

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/341327632>

Object Oriented Paradigm (OOP)

Article · January 2017

CITATIONS

0

READS

69

1 author:



[Alvaro F. Pinheiro](#)

University of Pernambuco

29 PUBLICATIONS 5 CITATIONS

SEE PROFILE

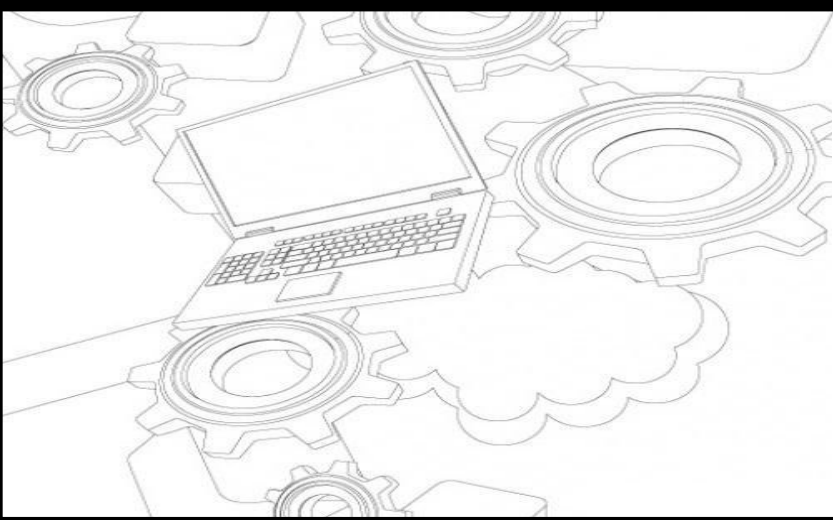
Álvaro Farias Pinheiro

Revista Fundamentos da Engenharia de Software

Paradigma Orientado a Objetos

P OO

Ano II – Número 4
Abril de 2017



Série Fundamentos da Engenharia de Software

Paradigma Orientado a Objetos

PINHEIRO, Álvaro Farias
Autor

Série Fundamentos da Engenharia de Software

Paradigma Orientado a Objetos

Publicação 2017

O autor acredita que todas as informações aqui apresentadas estão corretas e podem ser utilizadas para qualquer fim legal. Entretanto, não existe qualquer garantia explícita ou implícita, de que o uso de tais informações conduzirá sempre ao resultado desejado. Os nomes de sites e empresas, por ventura, mencionados, foram utilizados apenas para ilustrar os exemplos, não tendo vínculo nenhum com o livro, não garantindo a sua existência nem divulgação. Eventuais erratas estarão disponíveis para download no site de publicação.

As imagens utilizadas neste livro foram obtidas na Internet.

Dados da Publicação

Pinheiro, Álvaro Farias

Série Fundamentos da Engenharia de Software: Paradigma Orientado a Objetos

Ano II – Número 4 – Recife, Abril de 2017

Selo Editorial: Publicação Independente

1. POO Introdução

2. POO Conceitos

Série Fundamentos da Engenharia de Software

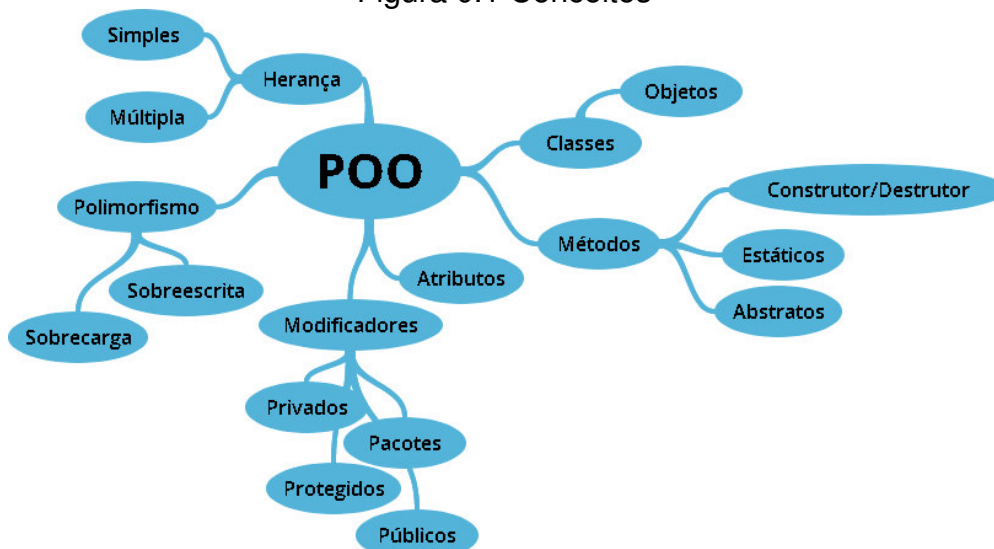
Paradigma Orientado a Objetos

Publicação Independente
Revista em português com o título
Paradigma Orientado a Objetos
Série Fundamentos da Engenharia de Software
Ano II – Número 4
Recife – Pernambuco – Brasil
Abril de 2017

Introdução

Paradigma Orientado a Objetos (POO) consiste em expressar os problemas como objetos, ao contrário da análise tradicional os quais eram em rotinas e dados, que aqui foram substituídos por métodos (comportamento) e atributos (propriedades). Assim, quando é colocado o problema de desenvolver um sistema na análise orientada a objetos, deve-se pensar como dividir esse problema em objetos.

Figura 0.1 Conceitos



Fonte: Próprio Autor

Classes

Pensando em um sistema acadêmico em POO teríamos Alunos, Administradores, Professores, Cursos, Turmas, etc. E a melhor maneira de conceituar estes termos é considerar um objeto do mundo real e mostrar como podemos representá-lo em termos conceitos em POO. Assim, um objeto é um conceito que usamos para representar uma entidade do mundo real. Exemplificando, um Aluno possui nome, data de nascimento, identidade, etc. E além dessas características (propriedades), possuem ações (métodos) como frequentar as aulas, fazer as avaliações, etc. Em termos de POO para podermos tratar os objetos temos que criar classes. Assim, uma classe representa um conjunto de objetos que possuem comportamentos e características comuns.

Na UML em primeiro lugar denomina-se uma classe e recomenda-se nomeá-la capitalizando-a e deixando no singular. E em segundo lugar denominam-se as propriedades, informações específicas relacionadas a uma classe de objeto, isto é características dos objetos que as classes representam. Em terceira e última parte, métodos, que são ações que os objetos de uma classe podem realizar. Assim tem-se um modelo para instanciar quantos objetos forem necessários. Dessa forma os objetos instanciados possuirão todas as características e comportamentos definidos pela classe. Então as classes especificam a estrutura (propriedades) e os comportamentos (operações) dos objetos, que são instâncias das classes.

Geralmente em um sistema de médio porte serão identificadas diversas classes que compõem o sistema. Neste contexto a UML surgiu como uma proposta de ser uma linguagem para modelagem de dados que usava diversos artefatos para representar o modelo de negócio; um destes artefatos é o diagrama de classes. Os diagramas de classes registram atributos e operações de uma classe e as restrições de como os objetos podem ser conectados, descrevendo também os tipos de objetos no sistema e os relacionamentos entre eles e esse podem ser associações e abstrações. Para poder representar a visibilidade dos atributos e operações, a UML usa os seguintes símbolos: + público, visível em qualquer classe; - privado, visível somente dentro da classe; # protegido, na classe e suas subclasses.

O relacionamento entre classes retrata as relações entre os objetos. Exemplo: um professor ministra uma disciplina para alunos numa sala. A UML reconhece três tipos mais importantes de relações: dependência, associação e generalização ou herança. Geralmente as classes não estão isoladas (coesas) e se relacionam entre si (acoplamento). O relacionamento e a comunicação entre as classes se definem em 3 tipos: Associações que podem ser de Agregação ou Composição; Generalização ou herança; e Dependências.

As associações são relacionamentos estruturais entre instâncias e especificam que objetos de uma classe estão ligados a objetos de outras classes, podendo ser unária, binária, etc. As associações podem existir entre classes ou entre objetos. Uma associação entre a classe Professor e a classe Disciplina (um professor ministra uma disciplina) significa que uma instância de Professor (um professor específico) vai ter uma associação com uma instância de uma Disciplina. Esta relação significa que as instâncias das classes são conectadas, seja fisicamente ou conceitualmente.

Dependências são relacionamentos de utilização no qual uma mudança na especificação de um elemento pode alterar a especificação do elemento dependente. A dependência entre classes indica que os objetos de uma classe usam serviços dos objetos de outra classe. Generalização ou herança, que pode ser simples ou múltipla (composta), serve para relacionar um elemento mais geral e um mais específico, onde o elemento mais específico herda as propriedades e métodos do elemento mais geral. Como a relação de dependência, ela existe só entre as classes. Um objeto particular não é um caso geral de outro objeto, apenas classes podem receber esse conceito.

Agregação é um tipo de associação (é parte de ou todo-parte) onde o objeto parte é um atributo do todo, onde o objeto parte somente são criados se o todo ao qual estão agregados seja criado. Pedidos é composto por itens de pedidos. Composição é o relacionamento entre um elemento (o todo) e outros elementos (as partes) onde as partes só podem pertencer ao todo e são criadas e destruídas com ele.

O diagrama de classes lista todos os conceitos do domínio que serão implementados no sistema e as relações entre os conceitos. Ele é muito importante, pois define a estrutura do sistema a desenvolver. O diagrama de classes é consequência do prévio levantamento de requisitos, definição de casos de usos e classes. Como exemplo tem os passos de elicitação de requisitos com os stakeholders do sistema a ser desenvolvido, usando a técnica de entrevista com os administradores, professores, etc. que a partir

desses os objetos do sistema são definidos: Alunos, Professores, Turmas, Cursos, etc. Definição dos atores do sistema: aluno, professor, administrador, etc. Definição e detalhamento dos casos de uso: Matricular Aluno, Pagar Matrícula, etc. Definição das classes: alunos, professor, etc.

Atributo representa uma propriedade que todos os objetos da classe têm, porém cada objeto terá valores particulares para seus atributos. A UML, o nome de um atributo é um texto que deve capitalizar todas as primeiras letras de cada palavra no nome menos a primeira palavra. Todos os métodos têm que respeitar exatamente a assinatura que é composta pelo nome, número de parâmetros, tipos de dados e ordem. Um método não pode acrescentar ou cortar um parâmetro. Para mandar a mensagem corretamente, devesse saber qual é a classe do objeto, já que cada classe tendo método com assinatura diferente.

Objetos

Agora que sabemos o conceito e como codificar uma classe, vamos entender sobre objeto. Segundo os pais da UML (Rumbaugh, Jacobson e Booch) um objeto é uma instância de uma classe, isto é, trata-se de uma cópia da classe na memória em tempo de execução (runtime), sendo assim, um elemento específico que possui valores (estados) nos atributos (campos). A UML representa um objeto usando um retângulo que o representa, onde o nome da instância do objeto é seguido de dois pontos, seguido do nome da classe, com essa formação sublinhada.

Métodos

A classe principal em Java temos um método principal "main" que por default recebe um array de caracteres que poderá ser preenchido quando o programa principal for carregado. Ao carregá-lo uma mensagem irá aparecer na tela "Método chamado!", pois quando um objeto da classe é instanciado o método exemplo é invocado.

Figura 1.2 Tipos de Métodos em POO



Fonte: Próprio Autor

Relacionamentos

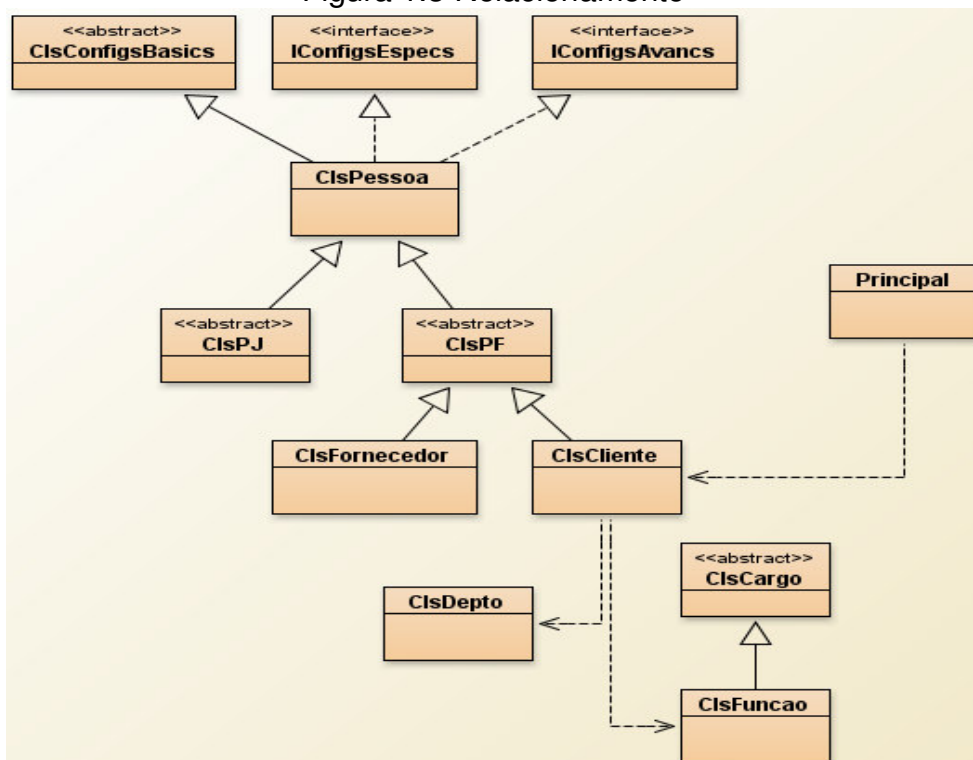
O que vamos estudar agora é como em Java se codifica os relacionamentos, isto é as associações e heranças. Também iremos codificar os três tipos de classes: Concreta, Abstrata e Interface. Observe o diagrama de class:

Temos a seguinte modelagem. A classe principal está associada à classe cliente, esta por sua vez está associada às classes departamento e função. A

classe função estende a classe abstrata cargo. A classe cliente é uma subclasse da classe pessoa-física que também é pai da classe filha fornecedor.

Ambas as classes, cliente e fornecedor são concretas, assim permitindo que se instanciem objetos dessas classes.

Figura 1.3 Relacionamento



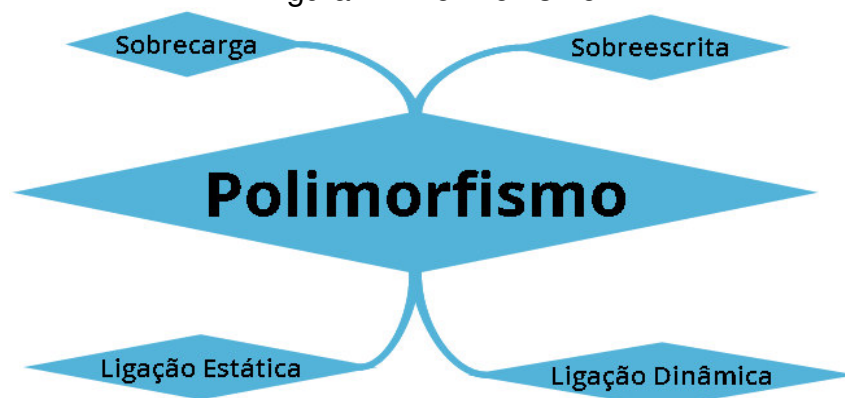
Fonte: Próprio Autor

Já as classes pessoa-física e pessoa-jurídica são abstratas, assim sendo possuem métodos não implementados e por consequência não permitindo que se instanciem objetos diretamente dessas classes. Essas duas últimas classes são filhas da classe concreta pessoa que estende a classe abstrata configurações básicas e implementa (realiza) as interfaces configurações especiais e configurações avançadas.

Polimorfismo

É a capacidade de classes mais abstratas possuírem comportamentos diferentes das classes concretas, isto é, duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura), mas comportamentos distintos, especializados para uma das classes derivadas. A decisão sobre qual o método que deve ser selecionado, de acordo com o tipo da classe derivada, é tomada em tempo de execução, através do mecanismo de ligação tardia. Observe a figura 10.5.

Figura 1.4 Polimorfismo



Fonte: Próprio Autor

Mas para ser considerado polimorfismo, é necessário que os métodos tenham exatamente a mesma identificação, quer dizer, o mesmo nome de método, sendo utilizado o mecanismo de redefinição de métodos, isto é, sobreescrita dos métodos.

Override ou Sobreescrita, a redefinição ocorre quando um método cuja assinatura já tenha sido especificada recebe uma nova definição, ou seja, uma nova implementação em uma classe derivada. O mecanismo de redefinição, juntamente com o conceito de ligação tardia, é a chave para a utilização do polimorfismo. É importante observar que, quando polimorfismo está sendo utilizado, o comportamento que será adotado por um método só será definido durante a execução.

Overload ou Sobrecarga, um método aplicado a um objeto é selecionado para execução através da sua assinatura e da verificação a qual classe o objeto pertence. Dois ou mais métodos de uma mesma classe podem ter o mesmo nome, desde que suas listas de parâmetros sejam diferentes, constituindo assim uma assinatura diferente. Tal situação não gera conflito, pois o compilador é capaz de detectar qual método deve ser escolhido a partir da análise dos tipos de argumentos do método.

Early Binding ou Ligação Prematura, quando um método é invocado durante a compilação do programa, o mecanismo de ligação é prematura.

Late Binding ou Ligação Tardia, quando a definição do método que será efetivamente invocado só ocorre durante a execução do programa, o mecanismo de ligação usado é o de tardia também conhecido pelos termos dynamic binding ou run-time binding.

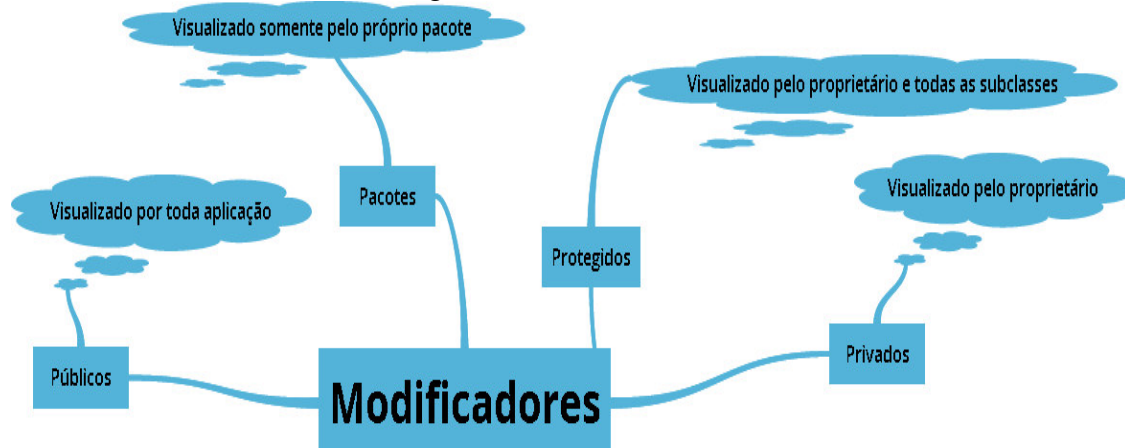
Em Java, todas as determinações de métodos a executar ocorrem através de ligação tardia exceto em dois casos: métodos declarados como final não podem ser redefinidos e, portanto não são passíveis de invocação polimórfica da parte de seus descendentes e métodos declarados como private são implicitamente finais.

Modificadores

`public boolean equals(Object obj)` é um método proveniente da classe `Object`. Por default, sua comparação utiliza o operador `==` para comparar os dois objetos. Mas o objetivo aqui é entender a palavra reservada `public`.

Existem quatro modificadores básicos: public, private, protected e package. A figura abaixo explica a visibilidade que cada um desses modificadores aplicam sobre as classes e métodos.

Figura 1.5 Modificadores



Fonte: Próprio Autor

Livros da série Fundamentos da Engenharia de Software



Fundamentos da Engenharia de Software: Conceitos Básicos é uma coletânea de disciplinas que integradas servem para fundamentar o

entendimento da construção de projetos de software com qualidade, isto é, baseado em processos maduros e reconhecidos pela comunidade tecnológica. O objetivo deste livro é fornecer ao leitor as bases necessárias para o desenvolvimento de aplicações sejam Desktop, Web ou Mobile. Iniciando a leitura na Teoria da Computação, passando por Processos, Linguagens, Bancos de Dados e finalizando com Sistemas de Informação e Colaboração. Este livro pode ser lido capítulo a capítulo ou somente a disciplina desejada, pois sua elaboração consiste na compilação das disciplinas fundamentais da Engenharia de Software que são independentes, mas ao mesmo tempo se integram objetivando o desenvolvimento de aplicações.



Introdução à Banco de Dados. Neste são abordados os conceitos básicos de bancos de dados e seus sistemas gerenciadores, mas com o foco

na arquitetura relacional, porque ainda hoje o mercado faz uso em larga escala desses bancos de dados, mesmo que o paradigma predominante seja o orientado a objetos e que, já existam a um bom tempo bancos orientados a objeto, até mesmo os bancos objetos-relacionais que são um híbrido entre essas duas arquiteturas, o que predomina ainda é o relacional, assim, este material é focado na linguagem de consulta estruturada para os SGBD-Rs do mercado, com foco na comparação de cinco dos mais utilizados bancos relacionais, os quais são: Oracle, SQLServer, MySQL, SQLBase e Interbase.



Este livro é sobre processos de desenvolvimento de software, evidenciando a necessidade de qualidade na construção de sistemas, conceituando a diferença entre desenvolvimento Adhoc e com processo. Para isso é realizado a introdução à engenharia de requisitos abordando as técnicas para a elicitação de requisitos que forneçam subsídios necessários para uma construção de software com maior qualidade, enfatizando a necessidade de se aplicar na construção de qualquer sistema as técnicas de análise e modelagem, evidenciando o uso da linguagem da Linguagem de Modelagem Unificada (UML) para diagramar um projeto de software, explicando a necessidade do uso de modelos na construção, entrando com detalhes na análise orientada a objetos, com o objetivo de explorar os seus conceitos de requisitos e modelagem integrados. Este material é finalizado com a introdução à medidas de esforço de desenvolvimento, técnica necessária para responder as perguntas básicas de qualquer desenvolvimento: Qual o prazo e custo? E para responder a essas questões é abordado o uso da métrica análise de ponto de função.



Este livro aborda os sistemas que são classificados como informação, a exemplo, sistemas de apoio a decisão, sistemas estratégicos, sistemas

gerenciais e sistemas transacionais. A produção deste material que compõe o volume 4 da coleção Fundamentos da Engenharia de Software é resultado da compilação das aulas produzidas nas disciplinas que compõem os capítulos deste livro.



A motivação deste livro é exemplificar os conceitos de Padrões de Projetos utilizando a linguagem de programação Java, sendo a construção uma

compilação das aulas produzidas com o intuito de facilitar o entendimento do assunto abordando os seguintes temas: Paradigma Orientado a Objetos que introduz o leitor nos conceitos do POO; Linguagem de Modelagem Unificada para apresentar a simbologia UML dos conceitos de POO; Linguagem de Programação Java apresentando essa poderosa linguagem de programação orientada a objetos para exemplificar os padrões de projeto; e Padrões de Projetos que neste livro aborda os mais referenciados nas academias, sendo eles o GRASP e GoF.



Este livro é o resultado do uso da ferramenta MS Project da Microsoft utilizada na aplicação dos conceitos de gestão de projetos do PMBOK com as premissas da engenharia de testes para aquisição de qualidade nos produtos de software.



desenvolvimento e lógica de programação.

Este livro aborda basicamente os conceitos básicos de programação como autômatos, tipos de linguagens, princípios dos compiladores, paradigmas de



dicas de codificação para a linguagem de marcação ASPX, para a linguagem de script mais utilizada pelos navegadores o JavaScript com exemplos de CSS e principalmente dicas de código para a linguagem de programação CSharp e de banco de dados SQL com foco no SQLServer.

Este livro introduz nas tecnologias Web abordando os conceitos básicos para desenvolvimento para Internet com a apresentação da plataforma Dot Net e exibindo