Fast-NU-ISB Spring 2022

Lab 11 –Protected Variation- Handling Persistence

In this lab we will learn about how to connect and retrieve information from MySQL Server Database and display in Console. We shall use Console for simplicity. In short this is what we do:

1. Install and run XAMP server to get server environment on local system
2. Download mysql connector jar file
3. Create our Java Console App in Eclipse.
4. Connect To Database
5. Retrieve Data using sql queries
6. Display data in Console

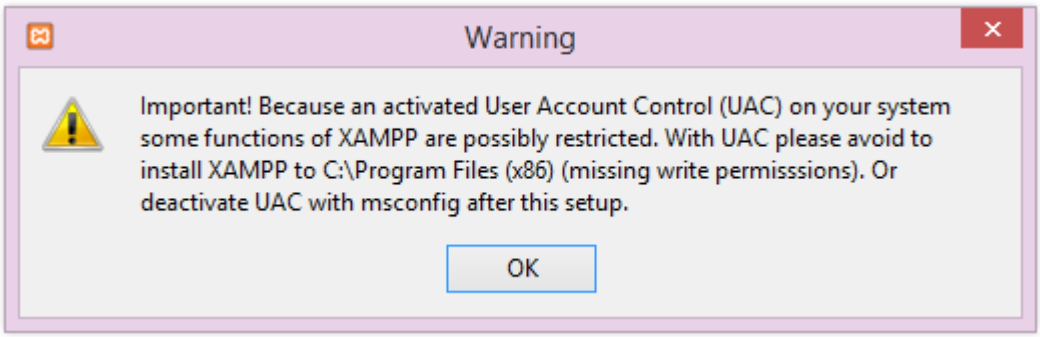
Install Xampp localhost server in windows 7,8.1,10

Xampp is one of the open source softwares freely distributed by **apachefriends.org to** develop php web applications. Xampp provides us the server environment on our local system.

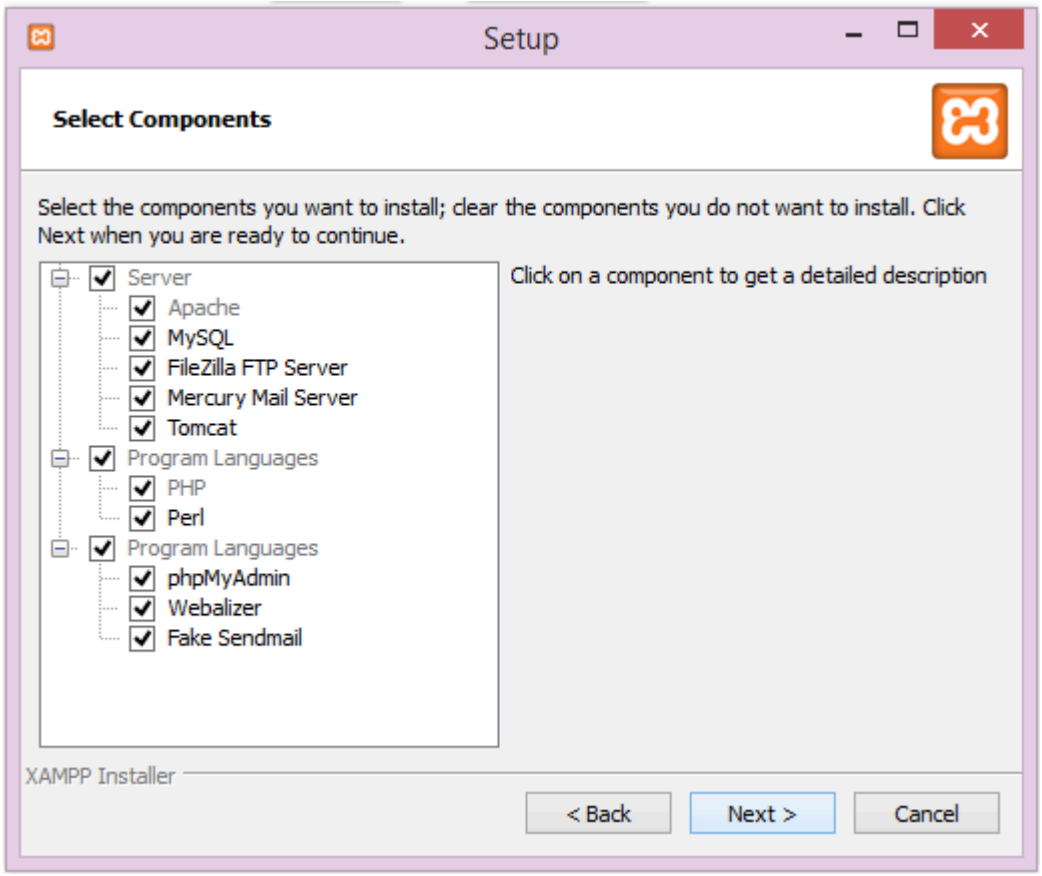
1. Download Xampp from its official website apachefriends.org

2. Double click on your downloaded setup to run it.

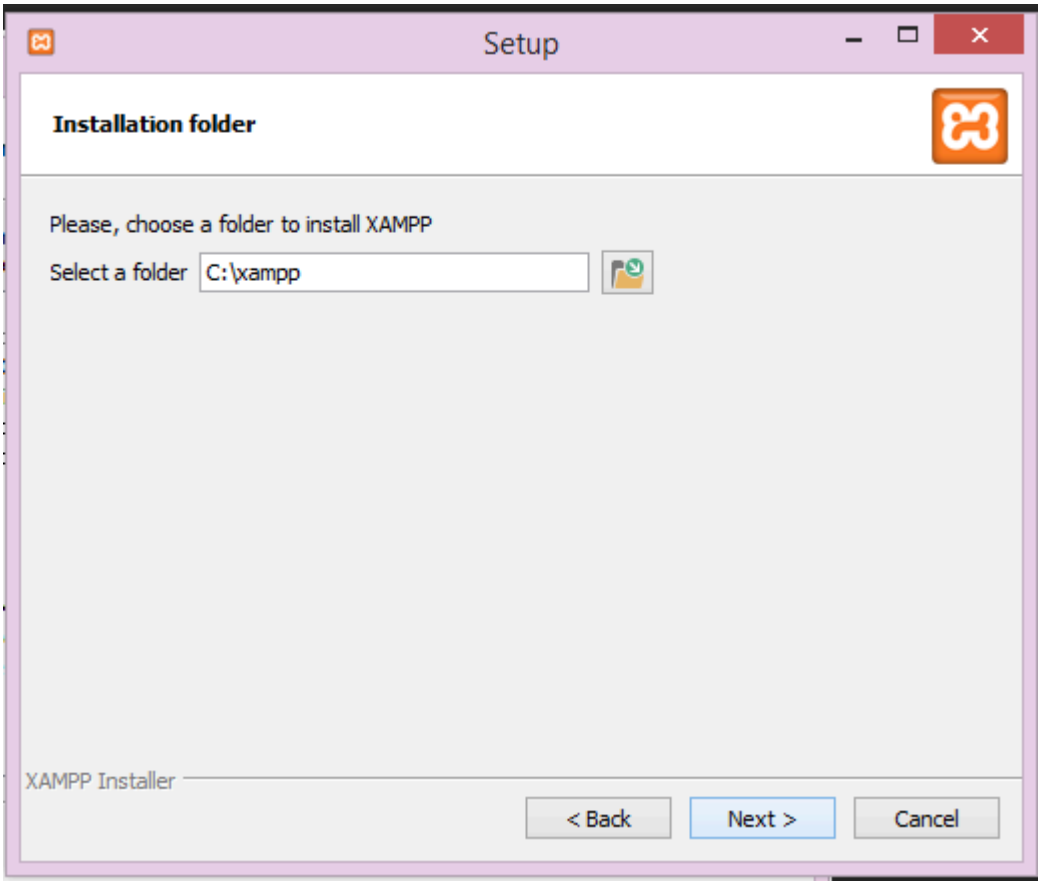
3. Now it will show you an important notice. Clock OK to continue



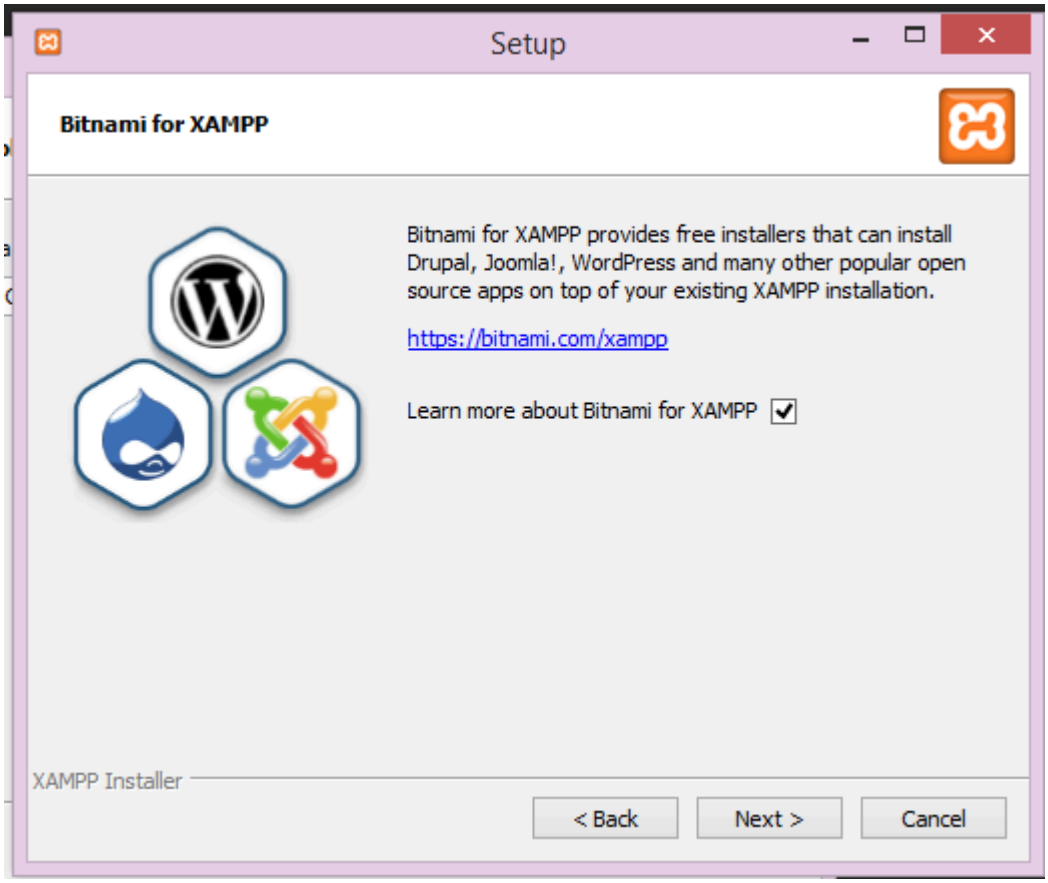
4. Select all components (Server- Apache, MySQL, FileZilla FTP Server, Mercury Mail server, Tomcat | Program Languages- PHP, Perl | Program Languages- phpMyAdmin, Webalizer, Fake Sendmail.) Click on next button



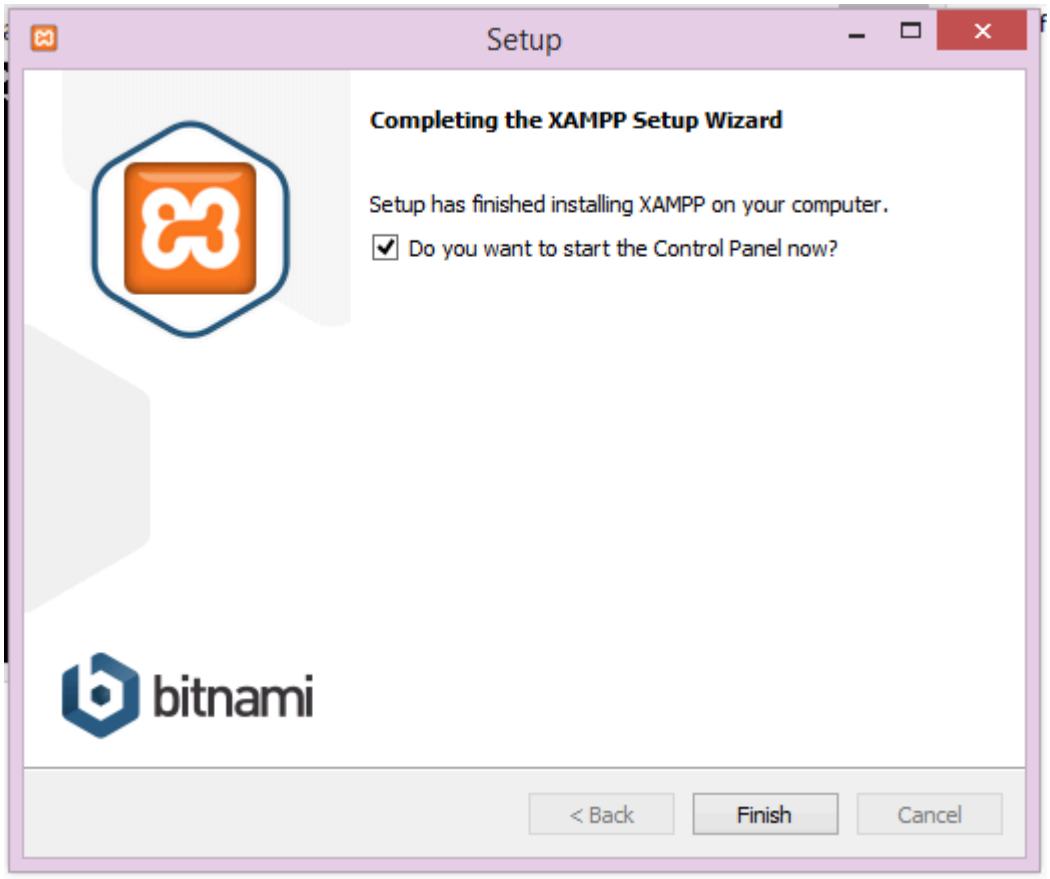
5. Select Xampp installation folder By default its installs into **C:\xampp .** Click on Next button



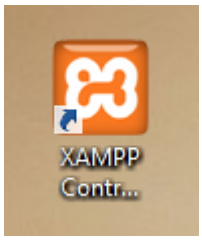
7. Now on the next installation window it will show a notification. Click Next to install



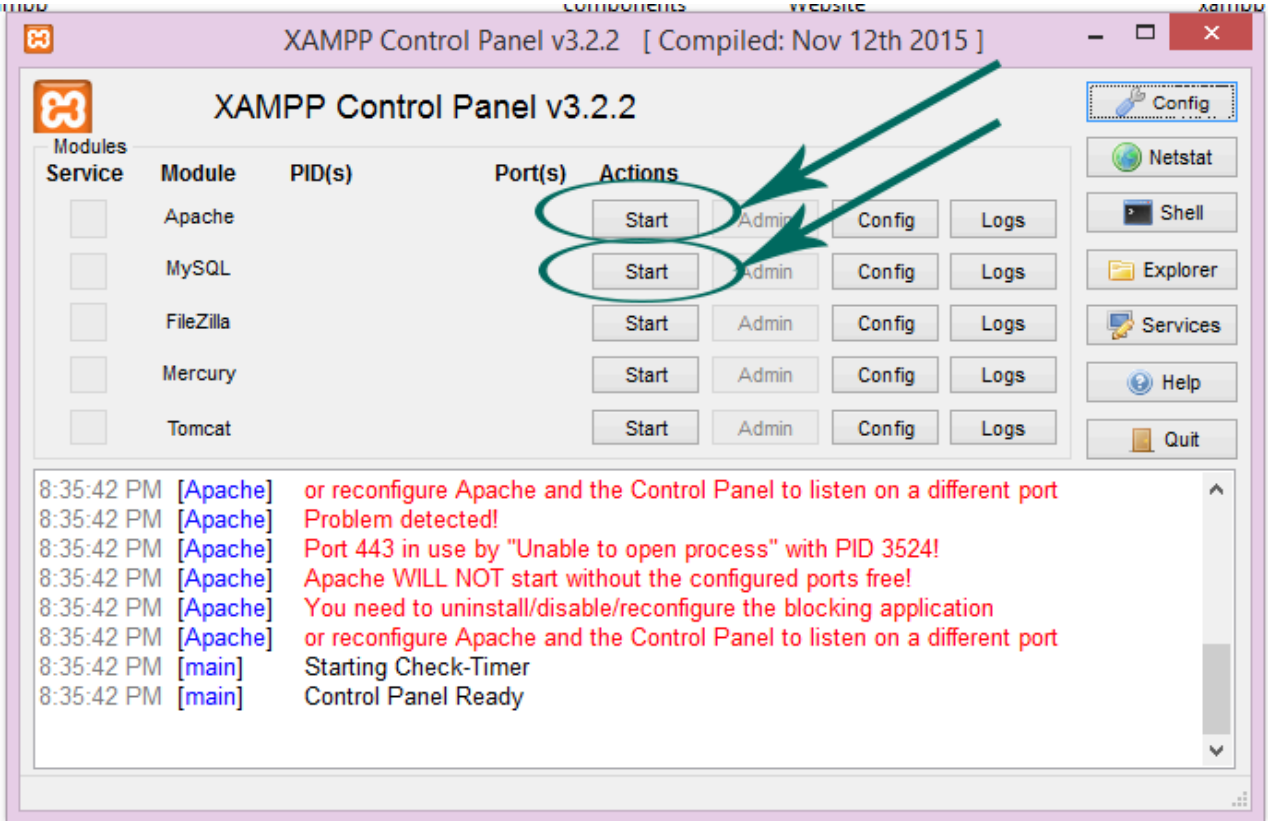
8. After done installation it will show Completing the Xampp setup wizard window



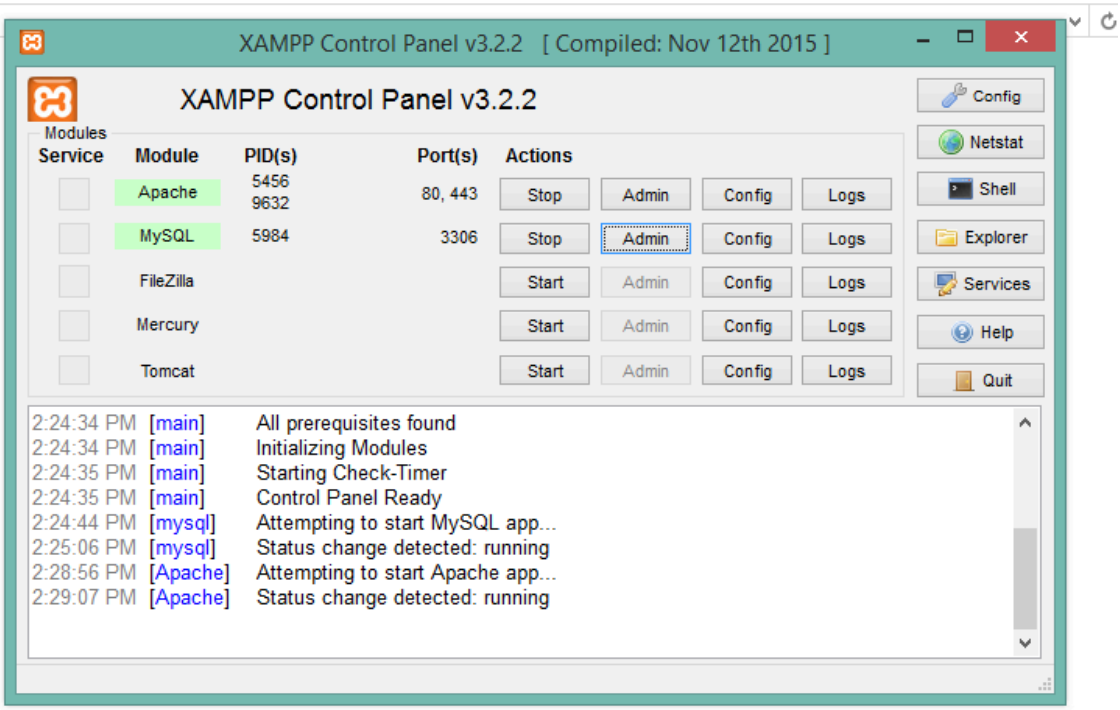
9. Now start XAMPP using xampp control panel icon which is automatically visible on your desktop screen

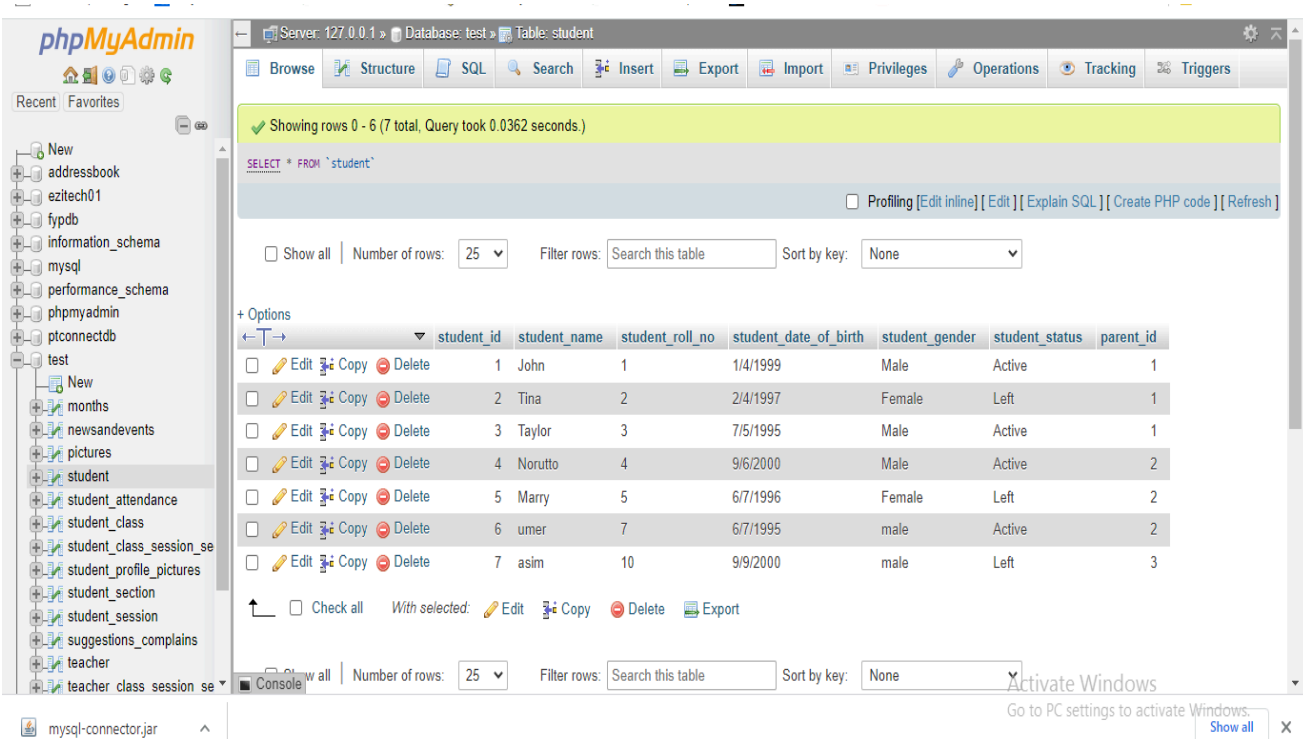


10. Double click on Xampp control panel icon to run it and it will show you its complete control panel window. **Click on Apache start button and MySQL start button to start them.**



11. Click on Admin button next to MySQL to start localhost phpmyadmin where database can be managed

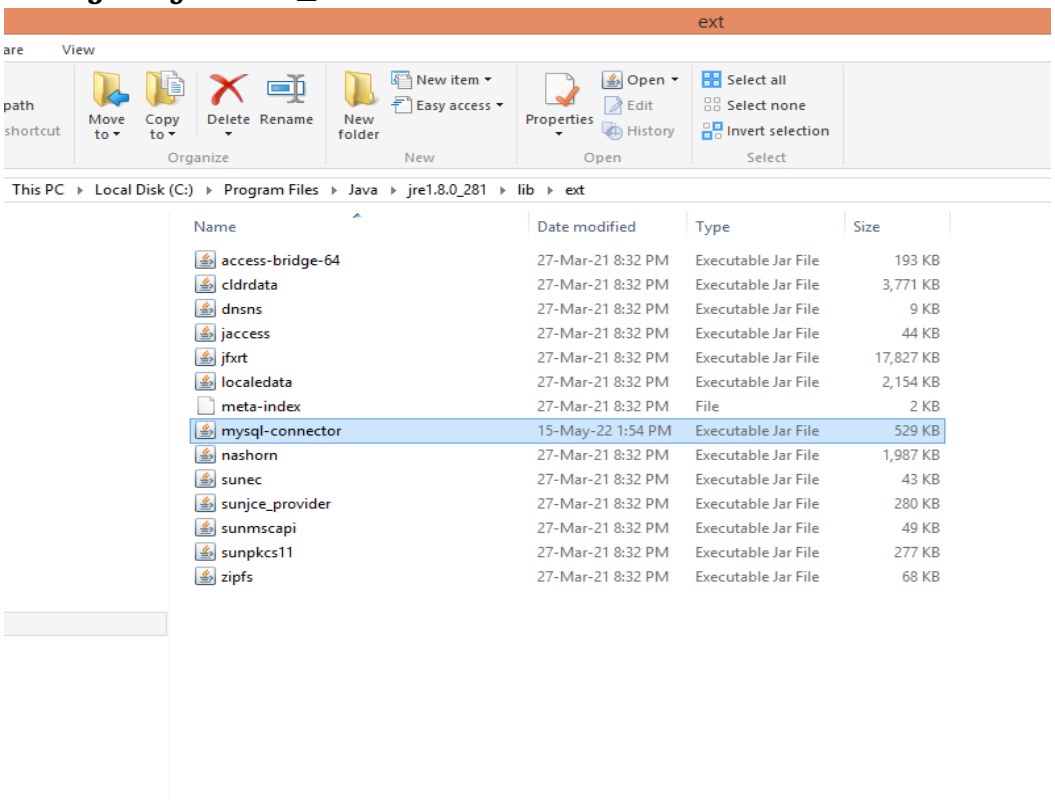




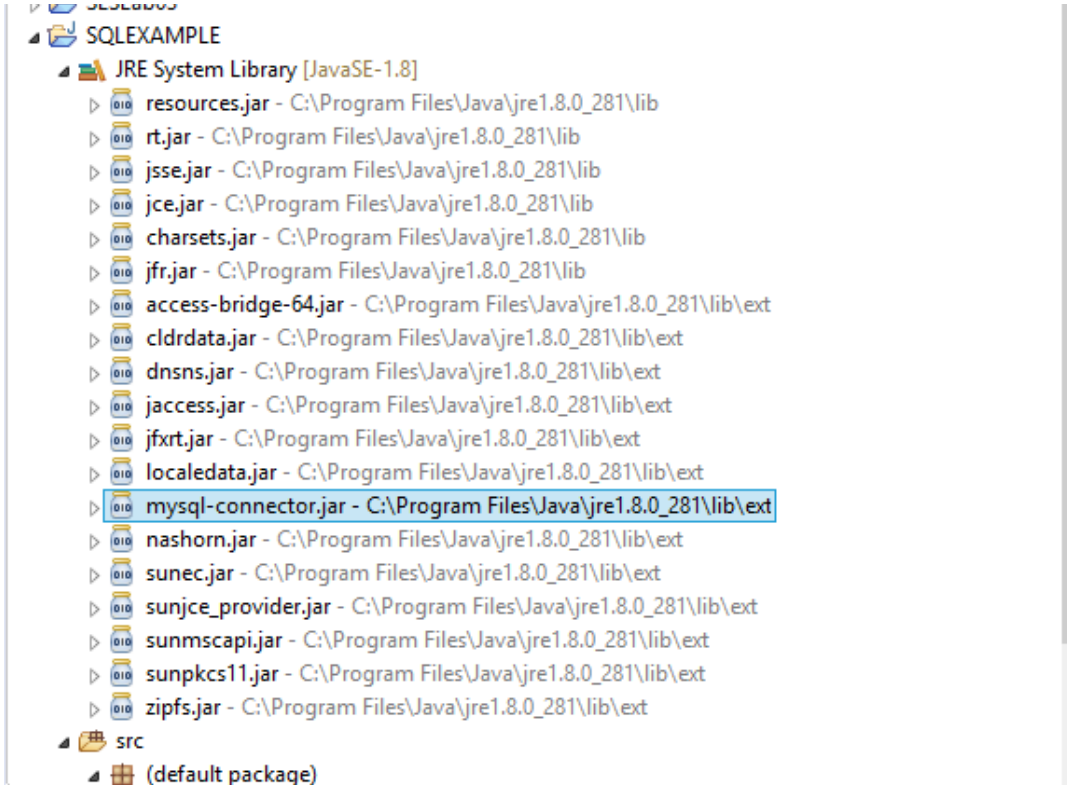
Connect Eclipse IDE with MySQL Server using mysql Connector

1. Download mysql-connector.jar file from the link https://goo.gl/ftjWmK

2. Copy the downloaded jar file and paste it to **C:\Program Files\Java\jre1.8.0_281\lib\ext** folder



3. Open Eclipse IDE and create a new java project, you will be able to see the jar file inside JRE Library folder



4. Create an interface `PersistenceHandler` to handle connection with storage. This interface should have different abstract methods for Create, Read, Update and Delete operations

```
public interface PersistenceHandler {


   abstract void connectDB();
   abstract void saveRecord();
   abstract void updateRecord();
   abstract void deleteRecord();
   abstract void readRecord();
}
```

5. Create concrete class of DBHandler that should implement the PersistenceHandler interface to handle connection with database. One can also create other sub classes to create connection with different other storage elements such as fileHandler class but all those should implement the interface to achieve polymorphic behavior of CRUD operations during runtime.

```
public class DBHandler implements PersistenceHandler{
   Connection con=null;
   @Override
    public void connectDB() {
          // TODO Auto-generated method stub

          }
   @Override
    public void readRecord() {
          // TODO Auto-generated method stub

          }
   @Override
    public void saveRecord() {
          // TODO Auto-generated method stub

    }
    @Override
    public void updateRecord() {
          // TODO Auto-generated method stub

    }
    @Override
    public void deleteRecord() {
          // TODO Auto-generated method stub

     }
```

6. The `DBHandler` class should modify the method connectDB that creates connection with mysql server. The class must import java.sql libraries.

```
import java.sql.*;

public class DBHandler implements PersistenceHandler{
  Connection con=null;
  public void connectDB() {
        try {
              Class.forName("com.mysql.jdbc.Driver");

                  con=DriverManager.getConnection("jdbc:mysql://local
                  host:3306/test");
                  if(con == null) {
                    System.out.println("DB connection failed");}
                  else
                    System.out.println("DB connection successful");


        }
        catch(Exception e) {
              System.out.println("exception: "+e);
        }
```

```
  }
}
```

7. Modify the readRecord method to read all male students from database. Add line con.close() inside the function to close connection with database after each operation.

```
public void readRecord() {
      Statement stmt;
      try {
      stmt = con.createStatement();
String sql="Select * from student where student_gender='Male'";
      ResultSet rs= stmt.executeQuery(sql);
      while(rs.next()) {

System.out.println("-----------------------------------------
-----------------------------------------------------------
----------------------------------------");
      System.out.println(rs.getInt(1)+"\t| "+rs.getString(2)+"\t|
      "+rs.getString(3)+"\t| "+rs.getString(4)+"\t|
      "+rs.getString(5)+"\t| "+rs.getString(6));

System.out.println("-----------------------------------------
-----------------------------------------------------------
----------------------------------------");
      }
      con.close();
            }
       catch (SQLException e) {

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

8. Create a Main class to call appropriate handler based on user's choice and perform operations on data.

```
import java.util.Scanner;

public class Main {
  Scanner sc;
  public static void main(String[] args) {

      PersistenceHandler handler;
      int choice = 0;
      Scanner sc= new Scanner(System.in); //System.in is a
      standard input stream.
```

```
            System.out.print("Chose storage option (0 for file and 1
            for database)- ");
            int a= sc.nextInt();

            switch(choice) {
                  case 1:
                        handler=new DBHandler();
                        break;
                  default:
                        System.out.print("No other storage is available
                        right now");
            }

            handler=new DBHandler();
            handler.connectDB();
            handler.readRecord();

      }}
```

How to do basic database operations (CRUD - Create, Retrieve, Update and Delete) using JDBC (Java Database Connectivity) API. These CRUD operations are equivalent to the INSERT, SELECT, UPDATE and DELETE statements in SQL language.

JDBC Execute CREATE Statement Example

```
 CREATE TABLE `users` (
    `user_id` int(11) NOT NULL AUTO_INCREMENT,
    `username` varchar(45) NOT NULL,
    `password` varchar(45) NOT NULL,
    `fullname` varchar(45) NOT NULL,
    `email` varchar(45) NOT NULL,
    PRIMARY KEY (`user_id`)
);
```

JDBC Execute INSERT Statement Example

```
String sql = "INSERT INTO Users (username, password, fullname, email) VALUES (?,
 ?, ?, ?)";

PreparedStatement statement = conn.prepareStatement(sql);
statement.setString(1, "bill");
statement.setString(2, "secretpass");
statement.setString(3, "Bill Gates");
statement.setString(4, "bill.gates@microsoft.com");

int rowsInserted = statement.executeUpdate();
if (rowsInserted > 0) {
```

```
    System.out.println("A new user was inserted successfully!");
}
```

JDBC Execute SELECT Statement Example

```
String sql = "SELECT * FROM Users";

Statement statement = conn.createStatement();
ResultSet result = statement.executeQuery(sql);


int count = 0;

while (result.next()){
    String name = result.getString(2);
    String pass = result.getString(3);
    String fullname = result.getString("fullname");
    String email = result.getString("email");

    String output = "User #%d: %s - %s - %s - %s";
    System.out.println(String.format(output, ++count, name, pass, fullname,
email));
}
```

JDBC Executing UPDATE Statement Example

```
String sql = "UPDATE Users SET password=?, fullname=?, email=? WHERE username=?";

PreparedStatement statement = conn.prepareStatement(sql);
statement.setString(1, "123456789");
statement.setString(2, "William Henry Bill Gates");
statement.setString(3, "bill.gates@microsoft.com");
statement.setString(4, "bill");

int rowsUpdated = statement.executeUpdate();
if (rowsUpdated > 0) {
    System.out.println("An existing user was updated successfully!");
}
```

JDBC Execute DELETE Statement Example

```
String sql = "DELETE FROM Users WHERE username=?";

PreparedStatement statement = conn.prepareStatement(sql);
statement.setString(1, "bill");

int rowsDeleted = statement.executeUpdate();
if (rowsDeleted > 0) {
    System.out.println("A user was deleted successfully!");
}
```

Tasks

A departmental store needs a software to manage its business. The company have many products under different categories. SalesDB is a database which stores all records of sales, products, payment etc.

The **product** table has following view inside database:

name	category	manufacturing_year	price	status	company	size	color
Audio Technica ATH	Headphones	2018	300	active	Sound World		black
Polk BOOM	Bluetooth speakers	2022	1200	available	Sound World		white
Safari water bottle	crockery	2019	800	available	WEST Point	small	blue
Cookies cracker	biscuit	2022	50	available	Magnet	Half role	
Mars	candle	2019	20	available	Light Palace	small	multi
Glow	Lamp	2020	1500	Available	Light Palace	medium	purple

Panadol	medicine	2017	20	available	Glaxo		

Create a java application that should be able to:

SMB extension	electronics	2017	500	available	E masters	medium	green

- Save a new product each time the stock gets updated.
- Search some product by name, category, price limit etc.
- change the product status to expired for all products having category medicine and manufactured before 2018

Suppose the store decides to remove the crockery section, the system should delete all related products from database also.