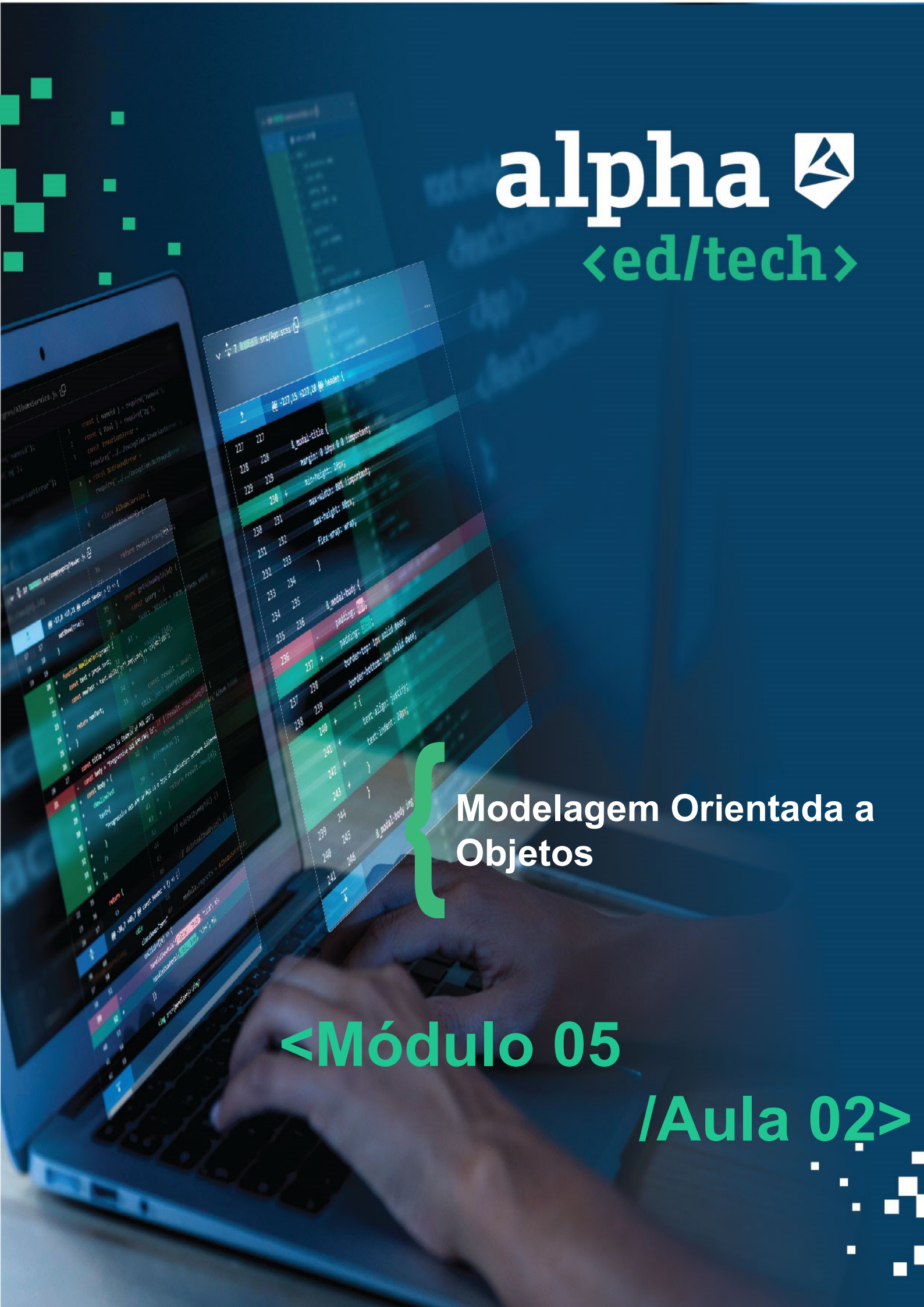

alpha
<ed/tech>
Modelagem Orientada a Objetos
<Módulo 05
/Aula 02>

Modelagem Orientada a Objetos

Histórico e Evolução

Foi discutido na aula sobre **Engenharia de Software** o histórico e evolução até chegar nos dias atuais com **Metodologias Ágeis** e **Modelagem OO**.

1. **Conceitos Iniciais de Orientação a Objetos:**
   - Década de 1960-1970: A ideia de **objetos**, **classes**, **encapsulamento** e **herança** começou a ser explorada. Linguagens como Simula e Smalltalk foram pioneiras nesse conceito.

2. **Desenvolvimento da Modelagem Orientada a Objetos:**
   - Década de 1980: A **orientação a objetos** começou a ser mais amplamente adotada. **Métodos e notações** foram desenvolvidos para modelar sistemas usando conceitos OO. Métodos como Booch, OMT (Object Modeling Technique) e OOSE (Object-Oriented Software Engineering) surgiram nesse período.

3. **Proliferação de Métodos OO:**
   - Início da década de 1990: Diversos métodos concorrentes de modelagem orientada a objetos foram propostos. Cada autor tinha suas próprias notações e abordagens, levando à necessidade de uma padronização.

4. **Chegada da UML:**
   - 1996: **Grady Booch, James Rumbaugh e Ivar Jacobson**, três dos principais contribuintes para métodos de modelagem orientada a objetos, uniram forças para criar uma linguagem de modelagem unificada. Isso resultou na criação da **UML (Unified Modeling Language)**. Eles ficaram conhecidos como **Os Três Amigos**.
   - Mas a grande força para adotar uma padronização como a **UML**, foi a **união de mais de 400 das maiores empresas de desenvolvimento de software, além de todos os autores de livros sobre o assunto**. Valendo um destaque especial para a **empresa Rational** que se tornou um padrão mundial na utilização da UML e OO. Na década de 1990 as modelagens chamadas de "Modelagem Estruturada de Sistemas" deram lugar à "Modelagem Orientada a Objetos". Em 1996 as diversas vertentes da Modelagem Orientada a Objetos foram organizadas na UML.
   - A **evolução da Orientação a Objetos (OO)** até a **criação da Linguagem de Modelagem Unificada (UML - Unified Modeling Language)** envolveu várias etapas, cada uma contribuindo para a melhoria na modelagem e desenvolvimento de sistemas. Observar que a Linguagem é Unificada para uma Modelagem dos sistemas padronizada.

5. **Objetivos da UML:**
   - **Padronização**: **Unificar os conceitos e notações usados em métodos OO** existentes para criar uma linguagem única e compreensível.
   - **Flexibilidade**: Oferecer suporte a diferentes abordagens de modelagem e ser aplicável a **diversos domínios**. (assim pode modelar um sistema gerencial, empresarial, industrial ou de controle de aviônicos ou lançamento de foquetes).
   - **Compreensibilidade**: Tornar a modelagem mais acessível a todos os ***stakeholders***, não apenas aos desenvolvedores.

Modelagem Orientada a Objetos

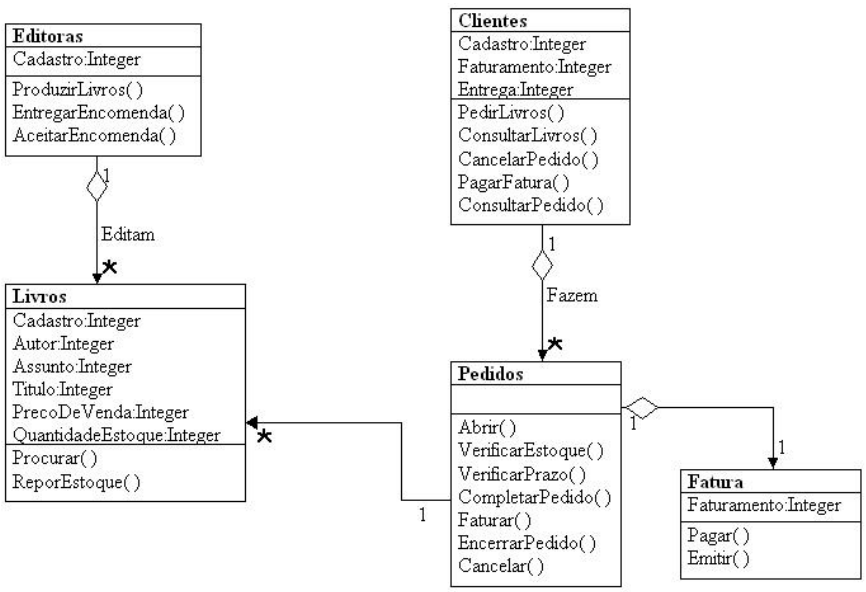
6. **Versatilidade da UML :**
   - Diagramas Específicos: A UML oferece uma variedade de diagramas, incluindo diagramas de classe, diagramas de sequência, diagramas de caso de uso, entre outros, para representar diferentes aspectos de um sistema.

7. **Adoção Generalizada :**
   - Final da década de 1990 e início dos anos 2000: A UML foi amplamente adotada pela indústria de software. Ferramentas de modelagem e desenvolvimento começaram a oferecer suporte direto à UML. Neste ponto do histórico é fundamental lembrar da empresa Rational, com diversas ferramentas CASE (Computer Aided Software Engineenring) que propiciavam o projeto (design, diagramas, testes, integração com IDEs) completo dos sistemas. As ferramentas estavam integradas e evoluindo com a versão da UML. **No ano de 2005 a IBM comprou a empresa Rational com todos seus funcionários e tornou as ferramentas produto da IBM**.

8. **Evolução Contínua:**
   - Desde então: A UML passou por várias revisões e continua sendo uma linguagem de modelagem padrão para sistemas orientados a objetos. Ela se adaptou às mudanças na indústria e às novas necessidades na modelagem de sistemas.

A criação da UML representou um marco importante na história da Engenharia de Software, fornecendo uma linguagem padronizada e compreensível que facilitou a comunicação e colaboração entre os envolvidos no desenvolvimento de software. **Seu legado persiste até hoje, continuando a ser uma ferramenta valiosa para a modelagem de sistemas complexos**.

A Modelagem Orientada a Objetos (OO) é uma abordagem para analisar, projetar e organizar sistemas de software com base nos princípios da Orientação a Objetos. Essa metodologia se baseia em conceitos fundamentais, como objetos, classes, herança, encapsulamento e polimorfismo, para estruturar e representar os elementos de um sistema de **forma mais próxima ao mundo real**.



Modelagem Orientada a Objetos

Conceitos-chave da Modelagem OO

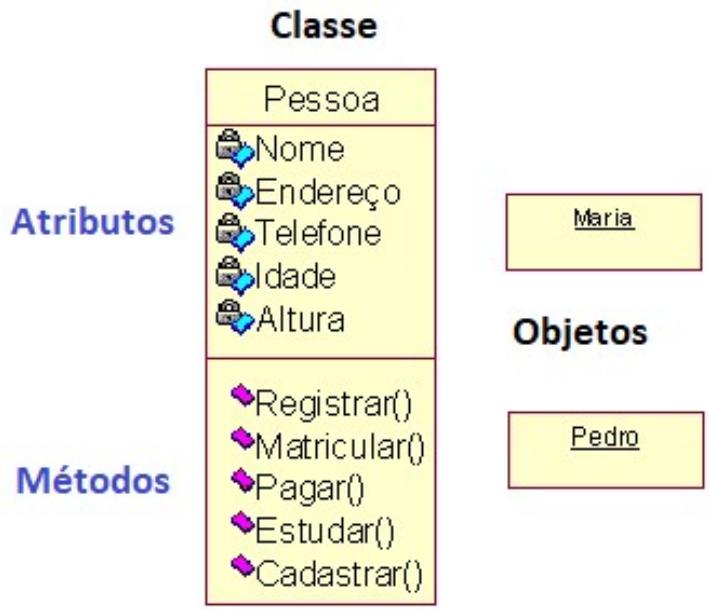
Utilizar a **orientação a objetos** no desenvolvimento de sistemas envolve a **aplicação de conceitos e práticas fundamentais** da Orientação a Objetos (OO) para **modelar, organizar e implementar software de maneira eficiente, modular e reutilizável**.

Abaixo, estão descritos os conceitos-chave da Modelagem OO, além das principais etapas e considerações para utilizar a orientação a objetos no desenvolvimento de sistemas:

1. **Objeto:**
   - Definição: Um objeto é uma **instância concreta de uma classe**. Pode ser uma entidade **física ou conceitual** que **possui características (atributos)** e **comportamentos (métodos)**.
   - Identificação de Objetos e Classes: Objetos e classes **relevantes no domínio do problema**.
   - Ação: Analisar os **requisitos do sistema** para identificar entidades, características e comportamentos que podem ser modelados como objetos e classes. Cartões CRC (Classe, Responsabilidade, Colaboração) são uma técnica eficaz no design orientado a objetos. Cada cartão representa uma classe e lista suas responsabilidades e colaborações com outras classes. Isso facilita a visualização das interações entre as classes, ajudando na identificação de papéis e relações no sistema. Essa abordagem favorece um design claro e coeso, promovendo uma comunicação eficiente entre desenvolvedores durante a fase de modelagem do software.

2. **Classe:**
   - Definição: Uma **classe** é um **modelo ou protótipo para criar objetos**. Ela define as características e comportamentos comuns a um grupo de objetos.
   - Definição de Classes e Atributos: Definir as classes que representarão os objetos no sistema.
   - Ação: Para cada classe identificada, determinar os atributos que representam as características dos objetos.

3. **Atributos:**
   - Definição: Atributos representam as **características de um objeto**. Por exemplo, em uma classe "Carro", os atributos podem incluir cor, modelo e ano de fabricação.



Modelagem Orientada a Objetos

Conceitos-chave da Modelagem OO

4. Métodos:

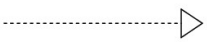
•Definição: Métodos são as ações ou operações que um objeto pode realizar. Eles definem o comportamento de um objeto. Por exemplo, em uma classe "Carro", métodos podem incluir "ligar", "desligar" e "acelerar".

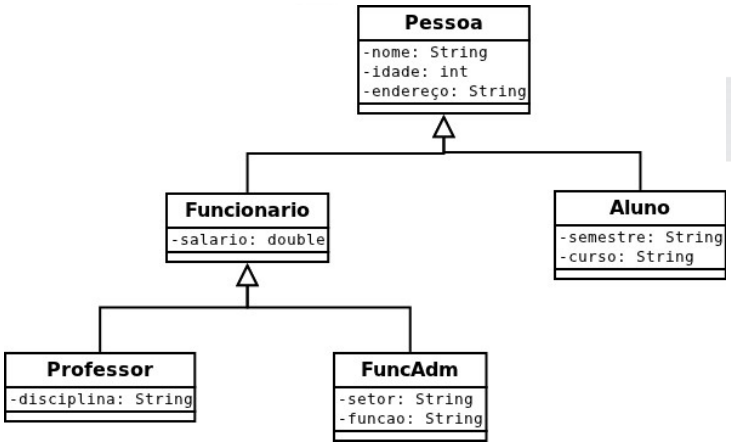
•Estruturação de Métodos: que definirão o comportamento das classes.

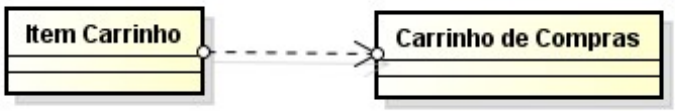
•Ação: Para cada classe, identificar os métodos que representam as ações ou operações que os objetos da classe podem realizar.

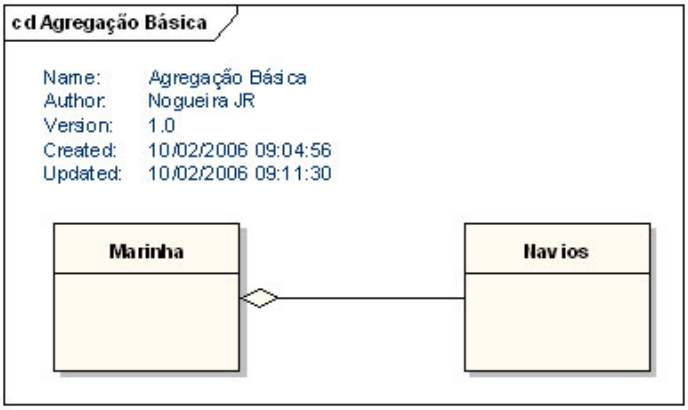
5. Relacionamentos entre Classes:

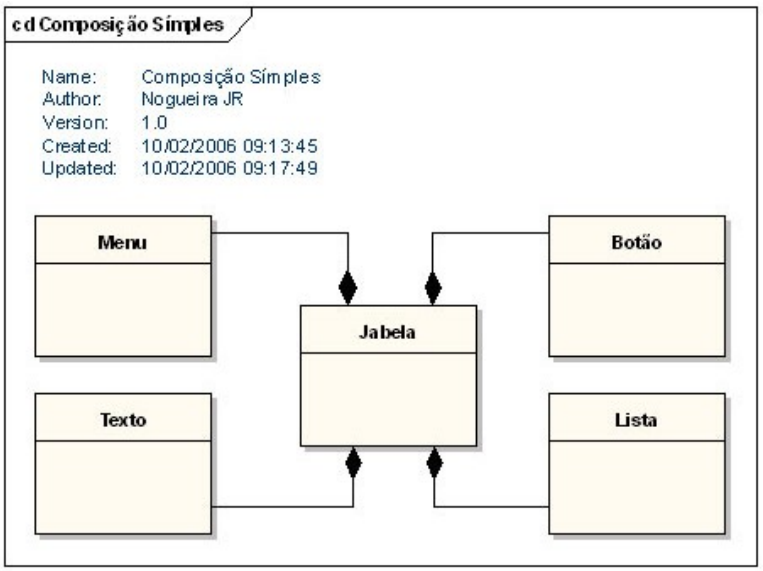
- Descrição: Estabelecer os relacionamentos entre as classes.
- Ação: Identificar associações, heranças, agregações ou composições que modelam as interações entre os objetos.

Relação	Representação Pictórica
Dependência	
Associação	
Generalização	
Realização	









http://www.linhadecodigo.com.br/artigo/943/uml-unified-modeling-language-generalizacao-agregacao-composicao-e-dependencia.aspx

Modelagem Orientada a Objetos

6. Herança:

•Definição: Herança permite que uma classe herde características e comportamentos de outra classe. Isso promove a **reutilização de código e a criação de hierarquias de classes**.

•Generalização: A Classe Pai é uma Generalização da Classe Filho, como por exemplo, um veículo é uma generalização de automóvel (veículo terrestre) e barco (veículo aquático).

•Especialização: A Classe Filho é uma Especialização da Classe Pai, como por exemplo, um avião a jato ou um avião a hélice é uma especialização da classe mais geral veículo aéreo.

7. Encapsulamento:

- Definição: Encapsulamento é o conceito de **esconder os detalhes internos** de uma classe e **expor apenas o que é necessário**. Isso protege a integridade dos dados e facilita a manutenção do código, aplicar o conceito de encapsulamento para proteger a integridade dos dados.
- Ação: Definir a **visibilidade de atributos e métodos**, tornando privados os detalhes internos das classes e expondo apenas o necessário.

8. Polimorfismo:

- Definição: Polimorfismo permite que objetos de diferentes classes sejam tratados de maneira uniforme. Isso ocorre quando uma classe pode ser usada como se fosse outra por meio de herança ou interfaces.
- Herança e Polimorfismo: Utilizar herança e polimorfismo para promover a **reutilização** e **flexibilidade**.
- Ação: Identificar hierarquias de classes onde a herança pode ser aplicada e utilizar polimorfismo para tratar objetos de classes diferentes de maneira uniforme.

9. Associação:

- Definição: Associação representa a relação entre duas ou mais classes. Pode ser unidirecional ou bidirecional e pode incluir multiplicidade, indicando quantos objetos estão envolvidos na associação.

10. Agregação e Composição:

- Definição: Agregação e composição são formas de representar relacionamentos entre objetos. Agregação é uma relação mais fraca, enquanto composição é uma relação mais forte, indicando que um objeto é parte de outro.

11. Visibilidade de Atributos e Métodos:

- Definição: Na modelagem de classes orientada a objetos, a visibilidade refere-se à acessibilidade de atributos e métodos de uma classe por outras classes. A escolha adequada da visibilidade é crucial para encapsular a implementação interna da classe, promovendo a segurança e a manutenção do código. Existem três níveis principais de visibilidade:
  - **Público (`public`)**: Atributos e métodos marcados como públicos são acessíveis de qualquer lugar, tanto dentro da classe quanto por outras classes. Eles são representados com o símbolo `+` em diagramas UML.
  - **Protegido (`protected`)**: Atributos e métodos protegidos são acessíveis dentro da classe que os define e por suas subclasses. Eles são indicados pelo símbolo `#` em diagramas UML.
  - **Privado (`private`)**: Atributos e métodos privados são acessíveis apenas dentro da classe que os define. Outras classes não podem acessá-los diretamente. Eles são representados com o símbolo `-` em diagramas UML.

Modelagem Orientada a Objetos

12. Multiplicidade:

•Na modelagem de classes orientada a objetos, a **multiplicidade em relacionamentos entre classes representa o número de instâncias de uma classe que podem se relacionar com o número correspondente de instâncias da outra classe**. Ela é indicada nos diagramas UML pelos números ou intervalos ao longo das linhas de associação. Por exemplo, em uma associação entre uma classe `Turma` e uma classe `Aluno`, uma multiplicidade de "1" no lado da `Turma` e "0..*" no lado do `Aluno` indicaria que uma turma pode ter exatamente um aluno, mas um aluno pode estar em zero ou mais turmas. Essa especificação ajuda a definir a cardinalidade e a natureza dos relacionamentos entre classes em um sistema orientado a objetos.

13. Diagramas UML:

- Definição: A Modelagem Orientada a Objetos muitas vezes utiliza diagramas UML (Unified Modeling Language) para visualizar e documentar a estrutura e o comportamento de um sistema.
- Ação: Criar diagramas de classes, diagramas de sequência, diagramas de atividade, entre outros, conforme necessário para representar diferentes aspectos do sistema.

14. Implementação:

- Descrição: Implementar as classes e métodos definidos na linguagem de programação escolhida.
- Ação: Codificar as classes, garantindo que a **estrutura e o comportamento estejam alinhados com o modelo orientado a objetos**.

15. Testes:

- Descrição: Realizar testes para garantir a **corretude e robustez do sistema**.
- Ação: Desenvolver casos de teste que cubram diferentes cenários e interações entre objetos.

16. Manutenção e Evolução:

- Descrição: Facilitar a manutenção e evolução contínua do sistema.
- Ação: Atualizar o modelo orientado a objetos conforme novos requisitos surgem, mantendo a consistência entre o design e a implementação.

A Modelagem Orientada a Objetos é uma abordagem poderosa para o desenvolvimento de software, pois oferece uma representação mais intuitiva e natural dos sistemas, flexíveis e fáceis de entender, promovendo a reusabilidade de código e facilitando a colaboração entre membros da equipe. Facilitando o design modular, a manutenção e a evolução do software ao longo do tempo.