Interface ResultSet e seus métodos:
https://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html
Recuperar e modificar valores usando a interface ResultSet:
https://docs.oracle.com/javase/tutorial/jdbc/basics/retrieving.html
Interface PreparedStatement e seus métodos:
https://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html
executeUpdate() — método da interface Statement que executa uma atualização/inserção/eliminação na base de dados
executeQuery(String sql) — método da interface Statement que executa uma consulta na base de dados
Resulset() — tabela de dados que representa um conjunto de resultados da base de dados, geralmente gerado pela execução de uma instrução select.
Tipo de dados SQL:
https://docs.microsoft.com/pt-br/sql/t-sql/data-types/data-types-transact-sql
Consulta do material de apoio na classroom no tópico UFCD – 0817:
Resumo comandos SQL e tipo de dados MYSQL.

Utilizar algumas das principais operações sobre uma base de dados recorrendo à Interface Statement

Exercícios:

Nota: Deves executar cada uma das classe **OperacoesBD** quando terminadas as alíneas correspondentes, para verificares no interface do SGBD (phpmyadmin) se ocorreram essas operações na base de dados produto.

1- Cria, no MySQL, uma base de dados com o nome **Gestao_produtos**;

2- Cria no Eclipse um projeto com o nome **Gestao_produtos**;

3- Dentro do projeto anterior cria a classe **LigacaoBD** que permita a ligação à BD criada no exercício1.

4- No mesmo projeto, cria a classe **OperacoesBD1** que permita efetuar as operações seguintes:

   a. Criar a tabela **produtos** com os campos **codigo, nome, preco_s_iva, taxa_iva e preco_c_iva,** dados, respetivamente, do tipo inteiro (auto incremento e chave primária), cadeia de carateres, real, real e decimal (com tamanho(3,2)) ;

   b. Inserir 3 produtos na tabela produtos definindo a taxa de IVA, em dois produtos, com o valor 21% e no restante 6% e ainda o preço com IVA com 0 em todos os produtos.

5- No mesmo projeto, cria a classe **OperacoesBD2** que permita efetuar as operações seguintes:

   a. Alterar a taxa de IVA dos produtos de 21% para 23%;

   b. Eliminar o produto com código 2;

   c. Calcular o preço com Iva dos produtos inseridos nas tabelas.

6- No mesmo projeto, cria a classe **OperacoesBD3** que permita efetuar a operação seguinte:

   a. Listar todos os produtos da tabela apresentando os preços com o formato monetário de euro e taxa de Iva com os símbolos de percentagem.

Formatação de dados com a classe NumberFormat

A classe NumberFormat, faz parte do pacote java.text e permite formatar números conforme a localização geográfica em que nos encontramos, realizando a distinção entre o sinal de ponto, milhar e de decimal, também identifica a posição do sinal do número e identifica o prefixo que indica a moeda em caso de valores monetários.

Para a sua utilização é necessário **importar a classe**:

```
import java.text.NumberFormat;
```

Os principais métodos do NumberFormat são:

getNumberInstance()
Retorna a instância de um objeto com base no formato da localidade padrão. É utilizado para números

getCurrencyInstance()
Usado para formatar moedas

getIntegerInstance()
Usado para formatar números ignorando casas decimais

getPercentInstance()
Usado para formatar frações para percentagem, por exemplo 0,15 é formatado e mostrado como 15%

Exemplos:

```
import java.text.NumberFormat;
public class Formata {
    public static void main(String[] args) {
         NumberFormat z = NumberFormat.getCurrencyInstance();
        System.out.println(z.format(12345678.90));
        // imprime: 12.345.678,90 €
    }
}
```

Exemplo para uma situação em que há vários números a serem formatados.

```
import java.text.NumberFormat;

public class Formata2 {
    public static void main(String[] args) {

        double n[]={523.34, 54344.23 ,95845.223 ,1084.895};
        NumberFormat z = NumberFormat.getCurrencyInstance();
```

```
 for (int a = 0; a < n.length; a++) {
            if(a != 0)
                System.out.print(", ");
                System.out.print(z.format(n[a]));
        }
        System.out.println();
        /*
        Imprime:
        523,34 €, 54.344,23 €, 95.845,22 €, 1.084,89 €        */
    }
}
```

Criar uma formatação personlizada utilizando o objeto DecimalFormat.

```
import java.text.DecimalFormat;

public class Formata {

    public static void main(String[] args) {
        double valor = 2000.0;
        double vezes = 3.0;
        double prestacao = valor/vezes;

        DecimalFormat df = new DecimalFormat("0.##");
        String dx = df.format(prestacao);
        System.out.print(dx);
    }
    // imprime:  666,67
}
```

```
0 - reserva de dígitos fixas
# - reserva de digítos que podem ou não ser inseridos
. - separador do decimal
, - separador de grupos
```

Algumas strings de formatação que podem ser utilizadas com o Decimal Format

Máscara de formatação	Formato impresso	Descrição
,##0,00	1,242.50	Separa grupo dos milhares com vírgulas, se o número é menor que um mostra zeros na frente.
$,##0.00;($,##0.00)	($1,535,50)	Números negativos entre parênteses. mostra $
0.#####	1244.5	Se número entre -1 e 1 mostra zero na frente e não mostra zeros no final.

Classe NumberFormat e seus métodos:
https://docs.oracle.com/javase/7/docs/api/java/text/NumberFormat.html

EXERCÍCIO: Escreve o código relativo ao 1.º exemplo. Adiciona as instruções necessárias de forma a obter o mesmo resultado, mas agora, utilizando a classe DecimalFormat.