# Análise Orientada a Objetos - Aula 01

Benay T. da L. de Carvalho - Universidade Norte do Paraná
RA: 3681235802

**Paradigma Orientado a Objetos**

No desenvolvimento de Software há diversos paradigmas - que são conjuntos de preceitos e métodos que servem para organizar e nos aproximar dos problemas com diferentes formas. Os mais comuns paradigmas são: **Imperativo, Procedural, Funcional, Declarativo e Orientado a Objetos.**

**Pontos Principais da Orientação a Objetos**

**1**. Gira em torno de classes, objetos, métodos e atributos.

**2**. Classes são as classificações dos objetos. A classe pessoa pode ter o objeto funcionário que terá os atributos: Nome, idade, gênero, número de identificação, etc. E os métodos são as ações que o objeto pode fazer, nessa situação por exemplo, o funcionário pode "bater o ponto".

**3**. Utiliza alguns tipos de relacionamentos entre classes, exemplos:

**3.1**. **Herança:** A herança funciona de forma a passar certas características (atributos) de uma classe (Super) para outra (Sub). Exemplificando, a classe liquído (Super) irá herdar para a classe suco (Sub) e para classe refrigerante (Sub), a consistência.

**3.2**. **Polimorfismo:** Digamos que ocorre uma herança entre classes e duas ou mais classes filhas herdam os mesmos elementos de uma classe pai, porém utilizam esses elementos de maneiras diferentes, isso é chamado de polimorfismo. Em suma, é quando diferentes classes têm métodos com mesmo nome mas funções diferentes.

**3.3**. **Encapsulamento:** É a limitação dos atributos e/ou métodos de uma classe pai que são herdados à uma classe filha. Por exemplo, digamos que iremos herdar alguns atributos do CEO para o Gerente de um estabelecimento e para evitar passar algumas permissões que apenas o CEO deve guardar, iremos limitar essa herança para ser herdado apenas o que o Gerente deve ter.

**Unified Modelling Language**

Ou UML é uma linguagem de modelagem que utiliza da Orientação a Objetos. É compatível com desenvolvimento de Software desde os requisitos até a implantação.

**Objetivos da UML**

- Modelar em diferentes linguagens e situações.
- Padrão para o desenvolvimento de software.
- Ser simples.
- Funcionar como forma de documentar para futuras alterações do sistema.

Os diagramas em UML possuem níveis de abstração, isso dita o quão complexo será o diagrama. Quanto maior o nível de abstração mais fácil será o entendimento.

**Níveis de abstração:**

- **Alto:** Ser o mais claro e simples possivel, serve para apresentar ao cliente
- **Médio:** Guiar o desenvolvimento apresentado, sem detalhar muito as classes, os objetos e as interações.
- **Baixo:** Demonstrar como será o sistema de forma precisa, especificando cada interação e cada método.

**Tipos de Diagramas**

1. **Diagramas Estruturais**

   1.1. **Classes:** Representa a estrutura e relação entre as classes que moldam os objetos. **Elementos:** Atributo, Operação e Associação.

   1.2. **Objetos:** Utiliza uma modelagem parecida com o diagrama de classes a diferença é que mostra os objetos que foram instanciados.

   1.3. **Pacotes:** Os pacotes são agrupamentos lógicos formado por pedaços do sistema. Os pacotes se relacionam com outros pacotes através de uma relação de dependência. Este diagrama é muito utilizado para ilustrar a arquitetura de um sistema.

   1.4. **Estruturas Compostas:** Destina-se a descrição dos relacionamentos entre os elementos. Utilizado para descrever a colaboração interna de classes, interfaces ou componentes para especificar uma funcionalidade. **Elementos:** Colaboração, parte, port, papéis.

1.5. **Componentes:** Ilustra como as classes deverão se organizar e relacionar. Pode-se explicitar a qual classe cada componente pertence. Utiliza o diagrama de classes como base.

1.6. **Implantações:** Descreve os componentes de hardware e Software e suas interações com outros elementos. Representa a configuração e arquitetura do sistema. **Elementos:** Nó (Peça física na qual o sistema será implantado), Artefatos(qualquer pedaço físico de informação instalado em um nó).

1.7. **Perfis.**

2. **Diagramas Comportamentais**

2.1. **Casos de Uso:** Descreve a funcionalidade proposta para um sistema. Representa as interações entre objetos e seus métodos de forma graficamente simples. Muito utilizado no levantamento de requisitos. **Elementos:** Ator, caso de uso e conexões(Include, extend, associação, generalização).

2.2. **Atividade:** Essencialmente é um fluxograma das atividades empregadas. Tendo um ênfase no fluxo de controle entre atividades. Diferentemente do caso de uso, no diagrama de atividades não há atores e o foco é somente nas atividades/métodos e suas conexões. **Os principais elementos são:** Atividades, sub-atividades, transição, decisão, ação, bifurcação, raia e envio/recepção de sinal.

2.3. **Máquina de Estados:** Representa o estado ou situação que um objeto pode se encontrar. Foca nas transições de estados de um objeto. **Elementos:** Estado (Condição de um objeto na qual ele executa certas atividades e aguarda certos eventos), Transição (Relação entre dois estados), Condição (Causa necessária para ocorrência de um estado).

3. **Diagramas de Interação (Fazem parte dos diagramas comportamentais)**

3.1. **Sequência:** Segue o diagrama de casos de uso, porém com o adicional de quê mostra as mensagens passadas entre objeto. Descreve a maneira como objetos colaboram em algum comportamento. **Elementos:** Linha de vida (Define o ponto de posição dos atores e objetos), Fragmento (Interações possiveis de um objeto para o outro).

3.2. **Colaboração:** Sendo isomórfico ao diagrama de sequência, a diferença do diagrama de colaboração é focar na organização dos objetos, enquanto o de sequência tem o foco na relação entre objetos e no seu tempo.

3.3. **Tempo:** Sendo semelhante ao dois anteriores, no diagrama de tempo há marcações de tempo para cada evento.

3.4. **Interação:** Diagramas de interação combinam diagramas de atividade com diagrama de sequência. É um tipo de diagrama de atividade que representa o envio ou o recebimento de dados entre um ator e um caso de uso.

### Ciclo de incrementação do Processo Unificado

O processo unificado funciona fazendo constantes incrementações no Software até chegar no resultado desejado. Esse ciclo pode ser representado por 4 fases, veja à seguir: **Concepção, Elaboração, Construção, Implementação.** Dentro das fase há sub-níveis, que são os seguintes: **Comunicação, Planejamento, Modelagem, Construção, Entrega e Incremento.**

A fase de **Concepção** engloba: Comunicação e planejamento. **Elaboração** engloba: Planejamento e modelagem. **Construção** engloba: Modelagem e Construção. **Implementação** engloba: Construção e entrega. e por fim, é feito o Incremento no software.

Diante de cada fase, é utilizado certos diagramas. E a cada fase que se passa, os diagramas da fase anterior ainda são utilizados, por isso, não irei escrever os mesmos a cada fase mas que fique compreendido.

### Fase de Concepção

- Diagrama de Casos de Uso.
- Diagrama de Sequência.
- Diagrama de Colaboração.
- Diagrama de Atividades.
- Diagrama de Máquina de Estados.

### Fase de Elaboração

- Diagrama de Classes.

### Fase de Construção

- Diagrama de Implantação.

### Fase de Implementação

- Diagrama de Componentes.

**Mecanismos Comuns da UML**

1. **Especificação:** É a parte de detalhamento sobre os métodos, descrição dos casos de uso, nome dos objetos, etc. Em suma, é a parte escrita dos diagramas.
2. **Adorno:** É a parte Gráfica-Visual dos diagramas, como os atores e os casos de uso por exemplo.
3. **Divisões.**
4. **Extensões.**

Curitiba
2023