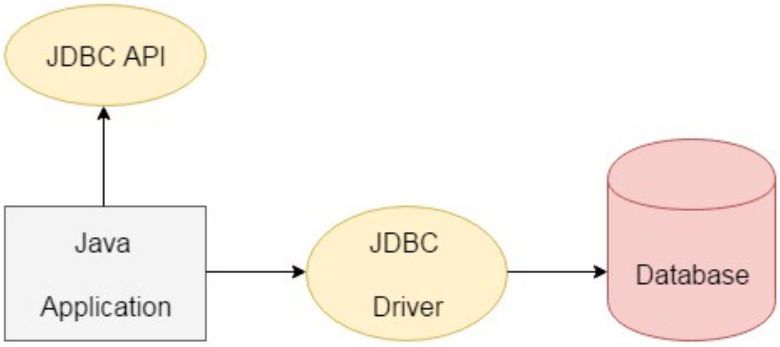
Java Database Connectivity (JDBC)

Bekerja dengan *database di* dalam bahasa pemrograman Java artinya berbicara tentang JDBC. Java JDBC API, lebih tepatnya, memungkinkan aplikasi anda untuk dapat melakukan DML (*data manipulation language*) dan DDL (*data definition language*) di berbagai jenis *database* yang ada, seperti: MySQL, PostgreSQL, MS SQL Server, Oracle, dll. Namun, sebelum anda dapat melakukan DML dan DDL, ada yang namanya JDBC *driver* yang mengubah perintah DML & DDL untuk dapat dikenali oleh *database*.

Ada 4 tipe JDBC *driver*:

- JDBC-ODBC *bridge* JDBC *driver*
- Partially Java code + *native code* JDBC *driver*
- All Java + *middleware* JDBC *driver*
- *Pure* Java



Ilustrasi 1: how Java App communicates to database

Ingat! JDBC *driver* **tidak sama dengan** JDBC API.

JDBC API berisikan:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface
- DriverManager class
- Blob class
- Clob class
- Types class

Adapun perintah DML:

- **S**elect – select * from employee where nip=’1234’
- **I**nsert – insert into employee(nip, nama, email) values(‘1234’, ‘bejo’, ‘bejo@del.ac.id’)

- **D**elete – delete from employee where nip=’1234’
- **U**pdate – update employee set nama=’ganteng’ where nip=’1234’

Sedangkan perintah DDL:

- **C**reate table – create table employee(nip varhcar(4), nama varchar(30), email varchar(12))
- **D**rop table – drop table employee
- **A**lter table – alter table employee add gender char(1)

Pada praktikum kali ini anda akan menggunakan *database* MySQL. Jadi, pastikan dahulu anda telah meng-*install* MySQL. Karena anda menggunakan *database* ini, anda harus terlebih dahulu mengunduh *mysql-connector* (driver) yang dapat anda temukan di cis.del.ac.id atau https://dev.mysql.com/downloads/connector/j/. Pastikan anda mengunduh *driver* yang benar yang sesuai dengan OS.

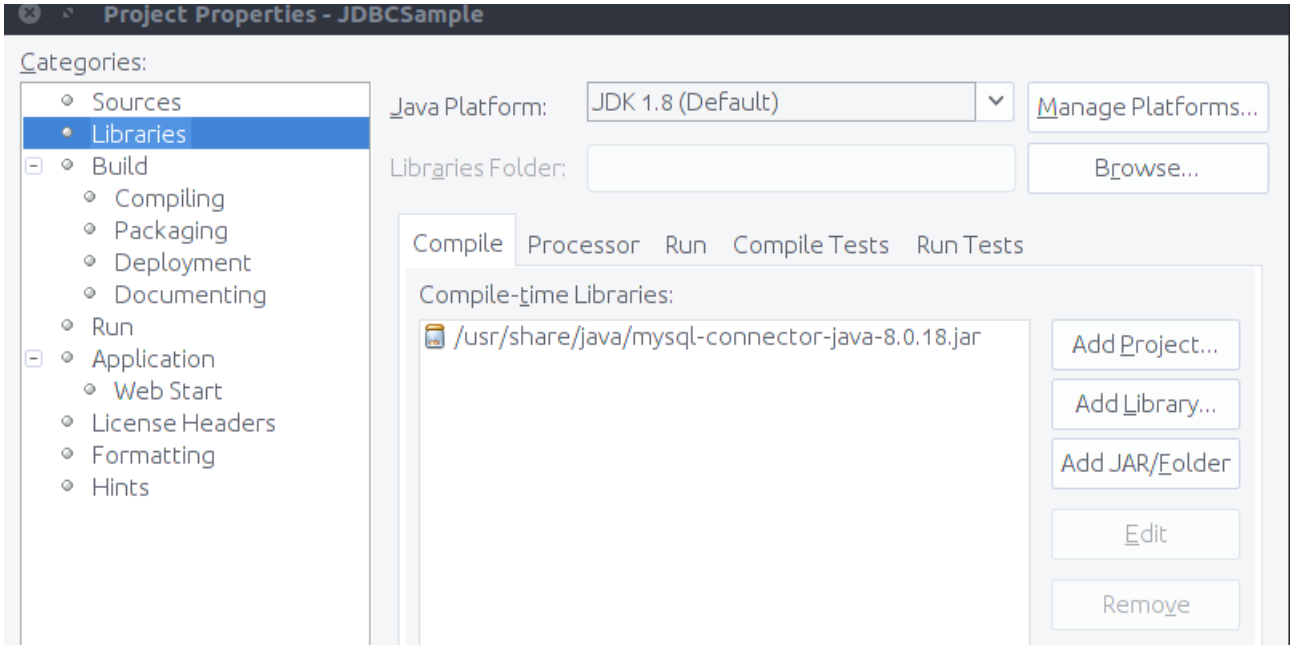
Buatlah terlebih dahulu *database* di MySQL anda dengan nama **PROFIL**.

```
create database PROFIL;
use PROFIL;
```

Setelah anda berhasil membuat *database* nya, saatnya anda membuat sebuah *table* dengan nama **TMahasiswa**.

```
CREATE TABLE TMahasiswa (
NIM VARCHAR(8) PRIMARY KEY,
NAMA VARCHAR(30) NOT NULL,
DOB DATE NOT NULL,
EMAIL VARCHAR(50)
)
```

Dari sisi Java, buka IDE Java pilihan anda (misal: NetBeans) lalu buat sebuah *project* baru dengan nama JDBCSample. *Select* proyek JDBCSample > *right click* > *Properties*. Dari menu *categories*, pilih *libraries*. Tambahkan *mysql-connector* yang telah anda unduh sebelumnya dengan cara menekan *button* “Add JAR/Folder”, lalu arahkan dimana *connector* itu berada.



Ilustrasi 2: how to add a new mysql connector

Setelah berhasil menambahkan *library* baru ke proyek anda, sekarang saatnya membuat sebuah kelas java. Beri nama kelas tersebut dengan **TMahasiswa.java**, lalu ketikan beberapa kode program di bawah ini:

```
6   package jdbcsample;
7
8   import java.sql.Date;
9
10
11
12  /**
13   *
14   * @author teamsar
15   */
16  public class TMahasiswa {
17      private String _nim, _nama, _email;
18      private Date _dob;
19
20      public void setNim(String nim){
21          _nim = nim;
22      }
23
24      public void setNama(String nama){
25          _nama = nama;
26      }
27
28      public void setEmail(String email){
29          _email = email;
30      }
31
32      public void setDob(Date dob){
33          _dob = dob;
34      }
35
36      public String getNim(){ return _nim; }
37      public String getNama(){ return _nama; }
38      public String getEmail(){ return _email; }
39      public Date getDob(){ return _dob; }
40  }
41
```

Lalu pada *file* java *driver* anda (JDBCSample.java), ketikan beberapa kode program seperti di bawah ini:

```
6   package jdbcsample;
7
8   import java.sql.Connection;
9
10  import java.sql.DriverManager;
11  import java.sql.PreparedStatement;
12  import java.sql.SQLException;
13  import java.text.ParseException;
14  import java.util.Scanner;
15  import java.text.SimpleDateFormat;
16  import java.util.Date;
17  import java.util.logging.Level;
18  import java.util.logging.Logger;
19
20  /**
21   *
22   * @author teamsar
23   */
24  public class JDBCSample {
25
26      /**
27       * @param args the command line arguments
28       */
29      public static void main(String[] args) throws SQLException, ParseException {
30          try {
31              // TODO code application logic here
32              String className = "com.mysql.cj.jdbc.Driver";
33              Class.forName(className);
34              System.out.println("Driver loaded successfully!");
```

```
35
            Connection con = DriverManager.getConnection("jdbc:mysql://172.22.42.4:3306/PROFIL",
37              "root", "newpassword123");
38
39          if(con.isClosed()) System.out.println("Connection is closed!");
40          else {
41              TMahasiswa objMhs = new TMahasiswa();
42              System.out.print("Masukan NIM: "); objMhs.setNim(new Scanner(System.in).nextLine());
43              System.out.print("Masukan nama mahasiswa: "); objMhs.setNama(new Scanner(System.in).nextLine());
44              System.out.print("Masukan DOB: ");
45              objMhs.setDob(convertToDateFromString(new Scanner(System.in).nextLine(), "dd/MM/yyyy"));
46              System.out.print("Masukan email: "); objMhs.setEmail(new Scanner(System.in).nextLine());
47
48              String _sql = "INSERT INTO TMahasiswa(NIM, NAMA, DOB, EMAIL) VALUES(?, ?, ?, ?)"; // template sql syntax
49              PreparedStatement pStmnt = con.prepareStatement(_sql);
50              pStmnt.setString(1, objMhs.getNim()); // index 1 pada kolom 1
51              pStmnt.setString(2, objMhs.getNama()); // index 2 pada kolom 2
52              pStmnt.setDate(3, objMhs.getDob()); // index 3 pada kolom 3
53              pStmnt.setString(4, objMhs.getEmail()); // index 4 pada kolom 4
54
55              int response = pStmnt.executeUpdate();
56              if (response > 0) System.out.println("Success save data");
57              else System.out.println("Unable to save the data");
58
59              con.close();
60          }
61      } catch (ClassNotFoundException ex) {
62          System.err.println(ex.getException());
63          Logger.getLogger(JDBCSample.class.getName()).log(Level.SEVERE, null, ex);
64      }
65  }
66
67  public static java.sql.Date convertToDateFromString(String date, String format) throws ParseException{
68      SimpleDateFormat sdf1 = new SimpleDateFormat(format);
69      Date dt = sdf1.parse(date);
70      return new java.sql.Date(dt.getTime());
71  }
72
73  }
```

Line ke-33 menunjukan bagaimana cara me-*load* JDBC *driver* yang telah ditambahkan ke dalam *library* anda.

Bila anda perhatikan pada *line* ke-35, disana terdapat cara bagaimana melakukan *open-connection* ke *database*. Gantilah *ip address* 172.22.42.4 dengan *ip address* anda, dan sesuaikan *username* dan *password* nya.

Penggunaan *PreparedStatement* untuk melakukan operasi DML dan atau DDL sangat disarankan, hal ini mengingat *PreparedStatement* memiliki banyak keunggulan dibandingkan dengan *Statement* biasa, seperti:

- Memungkinkan anda untuk membuat *query* dengan parameter yang dinamis. Lihat *line code* ke-48.
- *PreparedStatement* jauh lebih cepat dibandingkan *Statement* biasa yang ada di Java. Hal ini dikarenakan, *query sql* anda telah disimpan di dalam *cache database* dan dapat di-"*reuse*" setiap kali anda mengirimkan data ke *database*. Lihat cara penggunaan *PreparedStatement* di *line code* 49 – 55.
- *PreparedStatement* mencegah terjadinya *SQL Injection.*
- *PreparedStatement* lebih mudah terbaca dan dipahami bila dibandingkan dengan penggunaan *string* konkatenasi (*string concate*).
- Formating data (sebelum dikirimkan melalui parameter) jauh lebih mudah dan leluasa dilakukan dibandingkan dengan *Statement* biasa.

Setelah anda memasukan data-data melalui aplikasi java di atas, anda dapat melihat data yang baru saja anda entri melalui *query select* biasa dengan IDE pilihan:

```
mysql> select * from TMahasiswa;
+----------+---------------+------------+-------------------------+
| NIM      | NAMA          | DOB        | EMAIL                   |
+----------+---------------+------------+-------------------------+
| 11318068 | Bejo Sianturi | 1996-08-02 | bejo.sianturi@gmail.com |
+----------+---------------+------------+-------------------------+
1 row in set (0.00 sec)
```

Untuk membaca data dari *database*, anda juga dapat menggunakan *PreparedStatement*. Untuk lebih jelasnya, ubahlah kode program di atas yang telah dibuat sebelumnya.

```
6   package jdbcsample;
7
8   import java.sql.Connection;
9
10  import java.sql.DriverManager;
11  import java.sql.PreparedStatement;
12  import java.sql.ResultSet;
13  import java.sql.SQLException;
14  import java.text.ParseException;
15  import java.util.Scanner;
16  import java.text.SimpleDateFormat;
17  import java.util.Date;
18  import java.util.logging.Level;
19  import java.util.logging.Logger;
20
21  /**
22   *
23   * @author teamsar
24   */
25  public class JDBCSample {
26
27      /**
28       * @param args the command line arguments
29       */
30      public static void main(String[] args) throws SQLException, ParseException {
31          try {
32              // TODO code application logic here
33              String className = "com.mysql.cj.jdbc.Driver";
34              Class.forName(className);
35              System.out.println("Driver loaded successfully!");
36
37              Connection con = DriverManager.getConnection("jdbc:mysql://172.22.42.4:3306/PROFIL",
38                      "root", "newpassword123");
39
40              if(con.isClosed()) System.out.println("Connection is closed!");
41              else {
42                  TMahasiswa objMhs = new TMahasiswa();
43                  String _sql = "SELECT * FROM TMahasiswa WHERE NIM=?";
44                  PreparedStatement ps = con.prepareStatement(_sql);
45                  ps.setString(1, "11318068");
```

```
47                  ResultSet rs = ps.executeQuery();
48                  while(rs.next()){
49                      objMhs.setNim(rs.getString("NIM"));
50                      objMhs.setNama(rs.getString("NAMA"));
51                      objMhs.setDob(rs.getDate("DOB"));
52                      objMhs.setEmail(rs.getString("EMAIL"));
53                  }
54
55                  System.out.println(String.format("Mahasiswa %s dengan NIM %s lahir pada tanggal %s.",
56                          objMhs.getNama(), objMhs.getNim(), String.valueOf(objMhs.getDob())));
57                  System.out.println(String.format("Anda dapat menghubungi mahasiswa tersebut ke \"%s\"", objMhs.getEmail()));
58
59                  con.close();
60              }
61          } catch (ClassNotFoundException ex) {
62              System.err.println(ex.getException());
63              Logger.getLogger(JDBCSample.class.getName()).log(Level.SEVERE, null, ex);
64          }
65      }
66
67      public static java.sql.Date convertToDateFromString(String date, String format) throws ParseException{
68          SimpleDateFormat sdf1 = new SimpleDateFormat(format);
69          Date dt = sdf1.parse(date);
70          return new java.sql.Date(dt.getTime());
71      }
72
73  }
74
```

CHALLENGE!
Dengan memodifikasi program di atas, buatlah aplikasi yang dapat memasukan daftar mata kuliah yang ada di IT Del, kemudian mata kuliah apa saja yang diambil seorang mahasiswa. (Note: buatlah dua table baru dengan nama **TMatakuliah** dan table pembantu **TmhsMatkul**). Tampilkan ke dalam layar mata kuliah yang diambil tiap-tiap mahasiswa.

Submission dalam bentuk project java dan juga *syntax sql* yang di zip menjadi satu.