



**UNIVERSIDADE PAULISTA**

Professor: Me. Pablo I. Gandulfo

Data: 01/08/2020

Versão: 1.1

# Aplicações de Linguagem de Programação Orientada a Objetos

---

MÓDULO 6 e 7



**UNIVERSIDADE PAULISTA**

Professor: Me. Pablo I. Gandulfo

Data: 01/08/2020

Versão: 1.1

## Manipulação de dados com linguagem SQL

---

- Passos Envolvidos
- Comandos SQL:
  - Statement
  - PreparedStatement
  - CallableStatement
- Exemplo Completo de um CRUD
  - Classe Banco de Dados
- Metadados e DDL
- Exercícios

## DDL - Data Definition Language

---

- Os comandos SQL de Banco de Dados estão subdivididos entre DDL - Data Definition Language, DML - Data Manipulation Language, DQL - Data Query Language, DCL - Data Control Language e DTL - Data Transaction Language
- É possível executar qualquer um deles, em especial comandos de DDL, através do método:
  - Interface Statement
    - `boolean execute(stringSql)`

## Exemplo de Alteração de Metadados – Método main()

```
Class.forName("com.mysql.cj.jdbc.Driver");
Connection conn = DriverManager.getConnection( "jdbc:mysql://localhost:3306/test?useSSL=false" +
"&serverTimezone=UTC&allowPublicKeyRetrieval=true", "root", "admin");
Statement stmt = conn.createStatement();

System.out.println("Criando a tabela 'TESTE' ...");
stmt.execute("CREATE TABLE TESTE (id INTEGER NOT NULL)");

try { Thread.sleep(5000); } catch (InterruptedException e) { }

System.out.println("Destruindo a tabela 'TESTE' ...");
stmt.execute("DROP TABLE TESTE");

stmt.close();
conn.close();
```

### Saída

Criando a tabela 'TESTE' ...  
Destruindo a tabela 'TESTE' ...

## Metadados e DDL

---

- São informações relacionadas com os dados, como formatação ou tamanho
- Também podem ser acessadas através da API JDBC:
  - Metadados do Banco de Dados:
    - Método [Connection].getMetaData() => retorna um objeto DatabaseMetaData
  - Metadados dos conjuntos de resultados (ResultSet's)
    - Método [ResultSet].getMetaData() => retorna um objeto ResultSetMetaData
- Os metadados podem ser manipulados no Banco de Dados através de comandos SQL do tipo CREATE, ALTER ou DROP

## Exemplo de Leitura de Metadados - Método main()

```
Class.forName("com.mysql.cj.jdbc.Driver");
Connection conn = DriverManager.getConnection( "jdbc:mysql://localhost:3306/test?useSSL=false" +
"&serverTimezone=UTC&allowPublicKeyRetrieval=true", "root", "admin");
System.out.println("Banco de Dados: " + conn.getMetaData().getDatabaseProductName() + " - " +
conn.getMetaData().getDatabaseProductVersion());
System.out.println("Driver JDBC: " + conn.getMetaData().getDriverName() + " - " +
conn.getMetaData().getDriverVersion());
PreparedStatement stmt = conn.prepareStatement("SELECT * FROM funcionario");
ResultSet rs = stmt.executeQuery();
for (int i = 1; i <= rs.getMetaData().getColumnCount(); i++) {
    System.out.println("ResultSet - Coluna " + i + ": " + rs.getMetaData().getTableName(i) +
"." + rs.getMetaData().getColumnName(i));
}
rs.close();
stmt.close();
conn.close();
```

### Saída

Banco de Dados: MySQL - 8.0.15  
Driver JDBC: MySQL Connector/J - mysql-connector-java-8.0.21  
(Revision: 33f65445a1bcc544eb0120491926484da168f199)  
ResultSet - Coluna 1: funcionario.matricula  
ResultSet - Coluna 2: funcionario.nome  
ResultSet - Coluna 3: funcionario.cargo



UNIVERSIDADE PAULISTA

Professor: Me. Pablo I. Gandulfo

Data: 01/08/2020

Versão: 1.1

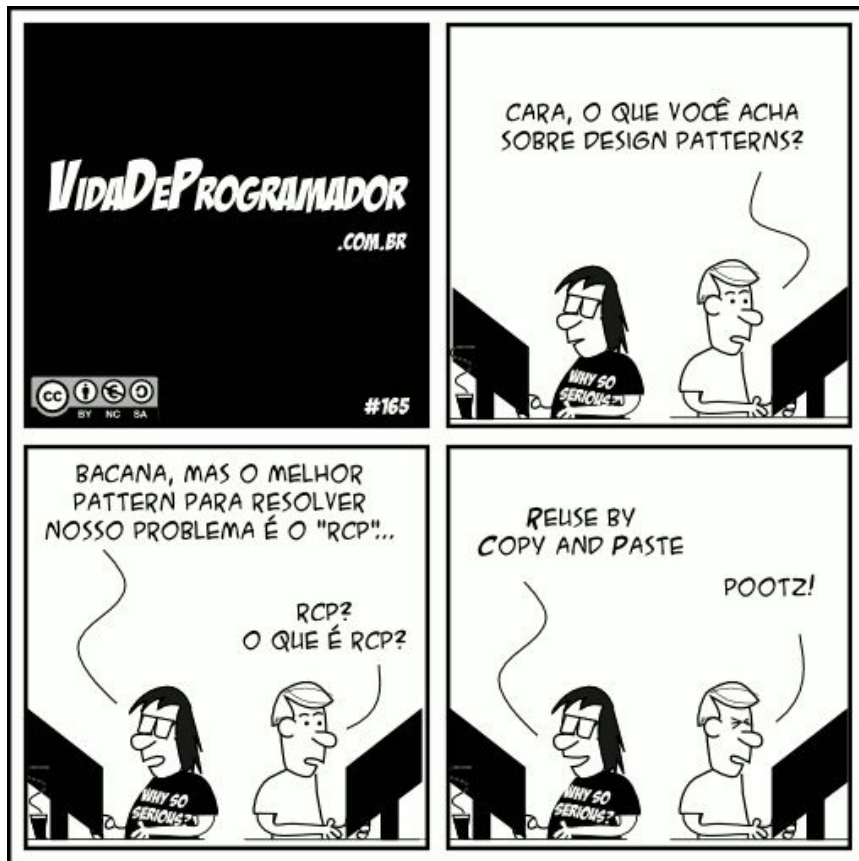
## Design Patterns – DAO (Data Access Object)

---

- Conceitos de Projeto (*Design*)
  - Padrões de Projeto (*Design Patterns*)
  - Arcabouços Conceituais (*Frameworks*)
  - Comparação e Diferenças
- Padrão de Projeto DAO - *Data Access Object*
- Exercícios

## Conceitos de Projeto

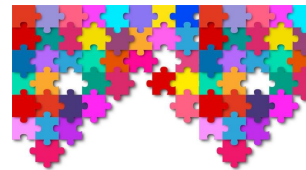
- Padrões de Projeto  
(*Design Patterns*)
- Arcabouços  
Conceituais  
(*Frameworks*)






## Conceitos de Projeto: Padrões de Projeto (*Design Patterns*)

---



- Padrões de projeto (*design patterns*) surgiram com a motivação de ajudar a solucionar problemas que ocorrem frequentemente
- **O que são?** A yellow lightbulb icon with a simple filament, symbolizing an idea or insight.
  - Uma combinação de:
    - Enunciado de um problema conhecido e recorrente de desenvolvimento em TI
    - Uma solução bem elaborada e inteligente, validada por várias pessoas
  - Possivelmente vem acompanhado de modelos da UML e código fonte, como referência

## Conceitos de Projeto: Padrões de Projeto (*Design Patterns*)

---

- **Para que servem?**

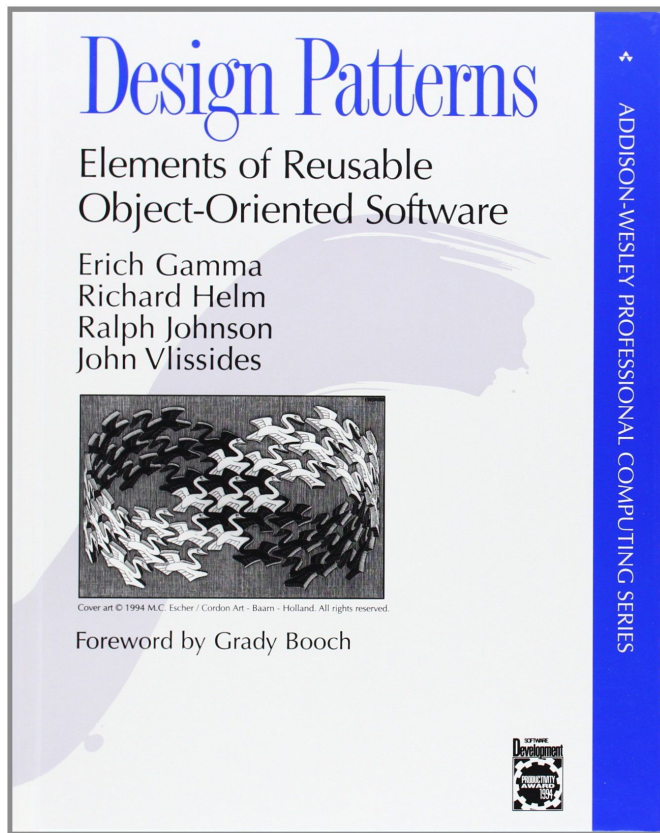
- Ao invés da equipe do projeto investir tempo tentando desenvolver uma solução para um problema relativamente complexo que já é amplamente conhecido no mercado, ela pode ganhar muito tempo adotando uma solução consolidada;
- Um grave problema no desenvolvimento de soluções “caseiras” (dentro do âmbito do projeto ou empresa) são os remendos feitos para permitir que essa solução possa ser utilizada em novos contextos não previstos. A adoção de uma solução de mercado diminui consideravelmente esse tipo de intervenção improvisada.

## Conceitos de Projeto: Padrões de Projeto (*Design Patterns*)

---

- Cada padrão de projeto descreve um problema que ocorre várias vezes ao nosso redor, e a solução para o problema de uma maneira que você pode usá-la diversas vezes (solução alto nível)
- O seu valor está no fato que são soluções utilizadas e testadas, indicando uma maior confiança em sua eficácia
- O conjunto dos mais conhecidos padrões de projeto estão catalogados no livro “Design Patterns: Elements of Reusable Object-Oriented Software” (também conhecido como “Design Patterns Bible”), escrito por Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides, os chamados “Gang of Four” (ou GoF)

## Livro “Design Patterns: Elements of Reusable Object-Oriented Software”



Apresentam 23 *design patterns* organizados em três grupos:

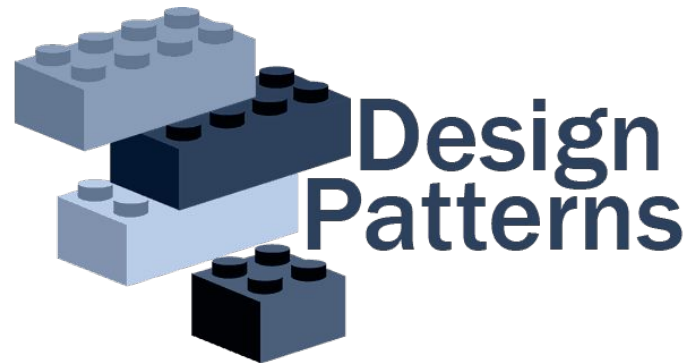
- Padrões de Criação (*Creational Patterns*): tratam da forma como os objetos são construídos
  - Exemplos: Factory, Prototype e Singleton
- Padrões Estruturais (*Structural Patterns*): tratam das relações entre objetos e como eles interagem entre si para formarem grandes objetos complexos
  - Exemplos: Facade, Proxy, Composite e Decorator
- Padrões Comportamentais (*Behavioral Patterns*): tratam da comunicação entre os objetos, especialmente em termos de responsabilidades e algoritmos
  - Exemplos: Command, Iterator e Strategy

## Conceitos de Projeto: Padrões de Projeto (*Design Patterns*) - Exemplo

---

- **Singleton**

- É um *design pattern* do tipo criacional que garante que uma classe seja instanciada apenas uma vez durante a execução do sistema
- Qual é a vantagem disso?
  - A razão mais comum é para controlar o acesso a algum recurso compartilhado, por exemplo, uma base de dados ou um arquivo
  - Pode também ser utilizado como um ponto central para armazenar e consultar “variáveis globais”



## Conceitos de Projeto: Arcabouços Conceituais (*Frameworks*)

---

- *Framework* é uma solução para uma família de problemas semelhantes, usando um conjunto de classes e interfaces com o objetivo de decompor a família de problemas, além de descrever como os objetos dessas classes colaboram para cumprir suas responsabilidades
- O conjunto de classes deve ser flexível e extensível, para permitir a construção de várias aplicações com pouco esforço, especificando apenas as particularidades de cada aplicação

## Conceitos de Projeto: Diferenças entre Padrões de Projeto e Arcabouços Conceituais

---

- Padrões de Projeto são mais abstratos do que *Frameworks*
  - Um *framework* inclui código, um padrão de projeto não (só um exemplo do uso)
  - Devido à presença de código, um *framework* pode ser estudado a nível de código, executado, e reusado diretamente
- Padrões de Projeto são elementos arquiteturais menores do que *Frameworks*
  - Um *framework* típico contém vários padrões de projeto, mas o contrário nunca ocorre
  - Exemplo: padrões de projeto são frequentemente usados para documentar *frameworks*
- Padrões de Projeto são menos especializados do que *Frameworks*
  - *Frameworks* sempre têm um domínio de aplicação particular, enquanto que padrões de projeto não ditam uma arquitetura de aplicação particular

## Padrão de Projeto DAO – *Data Access Object*

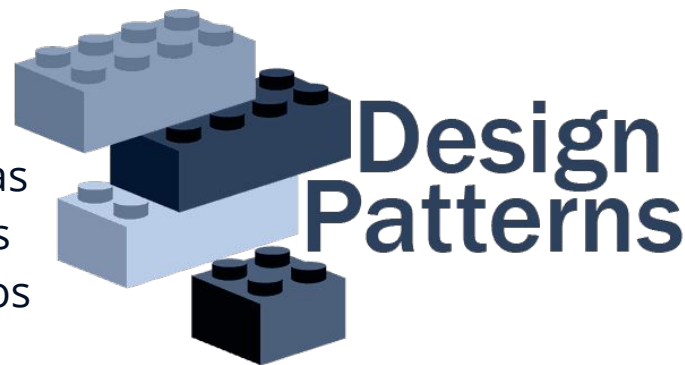
---

- **DAO**

- É um *design pattern* do tipo estrutural que fornece um padrão que se adapta a diferentes esquemas de armazenamento, sem afetar seus clientes ou componentes de negócios

- Qual é a vantagem disso?

Ao consultar ou armazenar dados numa fonte de dados, seja um banco de dados, arquivo, LDAP, web service REST ou outro meio, a forma de acesso depende diretamente do tipo de fonte de dados utilizado. Por exemplo, a forma de ler ou gravar dados num arquivo é diferente de uma tabela de um BD relacional Oracle, que também é diferente de um BD não relacional MongoDB, etc..





## Padrão de Projeto DAO - *Data Access Object*

---

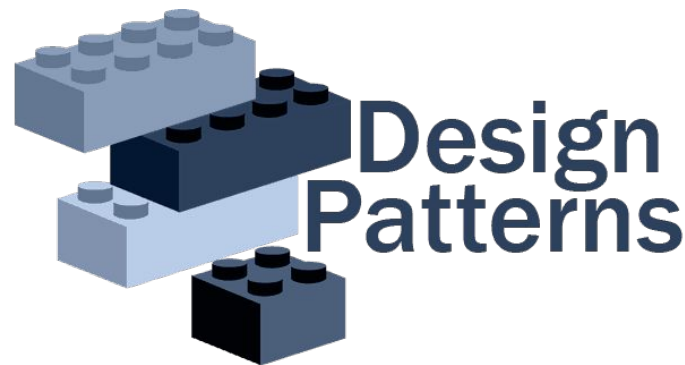
- **DAO**

- Qual é a vantagem disso? (Cont.)

...

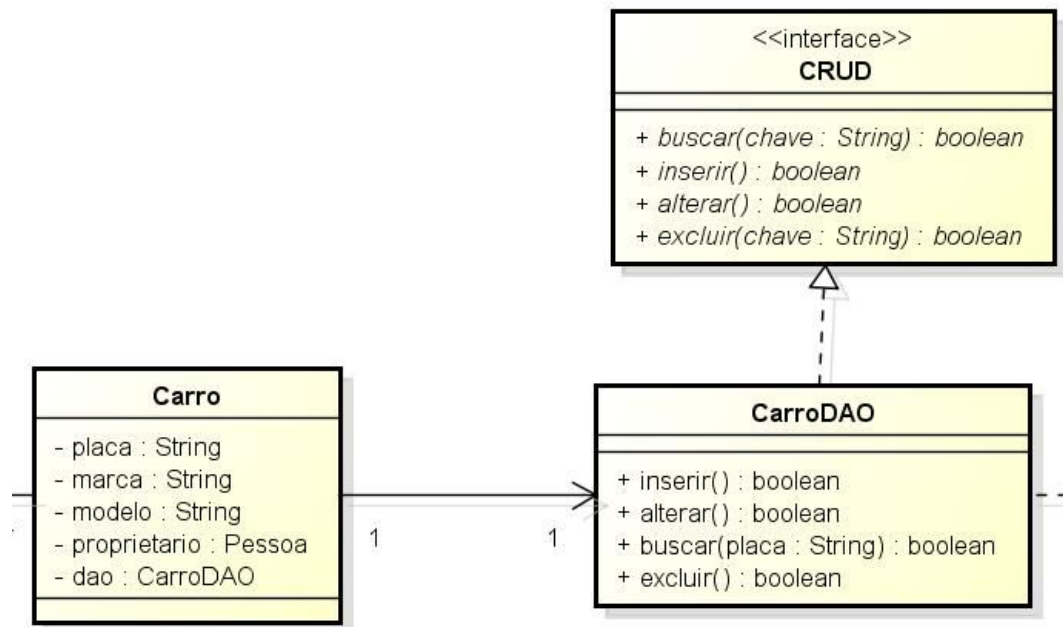
O Data Access Object (DAO) serve para abstrair e encapsular todo o acesso à fonte de dados através de uma interface simples e padronizada.

Caso a fonte de dados mude, o código interno de acesso provavelmente mudará, mas não o código que o utiliza.



## Padrão de Projeto DAO - *Data Access Object*

- Diagrama de Classes de Exemplo:



## Exercício 1

---

- Crie a interface genérica DAO e a implementação dessa interface que permita realizar as seguintes operações na tabela Funcionario ou Celular:
  - Object consultar(int id)
  - boolean incluir(Object o)
  - boolean alterar(Object o)
  - boolean excluir(int id)
- Desenvolva um código que acione esses métodos e ilustre o uso desse DAO