TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Nhập môn Công nghệ phần mềm
Introduction to Software Engineering
(IT3180)

Bài tập tuần 02

Vòng đời phần mềm &
Lập trình với cơ sở dữ liệu

Giang vien

Mục tiêu

- Thực hiện các bài tập (câu hỏi) về nội dung Vòng đời phần mềm
- Xác định các đầu vào (input) và kết quả (output) cho nhóm yêu cầu nghiệp vụ của bài toán (case study)
- Lập trình với cơ sở dữ liệu:
  - Xây dựng ứng dụng Java trên công cụ Netbeans
  - Thiết kế, xây dựng cơ sở dữ liệu với phpMyAdmin trên MySQL
  - Lập trình ứng dụng Java kết nối MySQL, thực thi các truy vấn SQL

Đánh giá

- Hoàn thành các bài tập về nội dung Vòng đời phần mềm, nắm được đặc điểm chính của các mô hình phát triển phần mềm khác nhau.
- Các nhóm sinh viên xác định được thông tin cơ bản (input / output) cho nghiệp vụ mà nhóm đã lựa chọn để phát triển
- Hoàn thành bài thực hành lập trình với cơ sở dữ liệu

Bài 1.1

a) Mô hình bản mẫu (prototyping model) của phát triển phần mềm là …

1. Một cách tiếp cận hợp lý khi yêu cầu được định nghĩa rõ ràng
2. Một cách tiếp cận hữu ích khi khách hàng không thể định nghĩa yêu cầu rõ ràng
3. Cách tiếp cận tốt nhất cho những dự án có đội phát triển lớn
4. Tất cả các phương án trên đều sai

Bài 1.1

b) Bước đầu tiên trong vòng đời phát triển phần mềm (Software Development Life Cycle) là?

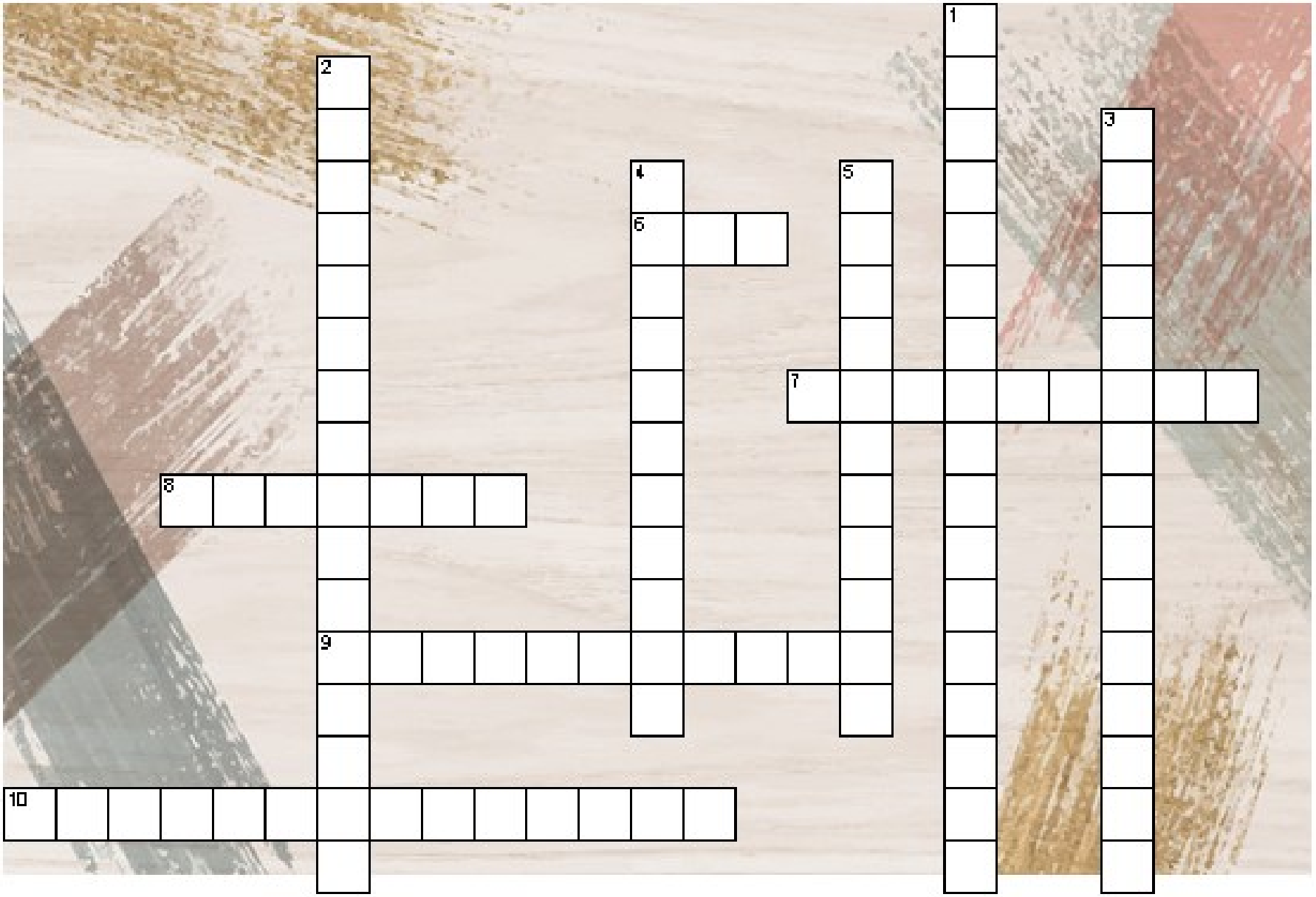
1. Xác định các nhu cầu và ràng buộc
2. Viết phần mềm
3. Vận hành hệ thống để loại bỏ các khiếm khuyết
4. Nâng cao sản phẩm sau khi đã triển khai

Bài 1.1

c) Tình huống nào không phù hợp để có thể áp dụng mô hình thác nước? (chọn nhiều)

1. Khó khăn trong việc bổ sung yêu cầu mới trong các pha sau của tiến trình
2. Các yêu cầu được xác định rõ ràng, đầy đủ ngay từ đầu
3. Khách hàng muốn có sản phẩm vận hành được sớm
4. Khó thu thập đầy đủ yêu cầu ngay ở giai đoạn đầu của dự án

Bài 1.2 Giải ô chữ



Bài 1.2 Giải ô chữ

Các gợi ý cho ô chữ:

Across

6. Mô hình phát triển tăng dần từng bước, chia thành các team thực hiện đầy đủ các pha. Mỗi chu trình phát triển rất ngắn từ 60-90 ngày
7. Các yêu cầu có tính ổn định cao. Các pha của tiến trình phát triển thực hiện tuần tự từng bước, không quay trở lại pha trước đó
8. Giai đoạn kiểm tra logic bên trong và chức năng bên ngoài để đảm bảo kết quả đầu ra chính xác
9. Giai đoạn đáp ứng những thay đổi, nâng cấp phần mềm đã phát triển do sự thay đổi của môi trường, nhu cầu
10. Có ưu điểm là khả năng tái sử dụng phần mềmthông qua các hoạt động: phân tích yêu cầu, thiết kế với sử dụng lại, phát triển và tích hợp, thẩm định

Down

1. Thời kỳ tính từ khi phần mềm được sinh (tạo) ra cho đến khi chết đi
2. Sự kết hợp mô hình tuần tự và ý tưởng lặp lại của chế bản mẫu
3. Tập hợp các hành động, phương thức, thực hành, thay đổi mà người ta dùng để duy trì và phát triển phần mềm và các thành phần liên quan
4. Mô hình có thể sữ dụng như "hệ sơ khai" để thu thập yêu cầu
   người dùng qua các thiết kế nhanh
5. Rủi ro được xem xét kỹ ở mỗi lần lặp tiến trình

Bài 1.2 Giải ô chữ

Hướng dẫn:

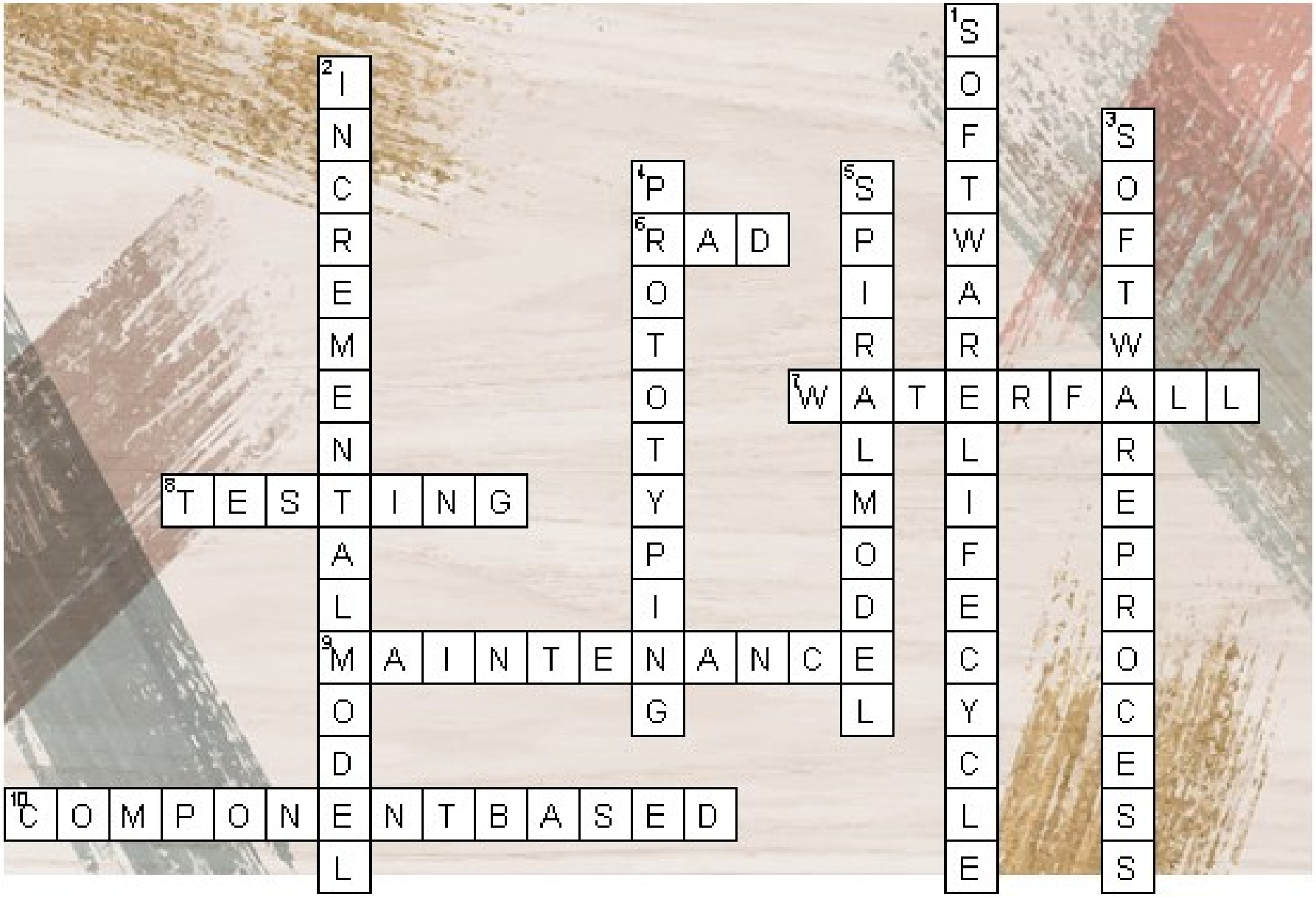
Across

6. Mô hình phát triển tăng dần từng bước, chia thành các team thực hiện đầy đủ các pha. Mỗi chu trình phát triển rất ngắn từ 60-90 ngày [RAD]
7. Các yêu cầu có tính ổn định cao. Các pha của tiến trình phát triển thực hiện tuần tự từng bước, không quay trở lại pha trước đó [WATERFALL]
8. Giai đoạn kiểm tra logic bên trong và chức năng bên ngoài để đảm bảo kết quả đầu ra chính xác [TESTING]
9. Giai đoạn đáp ứng những thay đổi, nâng cấp phần mềm đã phát triển do sự thay đổi của môi trường, nhu cầu [MAINTENANCE]
10. Có ưu điểm là khả năng tái sử dụng phần mềmthông qua các hoạt động: phân tích yêu cầu, thiết kế với sử dụng lại, phát triển và tích hợp, thẩm định [COMPONENTBASED]

Down

1. Thời kỳ tính từ khi phần mềm được sinh (tạo) ra cho đến khi chết đi [SOFTWARELIFECYCLE]
2. Sự kết hợp mô hình tuần tự và ý tưởng lặp lại của chế bản mẫu [INCREMENTALMODEL]
3. Tập hợp các hành động, phương thức, thực hành, thay đổi mà người ta dùng để duy trì và phát triển phần mềm và các thành phần liên quan [SOFTWAREPROCESS]
4. Mô hình có thể sữ dụng như "hệ sơ khai" để thu thập yêu cầu
   người dùng qua các thiết kế nhanh [PROTOTYPING]
5. Rủi ro được xem xét kỹ ở mỗi lần lặp tiến trình [SPIRALMODEL]

Bài 1.2 Giải ô chữ



Bài 1.3 So sánh các mô hình phát triển PM

Các mô hình: Thác nước, Chế thử, Gia tăng, Xoắn ốc, Phát triển dựa trên thành phần

	Đặc điểm chính	Ưu điểm	Nhược điểm	Tình huống áp dụng phù hợp
Mô hình Thác nước				
Mô hình Chế thử				
Mô hình Gia tăng				
Mô hình Xoắn ốc				
Mô hình dựa thành phần				

Bài 1.3 So sánh các mô hình phát triển PM

Hướng dẫn

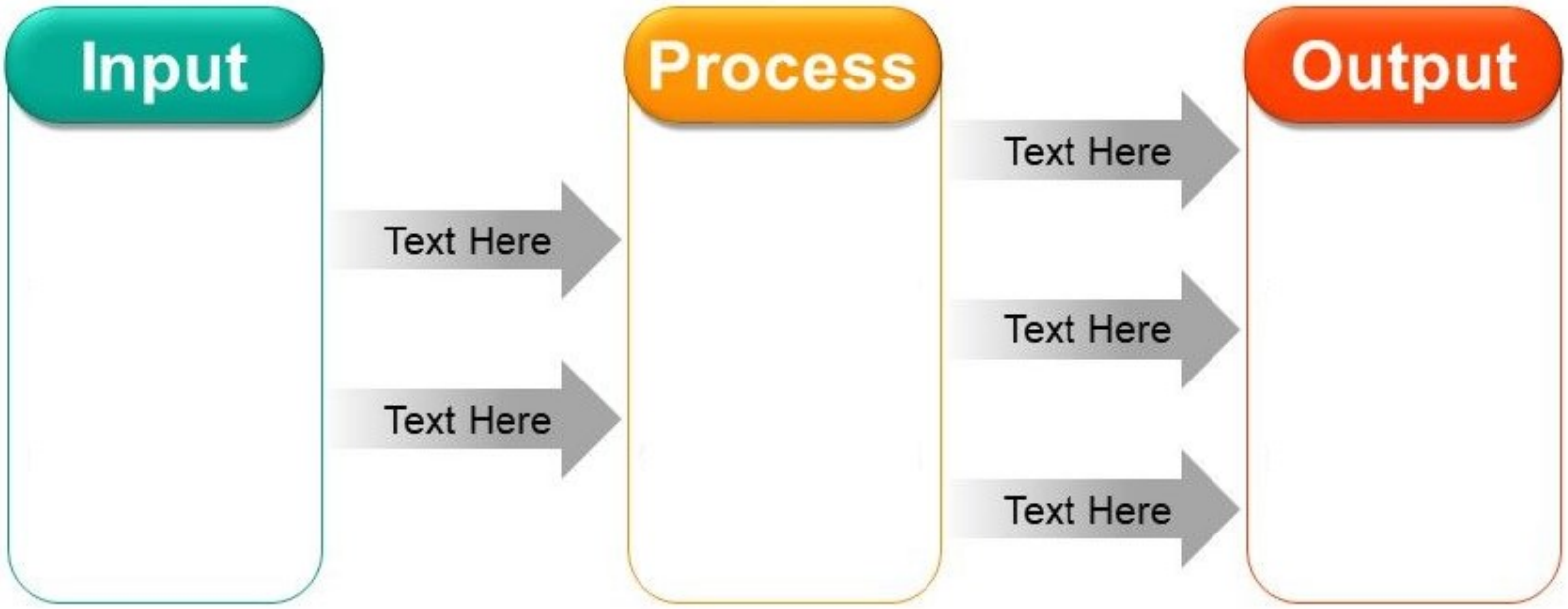
	Đặc điểm chính	Ưu điểm	Nhược điểm	Tình huống áp dụng phù hợp
Mô hình Thác nước	Tuyến tính, không quay lui,…	…	…	…
Mô hình Chế thử	Lặp, xây dựng bản mẫu,…	…	…	…
Mô hình Gia tăng	Lặp, tăng dần,…	…	…	…
Mô hình Xoắn ốc	Lặp, quản trị rủi ro,…	…	…	…
Mô hình dựa thành phần	Tái sử dụng,…	…	…	…

Bài 1.4

- Phân tích thông tin cơ bản (input / output) cho nghiệp vụ bài toán (case study)
  - Các nhóm thảo luận để xác định đầu vào (input) và kết quả (output) cho nhóm yêu cầu nghiệp vụ của bài toán (case study) mà nhóm đã lựa chọn.
  - Phân rã mô tả của nhóm yêu cầu nghiệp vụ thành các nghiệp vụ nhỏ hơn.
  - Với mỗi nghiệp vụ nhỏ này thực hiện phân tích các thông tin cụ thể

Bài 1.4

- Có thể vẽ sơ đồ thể hiện kết quả phân tích:



Thực hành lập trình với cơ sở dữ liệu

- JDBC là gì?
  - Là các API Java chuẩn tắc để truy cập CSDL quan hệ
    - Ứng dụng không cần quan tâm tới chi tiết cụ thể của CSDL
  - Nằm trong Java SE (J2SE)
    - Java SE 6 có phiên bản JDBC 4

Thực hành lập trình với cơ sở dữ liệu

- JDBC API
  - Định nghĩa một tập các Java Interfaces, được cài đặt bởi các vendor khác nhau, thành các JDBC Drivers
    - Các ứng dụng sử dụng tập các giao diện này để thực hiện các thao tác với CSDL → Tính portability
  - Phần lớn API của JDBC nằm trong gói java.sql.*
    - DriverManager, Connection, ResultSet, DatabaseMetaData, ResultSetMetaData, PreparedStatement, CallableStatement và Types
  - Một số chức năng nâng cao khác nằm trong gói javax.sql
    - DataSource

Thực hành lập trình với cơ sở dữ liệu

- JDBC Driver
  - Là cài đặt cụ thể của các JDBC interfaces
    - Tất cả các database server đều có JDBC driver(s) tương ứng

Thực hành lập trình với cơ sở dữ liệu

- Database URL
  - Được sử dụng để tạo một kết nối tới database
    - Có thể chứa server, port, protocol, etc.
  - Cú pháp:
    - jdbc:subprotocol_name:driver_dependant_databasename
  - Ví dụ:
    - Oracle thin driver
      - jdbc:oracle:thin:@machinename:1521:dbname
    - Derby
      - jdbc:derby://localhost:1527/sample
    - Pointbase
      - jdbc:pointbase:server://localhost/sample

Thực hành lập trình với cơ sở dữ liệu

- Các bước lập trình với JDBC
  - B1. Load JDBC driver cho từng loại CSDL
  - B2. Lấy đối tượng Connection
  - B3. Lấy đối tượng Statement
  - B4. Thực hiện câu truy vấn, câu lệnh update
  - B5. Đọc kết quả trả về
  - B6. Đọc các Meta-data (tùy chọn)
  - B7. Đóng đối tượng Statement và đối tượng Connection

B1. Load JDBC driver cho từng loại CSDL

- **Để load về driver cho CSDL và đăng ký nó với DriverManager, cần load class tương ứng**
  - **Class.forName(<database-driver>)**

```
try {
    // This loads an instance of the Pointbase DB Driver.
    // The driver has to be in the classpath.
    Class.forName("org.apache.derby.jdbc.ClientDriver");

}catch (ClassNotFoundException cnfe){
     System.out.println("" + cnfe);
}
```

B2. Lấy ra đối tượng Connection

- Lớp DriverManager chịu trách nhiệm tạo kết nối tới CSDL
  - Sử dụng DataSource là cách hay dùng hơn khi muốn lấy ra một đối tượng connection (trình bày ở phần sau)
- Ví dụ tạo kết nối tới CSDL như sau:

```
try {
     Connection connection = DriverManager.
  getConnection("jdbc:derby://localhost:1527/sample"
  , "app"," app ");
} catch(SQLException sqle) {
     System.out.println("" + sqle);
}
```

DriverManager & Connection

- **java.sql.DriverManager**
  - getConnection(String url, String user, String password) throws SQLException
- **java.sql.Connection**
  - Statement createStatement() throws SQLException
  - void close() throws SQLException
  - void setAutoCommit(boolean b) throws SQLException
  - void commit() throws SQLException
  - void rollback() throws SQLException

B3. Lấy ra đối tượng Statement

- **Tạo một đối tượng Statement từ đối tượng Connection**
  - **java.sql.Statement**
    - ResultSet executeQuery(string sql)
    - int executeUpdate(String sql)
  - **Ví dụ:**
    - Statement statement = connection.createStatement();
- **Cùng đối tượng Statement có thể được dùng cho nhiều queries không liên quan tới nhau**

B4. Thực thi các câu truy vấn/các lệnh

- **Từ đối tượng Statement, 2 lệnh được sử dụng nhiều nhất là**
  - **(a) QUERY (SELECT)**
    - ResultSet rs = statement.executeQuery("select * from customer_tbl");
  - **(b) ACTION COMMAND (UPDATE/DELETE)**
    - int iReturnValue = statement.executeUpdate("update manufacture_tbl set name = 'IBM' where mfr_num = 19985678");

B5. Xử lý kết quả nhận về

- **Duyệt trên ResultSet để xử lý thông tin**
  - **java.sql.ResultSet**
    - boolean next()
    - xxx getXxx(int columnNumber)
    - xxx getXxx(String columnName)
    - void close()
- **Đầu tiên, con trỏ lặp nằm ở trước hàng đầu tiên**
  - **LTV cần gọi phương thức next() để chuyển con trỏ đến hàng đầu tiên**

B5. Xử lý kết quả nhận về (2)

- **Khi đã có ResultSet, LTV dễ dàng xử lý dữ liệu**
  - **Lưu ý: Chỉ số của ResultSet bắt đầu từ 1**

```
while (rs.next()){
    // Wrong this will generate an error
    String value0 = rs.getString(0);

    // Correct!
    String value1 = rs.getString(1);
    int    value2 = rs.getInt(2);
    int    value3 = rs.getInt("ADDR_LN1");
}
```

B5. Xử lý kết quả nhận về (3)

- **Muốn lấy dữ liệu từ ResultSet, sử dụng phương thức getXXX() cho phù hợp**
  - getString()
  - getInt()
  - getDouble()
  - getObject()
- **Mỗi kiểu dữ liệu trong java.sql.Types, đều có phương thức getXXX tương ứng**

B6. Đọc metadata của ResultSet và metadata của CSDL (tùy chọn)

- **Khi đã có đối tượng ResultSet hoặc Connection, LTV có thể lấy về metadata của CSDL hoặc của câu truy vấn**
- **→ Đem lại thông tin hữu ích về dữ liệu lấy về, hoặc về CSDL đang sử dụng**
  - **ResultSetMetaData rsMeta = rs.getMetaData();**
  - **DatabaseMetaData dbmetadata = connection.getMetaData();**
    - Có khoảng 150 phương thức trong lớp DatabaseMetaData

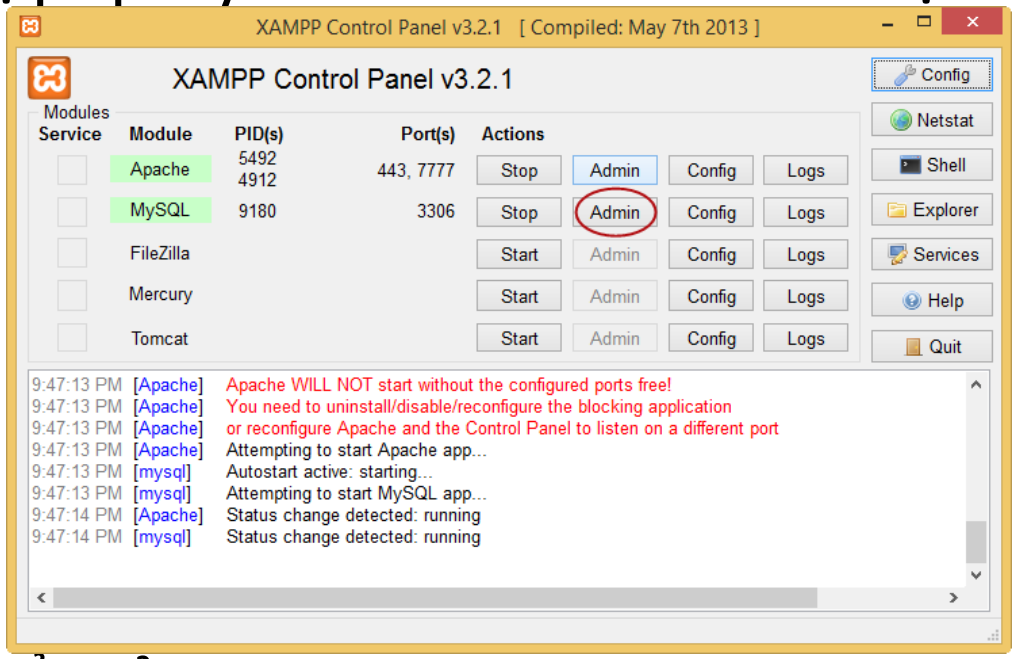
Ví dụ về ResultSetMetaData

```
ResultSetMetaData meta = rs.getMetaData();
//Return the column count
int iColumnCount = meta.getColumnCount();

for (int i =1 ; i <= iColumnCount ; i++){
  System.out.println("Column Name: " + meta.getColumnName(i));
  System.out.println("Column Type" + meta.getColumnType(i));
  System.out.println("Display Size: " +
    meta.getColumnDisplaySize(i) );
  System.out.println("Precision: " + meta.getPrecision(i));
  System.out.println("Scale: " +  meta.getScale(i) );
}
```

Nội dung thực hành

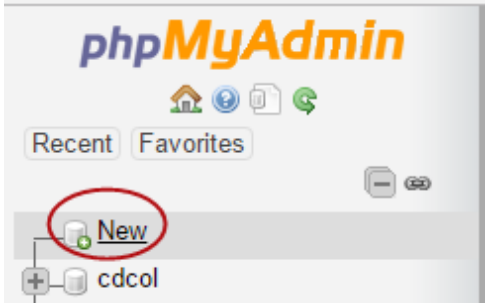
- **Bước 1:** Khởi động XAMPP Control Panel → Start các dịch vụ Apache và MySQL → chọn Admin để mở công cụ quản trị phpMyAdmin cho cơ sở dữ liệu MySQL

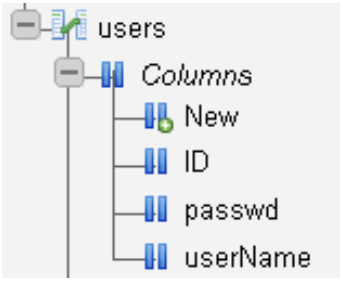


- Hoặc có thể mở trình duyệt và truy cập vào: http://localhost/phpmyadmin

Nội dung thực hành

- **Bước 2:** Tạo cơ sở dữ liệu
  - Tại trang quản trị phpMyAdmin → chọn New
  - Nhập tên database và chọn Create (nhớ chọn Collation là utf8_unicode_ci)
  - Tạo một bảng dữ liệu và nhập các dữ liệu mẫu vào bảng

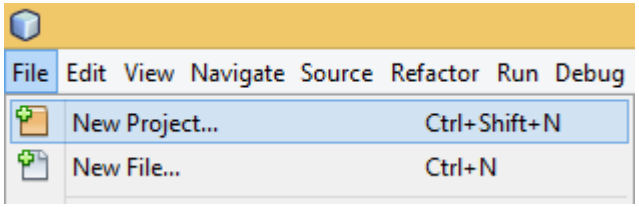


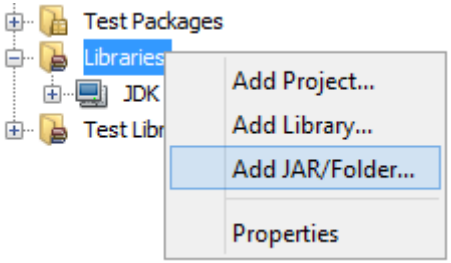


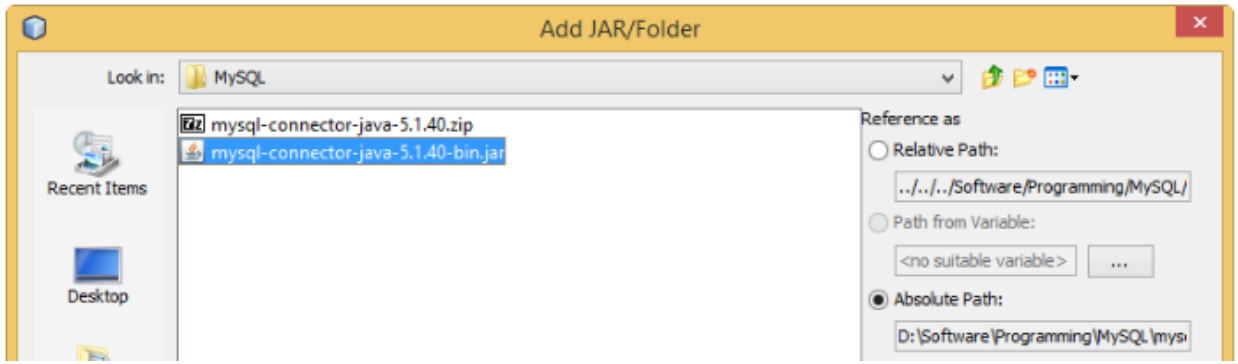
Name	Type	Collation	Attributes	Null	Default	Comments	Extra
ID	int(11)			No	*None*		AUTO_INCREMENT
userName	varchar(100)	utf8_unicode_ci		Yes	*NULL*		
passwd	varchar(100)	utf8_unicode_ci		Yes	*NULL*		

Nội dung thực hành

- **Bước 3:** Xây dựng ứng dụng Java kết nối với MySQL
  - Khởi động Netbeans -> File -> New Project …
  - Tải JDBC Driver cho MySQL và thêm vào project (https://dev.mysql.com/downloads/file/?id=465644)
  - Chọn tập tin jar -> chọn Open







Nội dung thực hành

- **Bước 3:** Xây dựng ứng dụng Java kết nối với MySQL
  - Viết code để truy vấn dữ liệu từ bảng trong cơ sở dữ liệu và hiển thị ra console

```
Connection conn = null;
Statement st = null;
ResultSet rs = null;
try {
    String dbURL = "jdbc:mysql://localhost/tên_database";
    String username = "root";
    String password = "";
    //Tạo kết nối
    conn = DriverManager.getConnection(dbURL, username, password);
    if (conn != null) {
        System.out.println("Kết nối thành công");
    }

    // Câu lệnh xem dữ liệu
    String sql = "select * from tên_bảng";
    // Tạo đối tượng thực thi câu lệnh Select
    st = conn.createStatement();
    // Thực thi
    rs = st.executeQuery(sql);

    // Nếu không có dữ liệu trong bảng
    if (rs.isBeforeFirst() == false) {
        JOptionPane.showMessageDialog(this, "Bảng không có dữ liệu!");
        return;
    }
    // Xử lý dữ liệu
    while (rs.next()) {
        //Sử dụng phương thức rs.getXX("Tên cột trong bảng")
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```

Nội dung bài tập tự làm

- **Thực hành xây dựng chương trình java với các lệnh SQL cơ bản (SELECT, INSERT, UPDATE, DELETE)**