**Unidade 4**

Regina Fedozzi

**Paradigma orientado a objetos**

mostrar a você a importância de como modelar um software usando o processo unificado iterativo e incremental utilizando UML (*Unified Modeling Language* – Linguagem de Modelagem Unificada e

processos unificados (PU).Lhe será apresentado a você o paradigma e os conceitos principais de orientação a objetos e as características do processo unificado (PU);E vamos aprender como desenvolver os principais diagramas da UML.

Pensamos em um paradigma como um padrão de pensamento que guia um conjunto de atividades relacionadas. Na engenharia de software, consideramos um paradigma como um modelo que já foi testado e segue alguns princípios para a resolução de um problema computacional.

Para ajudá-lo nessa empreitada, nesta seção estudaremos os fundamentos do paradigma de orientação a objeto e os seus princípios de como solucionar problemas olhando para cada situação e aprender a visualizar padrões.

Segundo Tucker e Noolan, (2010, p. 3) “[…] um paradigma de progra­mação é um padrão de resolução de problemas que se relaciona a um deter­minado gênero de programas e linguagens”.

Conforme podemos observar, padrão e modelo são termos utilizados como sinônimos, e um modelo nada mais é que uma representação do mundo real.

Conceitos fundamentais de orientação a objetos, tais como: abstração (objeto), classe, herança e polimorfismo.

Entre os conceitos e princípios básicos de POO (Programação Orientada a Objetos) destacaremos:

A **abstração** é a ideia central do paradigma orientado a objetos. No processo de abstração, nos referimos a um **objeto** sem nos preocuparmos com detalhes, como cor, tamanho, código e validade, entre outros. Suponha que você ouviu o termo cadeira: você pensa na ideia de como é uma cadeira, e isso é uma abstração.

A **classe** é a representação da abstração; é o momento em que você define as características que toda cadeira deverá ter e quais ações que ela poderá fazer.

As denominações técnicas para as características são **atributos**, e as ações ou comportamentos chamamos de **métodos**.

Assim, a nossa ideia de cadeira se concretiza quando construímos a classe Cadeira e definimos seus atributos, por exemplo: nossa classe Cadeira tem pés, rodinhas, encosto, assento e cor – esses são seus atributos. E ela move o encosto, eleva o assento e anda – esses são seus métodos. Sobre a classe, ainda podemos afirmar que é algo abstrato, mas tornou-se um uma definição do projeto de uma cadeira. Quando uma cadeira é construída a partir desta classe, então temos um objeto do tipo Cadeira, ou seja, um objeto da classe cadeira.

O **objeto** é a materialização de forma única de uma classe, e a esta materialização damos o nome de instância da classe. Portanto, um objeto é uma instância de uma classe. Os objetos são únicos e distin­guem-se pelos valores dos seus atributos e ações (comportamentos), apesar de pertencerem a uma mesma classe.

O **objeto** é a materialização de forma única de uma classe, e a esta materialização damos o nome de instância da classe. Portanto, um objeto é uma instância de uma classe. Os objetos são únicos e distin­guem-se pelos valores dos seus atributos e ações (comportamentos), apesar de pertencerem a uma mesma classe.

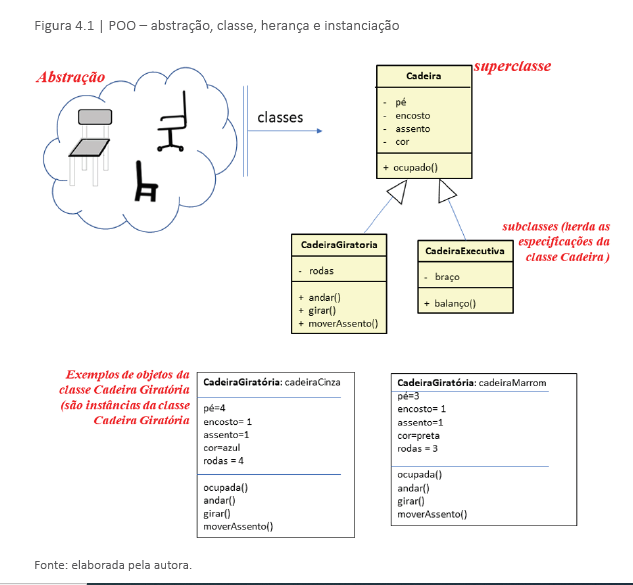
Assim, uma **classe** é a representação de um conjunto de objetos; em outras palavras, é a representação da abstração do objeto com suas características e comportamentos.

Ao construírmos a *cadeiraCinza* a partir da classe Cadeira que definimos no item (3), temos um objeto do tipo Cadeira, com cor cinza, rodinhas, pés, encosto e assento, que move o encosto e pode andar. A *cadeiraCinza* é única. Em um programa, temos a classe Cadeira, e ao instanciarmos a *cadeiraCinza* a partir da classe Cadeira, ela terá um espaço na memória do compu­tador, no qual serão armazenados seus atributos e métodos e um endereço único que a identifica, isto é, o objeto é a materialização de um exemplar do tipo da classe que o define.

A **herança** “permite criar novas classes a partir de classes já existentes, sem duplicar nenhum código” (REZENDE, 2002, p. 214).

No processo de abstração podemos definir classes abrangentes, e durante o processo de modelagem serão refinadas e construídas subclasses que poderão herdar as características e comportamentos da classe genérica. A classe genérica denominamos de **superclasse**, e a classe que herda as características da superclasse chamamos de **subclasse**,

É importante ressaltar que a subclasse pode acrescentar novas carac­terísticas e comportamentos e alterar as já existentes, pois a subclasse é uma nova classe. Essas alterações têm efeito somente na nova classe especificada.



O **encapsulamento** consiste na junção de partes isoladas de um programa, e essas partes podem ser acessadas separadamente.

Na POO, o encapsulamento tem capacidade de tornar a visibilidade das informações e os detalhes da implementação dos métodos de uma classe oculta ou restrita.

Em outras palavras, ocultar do usuário da classe como ela faz uma determinada ação ou como os dados são representados.

Segundo Rezende “[…] é o processo de dissimulação de todos os detalhes de um objeto que não contribuem para suas características essenciais”

Um exemplo de aplicação do conceito de encapsulamento pode ser visto quando você usa o caixa eletrônico (um objeto). Você apenas informa sua identificação e senha para acesso, e os serviços são liberados para seu uso. Você não sabe como o caixa fez a validação, onde estava armazenado sua identificação, como ele descriptografou sua senha etc. Isso é um exemplo de encapsulamento, e você pode ter acesso aos dados encapsulados pela classe apenas por meio de mensa­gens com o objeto: você solicita e ele responde.

“-“ (um traço) para indicar que o atributo ou o método está encapsulado.

“+” (adição) indica que o atributo ou método é visível por outros objetos, ou seja, é público

O **polimorfismo** “[…] significa que a mesma operação [método] pode atuar de modos diversos em classes diferentes” (REZENDE, 2002, p. 214). O polimorfismo é uma característica da POO que dá às classes flexibilidade na construção da solução e no reuso do código.

Agora que você já conheceu os principais conceitos e princípios de POO, será mais fácil perceber as vantagens deste paradigma, pois ele envolve a análise, o projeto e a implementação.

A POO (Programação Orientada a Objeto) aplica os conceitos de OO no desenvolvimento do código.

Um instrumento de OO utilizado na análise é o caso de uso

Já a A/POO (Análise e Projeto Orientado a Objeto) aplica os conceitos de OO na análise e na elaboração do projeto, que são fases que antecedem a programação.

Um instrumento de OO utilizado no projeto é a UML (Linguagem de Modelagem Unificada).

Sommervile explica que um projeto que venha a utilizar o paradigma orientado a objeto deve aplicar em todo o seu processo de desen­volvimento esta estratégia, atentando para as práticas utilizadas no projeto e na programação (SOMMEVILLE, 2009).

Quando construídos corretamente, sistemas orientados a objetos são flexíveis a mudanças, possuem estruturas bem conhecidas e proveem a oportunidade de criar e implementar componentes com alto grau de reutili­zação” (RAMOS, 2017, [s.p.]).

Assim, podemos identificar as seguintes vantagens na Orientação a Objetos:

1. Reutilização de código.
2. Utilização de um único padrão conceitual para a análise, o projeto e a implementação.
3. O tempo de desenvolvimento do software é mais rápido.
4. A construção de sistemas mais complexos é simplificado pelo fato de cada objeto ser simples, fácil de testar e integrar ao demais objetos. Este fator também alonga o ciclo de vida do software, pois as horas demandadas para o desenvolvimento do software são gastas na manutenção e no incremento de novas funcionalidades, e o conceito de objeto e os mesmos interagirem entre si, facilitando implementar novos recursos; logo o tempo de vida das aplicações é mais longo.
5. Pode ter os custos de desenvolvimento reduzidos.

**Classe:** representação de um conjunto de objetos. Em outras palavras, é a representação da abstração do objeto com suas características e comportamentos.

**Objeto:** é a materialização da classe, qualquer coisa do mundo real.

**Herança:** aplicamos o conceito de herança quando elaboramos uma classe que herda as características e operações de outra classe.

**Polimorfismo:** quando objetos de um mesmo tipo de classe apresentam comportamentos diferentes. Dois carros são objetos da classe carro, porém um usa embreagem para mudar as marchas e outro não.

**Encapsulamento:** isola os atributos da classe e suas ações, garantindo a integridade e a ocultação dos dados e ações. O relacionamento entre as classes passa a ser feito por mensagens. Por exemplo: no objeto controle remoto da televisão, você pressiona o comando e ele transmite a mensagem para o aparelho de televisão, que executa o pedido, mas você não precisa ter conhecimento de como isso é feito.