

|  |  |
| --- | --- |
| **Unit Code** | CSC 3140 |
| **Unit Title** | **Advanced Programming Techniques** |
| **Semester** | Spring 2023 |

Student Id:

Table of Contents

[Introduction 3](#_Toc132206662)

[Problem 1 4](#_Toc132206663)

[Analysis 5](#_Toc132206664)

[Implementation and Explanation 5](#_Toc132206665)

[Results: 12](#_Toc132206666)

[Problem 2 13](#_Toc132206667)

[Analysis 13](#_Toc132206668)

[Implementation and Explanations 13](#_Toc132206669)

[Results 16](#_Toc132206670)

# Introduction

In this project, we were given two challenges to analyze, code, and test using HTML, Java, and SQL, and doing the same task on servlets as well. The report below will illustrate the step-by-step effort, including the assumptions used to complete each job with complete code and result outputs. For this assignment I have used

* Visual studio code
* MySQL

# Problem 1

Employee Database Management System using SQL and Java

Design and implement an Employee Database Management System using SQL and Java programming language. The system should allow users to *add*, *modify*, *delete*, and *search* employee records.

Perform the following tasks:

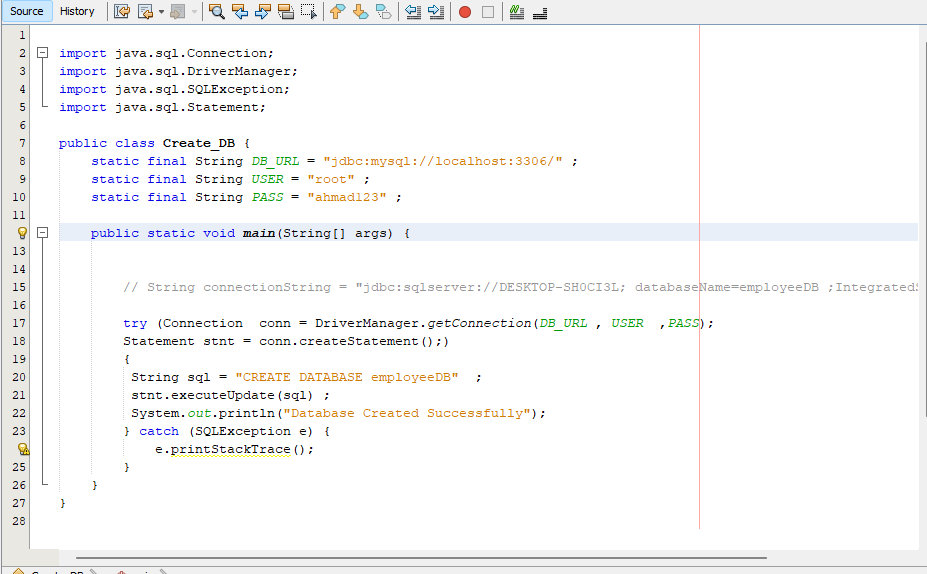
1. Create a Database called “employeedb” in MySQL Workbench.

* 1. 2. Create a table called “employee” in employeedb with columns as below: a) firstname (varchar)
  2. b) midname (varchar)
  3. c) lastname (varchar)
  4. d) phone (varchar)
  5. e) country (varchar)
  6. f) department (varchar)
  7. g) jobtitle (varchar)
  8. h) salary (double)
  9. 3. Insert a test dataset of employee records.
  10. 4. Implement a Java program that connects to the database and tests the functionality for adding and modifying employee records.
  11. 5. Implement delete functionality that deletes an employee based on a given employee's last name.
  12. 6. Implement a search functionality that allows the user to search for employees based on specific criteria, such as phone number.

## Analysis

We are assigned the responsibility of creating a database using MySQL and creating a table of employees with specific properties. Following the database, we would need to test the database using Java. For this task I have used java for the complete process and in the end combine my work in one file to test all the functions at once.

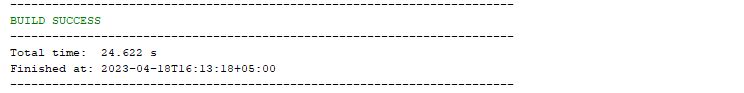
## Implementation and Explanation



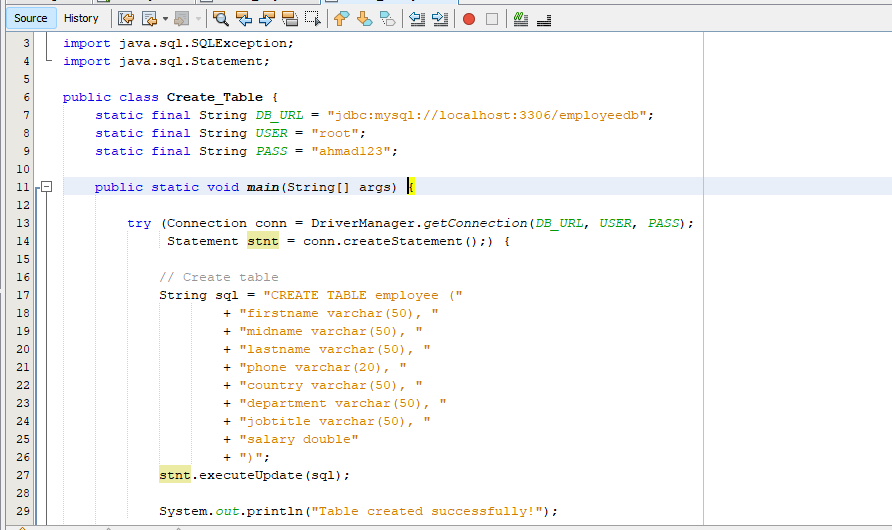
This code connects the database using java and we can see that out database empoyeeDB has been created successfully.

* The import lines at the start of the program import the classes required for JDBC operations from the java.sql package.
* A main method is created for the Create DB class.
* The variables DB\_URL, USER, and PASS are used to indicate the URL, username, and password required for connecting to the MySQL server.
* A try-with-resources block is used within the main function to immediately exit the database resources once they have been used. The Driver Manager class's get Connection function is used to connect to the MySQL server.
* The create Statement method is used to generate a Statement object from the connection.
* A SQL CREATE DATABASE command is saved in the SQL variable as a string.
* The query object's execute Update function serves to run the SQL query and build the database.
* To show that a database was successfully built, a message is written to the console.
* If there are any issues during the database's creation process, an SQL Exception is thrown and its stack trace is reported to the console.

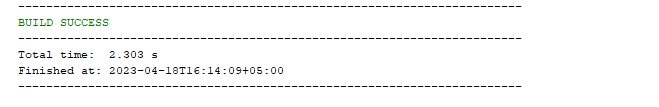
The database file with specific name is created.

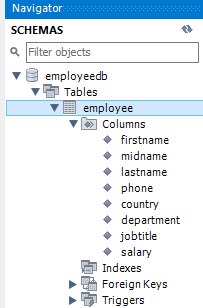


Now the next step is to create the table of employee and insert data inside it. We can do that using the code below.

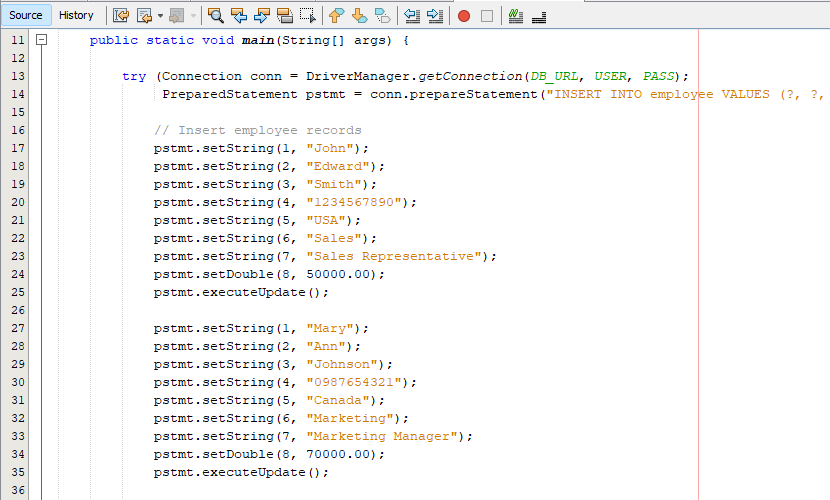


* The DB\_URL constant now refers to the database "employeedb" that was established in MySQL Workbench.
* A SQL CREATE TABLE statement is described as a text containing the columns and data types supplied.
* The executeUpdate method is used to run the SQL query to build the "employeedb" database's "employee" table.
* To show that the table was successfully constructed, a message is displayed to the console.



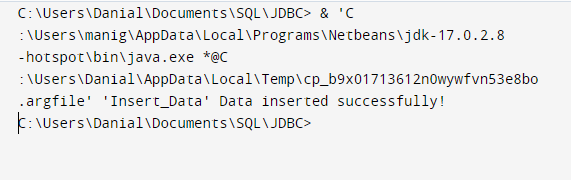


Next up is data insertion

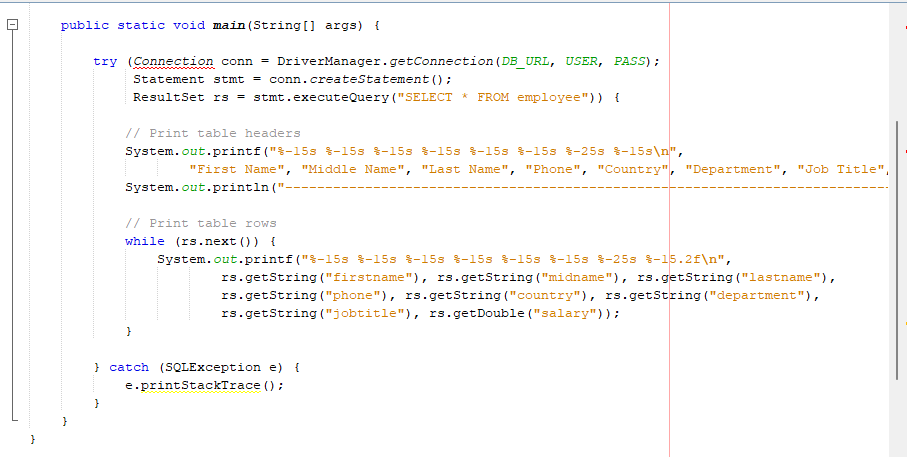


* The DB\_URL constant now refers to the database "employeedb" that was established in MySQL Workbench.
* To insert parameterized data into the "employee" database, a PreparedStatement object is used instead of a Statement object.
* Three employee entries are added into the "employee" database by setting the parameters in the PreparedStatement with the setString and setDouble procedures.

Using the following pattern we can insert random values that we want into the table.

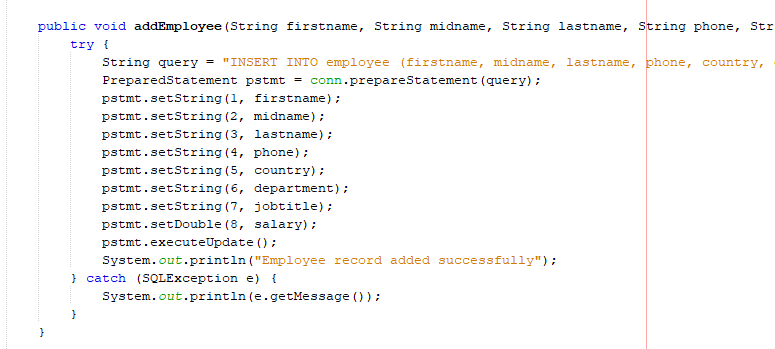


To verify that the table has been updated we can show the table from the code below:



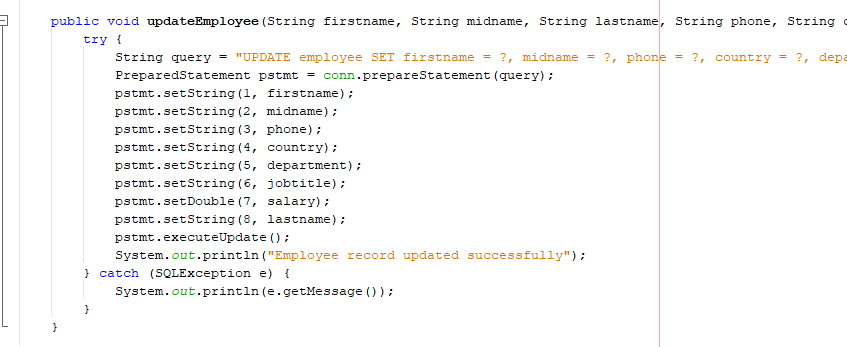
* The result of the SELECT query that obtains all data from the "employee" table is stored in a ResultSet object.
* The printf method is used to print formatted strings to the console, which improves table alignment and readability.
* The values of the columns in each row of the ResultSet are retrieved using the getString and getDouble functions, and these values are written to the terminal using printf.

Let’s Test the remaining functions by putting all the functions to one file:



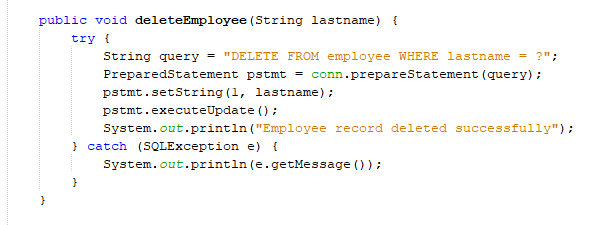
The procedure begins by generating a SQL query string that adds a new record into the employee database using the parameters specified. The conn connection object is then used to build a PreparedStatement object, and the arguments are set using the PreparedStatement object's setString and setDouble methods. Finally, the query is executed and the new employee record is added to the database using the executeUpdate function.

If an error occurs while the SQL query is being executed, the method captures the SQLException and sends the error message to the console. Otherwise, a success message is printed to the console indicating that the employee record was successfully added.



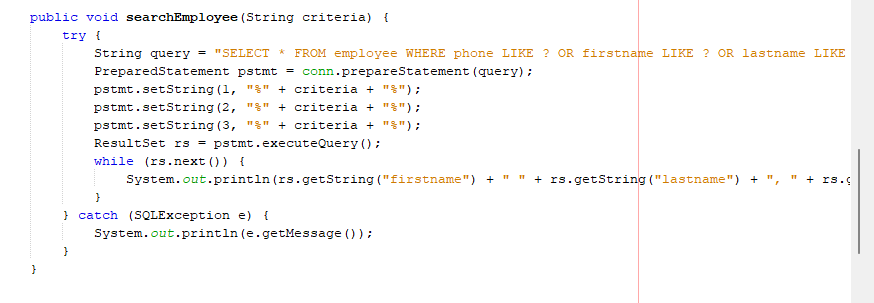
The updateEmployee() function refreshes an employee's information in the database. It accepts the following inputs as arguments: firstname, middlename, lastname, phone, country, department, jobtitle, and pay.

Then, using the change statement, creates a SQL query to change the employee's record in the database table employee. The query assigns the firstname, middlename, phone, country, department, jobtitle, and salary columns in the table to the method's parameters. It specifies which record to update using the WHERE clause, depending on the lastname column matching the lastname input supplied to the function.



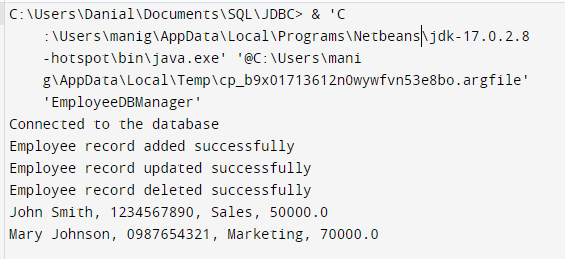
The deleteEmployee method is used to remove an employee record from the linked database's employee table based on the employee's lastname.

This function accepts a lastname input to determine which employee record should be destroyed. It then constructs a PreparedStatement object with a SQL DELETE statement that specifies the employee table and the lastname argument, then uses the setString function to set the value of the lastname field. Finally, the PreparedStatement object's executeUpdate function is used to execute the DELETE statement and delete the employee entry from the database.



The searchEmployee method accepts a string criterion as input and searches the database for employees. The method creates a SQL query to look for workers with the same phone number, first name, or last name as the supplied parameters.The setString function is then used to build a PreparedStatement object and set the criterion parameter to the query placeholders. The query is then executed using the executeQuery method, which provides a ResultSet object containing the rows of the query result. The ResultSet object is then iterated through using the next method, which moves the cursor to the next row and returns true if there are additional rows. The function retrieves column values for each row by utilising the getString and getDouble methods of the ResultSet object.Then the output is shown in the output console.

## Results:



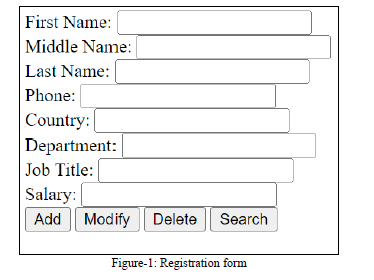
# Problem 2

Re-implement problem-1 by using servlets.

The Servlet performs the following tasks:

1. Use HTML to design a registration form similar to one in the figure-1.

2. Servlet will add, update, delete, or search the employee information.



## Analysis

After completing the problem 1 now our task was to use servlet for the same problem. Servlet is a Java program that runs on a web server to process requests and create answers. It is a method for creating Java web apps. Servlets are a component of the Java EE (Enterprise Edition) platform that offer a powerful, efficient, and adaptable way of producing dynamic web content. Now we will be using servlet to do the above task.

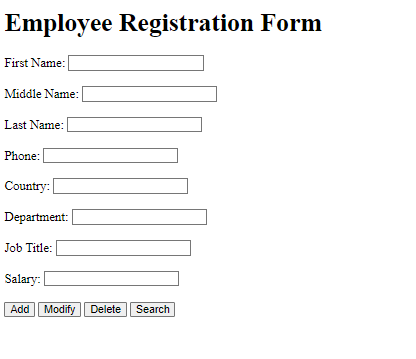
## Implementation and Explanations

We basically need 2 main parts for this

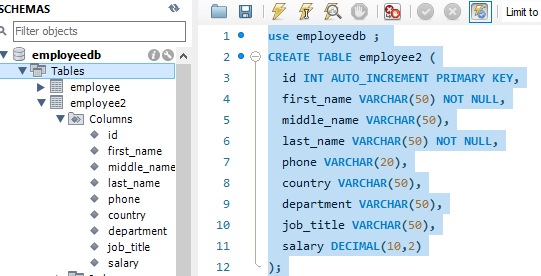
* MySQL
* servlets code

Registration form created using the below html code:

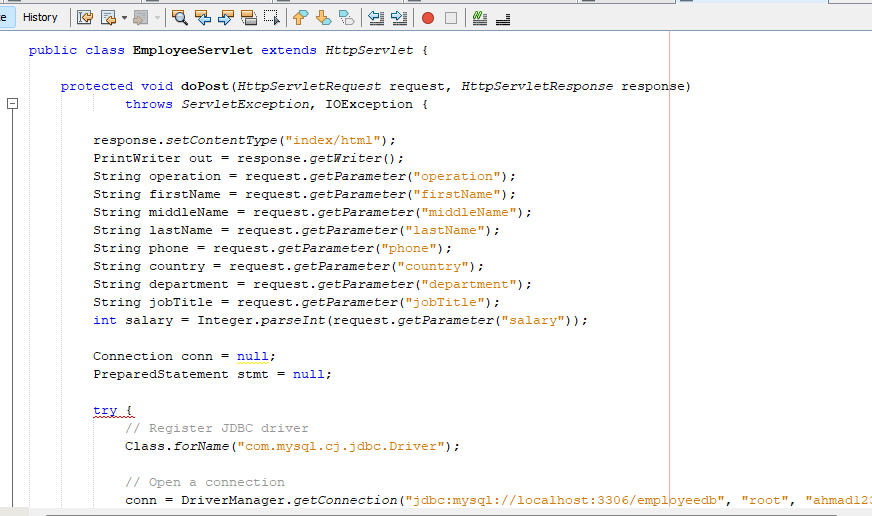




This is our main page. I everything that will be written in this will be handles by do get() method and next the do post will update the SQL database just like before .

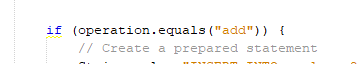


So, we have used the same database but just to separate things out we have used another table names employee2 and we can see these attributes will be affected after running our java servlet program. So, let’s see the java part next.



This method will get the data from the html form index.html and then connect to the database as shown below and make changes in the employee2 table. The form values will be extracted and the values of each form field using the getParameter method of the HttpServletRequest object.

I have used if else to check which button is pressed so that the following action can be performed:



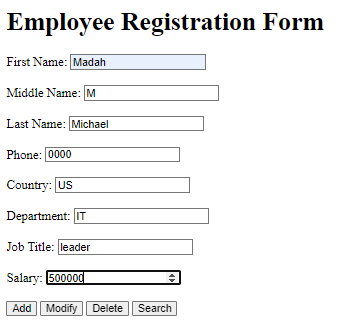




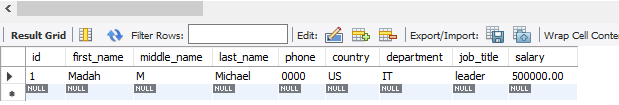


## Results

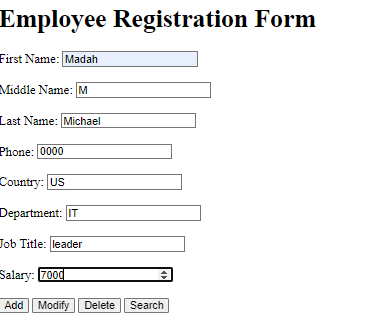
For add :

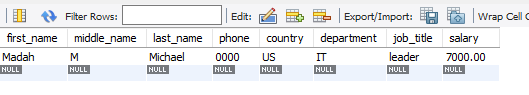


The following will be added to the table in the employee2 table:

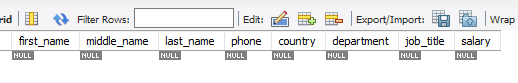


For modify:





For delete:



So, we have seen the changes in the database using servlet java html.