

# Domain Driven Design

## Eliane Marion

FIAP

# AGENDA DE TRABALHO



Conexão com  
Banco de Dados



Design Patterns



01

DESIGN PATTERNS

# DESIGN PATTERNS

- São arquiteturas testadas para construir softwares orientados a objetos flexíveis e sustentáveis.
- Ajudam a reduzir substancialmente a complexidade do processo de design.

- Os Design Patterns são uma coleção de padrões de projeto de software que contém soluções para problemas conhecidos e recorrentes no desenvolvimento de software descrevendo uma solução comprovada para um problema de projeto recorrente.

# DESIGN PATTERNS

São soluções típicas para problemas comuns em projeto de software. Cada padrão é como uma planta de construção que você pode customizar para resolver um problema de projeto particular em seu código.

## **Benefícios dos padrões**

Padrões são um conjunto de ferramentas para soluções de problemas comuns em design de software. Eles definem uma linguagem comum que ajuda sua equipe a se comunicar mais eficientemente.

## **Classificação**

Padrões de projeto diferem por sua complexidade, nível de detalhamento e grau de aplicabilidade. Além disso, eles podem ser categorizados por seu propósito e divididos em três grupos

# PADRÃO SINGLETON

- Tem como definição garantir que uma classe tenha apenas uma instância de si mesma e que forneça um ponto global de acesso a ela.
- Garantir que apenas um objeto exista, independente do número de requisições que receber para criá-lo.

## Aplicações

- Um único banco de dados
- Um único acesso ao arquivo de log
- Um único objeto que representa um vídeo
- Uma única fachada (Façade pattern)
- Objetivo: garantir que uma classe só tenha uma instância

# PADRÃO MVC

Padrão de desenvolvimento de software baseado em 03 camadas:

- Modelo (Model)
- Visão (View)
- Controladora (Controller)

## **Objetivo:**

- Separar a lógica da aplicação da apresentação das informações ao usuário

Pode-se alterar as telas ou componentes visuais do sistema sem modificar as classes responsáveis pela a lógica da aplicação (modelo e controladoras), e vice-versa.

Diminui-se o tempo de manutenção de funcionalidades devido a alta coesão (classes com responsabilidades e objetivos bem definidos) e ao baixo acoplamento.

# PADRÃO MVC

- **Modelo**: Concentra as classes de domínio (entidades) da aplicação, além das classes de negócio e de acesso a dados;
- **Visão**: Responsável pelo layout da aplicação (telas em HTML, por exemplo) e seus componentes visuais;

- **Controladora**: Direciona o fluxo de dados entre as camadas de visão e de modelo da aplicação.



# PADRÃO DAO (Data Access Object)

## Objetivo:

- Encapsular o acesso a dados em uma classe separada da aplicação

O padrão DAO em conjunto com padrões de projeto que atuam como fábricas de objetos (Factory e Abstract Factory) possibilita a implementação de acesso para diferentes mecanismos de persistência.

## Características

- Centralização do código de acesso/manipulação de dados da aplicação.
- Separação da lógica de negócio da persistência.
- Tornar transparente o acesso aos dados nas aplicações.
- Possibilitar acesso a diferentes fontes de dados de forma transparente para o usuário.

# PADRÃO DAO (Data Access Object)

O padrão de projeto DAO surgiu da necessidade de separar a lógica de negócios da lógica de persistência de dados. As classes DAO são responsáveis por trocar informações com o SGBD e fornecer operações CRUD e de pesquisas.

- É possível compara o design pattern DAO com a camada Model do padrão MVC.
- Model x View x Controller



# 02

JDBC

# COMO CONECTAR O JAVA AO BANCO DE DADOS?

Qualquer linguagem pode se conectar a qualquer banco?

O que é necessário para fazer a conexão?

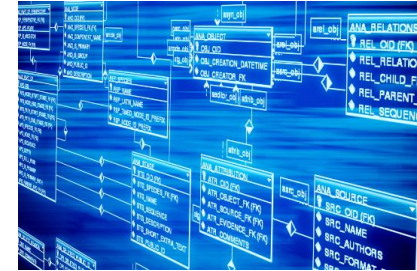
Como inserir, alterar, consultar dados do banco de dados?

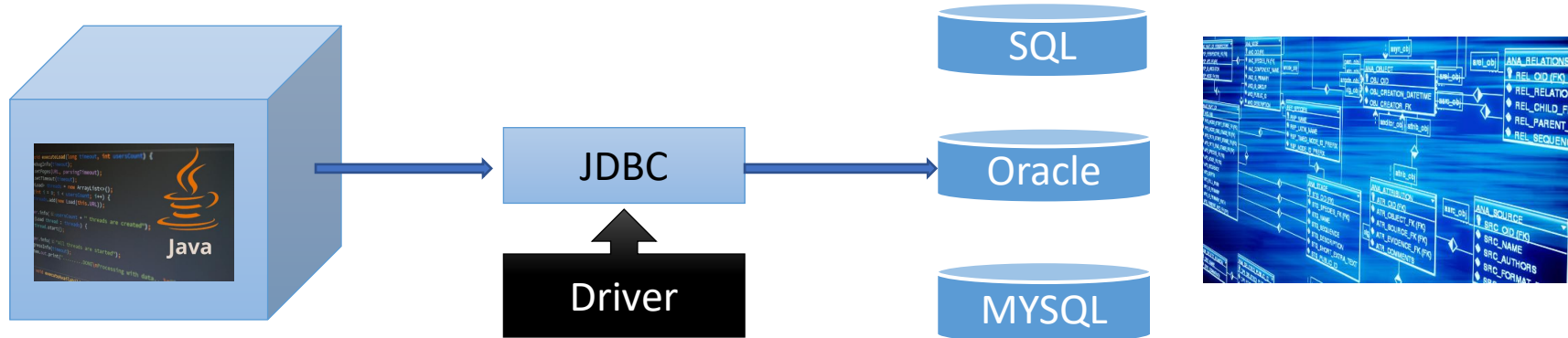
É seguro fazer a aplicação e o banco se conectarem?





# MYSQL





# JDBC - Java Database Connectivity

- Criado em 1997
- Especificação para acesso a banco de dados relacionais em Java
- Usa o driver do banco de dados (implementações do JDBC) para estabelecer a conexão.

- Abstração do protocolo de comunicação com o banco de dados
- Impacto mínimo na aplicação ao trocar de banco de dados
- Pattern DAO ajuda a isolar o código da API do JDBC

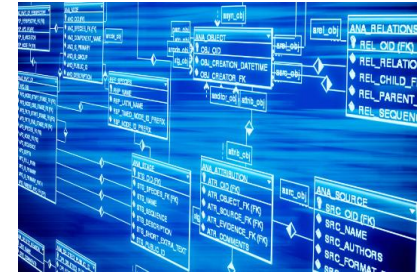
# JDBC - Java Database Connectivity



## DESVANTAGENS

- Código muito verboso.
- Alto acoplamento com o banco de dados
- Atualmente utilizamos outros frameworks para fazer a conexão (2º ano)







# CONVERSANDO COM O BANCO DE DADOS



Java Database Connectivity - JDBC

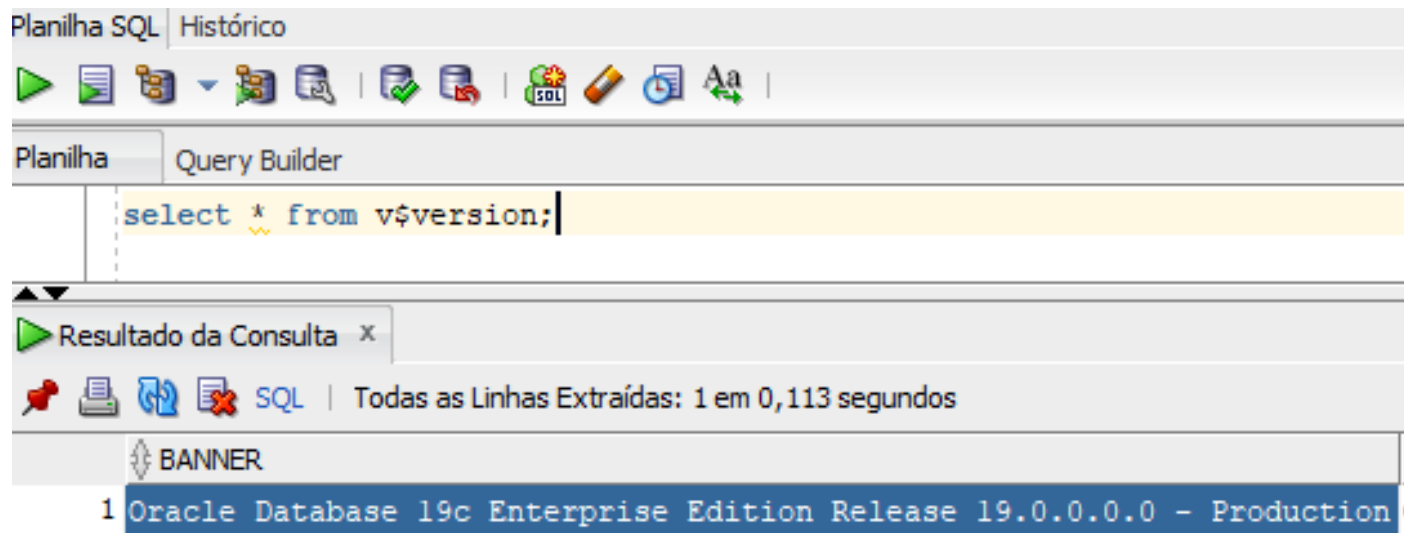


API que contém recursos para conectar o Java com uma série de Banco de Dados baseado em SQL.

Para verificar a versão do Oracle Database faça um `Select * From v$version;`



# VERIFICANDO A VERSÃO DO ORACLE



Para verificar a versão do Oracle Database faça um `Select * From v$version;`





# Fazendo download da versão do oracle

[JDBC and UCP Downloads page \(oracle.com\)](https://www.oracle.com/technetwork/database/enterprise/downloads/jdbc-ucp-downloads-page.html)

## Oracle Database 21c (21.6.0.0.1) JDBC Driver & UCP Downloads

Supports Oracle Database versions - 21c, 19c, 18c, and 12.2

Patched on top of 21.6.0.0. Contains OAuth2 support in JDBC Driver specifically for Microsoft Azure Active Directory (AD) token Authentication.

Name	Download	JDK Supported	Description
Oracle JDBC driver	 ojdbc11.jar	Implements JDBC 4.3 spec and certified with JDK11 and JDK17	Oracle JDBC driver except classes for NLS support in Oracle Object and Collection types. (5,181,226 bytes) - (SHA1: 699ca7fb1367bb87dae03c87c7be60bc2d27636e)
Oracle JDBC driver	 ojdbc8.jar	Implements JDBC 4.2 spec and certified with JDK8 and JDK11	Oracle JDBC driver except classes for NLS support in Oracle Object and Collection types. (5,088,810 bytes) - (SHA1: 1888acd4bcb1457f1a10a3333a3d104c23283cd9)



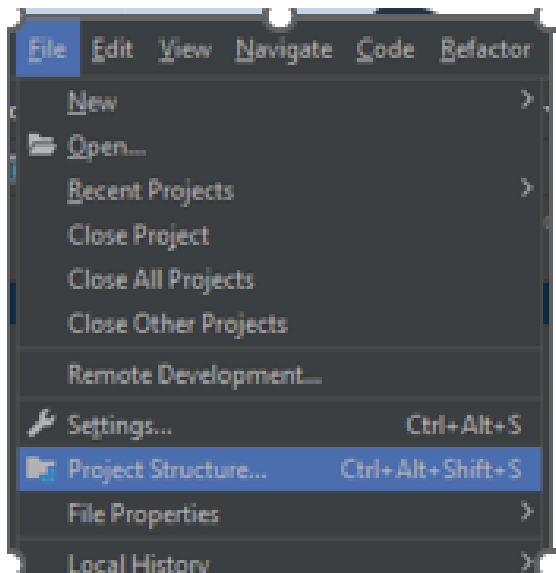
# 03

Fábrica de Conexões



# Adicionando o driver no IntelliJ

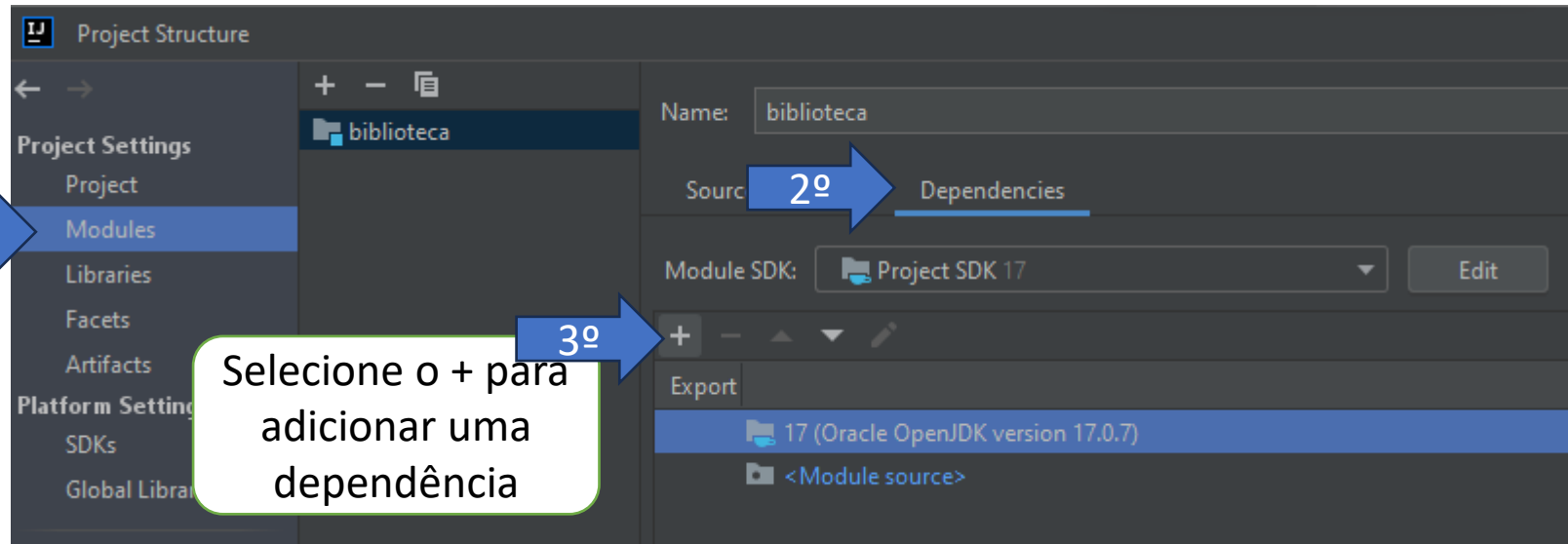
A primeira etapa é incluir o arquivo que fizemos download (jar) dentro do nosso projeto. Selecione a opção File -> Project Structure.





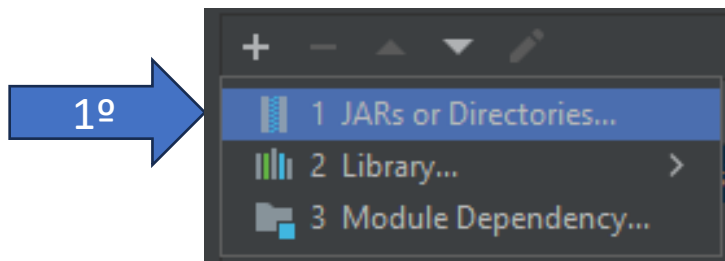
# Adicionando o driver no IntelliJ

Clique sobre Modules, em seguida selecione a opção Dependencies



# Adicionando o driver no IntelliJ

Selecione JARs or Directories, em seguida selecione o arquivo ojdbc11.jar que foi copiado para a pasta lib.

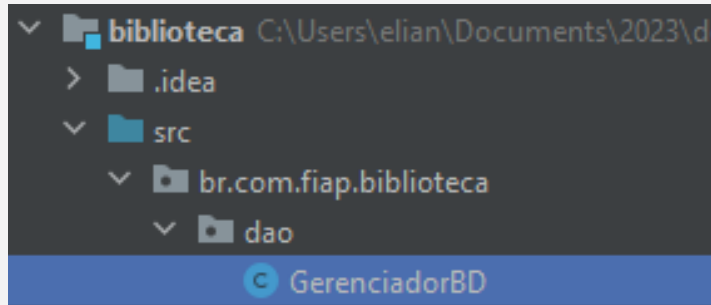




# Conexão de Banco de Dados

Dentro do pacote `br.com.fiap.biblioteca.dao` criar uma classe chamada **GerenciadorBD**

{ }



```
package br.com.fiap.biblioteca.dao;

import ...

no usages

public class GerenciadorBD {

}
```



# A classe GerenciadorBD

- Será utilizada toda vez que quisermos estabelecer uma conexão com o banco de dados.
- Usa o driver para estabelecer a conexão. Retorna essa conexão ativa.

GerenciadorBD.java X

```
1 package DAO;
2
3 import java.sql.Connection;
4
5 public class GerenciadorBD {
6     public static Connection obterConexao() {
7         Connection conexao = null;
8         return conexao;
9     }
10 }
11
12
```

- Classe do pacote java.sql;
- Contém métodos para gerenciar a conexão com o banco

# A classe GerenciadorBD

```
public class GerenciadorBD {  
    public static Connection obterConexao() {  
        Connection conexao = null;  
        try {  
            conexao = DriverManager.getConnection("jdbc:oracle:thin:@oracle.fiap.com.br:1521:orcl", "USUARIO", "SENHA");  
            //banco utilizado: conexão:porta, usuário, senha  
        } catch(SQLException erro) {  
            erro.printStackTrace();  
        }  
        return conexao;  
    }  
}
```

A classe que nos permitirá usar o driver para estabelecer a conexão é a classe `DriverManager`, também do pacote `java.sql`. O método que estabelece a conexão é o `getConnection`, que pede uma string de conexão, seguida do usuário e da senha do banco.

# Testando a conexão

Com a classe GerenciadorBD criada e retornando uma Connection podemos agora criar um teste para verificar se nossa conexão está sendo estabelecida com o banco de dados.

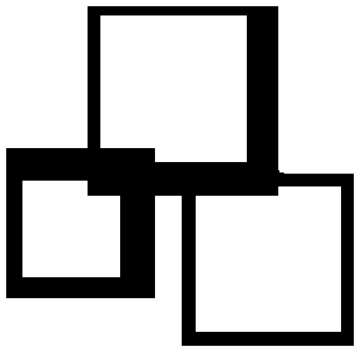
O primeiro passo é criar um pacote chamado teste e nele uma classe chamada TesteConexao com o método main.

```
public class TesteConexao {  
  
    public static void main(String[] args) {  
        if(GerenciadorBD.obterConexao() == null) {  
            System.out.println("Erro ao estabelecer conexão");  
        }  
        else {  
            System.out.println("Conexão estabelecida com sucesso!");  
        }  
    }  
}
```

# Testando a conexão

{ }

Ao executar esta classe o método `obterConexao` é chamado, ele através do drive tenta se conectar ao banco de dados configurado usando o usuário e senha fornecidos, se a conexão retornar null significa que houve algum problema e não foi possível estabelecer a comunicação com o banco de dados, portanto exibirá a mensagem: "Erro ao estabelecer conexão", caso contrário a conexão foi estabelecida e exibirá a mensagem "Conexão estabelecida com sucesso!"



**OBRIGADO**

*To be continued...*

