INSTITUTO FEDERAL
SANTA CATARINA

# Programação Orientada a Objetos

Prof. Hugo Marcondes
hugo.marcondes@ifsc.edu.br

Aula 01

Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina



# Complexidade



## Complexidade



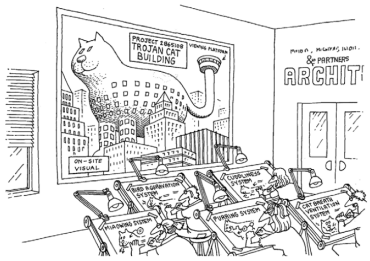
- O projeto de um sistema orientado a objetos visa principalmente tratar o projeto de sistemas complexos
- As principais atributos de um sistema complexo são:
  - Estrutura Hierarquizada
  - Elementos primitivos relativos
  - Separação de responsabilidades (concerns)
  - Padrões comuns
  - Formas intermediárias estáveis



## Estrutura Hierarquizada



Frequently, complexity takes the form of a hierarchy, whereby a complex system is composed of interrelated subsystems that have in turn their own subsystems, and so on, until some lowest level of elementary components is reached.
Booch, 2007




The architecture of a complex system is a function of its components as well as the hierarchic relationships among these components.



## Separação de Responsabilidades



- "Intracomponent linkages are generally stronger than intercomponent linkages. This fact has the effect of **separating the high-frequency dynamics** of the components—involving the internal structure of the components—**from the low- frequency dynamics**—involving interaction among components."
- A diferença entre as interações internas e externas de componentes provêem uma clara separação de responsabilidades (separation of concerns) entre as várias partes do sistema.



## Complexidade



- Elementos primitivos relativos
  - A escolha de quais os componentes em um sistema são primitivos é relativamente arbitrário e é em grande parte a critério do observador do sistema
- Padrões comuns
  - Sistemas hierárquicos são geralmente compostas por apenas alguns tipos diferentes de sub-sistemas em várias combinações e arranjos
- Formas intermediárias estáveis
  - Um sistema complexo que funciona é invariavelmente encontrado a partir da evolução de um sistema simples funcional . . . . "Um sistema complexo projetado do zero nunca funciona e não pode ser remendado para que funcione. Você tem que começar de novo, começando com um sistema simples que funcione."

## O papel da decomposição

"The technique of mastering complexity has been known since ancient times: divide et impera (divide and rule)"

- No projeto de sistemas complexos é essencial realizar a decomposição do sistema em partes menores, cada qual pode ser refinada independentemente.
  - Decomposição algorítmica
  - Decomposição orientada a Objetos



## Decomposição Algorítmica

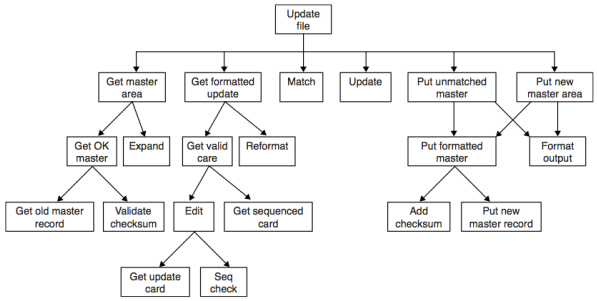



**Figure 1–3** Algorithmic Decomposition



## Decomposição orientada a Objetos

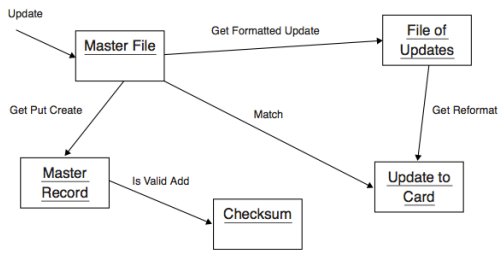



**Figure 1–4** Object-Oriented Decomposition



## Resumindo

- Software é inerentemente complexo; a complexidade de sistemas de software geralmente excedem a capacidade intelectual humana
- A tarefa do time de desenvolvimento de software é projetar a "ilusão da simplicidade"
- Complexidade geralmente é estruturada de forma hierárquica; Há basicamente dois tipos de hierarquias: "'é um[a]" e "é parte de".
- Sistemas complexos geralmente evoluem de formas intermediárias estáveis.
- A **cognição humana é fundamentalmente limitada**; esta limitação é abordada através do uso de **decomposição**, **abstração** e **hierarquia**.



## Resumindo

- Sistemas complexos podem ser vistos com foco em "coisas" ou "processos"; **Geralmente é mais adequado realizar a decomposição orientada a objetos**, na qual o mundo é visto como uma **coleção de objetos significativos que colaboram entre si para atingir um comportamento de alto-nível**.
- A **análise e projeto orientado a objetos** é o método que nos guia para uma **decomposição orientada a objetos**; O projeto orientado a objetos utiliza um processo e notação para a construção de sistemas complexos de software e oferece um conjunto rico de modelos nos quais podemos pensar sobre diferentes aspectos do sistema em consideração



# O Modelo de Objetos

## O modelo de Objeto

- A tecnologia orientada a objetos é construída através de um elemento fundamental, o modelo de objeto.
- O modelo de objeto engloba os seguintes princípios
  - Abstração
  - Encapsulamento
  - Modularidade
  - Hierarquia
  - Tipagem (typing)
  - Concorrência
  - Persistência



## Programação Orientada a Objetos

Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

- Programação orientada a objetos utiliza objetos, e não algoritmos, como blocos de construção fundamental
- Todo objeto é uma instância de uma classe
- As classes são relacionadas umas com as outras através de relações de herança (Hierarquia do tipo "é um")
- De forma geral, uma linguagem é considerada orientada a objetos se:
  - Suporta objetos como abstração de dados com uma interface de operações e um estado local "protegido"
  - Objetos tem um tipo associado (classe)
  - Tipos (classes) podem herdar atributos de supertipos (superclasses).



## Projeto Orientada a Objetos

Object-oriented design is a method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design.

- Leva a decomposição orientada a objetos
- Por modelo lógico, entende-se a estrutura de classes e objetos
- Por modelo físico, entende-se a arquitetura modular e de processos do sistema



## Análise Orientada a Objetos

Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.



## Elementos do modelo de objetos

- Dentre os elementos que compõe o modelo de objetos, temos 4 que são fundamentais:
  - Abstração
  - Encapsulamento
  - Modularidade
  - Hierarquia
- e 3 secundários:
  - Tipagem
  - Concorrência
  - Persistência

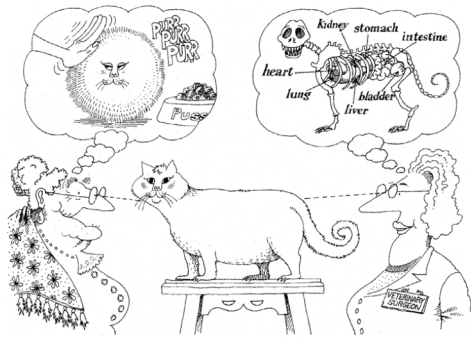


## Abstração

An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.

- Uma boa abstração é aquela que enfatiza os detalhes que são significativos para o usuário e suprime os detalhes que, ao menos no determinado contexto, não são relevantes.
- Decidir pelo conjunto adequado de abstrações para um domínio em específico é o principal desafio em um projeto orientado a objetos

## Abstração




Abstraction focuses on the essential characteristics of some object, relative to the perspective of the viewer.



## Abstração

- Um objeto é uma abstração de um determinado conceito no domínio do problema.
- As abstrações definem contratos entre si.
  - **Operações** que uma determinada abstração pode executar
  - Determina as assunções que podemos estabelecer em relação ao **comportamento** do objeto
  - Possuem **propriedades estáticas e dinâmicas**



## Exemplo

| **Abstraction:** Temperature Sensor |
|---|
| **Important Characteristics:**<br>temperature<br>location |
| **Responsibilities:**<br>report current temperature<br>calibrate |

**Figure 2–6** Abstraction of a `Temperature Sensor`



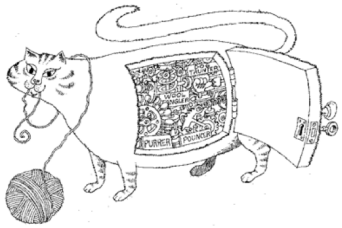
## Encapsulamento

Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation

- Abstração e encapsulamento são conceitos complementares: Abstração foca no comportamento observável de um objetos, enquanto encapsulamento foca na implementação que emerge este comportamento.
- Encapsulamento provê barreiras explícitas entre diferentes abstrações e desta forma leva a uma clara separação de responsabilidades.
- "For abstraction to work, implementations must be encapsulated" [53]. Na prática, isto significa que cada classe deve ter duas partes: uma interface e uma implementação.



## Encapsulamento



Encapsulation hides the details of the implementation of an object.



## Separação de Responsabilidades

| **Abstraction:** Heater |
|---|
| **Important Characteristics:**<br>location<br>status |
| **Responsibilities:**<br>turn on<br>turn off<br>provide status |

Related Candidate Abstractions: Heater Controller, Temperature Sensor

**Figure 2–9** Abstraction of a `Heater`

Separação de Responsabilidades

Não definimos como responsabilidade da abstração Heater manter uma temperatura fixa. Ao invés, deixamos esta responsabilidade para outro objeto (ex. Heater Controller), que deve colaborar com um sensor de temperatura e uma aquecedor para atingir o seu comportamento de alto nível.

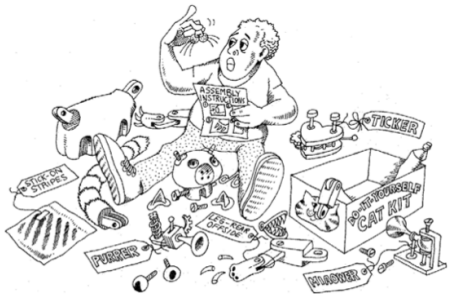
## Modularidade

Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules.

- Modularização consiste em dividir um programa em módulos que podem ser compilados separadamente, mas que possui conexões com outros módulos.
- Módulos servem como compartimentos em que podemos declarar classes e objetos do nosso projeto lógico.
- Se esforce para construir módulos que são coesos (agrupando abstrações logicamente relacionadas) e com acoplamento fraco (minimizando as dependências entre módulos).



## Modularidade



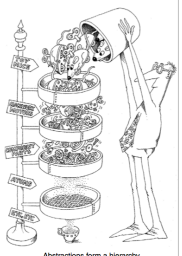

Modularity packages abstractions into discrete units.



## Hierarquia

Hierarchy is a ranking or ordering of abstractions.

- As principais hierarquias em sistemas complexos são:
  - Estrutura de classes - tipo "é um(a)"
  - Estrutura de objetos - tipo "'é parte de"
- Herança é a hierarquia do tipo "'é um(a)" mais importante
  - Define as relações entre classes
  - Hierarquia de abstrações em que subclasses herdam uma ou mais superclasses
- Tipicamente, uma subclasses, estende ou redefine a estrutura e comportamento de suas superclasses



Abstractions form a hierarchy.



## Herança Múltipla

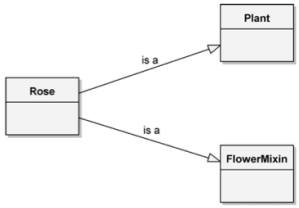



**Figure 2–10** The `Rose` Class, Which Inherits from Multiple Superclasses



## Herança Múltipla

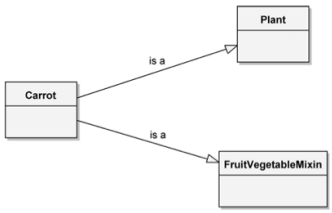



**Figure 2–11** The `Carrot` Class, Which Inherits from Multiple Superclasses



## Herança Múltipla

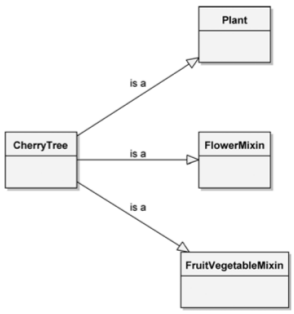



**Figure 2–12** The `CherryTree` Class, Which Inherits from Multiple Superclasses

## Herança Múltipla

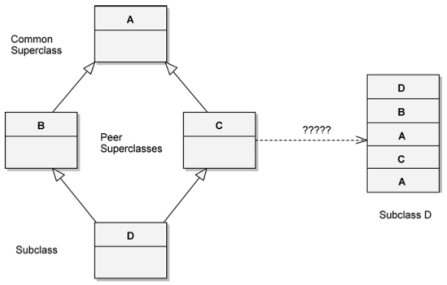



**Figure 2–13** The Repeated Inheritance Problem



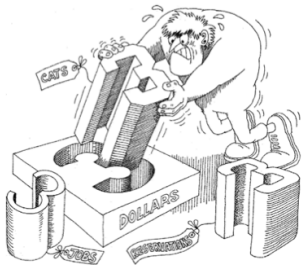
## Agregação

- Enquanto hierarquias do tipo "'é um(a)" denotam relações de generalização/especialização, hierarquias do tipo "é parte de" descrevem relações de agregação.
- Podemos argumentar que um jardim consiste em uma coleção de plantas, junto com o um plano de manejo. Em outras palavras, plantas são "parte de" um jardim, e o plano de manejo é "parte de" um jardim. Esta relação é denominada agregação
- Agregações permitem o agrupamento físico entre estruturas logicamente relacionadas, e herança permite que esses grupos sejam facilmente reutilizados entre diferentes abstrações



## Tipagem

Typing is the enforcement of the class of an object, such that objects of different types may not be interchanged, or at the most, they may be interchanged only in very restricted ways.

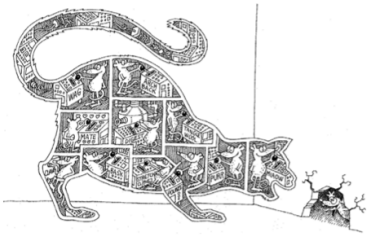



Strong typing prevents mixing of abstractions.



## Concorrência

Concurrency is the property that distinguishes an active object from one that is not active.



Concurrency allows different objects to act at the same time.



## Persistência

Persistence is the property of an object through which its existence transcends time (i.e., the object continues to exist after its creator ceases to exist) and/or space (i.e., the object's location moves from the address space in which it was created).

Persistence saves the state and class of an object across time or space.



## Resumindo...

- A maturação da engenharia de software tem levado ao desenvolvimento de análise, projeto e métodos de programação orientada a objetos, os quais abordam as principais questões de programação de grande porte.
- Há vários paradigmas de programação diferentes: orientada a procedimentos, orientada a objeto, orientada a lógica, orientada a regras, e orientado para a restrição.
- Uma abstração denota as características essenciais de um objeto que o distinguem de todos os outros tipos de objetos e, assim, proporciona fronteiras conceituais bem definidas em relação à perspectiva do espectador

## Resumindo...

- Encapsulamento é o processo de compartimentar os elementos de uma abstração que constituem a sua estrutura e do comportamento; encapsulamento serve para separar a interface contratual de uma abstração e sua implementação.
- A modularidade é a propriedade de um sistema que foi decomposto em um conjunto de módulos coesos e fracamente acoplados.
- Hierarquia é um ranking ou ordenação de abstrações.
- Tipos é a execução da classe de um objeto, de modo que objetos de diferentes tipos não podem ser trocados ou, no máximo, podem ser trocadas entre si apenas em formas muito restritas.



# Classes e Objetos



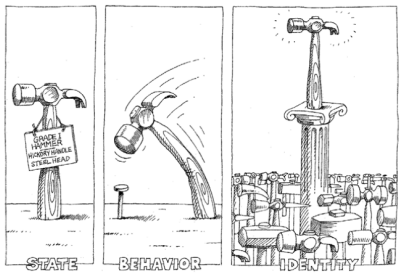
## Objetos

- Da perspectiva da cognição humana, um objeto pode ser:
  - Uma coisa tangível/visível
  - Algo que pode ser compreendido intelectualmente
  - Algo em que uma ação pode ser direcionado
- Um objeto representa um indivíduo, uma unidade, item ou entidade identificável, seja real ou abstrata, com um papel bem definido no domínio do problema



## Objetos

An object is an entity that has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. The terms instance and object are interchangeable.




An object has state, exhibits some well-defined behavior, and has a unique identity.



## Estado de objetos

- O estado de um objeto engloba todas as suas propriedades (geralmente estáticas) além dos valores (geralmente dinâmicos) destas propriedades
- Uma propriedade é uma característica, qualidade que contribui para tornar um objeto único
  - Propriedades são geralmente estáticas
    - Em algumas circunstâncias, uma propriedade de um objeto pode mudar (ex. machine learning)
  - Toda propriedade possui um valor. Este valor pode ser um valor quantitativo, ou pode denotar outro objeto
- O fato de todo objeto possuir um estado, implica que todo objeto ocupa algum espaço (memória)



## Comportamento de objetos

- Nenhum objeto existe isoladamente.
- Comportamento é como um objeto age e reage, em termos de sua mudança de estado e passagem de mensagens
  - O comportamento de um objeto representa a sua atividade externa visível
- Uma operação é alguma ação que um objeto executa em outro de forma a provocar uma reação
  - Java - métodos
  - C++ - função membro
  - Smalltalk - mensagem

## Comportamento de objetos

- Uma mensagem é simplesmente uma operação que um objeto pode realizar em outro
  - Operação e mensagem são termos intercambiáveis
- A passagem de mensagens é uma parte da equação que define o comportamento de um objeto.
- O estado do objeto também afeta o seu comportamento
  - O estado do objeto representa o resultado cumulativo de seu comportamento



## Operações de objetos

- Uma operação denota um serviço que é provido para outros objetos
  - **Modificadores**: uma operação que altera o estado de um objeto.
  - **Selecionadores**: uma operação que acessa o estado de um objeto (sem alterá-lo)
  - **Interadores**: uma operação que permite o acesso a "partes" de um objeto em uma ordem bem definida
  - **Construtores**: uma operação para criar e inicializar o estado de um objeto
  - **Destrutores**: uma operação para livrar o estado de um objeto e "destruí-lo".



## Operações de objetos

- Todos os métodos associados a um objeto particular compreendem o seu protocolo. O protocolo de um objeto define assim o envelope do comportamento permitido de um objeto e assim compreende toda a visão estática e dinâmica do objeto.
- Um papel é uma máscara que um objeto usa e assim define um contrato entre uma abstração e seus clientes
- A existência de estado dentro de um objeto significa que a ordem em que as operações são invocadas é importante
  - Cada objeto é como uma minúscula máquina independente
- Objetos ativos ou passivos. Um objeto ativo é aquele que abrange seu próprio segmento de controle, enquanto que um objeto passivo não



## Identidade do Objeto

- Identidade é uma propriedade que distingue um objeto de todos os demais
- A identidade única de cada objeto é preservada ao longo do tempo de vida do objeto, mesmo que seu estado seja modificado.



## Relacionamento entre Objetos

- A relação entre quaisquer dois objetos abrange os pressupostos que cada um faz sobre o outro, incluindo quais operações podem ser executadas e os comportamentos resultantes
- Associações
  - Conexão física ou conceitual entre objetos
  - Denota a associação específica através do qual um objeto (o cliente) se aplica aos serviços de outro objeto (o fornecedor)
- Agregações
  - Enquanto associações denotam uma relação cliente/fornecedor ou "peer-to-peer", agregações denotam a hierarquia todo/parte, com a capacidade de navegar a partir do todo (também chamado o agregado) para as suas partes
  - Agregação é um tipo especializado de associação



## Relacionamento entre objetos

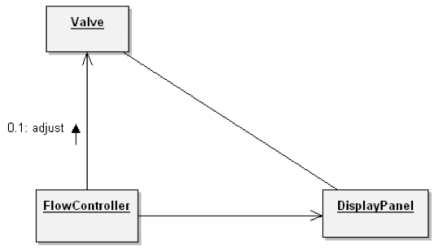



**Figure 3–5** Links

## Relacionamento entre objetos

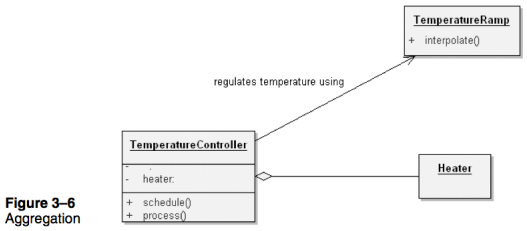
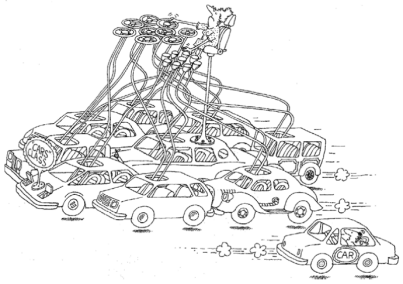



**Figure 3–6**
Aggregation



## Classes de Objetos

- Uma classe é um conjunto de objetos que compartilham uma estrutura, um comportamento, e uma semântica comum



A class represents a set of objects that share a common structure and a common behavior.



## Interface e Implementação

- A classe serve como um contrato entre abstrações e todos os seus clientes, definido a sua interface.
- A interface de uma classe pode ser dividida em:
  - **Pública**: uma declaração acessível a todos os clientes
  - **Protegida**: uma declaração acessível apenas a própria classe e suas sub-classes
  - **Privada**: uma declaração acessível apenas a própria classe



## Interface

- As constantes e variáveis que formam a representação de uma classe é conhecida de diversas formas, de acordo com a linguagem de programação utilizada
  - instance variable (Smalltalk)
  - field (Java)
  - member data (C++)
- O estado de um objeto deve ter alguma correspondência em sua classe, e é tipicamente expressada através de suas constantes e variáveis, declaradas para interface privada ou protegida da classe.
  - Encapsulamento da representação do estado de um objeto!



## Relações entre Classes

- Classes, como objetos, não existem isoladamente
  - Abstrações chaves são geralmente relacionadas em variadas formas
- Três tipos básicos de relacionamentos entre classes
  - Generalização / Especialização - "é um(a)"
  - Relação "é parte de"
  - Associação - dependência semântica entre classes



## Associações

- É a relação mais genérica e geralmente a mais fraca semanticamente
  - Geralmente expressam dependências entre as abstrações
- Durante a análise é importante capturar tais dependências semânticas, seus papéis e sua cardinalidade

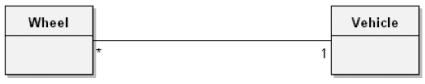
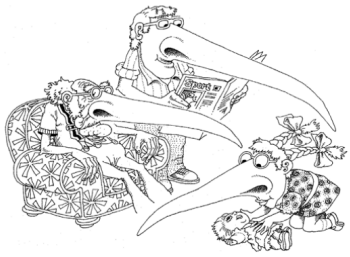



**Figure 3–7** Association

## Herança

- Herança expressa as relações de generalização e especialização entre classes



A subclass may inherit the structure and behavior of its superclass.



## Herança

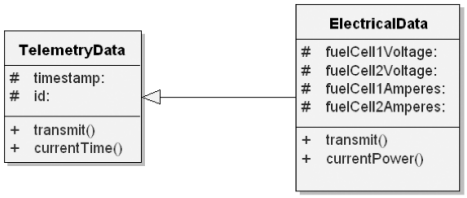



**Figure 3–8** `ElectricalData` Inherits from the Superclass `TelemetryData`

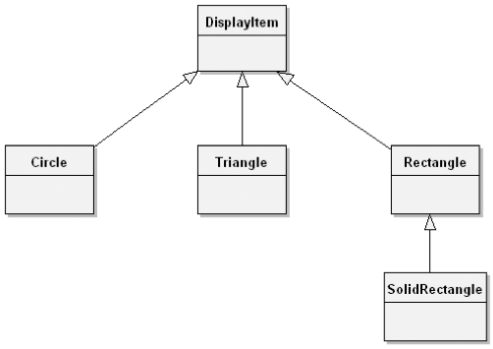


## Polimorfismo

- Polimorfismo é um conceito na teoria de tipos, onde um nome pode denotar instancias de diferentes classes, desde que estas estejam relacionadas por alguma superclasse (herança)
- Qualquer objeto denotado por este nome é capaz de responder a um conjunto de operações de formas distintas
- Polimorfismo é útil quando temos diversas classes com o mesmo protocolo
- Polimorfismo é implementado através do conceito de ligação tardia. Na presença de polimorfismo, a ligação de um método com o seu "nome" não é determinada até a execução do programa.

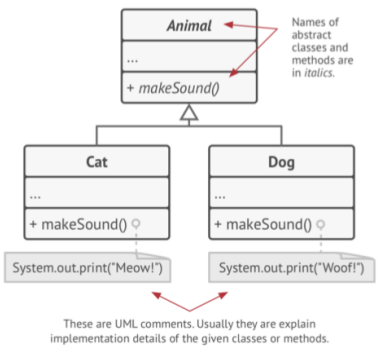


## Polimorfismo






## Polimorfismo




```
1  bag = [new Cat(), new Dog()];
2
3  foreach (Animal a : bag)
4    a.makeSound()
5
6  // Meow!
7  // Woof!
```



## Agregação

- Relações de agregação denotam a relação de todo-parte.
- Os objetos contidos podem existir sem serem parte do objeto que os contém.

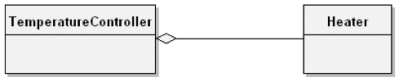
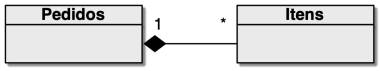



**Figure 3–12** Aggregation

## Composição

- Um tipo de agregação onde objetos contidos não fazem sentido fora do contexto do objeto que os contém.






## Análise e Projeto

- Durante a análise e dos estágio iniciais do projeto, o desenvolvedor deve:
  - Identificar as classes que formam o vocabulário do domínio do problema.
  - Definir a estrutura pela qual os objetos trabalham em conjunto para prover o comportamento que satisfaça os requisitos do problema



## A qualidade de uma Abstração

- Como podemos avaliar se uma classe ou objeto é bem projetado ?
  - Acoplamento
    - Dependência entre módulos
  - Coesão
    - Grau de conectividade entre elementos de um mesmo módulo
  - Suficiência
    - O módulo captura suficientemente as características das abstrações?
  - Completude
    - A interface captura todas as características de forma significativa ?
  - Operações primitivas
    - Suas operações são primitivas ? Podem ser implementadas eficientemente com a representação da abstração ?



## Resumindo...

- Um objeto tem estado, comportamento e identidade
- A estrutura e o comportamento de objetos semelhantes são definidos na sua classe comum
- O estado de um objeto engloba todas as propriedades (normalmente estáticas) do objeto mais os valores correntes (usualmente dinâmicas) de cada uma destas propriedades.
- Comportamento é como um objeto age e reage em termos de suas alterações de estado e passagem de mensagens.
- Identidade é a propriedade de um objeto que o distingue de todos os outros objetos.



## Resumindo...

- Uma classe é um conjunto de objetos que compartilham uma estrutura comum e um comportamento comum
- Os três tipos de relacionamentos incluem associação, herança e agregação
- Abstrações-chave são as classes e objetos que formam o vocabulário do domínio do problema
- Um mecanismo é uma estrutura em que um conjunto de objetos trabalham juntos para fornecer um comportamento que satisfaça alguma exigência do problema
- A qualidade de uma abstração pode ser medida por seu acoplamento, coesão, suficiência, completude e primitivismo