Para efetuar a ligação entre os programas em Java e as bases de dados (qualquer que seja o SGBD) é necessário adicionar uma biblioteca própria o – **JDBC** ou Java **D**ata**B**ase **C**onnectivity (conjunto de classes Java compactadas num ficheiro com extensão jar), que permitirá, a partir dos programas, o envio de instruções SQL para a base da dados.
O JDBC Driver a usar será o ***Connector/J.***

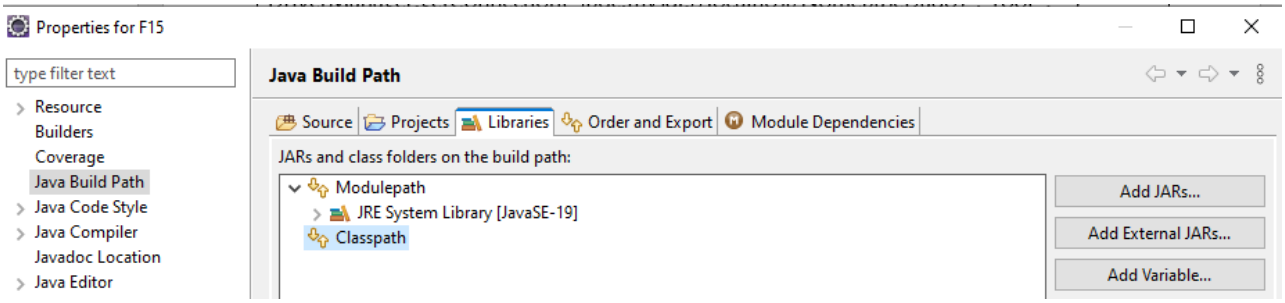
Instruções para adicionar o Connector/J:

1. Fazer o download do connector/J em: https://dev.mysql.com/downloads/file/?id=476198
2. Descompactar o ficheiro **mysql-connector-java-<versão>.zip**;
3. Copiar o ficheiro para a pasta: C:\Program files\Java\jdk---version\lib

1.ª Ligação a uma BD

- É necessário criar duas classes: Uma para ligação/conexão com a BD e outra para testar a ligação/conexão.

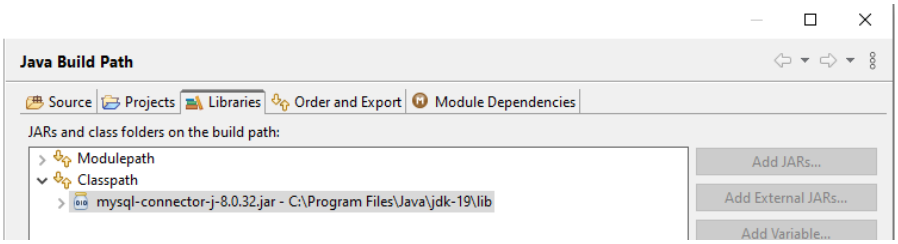
1. Cria um projeto no eclipse com o nome **LigarBD**;
2. Clica com o botão direito do rato sobre o projeto para adicionar o conetor mysql ao projeto:
   - Selecionar o item propriedades/**properties** e no menu da esquerda selecionar o item **Java Build Path** devendo visualizar-se uma janela como a seguinte



   - Selecionar *Classpath*

Para selecionar o ficheiro jar

- Clicar no botão à esquerda *"ADD External JARs... "*



- Clicar no botão Aply and Close

3. Cria uma classe sem o método main e dá-lhe o nome de ligacao (*ligacao.java*)
   Esta será a classe responsável por estabelecer a conexão/ligação com a BD.

4. O código que permite fazer a ligação a uma base de dados compreende as seguintes fases:

- Indicação de que será utilizado o Connector/J:

```
Class.forName("com.mysql.jdbc.driver")
```

- Indicação da ligação:

```
Connection con =
DriverManager.getConnection("jbdc:mysql://localhost/NomeBaseDados","root","")
```

Notas:
A base de dados encontra-se em *localhost* e denomina-se *NomeBaseDados* . O acesso é feito através de uma conta no *MySQL*, cujo utilizador é *root* e com palavra-passe nula ("");

Tanto a linha de obtenção do *Driver* como a de obtenção da conexão/ligação deve ser inserida dentro do um bloco try/catch, pois as classes de *.sql* são exceções verificadas.

- Fechar a ligação.

Código para efetuar a ligação à base de dados VendasCD (com tratamento de erros)

Exercício: Criar uma nova classe sem método main de nome **ObterLigacao**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ObterLigacao {

// Criar uma função de nome OL que proceda à ligação à BD desejada
  public Connection OL(){
       System.out.println("Testar o acesso a BD MySQL");
       Connection con=null;
       try{
       //Indica que será utilizado o driver connector/J

        Class.forName("com.mysql.jdbc.Driver");



/* Liga ao sevidor BD Local, com o utilizador root e pwd nula, acedendo à bd:
VendasCD*/

       con = DriverManager.getConnection("jdbc:mysql://localhost/VendasCD", "root", "");
       System.out.println("Ligação efetuada com sucesso");
}
```

```
/*tratamento de exceções método forName da classe "Class" - caso a BD não exista*/

catch(ClassNotFoundException cnfe){
                    System.out.println("ClassNotFoundException");
             }
 /* trata a exceção lançada pelo método getConnetion da classe "DriverManager"*/

             catch(SQLException sqle){
                    System.out.println( sqle.getMessage());
             }
             return con;  // caso não haja erros é efetuada a ligação
      }
/* método FecharLigacao para fechar a ligação à BD */

      public void FecharLigacao(Connection con){
             try{
                    con.close();
                    System.out.println("\nLigação fechada com sucesso!");
             }
             catch(SQLException sqle){
                    System.out.println("SQlExeption");
}}}
```

Execução de comandos sobre uma base de dados

Como sabes, os dados de uma base de dados são consultados e manipulados através de comandos SQL. Para que um programa em Java possa enviar estes comandos à BD, existem 3 interfaces Java disponíveis:

- **Statement** — permite a execução dos comandos fundamentais de SQL (SELECT, INSERT, UPDATE, DELETE, entre outros);
- **PreparedStatement** — permite guardar instruções SQL pré-compiladas, quando a BD suporta este recurso;
- **CallableStatement** — permite executar procedimentos e funções armazenados na BD, quando a BD suporta este recurso;

Diferença entre Statement e PreparedStatement

Para executar a mesma consulta repetidas vezes mudando apenas os parâmetros de cada uma, a execução usando *PreparedStatements* será mais rápida e com menos carga sobre a BD, devido à sua flexibilidade.

Exemplo:

Com **Statement:**

```
Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT col1, col2, col3 FROM sua_tabela WHERE col1 = 'value1' AND
col3 = 1");
```

Com **PreparedStatement:**

```
PreparedStatement stmt = conn.preparedStatement("SELECT col1, col2 FROM sua_tabela WHERE col1 = ? AND col3
= ?");
stmt.setString(1, "value1");
stmt.setInt(2, 1);
ResultSet rs = stmt.executeQuery();
```

NOTA: os parâmetros da consulta têm que ser atribuídos através dos métodos `setInt()`, `setString()`, etc. presentes na interface `PreparedStatement` e não por concatenação de strings.

Esta classe irá utilizar os métodos da classe anterior.

Todas as classes do projeto que careçam de aceder à base de dados terão que, à semelhança desta classe, aceder aos métodos da classe ObterLigacao.class, tendo a seguinte estrutura:

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.SQLException;

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

	ObterLigacao obterLigacao=new ObterLigacao ();
	Connection con = obterLigacao.OL();

		//colocar aqui o código

	obterLigacao.FecharLigacao(con);
}

    }
```

Exercício: Cria uma nova classe com main de nome ***TestarLigacao*** e acrescenta-lhe o código escrito atrás.

Neste caso iremos consultar os dados da tabela editoras, pelo que, no local correspondente ao comentário acrescenta o seguinte código:

```
try{
/* Declara o objeto stm da interface Statement, necessário para enviar comandos à BD*/

	Statement stm = con.createStatement(); // iremos usar a interface Statement

/* consulta todos os registos da tabela editoras e guarda-os num objeto res da
interface ResultSet. O comando executeQuery executa uma pesquisa na BD*/
	ResultSet res = stm.executeQuery("SELECT * FROM editoras");

	while(res.next()){

	   System.out.print("Código da Editora: " + res.getInt("CodEditora"));
	   System.out.print(" Nome da Editora: " + res.getString("NomeEditora")+"\n");

	}

	}
	catch(SQLException sqle){
		System.out.println("SQlExeption");
	}
```


Bibliografia:

https://sourceforge.net/projects/ucanaccess/files/latest/download

https://www.youtube.com/watch?v=9msZFGLhUzI

https://www.youtube.com/watch?v=IEik8lbs1eM

https://www.devmedia.com.br/como-conectar-uma-aplicacao-java-a-um-banco-de-dados-access/28184

```
01.  //Aqui criamos uma função responsável pela conexão
02.  private void Conecta()
03.  {
04.       try {
05.         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
06.          }
07.       catch (Exception Excecao) {
08.           JOptionPane.showMessageDialog(null,"SQLException: " + Excecao.getMessage(),"Erro: Carga do banco de dados",JOptionPane.INFORMATION_MESSAGE);
09.           }
10.       try{
11.           Conexao = DriverManager.getConnection("jdbc:odbc:Curso","","");
12.           Comando = Conexao.createStatement();
13.          }
14.       catch (Exception Excecao) {
15.           JOptionPane.showMessageDialog(null,"SQLException: " + Excecao.getMessage(),"Erro: Problema ao Conectar",JOptionPane.INFORMATION_MESSAGE);
16.           }
17.  }
```

```
String caminho = "C:/Program Files/PTW/DiaSoft/DataBase/user.mdb";
                                                Driver d =
(Driver)Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
                                                Connection con =
DriverManager.getConnection("jdbc:odbc:Driver={Microsoft Access Driver
(*.mdb)};DBQ=" + caminho);



                                                Statement stm =
con.createStatement();
```

```
import java.sql.*;

Connection
conn=DriverManager.getConnection("jdbc:ucanaccess://C:/__tmp/test/zzz.accdb");
Statement s = conn.createStatement();
ResultSet rs = s.executeQuery("SELECT [LastName] FROM [Clients]");
while (rs.next()) {
    System.out.println(rs.getString(1));
}
```

https://www.codejava.net/java-se/jdbc/java-jdbc-example-connect-to-microsoft-access-database

```
 try {

            Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");
        }
        catch(ClassNotFoundException cnfex) {

            System.out.println("Problem in loading or "
                    + "registering MS Access JDBC driver");
            cnfex.printStackTrace();
        }

        // Step 2: Opening database connection
        try {

            String msAccDB = "D:/WORKSPACE/TEST_WORKSPACE"
                    + "/Java-JDBC/Player.accdb";
            String dbURL = "jdbc:ucanaccess://"
                    + msAccDB;

            // Step 2.A: Create and
            // get connection using DriverManager class
            connection = DriverManager.getConnection(dbURL);

            // Step 2.B: Creating JDBC Statement
            statement = connection.createStatement();

            // Step 2.C: Executing SQL and
            // retrieve data into ResultSet
            resultSet = statement.executeQuery("SELECT * FROM PLAYER");
```

```
            System.out.println("ID\tName\t\t\tAge\tMatches");
            System.out.println("==\t================\t===\t=======");

            // processing returned data and printing into console
            while(resultSet.next()) {
                System.out.println(resultSet.getInt(1) + "\t" +
                        resultSet.getString(2) + "\t" +
                        resultSet.getString(3) + "\t" +
                        resultSet.getString(4));
            }
        }
        catch(SQLException sqlex){
            sqlex.printStackTrace();
        }
        finally {
            // Step 3: Closing database connection
            try {
                if(null != connection) {
                    // cleanup resources, once after processing
                    resultSet.close();
                    statement.close();

                    // and then finally close connection
                    connection.close();
                }
            }
            catch (SQLException sqlex) {
                sqlex.printStackTrace();
            }
        }
    }
```