Pós-Graduação

Projetos ágeis e análise de sistemas

Tema 04 – Paradigma orientado a objetos: histórico, características e conceitos

Bloco 1

Juliana Schiavetto Dauricio

Objetivos

1. Apresentar, conceituar e contextualizar o paradigma da orientação a objetos.

2. Incentivar o aprendizado contínuo e aplicado para a análise de sistemas sob a perspectiva de projetos ágeis.

3. Abordar os principais aspectos da orientação a objetos de forma que o aluno possa conhecer e aprender a utilizar as ferramentas de análise de sistemas, minimizando custos, erros e cumprindo os prazos determinados para as entregas.

Introdução

Vamos reforçar os estudos do que é o paradigma de orientação a objetos e como este é empregado na análise e desenvolvimento de sistemas.

> A abordagem orientada a objeto considera um sistema como uma coleção de objetos autocontidos, que inclui tanto dados como processos. As metodologias tradicionais de análise e o *design* de sistemas são centrados em dados ou em processos (DENNIS *et. al.,* 2014, p. 490).

Como surgiu a Orientação a Objetos (OO)?

- **Década de 60:**
  - ***hardware* não tinha a mesma capacidade de armazenamento.**
  - **analistas e desenvolvedores conseguiam fazer apenas sistemas pequenos e pouco complexos.**
  - **surgiram as linguagens Simula e Smaltalk:**

*Já na década de 60, começaram a surgir linguagens como **Simula e Smaltalk** que permitiam sustentar o princípio de que os programas deviam ser estruturados com base no problema a ser resolvido, fundamentando a construção nos objetos de negócio em si.*

Década de 70:

- **conceito de programação modular;**
- **desenvolvimento do sistema era baseado em funções, que eram o ponto-chave para o programa,**
- **foco em procedimentos de execução das ações do programa.**

A análise e o desenvolvimento fundamentavam-se no computador e na linguagem de programação, isto é, na solução e não no problema a ser resolvido. O ponto central eram os procedimentos e as funções, implementados em blocos estruturados, com comunicação por passagem de dados (LIMA, 2011, p. 17).

Mas afinal, o que é um objeto?

"[...] tudo o que é manipulável ou manufaturável; tudo o que é perceptível por qualquer dos sentidos; coisa, peça, artigo de compra e venda; matéria; assunto; [...] (LIMA, 2011, p. 19).

Programação estruturada e orientada a objetos:

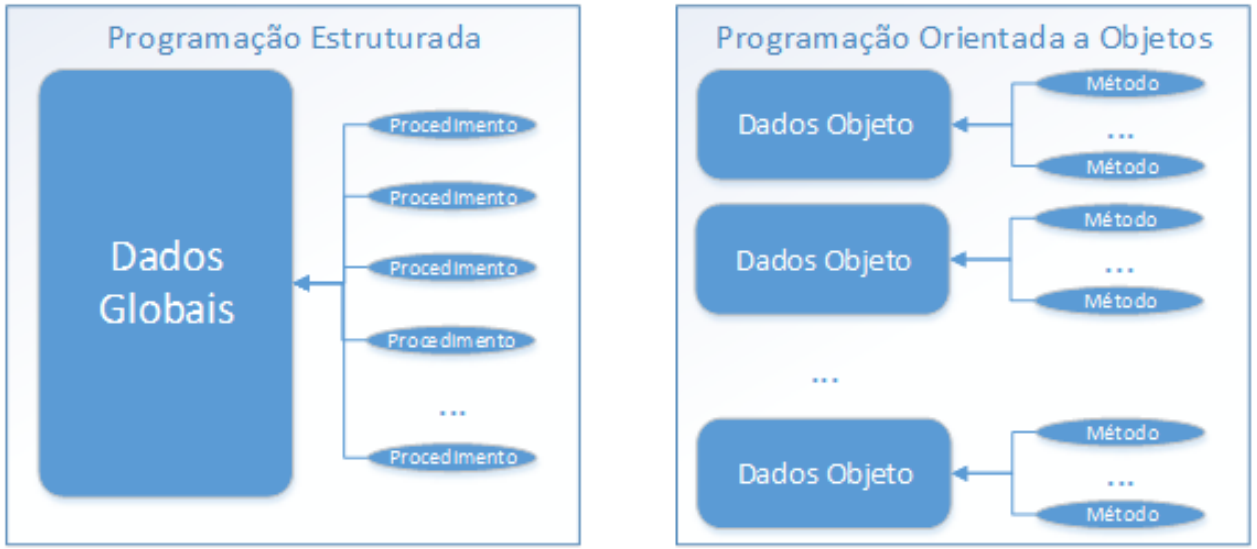


Figura 1: Paradigma de programação estruturada x Orientada a objetos.

Fonte: MACHADO, Henrique. Os quatro pilares da programação orientada a objetos. DevMedia. Disponível em: <http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Acesso em: 30 abr. 2016.

Programação estruturada

- **A figura 1 ilustra como é concebido um projeto de *software* que se fundamenta no paradigma de programação estruturada.**
- **Considera os elementos que provocarão ações do sistema como sendo de acesso global.**

Programação orientada a objetos:

- Todos os métodos e procedimentos estão diretamente relacionados ao próprio objeto, de forma a manter o seu estado ou comportamento.

Figura 2: Paradigma de programação estruturada x orientada a objetos.

Fonte: MACHADO, Henrique. Os quatro pilares da programação orientada a objetos. DevMedia. Disponível em: <http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Acesso em: 30 abr. 2016.

Quadro 1: Características de paradigmas de programação: estruturada x orientada a objetos

Programação orientada a objetos	Programação estruturada
Métodos	Procedimentos e funções
Instâncias de variáveis	Variáveis
Mensagens	Chamadas a procedimentos e funções
Classes	Tipos de dados definidos pelo usuário
Herança	Não disponível
Polimorfismo	Não disponível

Fonte**:** ABILIO, Igor. DevMedia. Programação orientada a objetos *versus* programação estruturada. Disponível em: <http://www.devmedia.com.br/programacao-orientada-a-objetos-versus-programacao-estruturada/32813>. Acesso em: 30 abr. 2016.

Programação orientada a objetos

- **A programação orientada a objetos considera os métodos de cada objeto.**
- **Ao invés de variáveis, um objeto contém atributos e métodos.**
- **A partir da análise de um objeto é possível visualizar com maior clareza o seu comportamento.**
- **Em OO utiliza-se o conceito de classes, ou seja, uma classe que pode conter mais de um objeto inclusive, com seus respectivos métodos.**
- **Então, é preciso modelar esse sistema.**

Programação orientada a objetos

- É preciso que analista saiba **identificar** a partir de um **exercício de abstração**, os **objetos do sistema**, quais serão as suas principais **funções** e ainda, quem são as **pessoas que interagem** com esta **ação**.

Identificando as classes

"Uma classe é o modo geral que usamos para definir e criar instâncias específicas ou objetos. Cada objeto é associado a uma classe" (DENNIS; WIXON; ROTH, 2014, p. 491).

Atenção a definição de um objeto

"Uma classe é o modo geral que usamos para definir e criar instâncias específicas ou objetos. **Cada objeto é associado a uma classe**." (DENNIS; WIXON; ROTH, 2014, p. 491).

Definição de uma classe

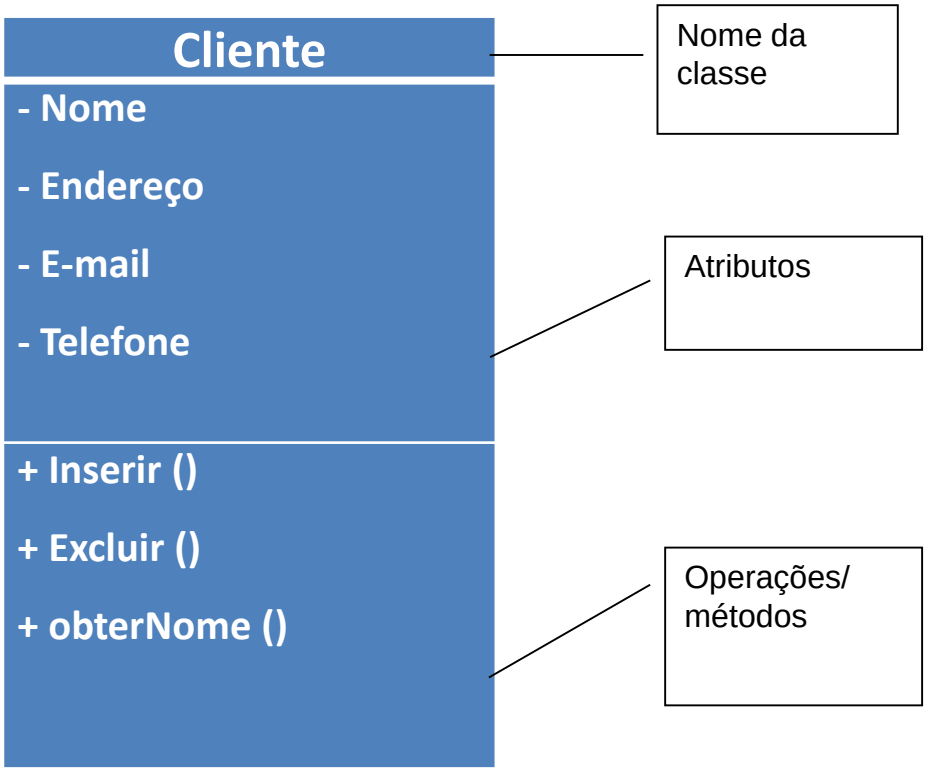


Figura 3: Exemplo de classe em OO
Fonte: O autor (2016).

Leia no material:

Ficam as recomendações de leitura do seu material didático.

Bons estudos!

Pós-Graduação

Projetos ágeis e análise de sistemas

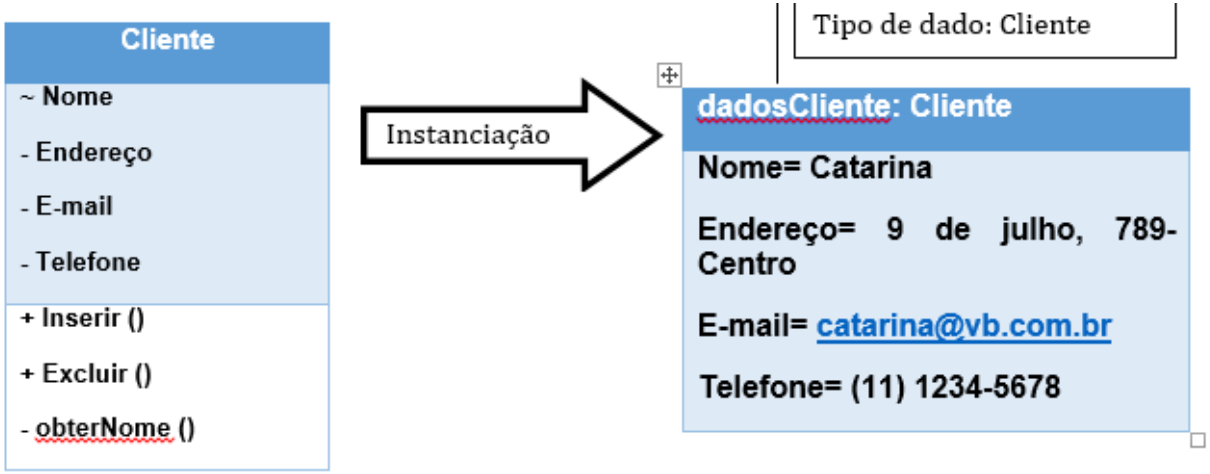
Tema 04 – Paradigma orientado a objetos: histórico, características e conceitos

Bloco 2

Juliana Schiavetto Dauricio

Classe e objeto

Figura 4: Instância



Fonte: Adaptado de Dennis *et. all.*, 2011, p. 491.

Uma classe também pode ser identificada pela classificação dos objetos a ela interligada, que têm o mesmo tipo de dado e comportamento. Nesse caso, cada objeto que pertence a uma classe é chamado de **instância de classe** (LIMA, 2011).

Visibilidade dos atributos e métodos

- **pública (+)** em que outras classes poderão ter acesso a esse atributo;
- **privada (-)** o atributo pode ser utilizado e visto apenas pela própria classe;
- **protegida (#)** o acesso se dá apenas pela própria classe e suas subclasses;
- **de pacote (~)** faz com que o atributo seja acessível pelas classes do pacote que a contém.

O que são métodos em OO?

Para invocar um método de um objeto, envia-se uma mensagem para ele especificando o nome do objeto, o método a ser executado e a lista de argumentos requeridos. Após a execução, o objeto pode ou não retornar um valor como resposta à mensagem recebida (LIMA, 2011, p. 23).

Principais conceitos em OO

Herança
Polimorfismo
Encapsulamento

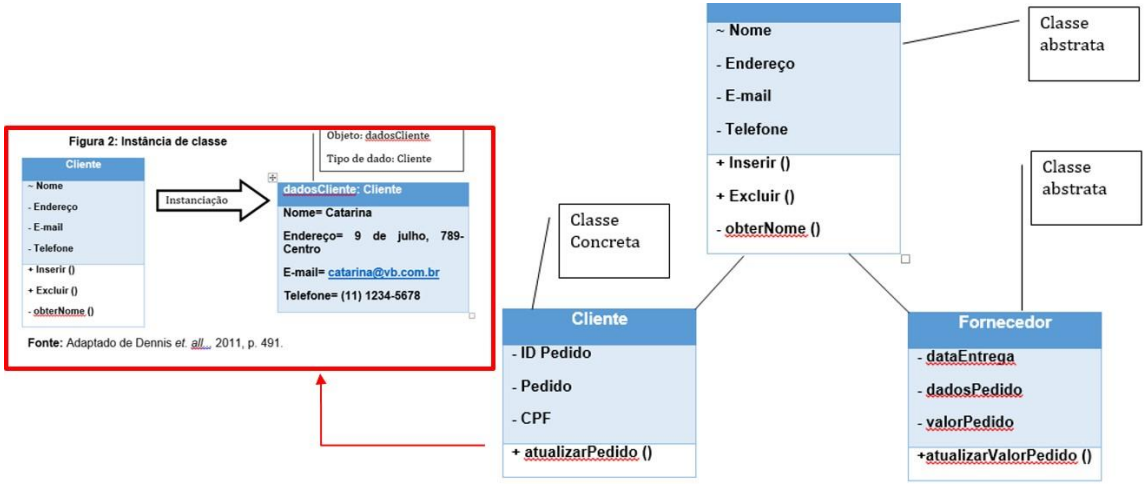
Herança

Quando estabelece uma relação de herança, significa que os atributos da **classe abstrata** ou superclasse também são atributos das subclasses, no entanto, estas também podem conter os seus próprios atributos e métodos, além dos herdados.

Herança

- O conceito de herança remete a uma **hierarquia**. Observe a figura a seguir que exemplifica este conceito:

Figura 5: Classe concreta x abstrata



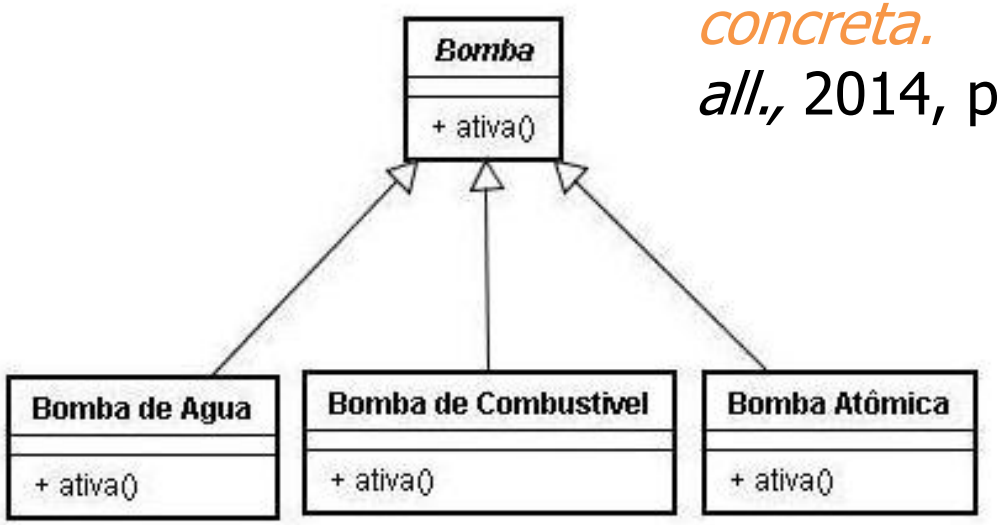
Fonte: Adaptado de Dennis *et. all.*, 2011, p. 494.

Herança: classe abstrata e classe concreta

- A maioria das classes em toda hierarquia levará às instâncias, e qualquer classe que tenha instâncias é denominada uma classe concreta. [...] Algumas classes não produzem instâncias porque são usadas simplesmente como modelos para outras classes mais específicas (em especial localizadas em um nível alto de uma hierarquia). Elas são classes abstratas (DENNIS, et al. 2014, p. 494).

Herança: classe abstrata e classe concreta

[..] qualquer classe que tenha instâncias é denominada uma classe concreta. (DENNIS *et. all.,* 2014, p. 494):



Polimorfismo

Polimorfismo é o princípio em que classes derivadas de uma mesma superclasse podem invocar operações que têm a mesma assinatura, mas comportamentos diferentes em cada subclasse, produzindo resultados diferentes, dependendo de como cada objeto implementa a operação (LIMA, 2011, p. 24).

Lima (2011, p. 24)

Polimorfismo

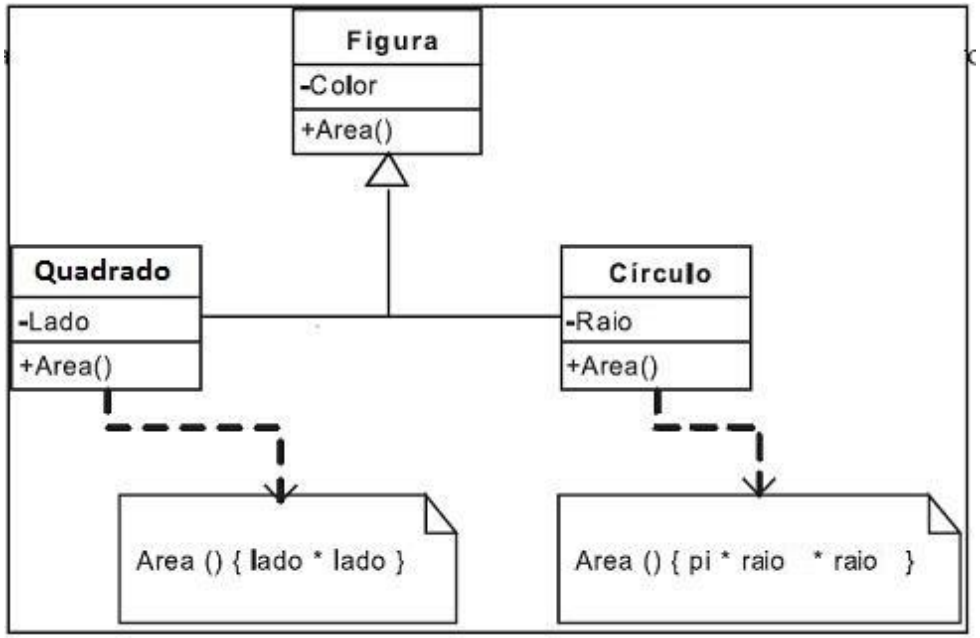


Figura 6: Polimorfismo
Fonte: VINÍCIUS, Thiago. Polimorfismo, classes abstratas e interfaces: Fundamentos da POO em Java. Disponível em: <http://www.devmedia.com.br/polimorfismo-classes-abstratas-e-interfaces-fundamentos-da-poo-em-java/26387>. Acesso em: 15 maio 2016.

Encapsulamento

"[...] é uma técnica que consiste em separar aspectos externos dos internos da implementação de um objeto, isto é, determinados detalhes ficam ocultos aos demais objetos e dizem respeito apenas ao próprio objeto" (LIMA, 2011, p. 2).

- As classes são do tipo privado (private) e serão acessadas apenas através de métodos setters e getters que alteram um valor e respectivamente, retornam um valor:

Listagem 2: Encapsulamento da classe Funcionario.

```
public class Funcionario {
	private double salario;
	private String nome;

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public void setSalario(double salario) {
		this.salario = salario;
	}

	public double getSalario() {
		return salario;
	}
}
```

Listagem 3: Classe Testadora dos métodos getters e setters.

```
public class TestaFuncionario {

	public static void main(String[] args) {
		Funcionario funcionario = new Funcionario();
		funcionario.setNome("Thiago");
		funcionario.setSalario(2500);

		System.out.println(funcionario.getNome());
		System.out.println(funcionario.getSalario());

	}
}
```

Figura 7: Encapsulamento
Fonte: VINÍCIUS, Thiago. Abstração, encapsulamento, e herança: pilares da POO em Java. Disponível em: <http://www.devmedia.com.br/abstracao-encapsulamento-e-heranca-pilares-da-poo-em-java/26366>. Acesso em: 16 maio 2016.

Obrigada!

Bons estudos.

Referências bibliográficas

SOMMERVILLE, Ian. Engenharia de Software. 9ª ed. São Paulo: Pearson Prentice Hall, 2011.

PIMENTEL, Manoel. DevMedia. Extreme Programming-conceitos e práticas. Disponível em: http://www.devmedia.com.br/extremeprogrammingconceitosepraticas/**1498** . Acesso em: 02/05/2016.

BRQ, Metodologias ágeis. Disponível em: http://www.brq.com/metodologias-ageis. Acesso em: 02/05/2016.