

# 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

<sup>2</sup> IFSC - Programação Orientada a Objetos

## Complexidade

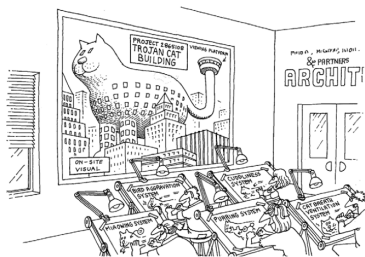
- O projeto de um sistema orientado a objetos visa principalmente tratar o projeto de sistemas complexos
- As principais atributos de um sistema complexo são:
  - Estrutura Hierarquizada
  - Elementos primitivos relativos
  - Separação de responsabilidades (concerns)
  - Padrões comuns
  - Formas intermediárias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

<sup>3</sup> IFSC - Programação Orientada a Objetos

## 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.

<sup>4</sup> IFSC - Programação Orientada a Objetos

## 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 varias partes do sistema.

5 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

## 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."

6 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

## 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

7 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

## Decomposição Algorítmica

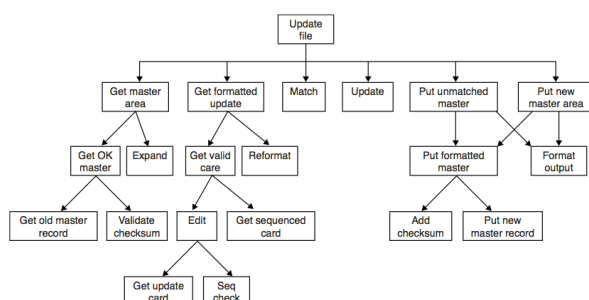


Figure 1-3 Algorithmic Decomposition

8 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

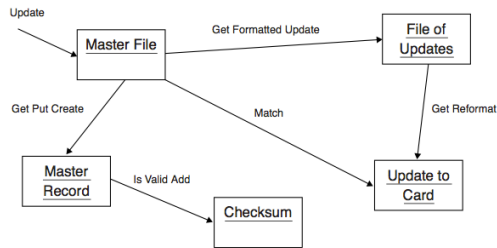


Figure 1-4 Object-Oriented Decomposition

## 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

13 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

---

## 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).

14 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

---

## 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

15 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

---

## 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**.

16 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

---

## 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

17 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

---

---

## 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 **ênfatiza** 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**

18 IFSC - Programação Orientada a Objetos

---

---

---

---

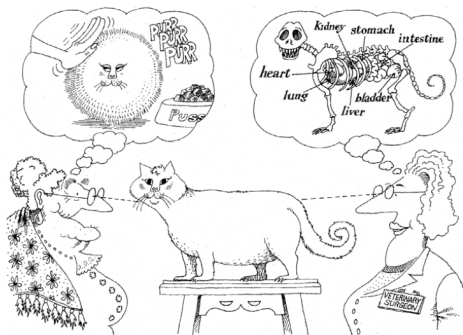
---

---

---

---

## Abstração



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

19 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

---

---

## 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**

20 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

---

---

<b>Abstraction:</b>	Temperature Sensor
<b>Important Characteristics:</b>	temperature location
<b>Responsibilities:</b>	report current temperature calibrate

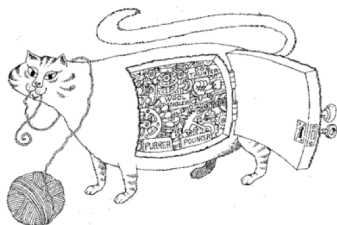
**Figure 2-6** Abstraction of a Temperature Sensor

## 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 objeto, 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.

<b>Abstraction:</b>	Heater
<b>Important Characteristics:</b>	location status
<b>Responsibilities:</b>	turn on turn off 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).

25 IFSC - Programação Orientada a Objetos

## Modularidade



Modularity packages abstractions into discrete units.

26 IFSC - Programação Orientada a Objetos

## Hierarquia



Hierarchy is a ranking or ordering of abstractions.

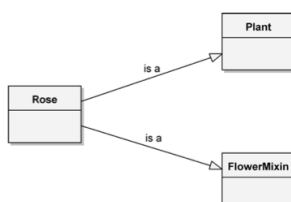
- As principais hierarquias em sistemas complexos são:
  - Estrutura de classes - tipo "é um{a}"
  - Estrutura de objetos - tipo "é parte de"
- Herança é a hierarquia do tipo "é um{a}" mais importante
  - Define as relações entre classes
  - Hierarquia de abstrações em que **subclasses** herdam uma ou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



Abstractions form a hierarchy.

27 IFSC - Programação Orientada a Objetos

## Herança Múltipla



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

<sup>28</sup> IFSC - Programação Orientada a Objetos

## Herança Múltipla

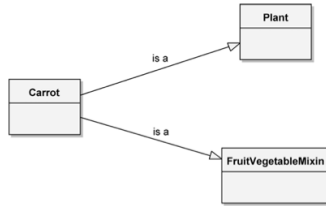


Figure 2-11 The Carrot Class, Which Inherits from Multiple Superclasses

---

---

---

---

---

---

---

---

## Herança Múltipla

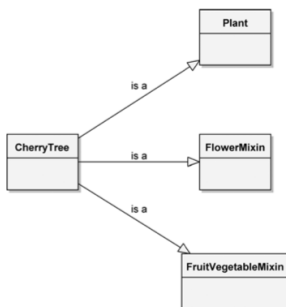


Figure 2-12 The CherryTree Class, Which Inherits from Multiple Superclasses

---

---

---

---

---

---

---

---

## Herança Múltipla

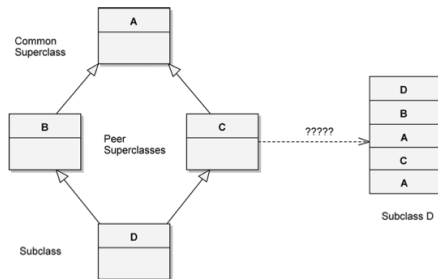


Figure 2-13 The Repeated Inheritance Problem

---

---

---

---

---

---

---

---

## 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

---

---

---

---

---

---

---

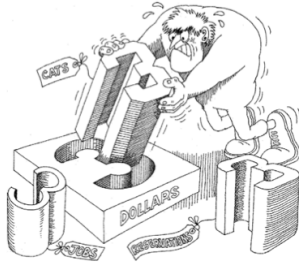
---



## 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.

33 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

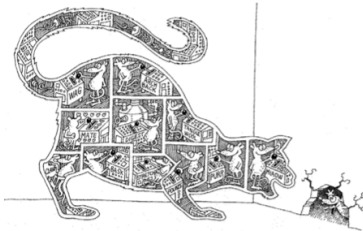
---

---

## 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.

34 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

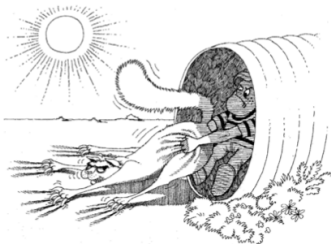
---

---

## 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.

35 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

---

---

## 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

36 IFSC - Programação Orientada a Objetos

---

---

---

---

---

---

---

---

## 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.

---

---

---

---

---

---