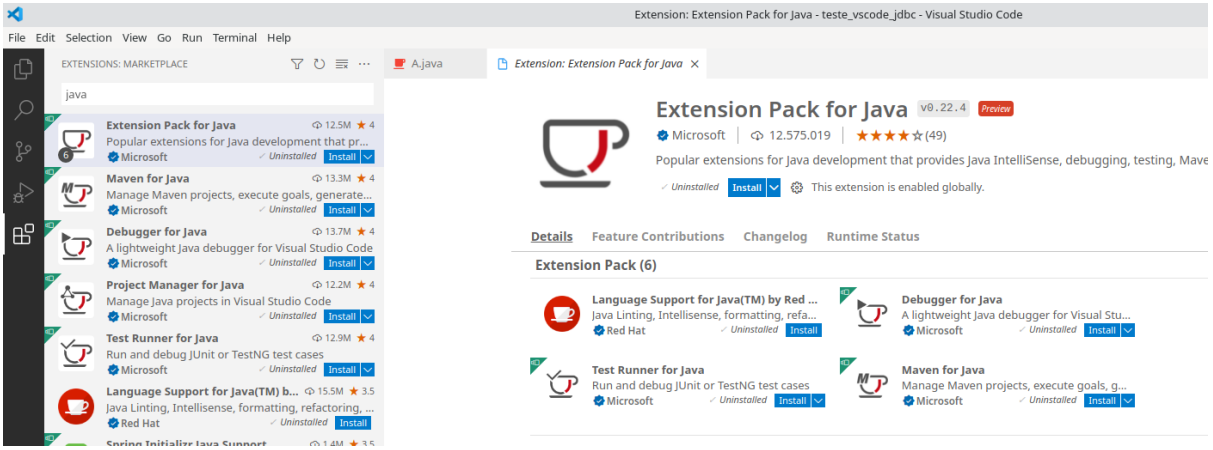
1 Introdução

Neste material, veremos como criar um projeto Java que se conecta a uma instância do MySQL utilizando o VS Code.

2 Desenvolvimento

2.1 (Extensão) No VS Code, certifique-se de que você possui a extensão **Extension Pack for Java**. Para fazer a sua instalação, abra o Marketplace do VS Code. Como mostra a Figura 2.1.1, busque por Java. Clique sobre o nome da extensão e então clique em **Install**.

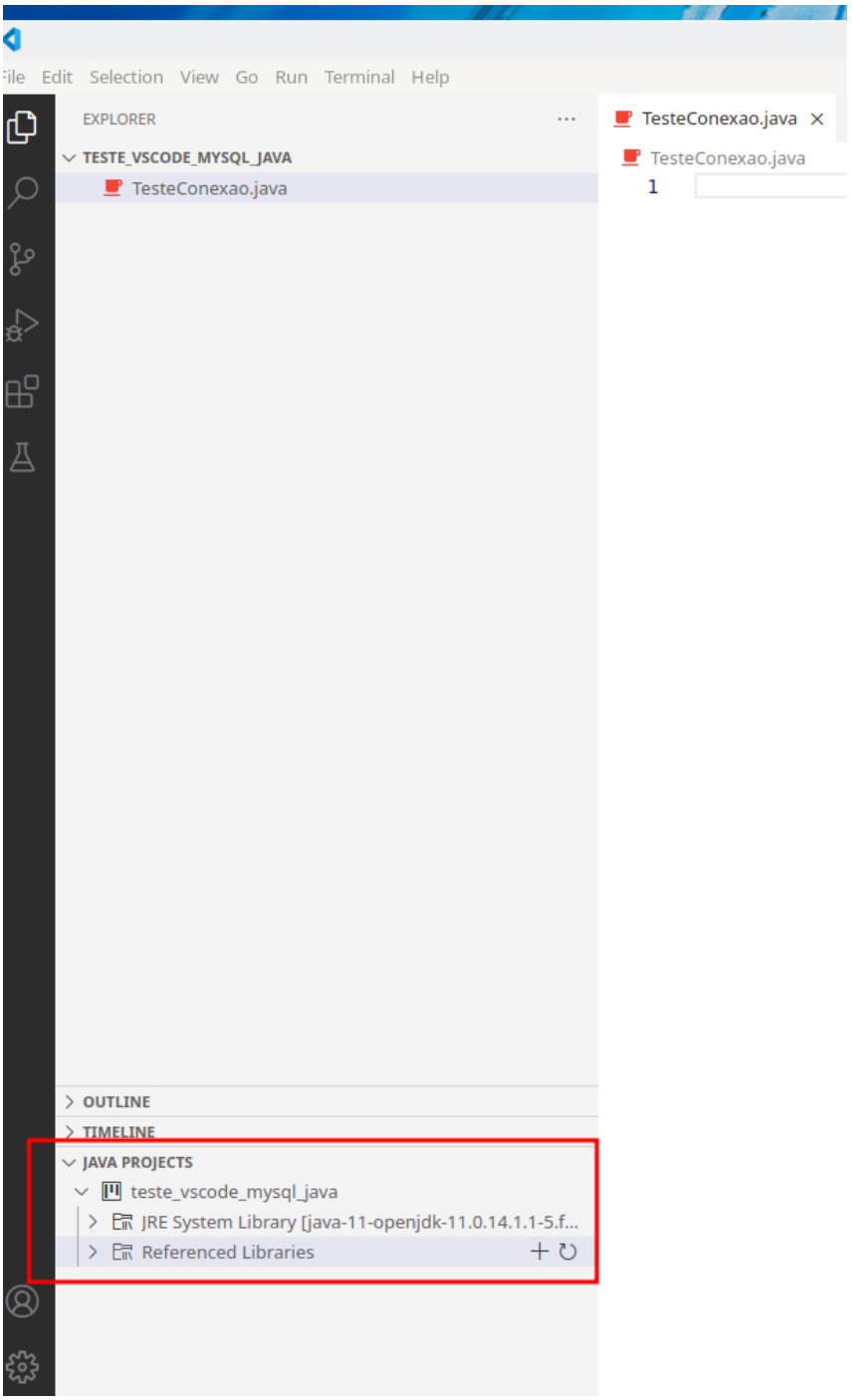
Figura 2.1.1



2.2 (Vincule o VS Code a uma pasta) Crie uma pasta apropriada para este projeto, inicialmente vazia. A seguir, no VS Code, clique **File >> Open Folder** e abra essa pasta.

2.3 (Arquivo java para teste) NO VS Code, crie um arquivo chamado **TesteConexao.java**. Logo depois de criar o arquivo, certifique-se de que a aba **Java Projects** apareceu. Ela costuma aparecer no canto inferior esquerdo do VS Code. Expanda os itens existentes até encontrar o item chamado **"Referenced Libraries"**. Não faça nada com ele por enquanto. Veja a Figura 2.3.2.

Figura 2.3.2

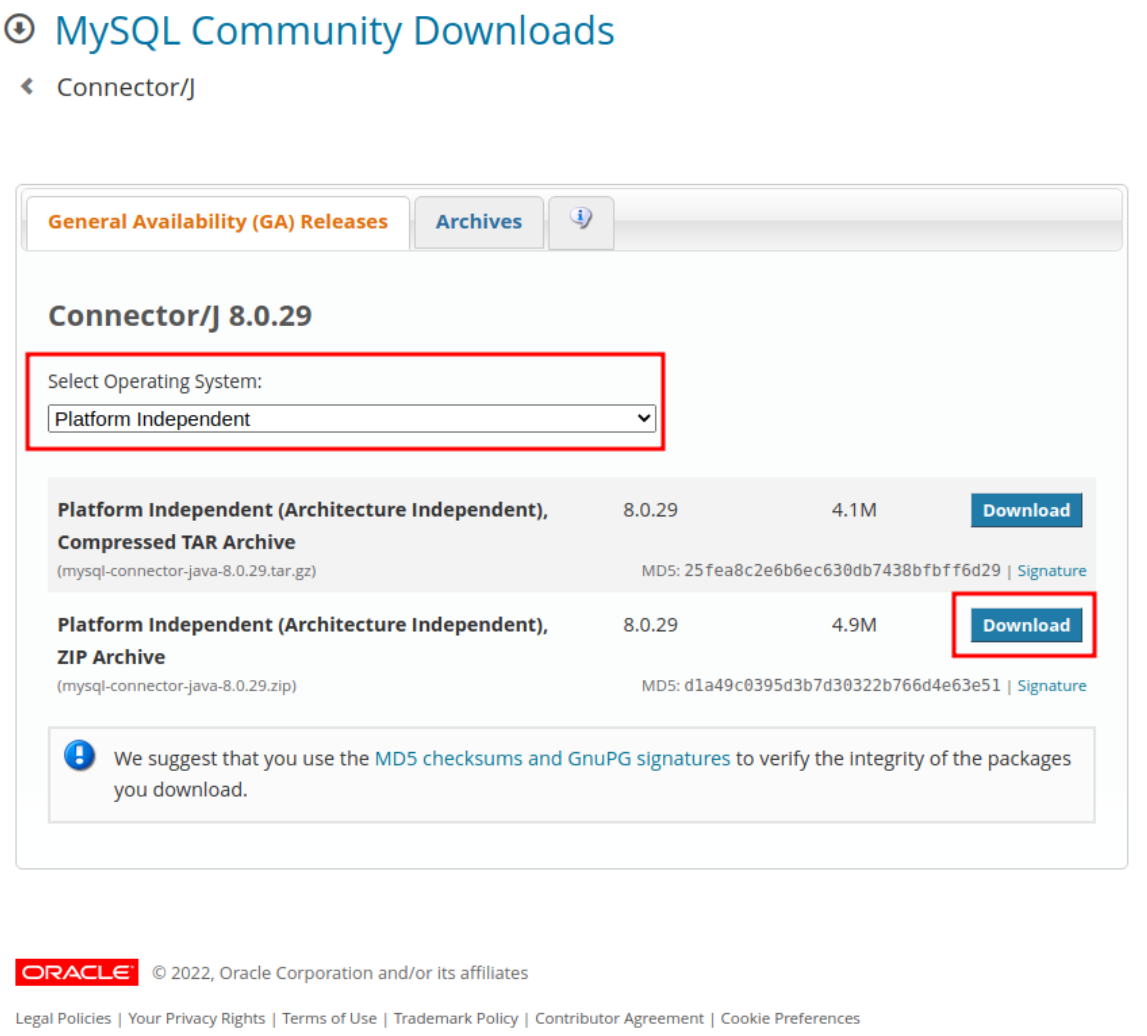


2.4(Download do Driver JDBC) Estamos diante de um cenário em que duas aplicações desejam se comunicar: nossa aplicação Java e o servidor MySQL. Isso será feito utilizando-se um protocolo próprio, que é implementado pelo próprio fabricante do MySQL. Essa implementação nos é entregue como uma coleção de classes empacotadas num arquivo de extensão **jar**. Ela leva o nome de “driver”. Visite o Link 2.4.1 para fazer o download.

Link 2.4.1
https://dev.mysql.com/downloads/connector/j/

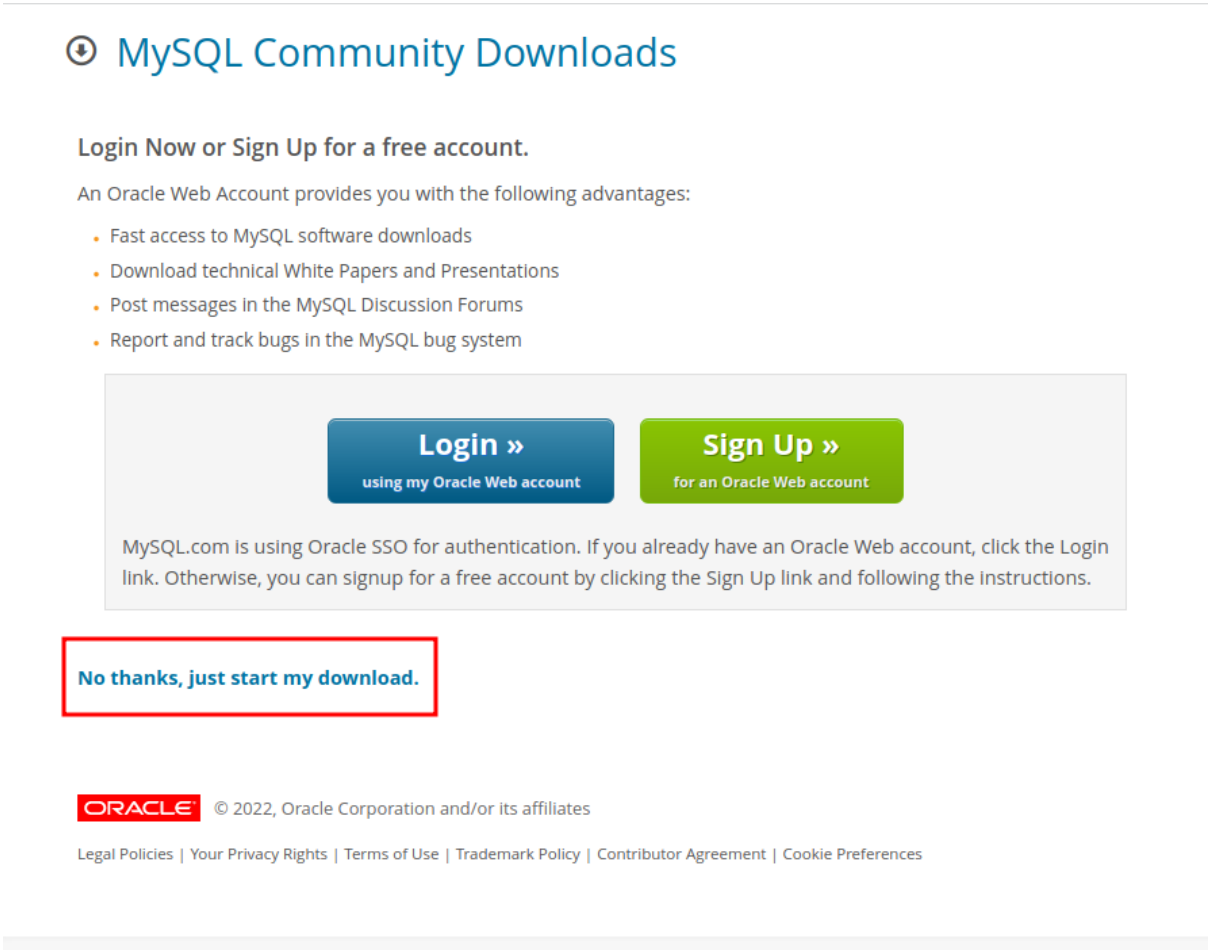
Escolha a opção **Platform Independent** e faça o download do arquivo em formato **zip**, como na Figura 2.4.1.

Figura 2.4.1



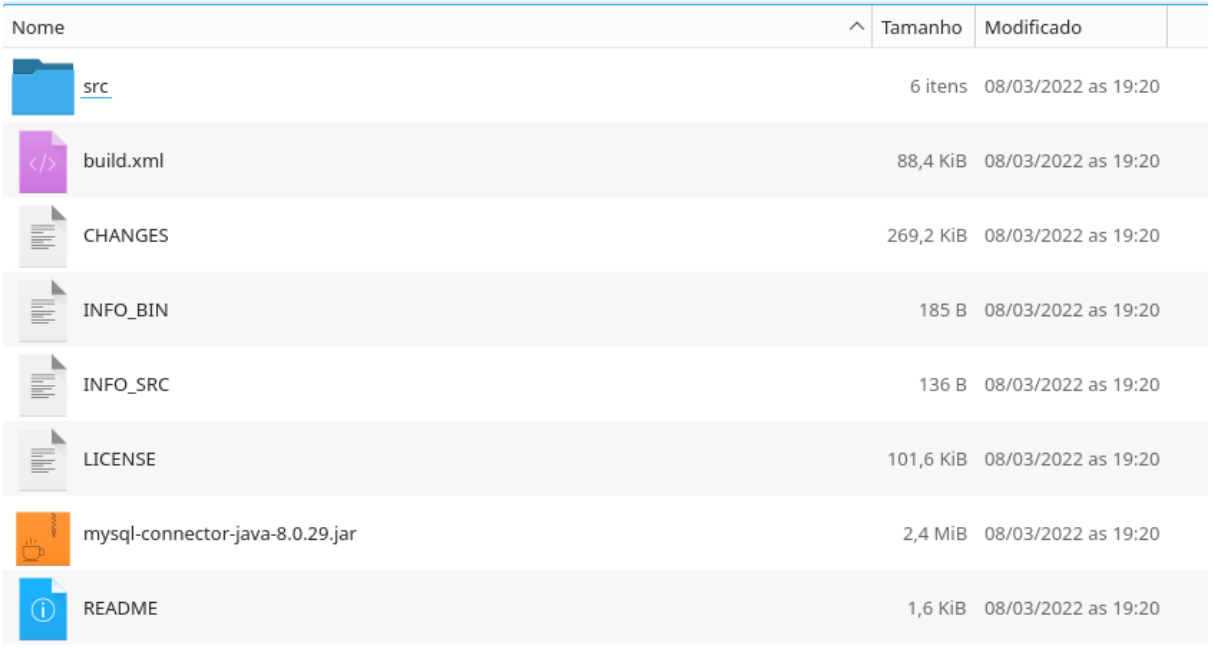
Na tela seguinte, exibida pela Figura 2.4.2, apenas clique em **No thanks, just start my download**.

Figura 2.4.2



Quando o download terminar, você precisará descompactar o arquivo. Uma vez que tenha feito isso, você se deparará com o conteúdo exibido pela Figura 2.4.3.

Figura 2.4.3

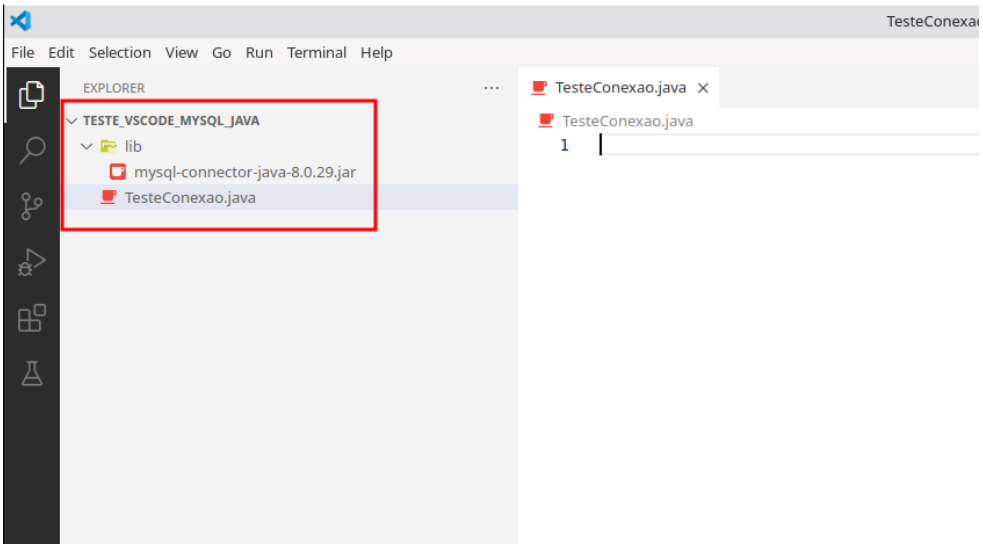


2.5 (Pasta lib para abrigar o arquivo .jar e adição de "biblioteca referenciada") O arquivo que desejamos é aquele de extensão **.jar**. Ele precisará ser adicionado ao **classpath** de nosso projeto. Há inúmeras formas de fazê-lo. Uma delas consiste nos seguintes passos

- Crie uma pasta chamada **lib** como subpasta daquela a que seu VS Code está vinculado. Você pode fazer isso a partir do próprio VS Code.
- Copie o arquivo **.jar** para dentro da pasta **lib**.

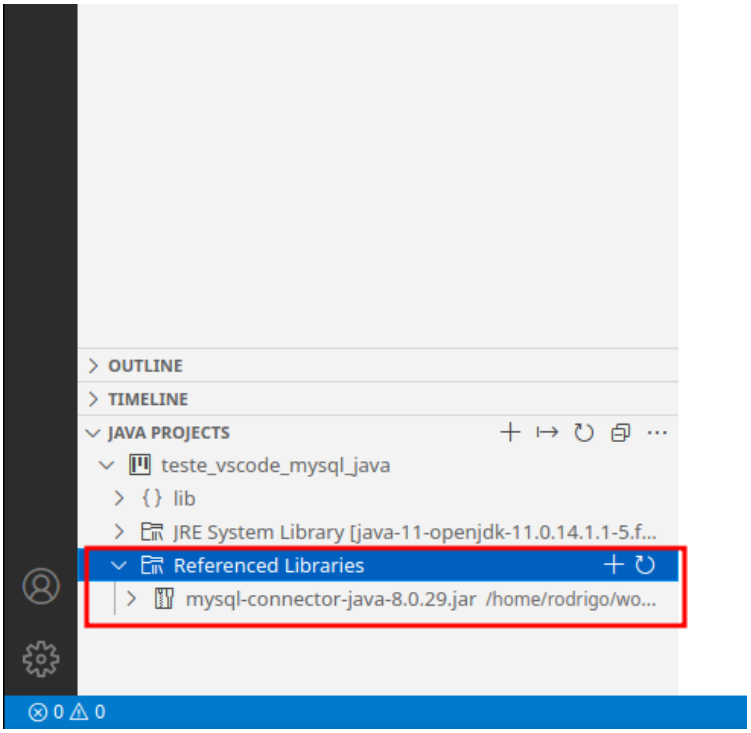
Por enquanto, o resultado esperado é aquele exibido pela Figura 2.5.1.

Figura 2.5.1



- No VS Code, no canto inferior esquerdo, clique no botão **+** que fica à frente de **Referenced Libraries**. Navegue até a pasta **lib** que você criou e adicione o arquivo **.jar**. Veja o resultado esperado na Figura 2.5.2
-

Figura 2.5.2



2.6 (Método main e exceções: coisas ruins podem acontecer) Estamos diante de um cenário em que dois programas de computador desejam se comunicar. Por um lado, temos o programa que escreveremos em Java. Por outro lado, o servidor MySQL. O programa em Java deseja enviar comandos SQL para o MySQL Server, na esperança de ele os receber, compilar, executar e devolver uma resposta indicando o resultado. Observe, entretanto, que há diferentes situações em que essa comunicação pode falhar.

- Esquecemos de adicionar o **driver** JDBC ao classpath de nossa aplicação. Ele é o responsável pela implementação do protocolo de comunicação entre as aplicações.
- Deixamos de instalar ou colocar o MySQL Server em execução.
- O MySQL Server está vinculado a uma porta diferente daquela que o programa em Java está tentando utilizar para enviar os comandos SQL.
- O programa em Java tenta usar um banco de dados que não existe no MySQL Server.
- O programa em Java envia um comando SQL que faz referência a uma tabela ou a uma coluna que não existe no banco de dados referenciado.

Todas essas situações são representadas como **exceções.** Quando temos um bloco de código que pode dar origem a uma exceção, desejamos ser capazes de especificar um outro cuja execução deve iniciar caso uma exceção seja detectada, da seguinte forma.

- “Tentamos” executar o bloco de código principal. Se nenhuma exceção acontecer, ele termina a sua execução e nada mais acontece.
- “Tentamos executar o bloco de código principal. Se uma exceção for detectada, o fluxo de execução desvia imediatamente para outro bloco de código que também especificamos, fazendo com que o fluxo alternativo de execução entre em funcionamento. Neste cenário, estamos **tratando a exceção** para que o programa não encerre a sua execução abruptamente.

A construção que desejamos utilizar para isso é a **try/catch**. Assim, defina o método main como mostra o Bloco de Código 2.6.1. Ele usa a API JDBC para tentar realizar uma conexão com o MySQL Server. o Bloco try é o principal. O catch é o alternativo cuja execução ocorre somente mediante uma exceção.

Bloco de Código 2.6.1

```
import java.sql.DriverManager;
import java.sql.Connection;
public class TesteConexao{
    public static void main(String[] args) {
        try{
            Connection conexao =
DriverManager.getConnection(
                //essa é a conhecida string de conexão
                "jdbc:mysql://localhost:3306/teste",
                "root",
                "1234"
            );
            if (conexao != null){
                System.out.println("Conexão estabelecida
com sucesso!");
            }
            else{
                System.out.println("Conexão não
estabelecida!");
            }
        }
        catch (Exception e){
```

```
            System.out.println("Exceção: " +
e.getMessage());
        }
    }
}
```

Observe que estamos fazendo referência a um banco de dados chamado **teste**. Como ele ainda não existe, o fluxo de execução desvia para o bloco catch e exibe uma mensagem parecida com aquela que a Figura 2.6.1 exibe.

Figura 2.6.1

2.7 (Criando um banco de dados no MySQL usando o MySQL Workbench) Abra o MySQL Workbench e crie um banco de dados com o comando exibido pelo Bloco de Código 2.7.1.

Bloco de Código 2.7.1

```
-- cria o banco
CREATE DATABASE teste;
-- explica ao MySQL Server que os comandos a seguir deverão ter impacto neste banco
USE teste;
```

2.8 (Criando uma tabela) A seguir, vamos criar uma tabela para testar operações CRUD (Create, Read, Update, Delete). Para tal, use o comando do Bloco de Código 2.8.1.

Bloco de Código 2.8.1

```
CREATE TABLE tb_pessoa(
        cod_pessoa INT PRIMARY KEY AUTO_INCREMENT,
   nome VARCHAR(200) NOT NULL,
   idade INT NOT NULL
);
```

2.9 (Inserindo pessoas com o MySQL Workbench) Use o comando do Bloco de Código 2.9.1 para inserir dados na tabela.

Bloco de Código 2.9.1

```
-- insere duas pessoas
INSERT INTO tb_pessoa (nome, idade) VALUES ('Pedro', 17), ('João', 22);
```

2.10 (Inserindo pessoas com a aplicação Java) De volta ao VS Code, ajuste seu método main para fazer a inserção dos dados de uma pessoa digitados pelo usuário. Veja o Bloco de Código 2.10.1.

Bloco de Código 2.10.1

```
import java.sql.DriverManager;
import java.sql.PreparedStatement;

import javax.swing.JOptionPane;

import java.sql.Connection;
public class TesteConexao{
    public static void main(String[] args) {
        try{
            Connection conexao = DriverManager.getConnection(
                //essa é a conhecida string de conexão
                "jdbc:mysql://localhost:3306/teste",
                "root",
                "1234"
            );
            if (conexao != null){
                System.out.println("Conexão estabelecida com sucesso!");
                String nome = JOptionPane.showInputDialog("Qual o nome?");
                int idade = Integer.parseInt(JOptionPane.showInputDialog("Qual a
idade?"));
                //o comando a ser executado é uma simples string
                //os símbolos ? reservam o lugar de valores a serem especificados
                String sql = "INSERT INTO tb_pessoa (nome, idade) VALUES (?, ?)";
                //uma estrutura que representa o comando é "preparada". Assim, ele
pode ser executado diversas vezes com dados diferentes
                PreparedStatement ps = conexao.prepareStatement(sql);
                //configuramos os valores a serem inseridos
                ps.setString(1, nome);
                ps.setInt(2, idade);
                //executamos o comando
                ps.execute();
                //fechamos a conexão para que o MySQL Server possa liberar recursos
                conexao.close();
                JOptionPane.showMessageDialog(null, "Dados inseridos com sucesso!");
            }
            else{
```

```
                System.out.println("Conexão não estabelecida!");
            }
        }
        catch (Exception e){
            System.out.println("Exceção: " + e.getMessage());
        }
    }
}
```