carro):
  - Marca: Volkswagen
  - Modelo: Gol
  - AnoModelo: 2000
- Exemplo (Modelo)
  - Nome: Gol
  - Motor: 1.0 8v Flex

3.3.1.2 - Comportamento do Objeto

- O comportamento determina como um um objeto pode responder a interações mediante à ativação de operações decorrentes de mensagens recebidas de outros objetos
- O Comportamento é determinado pelo conjunto de operações que o objeto pode realizar.
- Controle Academico
  - Maurício
  - Solicita matrícula
  - Retorna: 105018

3.3.1.2 - Comportamento do Objeto

- O comportamento é dado pelo conjunto de métodos definidos para o objeto
- Estes métodos normalmente obtem ou alteram o estado do objeto

3.3.1.3 – Identidade do objeto

- Cada objeto tem um único identificador, mesmo que seu estado seja idêntico ao de outro objeto
  - Maurício (objeto)
  - Controle Acadêmico (Sistema que está sendo construído)
  - Semestre (estado)
  - Matrícula 105018 (Propriedade de um aluno)
  - Lista de Semestres Cursados (candidato a objeto)
  - Semestre corrente (o mesmo que semestre)

3.3.1.3 – Identidade do objeto

- A identidade de um objeto é passada através de um número de referência
- Cada objeto no espaço de memória terá seu próprio número de referência (único)
- Um objeto pode se referenciar a outro
  - Mas lembre-se: neste caso não existem dois objetos

3.3.2 - Classe

- Uma classe é a descrição de um grupo de objetos

com:

- propriedades Semelhantes (atributos)
- mesmo comportamento (métodos)
- mesmos relacionamentos com outros objetos
  - (associações e agregações)
- e mesma semântica

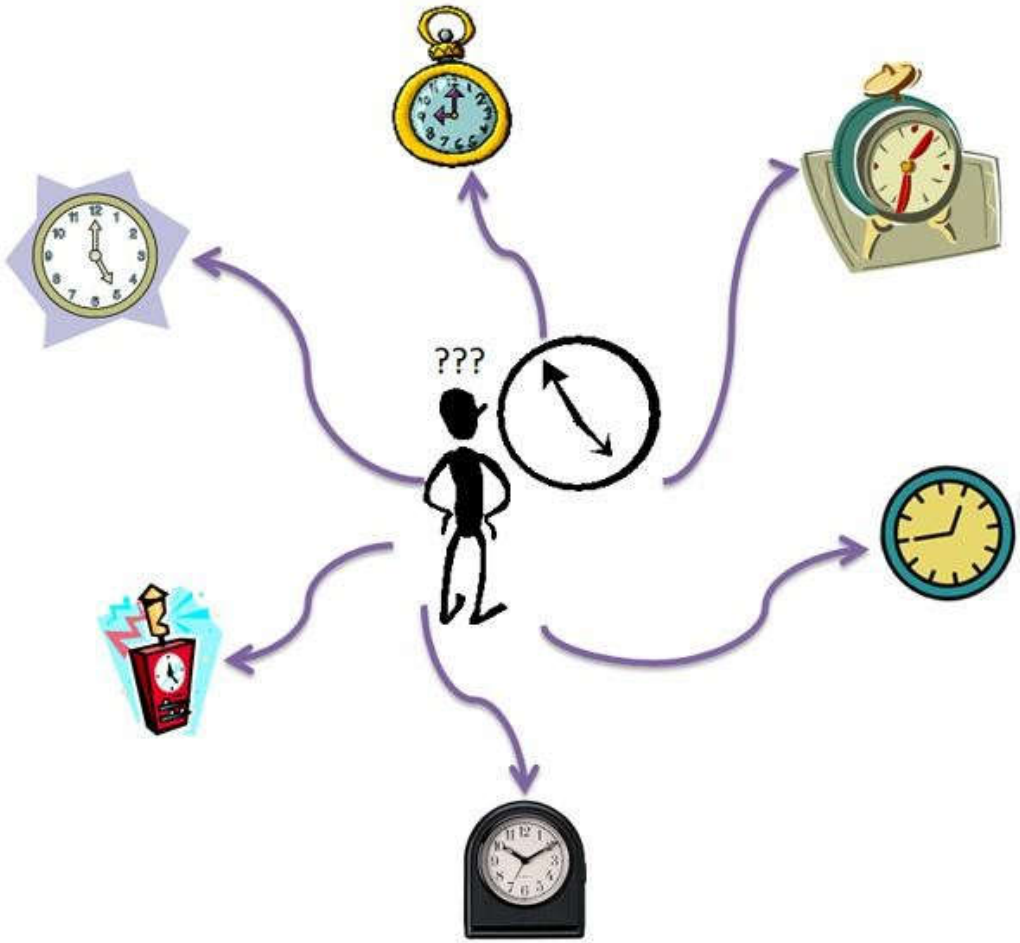
- Um objeto é uma instância de uma classe

3.3.2 - Classe

Relação de Classe com Objeto

- Uma classe é a definição de um objeto (é a forma para gerar objetos do mesmo tipo)
- Um objeto é a classe se tornando real (instância)

Classes e Objetos



3.3.2 - Classe

- Uma classe é uma abstração que
  - Enfatiza características relevantes
  - Abstrai outras características
  - Abstração: ajuda lidar com a complexidade

3.3.2 – Classes - Exemplos

- Professor
  - Atributos: Nome, Matrícula, Data de Contratação,Titulação
  - Métodos: DefineNome(), AlteraNome(), TempoServiço(), DefineTitulação(), AlteraTitulação(), …
- Objeto: (Professor) Matias
  - Nome: Luis Matias
  - Matricula: 8967.2
  - Data de Contratação: 20/09/2009
  - Titulação: Mestre

3.3.2 – Classes - Exemplos

- Turma:
  - ❑ Atributos: Cod, Nome, Local, Créditos, Horário, Capacidade
  - ❑ Métodos: DefineCod(), AlteraCod(), DefineNome(), AlteraNome(), NrCreditos(), AdicionaAluno(), EliminaAluno(), VerificaEstado(), . .

3.3.2.1- Definindo Classes

- Uma classe deveria capturar uma e somente uma abstração chave.

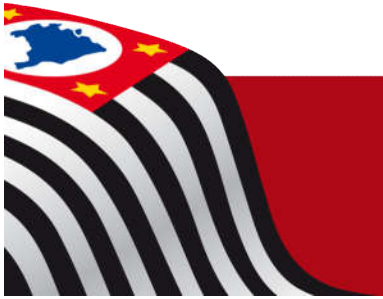
  - Abstração ruim: classe "Aluno" que conhece a informação do aluno e as disciplinas que aquele aluno está matriculado.

- Boa abstração: separar em uma classe para Aluno e uma classe para Disciplina

3.3.2.2 - Nomeando Classes

- Uma classe deveria ser um substantivo singular que melhor caracteriza a abstração
- Dificuldades na nomeação das classes podem indicar abstrações mal definidas

- Nomes deveriam surgir diretamente do domínio do problema



3.3.2.2 - Nomeando Classes - Estilo

- Um guia de estilo deveria ditar convenções de nomeação para classes
- Uma proposta simples:
  - Classes são nomeadas com um substantivo no singular
  - O nome de uma classe inicia com a primeira letra maiúscula
  - Não são utilizados símbolos de sublinhado ("_") nomes compostos de múltiplas palavras são organizados com todas as palavras juntas, onde a primeira letra de cada uma fica em maiúscula
- Exemplos: Aluno, Professor, ControleAcadêmico

3.3.3 - Atributo

- O estado de um objeto é dado por valores de atributos e por ligações que tem com outros objetos
- Todos os objetos de uma classe são caracterizados pelos mesmos atributos, ou variáveis de instâncias
- O mesmo atributo pode ter valores diferentes de objeto para objeto

3.3.3 - Atributo

- Atributos são definidos ao nível da classe, enquanto que os valores dos atributos são definidos ao nível do objeto
- Exemplos:
  - uma pessoa (classe) tem os atributos nome, data de nascimento e peso
  - João é uma pessoa (objeto da classe pessoa) com nome "João da Silva", data de nascimento "18/03/1973" e peso "70Kg"

3.3.4 - Métodos

- O comportamento invocável de objetos são os métodos
- Um método é algo que se pode pedir para um objeto de uma classe fazer
- Objetos da mesma classe tem os mesmos métodos
- Métodos são definidos ao nível de classe, enquanto que a invocação de uma operação é definida ao nível de objeto

3.3.4.1 - Método Construtor

- É um método utilizado para inicializar objetos da classe quando estes são criados
- Este método possui o mesmo nome da Classe e não tem nenhum tipo de retorno, nem mesmo void
- Normalmente é utilizado para inicializar o estado inicial do objeto

3.3.4.2 – Visibilidade do Método

- Atributos e Métodos Públicos (+)
  - São atributos e métodos dos objetos que podem ser visíveis externamente, ou seja, outros objetos poderão acessar os atributos e métodos destes objetos sem restrições
- Atributos e Métodos Privados(-)
  - Só podem ser acessados por operações internas ao próprio objeto
- Atributos e Métodos Protegidos (#)
  - Só podem ser acessados por operações internas ou de objetos que tenham a classe na hierarquia abaixo (subclasses)

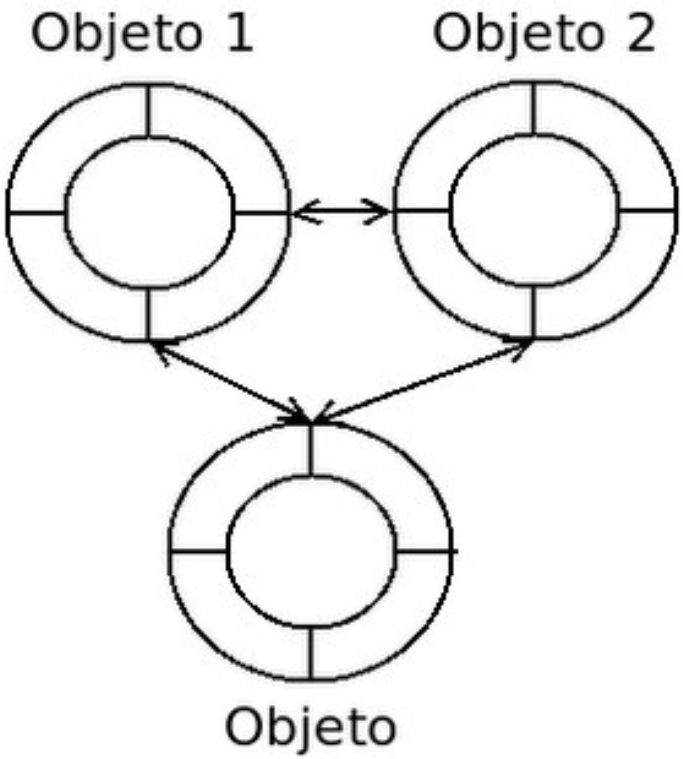
3.3.4.3 – Interface e Assinatura

- Interface
  - Os métodos públicos de uma classe definem a interface da classe. Os métodos privativos não fazem parte interface da classe pois não pode ser acessada externamente
- Assinatura
  - Deve ser definido o nome dos métodos, se tem tipo de retorno, e se recebe parâmetros, a este conjunto de informações sobre o método dá-se o nome de assinatura do método

3.3.5 - Mensagens

- Uma mensagem é uma solicitação feita por um objeto A a um objeto B
- Como resultado desta solicitação, o objeto B irá modificar seu estado ou irá retornar algum Valor
- O conceito de mensagem está diretamente associado ao conceito de operação . A interação entre os objetos é feita através da troca de mensagens

Troca de Mensagens



3.3.6 – Hierarquia de classes

- Quando vamos trabalhar com um grande conjunto de classes de objetos, é necessário organizar estas classes de maneira ordenada de modo que tenhamos uma hierarquia
- Em uma hierarquia de classes teremos as classes mais genéricas no topo, e as mais específicas na base.

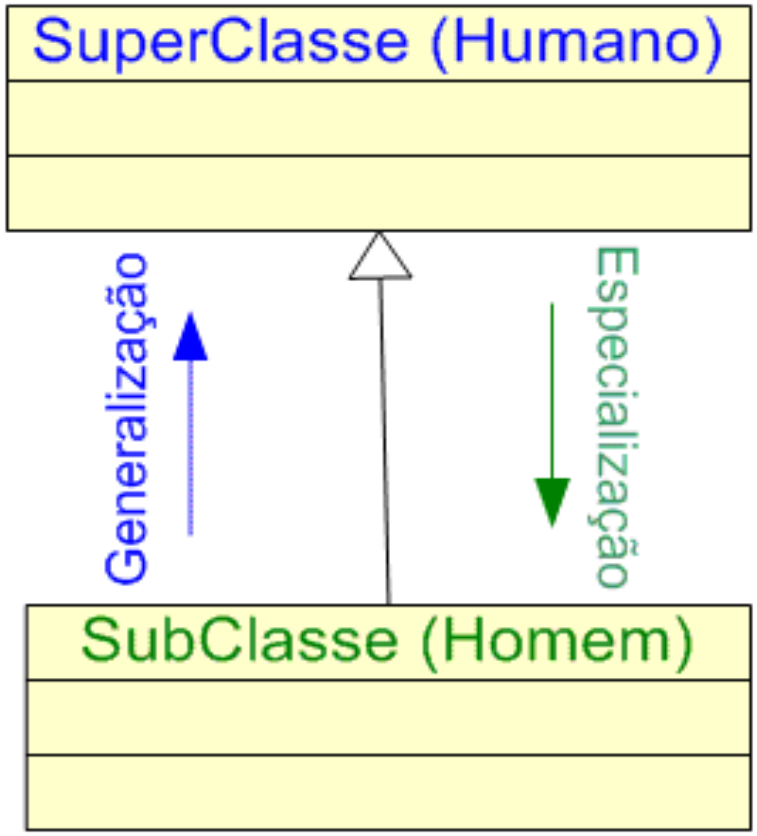
3.3.6 – Hierarquia de classes

- Automóveis
  - automóveis utilitários (camionetes leves)
    - utilitários urbanos
    - utilitários off-road
  - automóveis de passeio
    - passeio familia
    - passeio esportivo
  - automóveis de carga
    - carga inflamáveis
    - carga com frígirifico

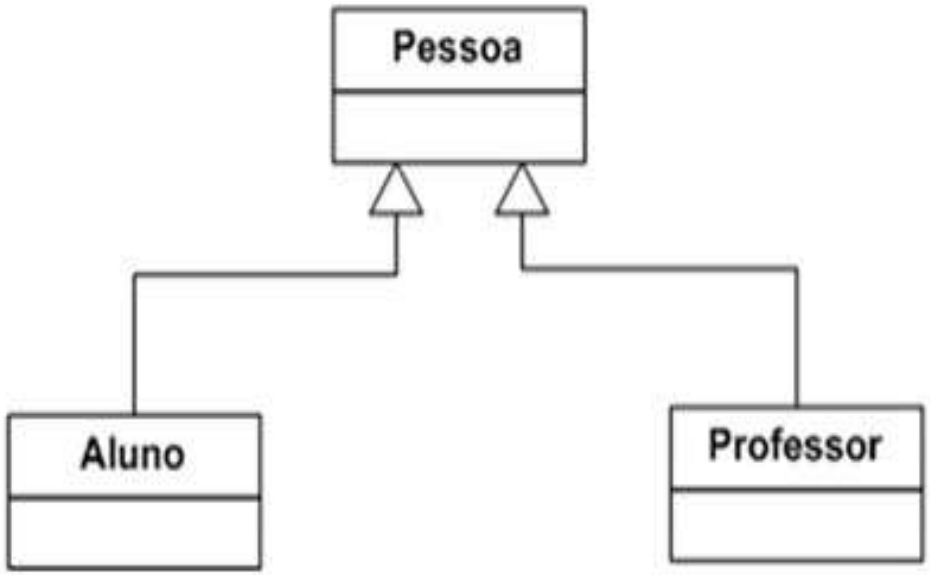
3.3.6.1 – Hierarquia de classes - Herança

- Em uma hierarquia de classes semelhantes podemos dizer que as classes mais específicas herdam as características das mais genéricas, ou seja, todo automóvel de passeio familia é um automóvel de passeio
- A classe de nível superior na associação de herança é chamada de super-classe e a inferior de sub-classe
- Portanto automóvel de passeio família é uma sub- classe de automóvel de passeio

SuperClasse e SubClasse



Herança



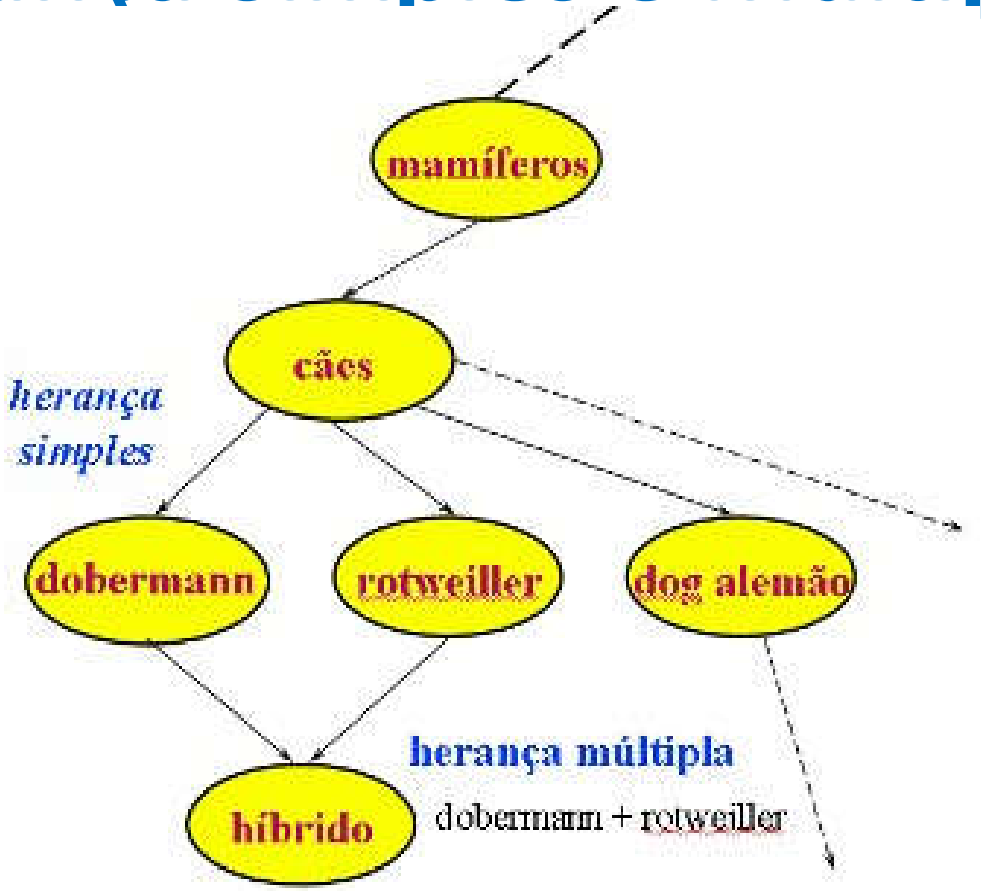
Herança: Aluno e Professor herdam de Pessoa

3.3.6.2 – Herança simples e múltipla

- Simples:
  - ❑ Se herda apenas de uma superclasse

- Múltipla
  - ❑ Existe a possibilidade de se herdar de várias classes

Herança Simples e Múltipla



3.3.7 - Agregação / Composição

- É o princípio que permite o desenvolvedor considerar algo muito grande através do enfoque TODO-PARTE
- Utilizado para definir regras de negócio
  - ❑ Se a instância do todo for removida, suas partes também deverão ser removidas (composição)
  - ❑ Se a instância do todo for removida, suas partes não necessariamente deverão ser removidas (agregação)

3.3.7.1 - Agregação

- Define uma "dependência fraca" entre o todo e suas partes
- Se o todo for removido, as suas partes poderão continuar existindo é um mecanismo que permite a construção de uma classe agregada a partir de outras classes componentes.
- *Usa-se dizer que um objeto da classe agregada (Todo) tem objetos das classes componentes (Parte)*

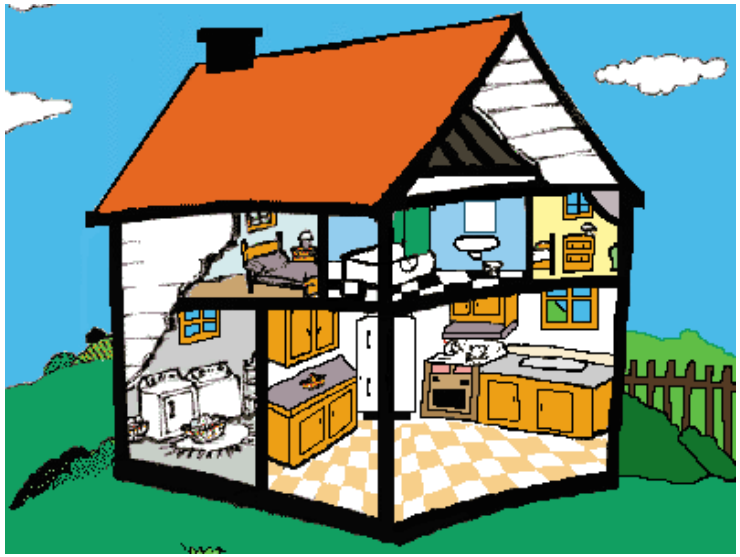
Agregação

Time
TODO

Atletas
Parte

3.3.7.2 - Composição

- Define uma "dependência forte" do todo e suas partes
- Se o todo deixar de existir, as suas partes também deixam de existir



Composição

Pedido
TODO

Se retirar o PEDIDO (todo), não faz sentido ter os itens (parte)

Itens do Pedido
Parte

3.3.8- Herança x Agregação x Composição

- Herança
  - "é um tipo de", as propriedades são herdadas de outra classe
- Agregação
  - "usam um(a)", um objeto usa um outro objeto
- Composição
  - "composto por", um objeto é composto por outros objeto

3.3.9 - Abstração

- Abstração é o processo através do qual detalhes são ignorados, para nos concentrarmos nas características essenciais
- A abstração nos leva a representar os objetos de acordo com o ponto de vista e interesse de quem os representa

3.3.9 - Abstração

- Para descrevermos um automóvel (do ponto de vista de um observador externo), identificamos a cor do mesmo, o número de portas, o tipo das rodas e pneus.

- Quando identificamos o automóvel apenas a partir destas características externas estamos fazendo uma abstração pois uma série de detalhes internos não estarão sendo descritos.

3.3.9 - Abstração

- Ex: Ao modelarmos um objeto avião no contexto de um sistema de marcação de passagens aéreas, não vai nos interessar a característica número de turbinas do avião, mas irá nos interessar a característica número de assentos disponíveis

- Ao ignorarmos algumas características não relevantes em um determinado contexto, estamos fazendo uma abstração

3.3.10 - Encapsulamento

- É o processo de ocultação das características internas do objeto
- O encapsulamento cuida para que certas características não possam ser vistas ou modificadas externamente

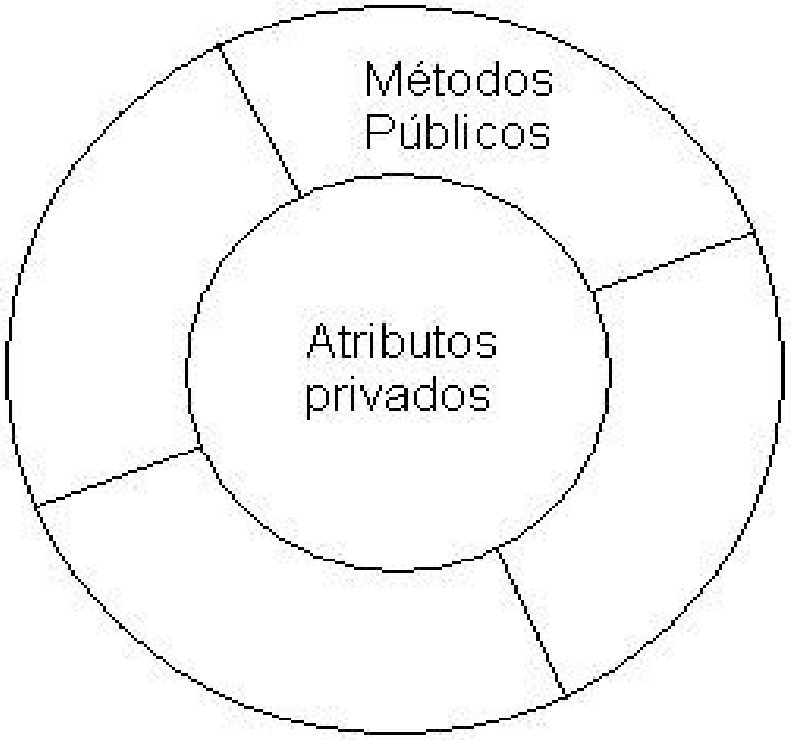
3.3.10 - Encapsulamento

- Exemplo
  - Podemos dizer que o motor de um automóvel está encapsulado, pois normalmente não podemos ver ou alterar características internas do motor
  - Podemos então utilizar um automóvel sem conhecer nada das complexidades do motor, que estão encapsuladas

3.3.10 - Encapsulamento

- No contexto de uma linguagem O O um objeto pode ser definido como um conjunto de "funções“ unidas aos dados que elas necessitam
- O encapsulamento "protege" os dados que estão "dentro" dos objetos, evitando assim que os mesmos sejam alterados erroneamente
- Os dados só poderão ser alterados pelas "funções” dos próprios objetos
- As "funções" dos objetos são chamadas de operações ou métodos

Encapsulamento



3.3.11- Polimorfismo

- É a capacidade de objetos diferentes reagirem segundo a sua função a uma ordem padrão.
- O comando "abre", por exemplo, faz um objeto entrar em ação, seja ele uma janela, uma porta ou uma tampa de garrafa
- O nome e a assinatura do método são iguais mas o comportamento muda

Polimorfismo

3.3.11.1 - Polimorfismo e Herança

- Uma superclasse Conta Comum define um método saque
- Uma subclasse de Conta Comum, Conta especial define um método saque

- Os dois tem a mesma assinatura e parâmetros
- Pelo polimorfismo, um objeto Conta Especial chamará o método mais "próximo" na hierarquia

3.3.11.1 – Polimorfismo e Herança

- Sobrecarga
  - Através do mecanismo de sobrecarga, dois métodos de uma classe podem ter o mesmo nome, desde que suas assinaturas sejam diferentes.
  - Tal situação não gera conflito pois o compilador é capaz de detectar qual método deve ser escolhido a partir da análise dos argumentos do método.

3.3.12 - Persistência

- Refere-se à habilidade de um objeto existir além da execução que o criou, ou seja, ser armazenado em memória secundária (permanente ou persistente).
- Não é o armazenamento apenas dos atributos (dados), mas também dos métodos para realizar os acessos .
- Muito encontrado em sistemas que manipulam banco de dados
  - Hibernate

4 - Porque utilizar Orientação a Objetos?

- Quando bem empregada, a Orientação a Objetos traz diversas vantagens:
  - reutilização,
  - confiabilidade,
  - modelo de sistema mais realístico,
  - facilidade de interoperabilidade e de manutenção,
  - aumento da qualidade,
  - maior produtividade
  - unificação do paradigma (da análise a implementação).
- A Orientação a Objetos é um paradigma que pode ser aplicado ao longo de todo o processo de construção do software.

5 - Unified Modeling Language (UML)

- UML é uma linguagem para especificação, construção, visualização e documentação de sistemas de software.
  - ❑ UML não é uma metodologia, não diz quem deve fazer o quê, quando e como.
  - ❑ UML pode ser usado segundo diferentes metodologias, tais como RUP (Rational Unified Process), FDD (Feature Driven Development), etc.
  - ❑ UML não é uma linguagem de programação

5 - Unified Modeling Language (UML)

- Modelos e Diagramas
  - Um modelo é uma representação em pequena escala, numa perspectiva particular, de um sistema existente ou a desenvolver
  - Ao longo do ciclo de vida de um sistema são construídos vários modelos, sucessivamente refinados e enriquecidos

- Um modelo é constituído por um conjunto de diagramas (desenhos) consistentes entre si, acompanhados de descrições textuais dos elementos que aparecem nos vários diagramas