Modelagem de Software

Introdução à Orientação a Objetos

Unified Modeling Language (UML)

Por que modelar?

- Testar antes de construir (simulação, provas formais, execução)
- Comunicação com stakeholders
- Gerenciamento da complexidade
- Separação de partes a serem tratadas em cada etapa

Modelo de Software - definição

Um modelo de software captura uma visão do sistema físico

Uma abstração do sistema, com um certo propósito

- Aspectos estruturais e comportamentais

Fases de um processo de desenvolvimento de software

- Engenharia de Requisitos
  - Levantamento
  - Análise
  - Validação
  - Gerenciamento
- Projeto
- Codificação
- Testes
- Implantação

Breve Histórico da UML (onde pesquisar: http://uml.org/)

- Surgiu da união de 3 métodos de modelagem
  - Método de Booch
  - OMT (*Object Modeling Technique*) de Jacobson
  - OOSE (*Object-Oriented Software Engineering*) de Rumbaugh
- 1ª versão
  - 1996
  - Os três amigos
- Adotada em 1997 pela OMG (*Object Management Group*)
- Versão 2.0 em 2005 (atual)

A UML está totalmente inserida no paradigma orientação a objetos...

É melhor compreendermos esse conceito antes

Modelagem

- Abstração – criação do minimundo
  - Essencial para identificação das classes
  - Consiste na seleção de aspectos de interesse do domínio do problema a modelar

- Orientação a Objetos
  - “Uma nova maneira de pensar os problemas utilizando modelos organizados a partir de conceitos do mundo real...” (Rumbaugh)
  - Primeira Linguagem OO: Simula 67 (1967)

Classes e Objetos

- Classe
  - Modelo, categoria
  - O projeto de um objeto
  - Representação de um conjunto de coisas reais ou abstratas que são reconhecidas como sendo do mesmo tipo
  - Possui atributos (características) e operações (comportamentos)

- Objeto
  - Membros, instâncias
  - Pode ser real ou abstrato
  - Representa um conceito da realidade humana (físico ou conceitual)
  - Físico: Um carro, uma pessoa, um livro
  - Conceitual: um diagrama de classes de uma sistema

Atributos

- Propriedades
- São as características de uma classe
  - Permitem diferenciar cada instância (objeto)
- Exemplo:
  - Considerando a classe Pessoa
  - nome, sexo, cor, anoNascimento;

Métodos

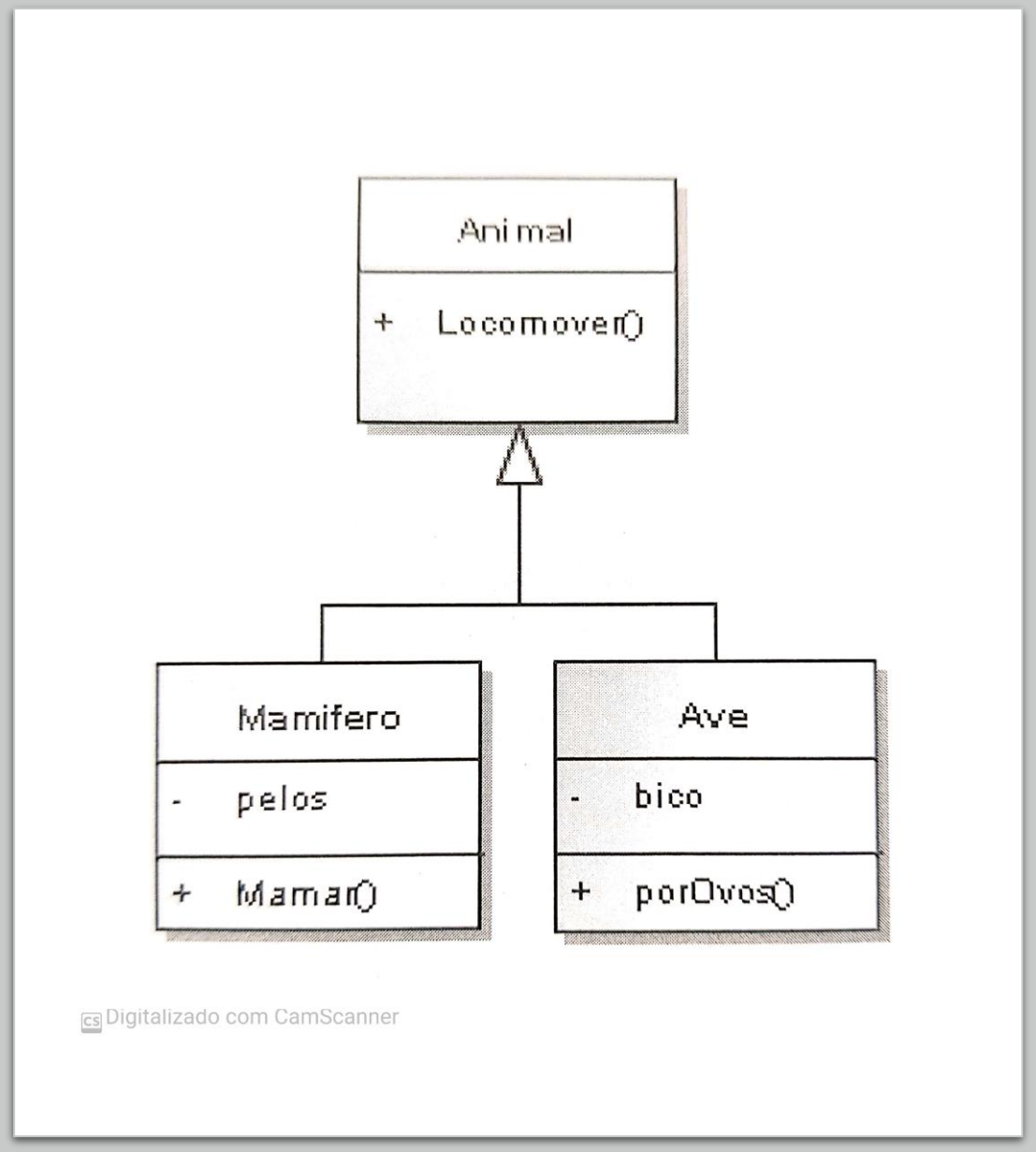
- Comportamentos
- Representam as atividades que uma classe pode executar
- Podemos comparar um método a uma função desenvolvida em uma linguagem de programação estruturada como C
- Os métodos podem (ou não) receber parâmetros e retornar (ou não) valores

Visibilidade

- Indica o nível de acessibilidade de um atributo ou método
- Basicamente, há quatro modos de visibilidade
  - Pública (+)
    - Objetos de quaisquer classes podem acessar os atributos e métodos
  - Protegida (#)
    - A classe, as classes do mesmo pacote e as subclasses podem ter acesso aos atributos e métodos
  - de pacote ou padrão (~)
    - Somente as classes do mesmo pacote têm acesso os atributos e métodos
  - Privada (-)
    - Apenas a própria classe tem acesso

Herança

- Permite o reaproveitamento de atributos e de métodos
  - → reutilização de códigos
  - Superclasse - Subclasse
- Superclasse – também chamada de “classe mãe”
  - Possui classes derivadas dela que são chamadas de subclasses
- Subclasse – também chamada de “classe filha”
  - Herda os métodos e os atributos da sua “classe mãe”



Polimorfismo

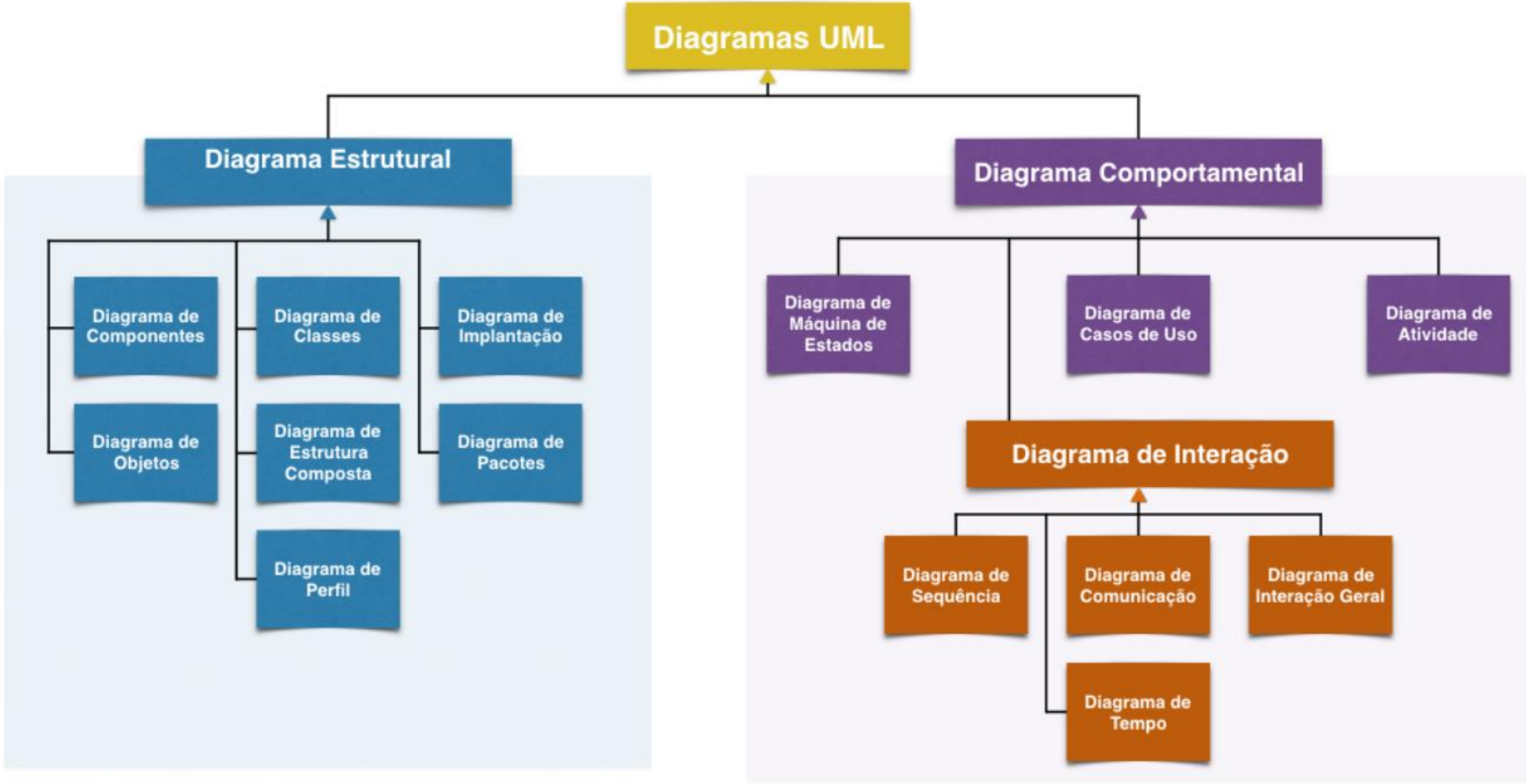
- Existem várias formas de implementação
  - Uma das formas de implementar é a redefinição de métodos previamente herdados de uma classe
    - dinâmico ou sobreposição
    - Os métodos, apesar de semelhantes, diferem de alguma forma da implementação utilizada na superclasse
  - Outra é a utilização de diferentes tipos ou quantidade de parâmetros
    - estático ou sobrecarga
    - Os métodos admitem parâmetros diferentes, mas o tipo de retorno é sempre o mesmo
  - Existe ainda o paramétrico
    - Uso de Generics

Agora, com vocês a UML

- Diagrama de Classes
- Diagrama de Objetos
- Diagrama de Componente
- Diagrama de Implantação
- Diagrama de Composição
- Diagrama de Pacotes
- Diagrama de Casos de Uso
- Diagrama de Sequência
- Diagrama de Comunicação
- Diagrama de Atividade
- Diagrama de Estados
- Diagrama de Interação
- Diagrama de Tempo

Por que tantos diagramas?

O objetivo é fornecer múltiplas visões do sistema a ser modelado, ou seja, permitir a sua análise sob diversos aspectos.



Diagramas Estruturais

- Classe
  - o mais utilizado na UML e serve de apoio a outros diagramas
  - mostra o conjunto de classes com seus atributos e métodos e os relacionamentos entre elas
- Objeto
  - esta relacionado ao diagrama de classes, depende desse e o complementa
  - fornece uma visão dos valores armazenados pelos objetos de um Diagrama de Classe em um determinado momento da execução do processo do software.
- Componentes
  - está associado à linguagem de programação e tem por finalidade indicar os componentes do software e seus relacionamentos
- Implantação
  - determina as necessidades de hardware e características físicas do sistema
- Pacotes
  - representa os subsistemas englobados de forma a determinar partes que o compõem
- Estrutura Composta
  - descreve a estrutura interna de um classificador (classe ou componente), detalhando partes internas e como se relacionam

Diagramas Comportamentais

- Casos de Uso
  - geral e informal para fases de levantamento e análise de requisitos do sistema
- Máquina de Estados
  - procura acompanhar as mudanças sofridas por um objeto dentro de um processo
- Atividades
  - descreve os passos a serem percorridos para a conclusão de uma atividade
- Interação: dividem-se em
  - Sequência
    - descreve a ordem temporal em que as mensagens são trocadas entre os objetos
  - Interação
    - variação do diagrama de atividades que fornece visão geral dentro do sistema ou processo do negócio
  - Comunicação
    - associado ao Diagrama de Sequência, complementando-o e concentrando-se em como os objetos estão vinculados
  - Tempo
    - descreve a mudança de estado ou condição de uma instância de uma classe ou seu papel durante o tempo.