Experiment#			
Date		Student ID	
		Student Name	

17. Java Database Connectivity

Aim/Objective: To understand the how a Java application can be connected with any database in the world for data persistence.

Description: The student will understand the concepts of JDBC.

Pre-Requisites: Classes, Objects and database

Tools: Eclipse IDE for Enterprise Java and Web Developers

Pre-Lab:

1) Explain the limitations of file system making us to use database?

① Redundancy & Inconsistency: File systems can't prevent duplicate or conflicting data, unlike data bases

② Data Integrity: Databases enforces rules to ensure data accuracy; file systems don't

③ Concurrency Issues: File systems struggle with multi-user access, risking data conflicts.

④ Limited Security: File system offer basic permissions, data bases provide detailed access control.

Experiment#		Student ID	
Date		Student Name	

2) Explain the concept of JDBC and its role in Java database connectivity. What are the main steps involved in performing database operations using JDBC?

JDBC concept & Role : JDBC in JAVA's API for connecting to databases, allowing java programs to excute SQL queries and interact with relational databases

main steps in JDBC :-

(1) Load Drivers : Load the specific database drivers
(2) Establish connection: prepare SQL statements for execution.
(3) Create state: Connect todatabase using Driver Manager
(4) close connections: Properly close ResultSet, Statement, and connection.

Experiment#		Student ID	
Date		Student Name	

In-Lab:

1) You are developing a Student Information Management System where student details such as student ID, name, address, and program can be managed through a web interface. Implement a servlet called StudentDetailsServlet that accepts student details through a form submission and stores these details in a database using JDBC. Perform the following steps to create and manage the student database:

Requirements

A. **Create the Student Database**:
   a. Write a JDBC program to create a Student database.
B. **Create the Registration Table**:
   a. Write a JDBC program to create a Registration table inside the Student database with fields id, name, address, and program, where id is the primary key.
C. **Insert Records**:
   a. Write a JDBC program to insert four records into the Registration table.
D. **Display Records**:
   a. Write a JDBC program to display the records inserted into the Registration table.
E. **Update Records**:
   a. Write a JDBC program to update the program of students whose id values are 100 and 101.
F. **Delete Records**:
   a. Write a JDBC program to delete the student record whose id is 101.

Procedure/Program:

```
import java.sql

Public class studentDatabaseManager{
Private static final String URL = "jbbc:mysql:
        //localhost:3306/student";
Private static final String USER: "root";
Private static final PASS = "Password";
```

```
Public static void main(String[] args) {
  try {
    createDatabase(),
    createTable(),
    insertRecords(),
    displayRecords();
    updateRecords();
    deleteRecords();
    displayRecords();
  } catch (Exception e) {
      e.printStackTrace();
  }
}
Public static void createDatabase() throws
    Exception {
try (Connection connection = DriverManager.getConnect
      "jdbc:mysql://localhost:3306/", USER,
                                      PASS),
    Statement.executeUpdate("CREATE DATABASE
      IF NOT EXISTS student");
    System.out.println("Database 'student'
      created.");
}
```

Experiment#		Student ID	
Date		Student Name	

```
public static void createTable() throws Exception{
try{Connection connection=DriverManager.get
        Connection(URL,USER,PASS);
Statement statement=connection.createStatement();
statement.executeUpdate("CREATE TABLE
        IF NOT EXISTS Registration ("+
    "id INT PRIMARY KEY,"+
    "name VARCHAR(50),"+
    "address VARCHAR(100),"+
    "program VARCHAR(50)]");
System.out.println("Table 'Registration'
        created.");
}}

Public static void insertARecord() throws Exception{
try(Connection connection=DriverManager.get
        Connection(URL,USER,PASS);
Prepared Statement preparedStatement=
        Connection.prepareStatement("
INSERT INTO Registration(id, name,
address, program) values(?,?)
```

```
object[][] data = {
    {100, "Alice", "123 Maple St", "Computer Science"},
    {101, "Bob", "456 Oak Ave", "Information Technology"},
    {102, "Charlie", "789 Pine Rd", "Software Engineering"},
    {103, "Daisy", "101 Cedar Blvd", "Data Science"},
};
for (Object[] record : data) {
Prepared Statement.setInt(1, (int) record[0]);
Prepared Statement.setString(2, (String) record[1]);
Prepared Statement.setString(3, (String) record[2]);
Prepared Statement.setString(4, (String) record[3]);
        Prepared Statement.addBatch();
    }
System.out.println("Records Inserted");
}
```

```
Public static void updateRecords() throws
Exception {
try {Connection connection = DriverManager.
getConnection(URL, USER, PASS);
Prepared statement preparedStatement =
connection.prepareStatement("UPdate
Registration SET program = ? WHERE id = ?");
Object[][] updates={
{"Cybersecurity", 102},
{"Artifical Intgeligence",101}
};
for(Object[] update : updates){
Prepared Statement.setString.setString(1,
(String) update[0]);
Prepared Statement: setInt(2, (Int)
update[1]);
Prepared Statement.addBatch();
}
System.out.Println("Records
Updated.");
```

Experiment#		Student ID	
Date		Student Name	

```
Public static void deleteRecord() throws
        Exception {
try (Connection connection = DriverManager.get
        Connection (URL, USER, PASS);
Prepared Statement preparedStatement =
        Connection.preparedStatement (
        "DELETE FROM Registration WHERE id=?")) {
        Prepared Statement.setInt (1, id);
        Prepared Statement.executeUpdate();
            System.out.println ("Record deleted");
}}}
```

Experiment#		Student ID	
Date		Student Name	

✓ **Data and Results:**

The database was successfully created with a "Registration" table, and records was inserted, displayed, updated, and deleted as required

✓ **Analysis and Inferences:**

Each JDBC operation (create, insert, update, delete) executed successfully. demons. trating effective database connectivity & manipulation using JDBC in JAVA

Experiment#			
Date		Student ID	
		Student Name	

VIVA-VOCE Questions (In-Lab):

1) **What is the role of JDBC in Java database connectivity?**

JDBC Role: Enables java applications to interact with databases.

2) **What are the different types of JDBC drivers? Explain their differences and advantages.**

JDBC Drivers: Types 1-4 (ODBC Bridge, Native-API, Network protocal, Thin Drivers); differs in efficiantly, portability and dependency.

Experiment#		Student ID	
Date		Student Name	

3) How do you establish a database connection using JDBC?

Database Connection: Use DriverManager.
getConnection() with URL, username, &
Password.

4) Explain the difference between Statement and PreparedStatement in JDBC.

Statement: Statement for simple quires
Proposed Statement: proposed statement for
Parameterized, efficient, secure queries

Experiment#			
Date		Student ID	
		Student Name	

5) How do you handle exceptions related to database operations in JDBC? What are some common JDBC-related exceptions?

Use try-catch for SQLException, handle
common errors like constraints
violation and Syntax issues.

Post-Lab:

1) Create a JDBC program to connect to a PostgreSQL database. Specifically, you need to create a "test" database in the PostgreSQL server and write a JDBC program that checks whether the connection to the database is successful. If the connection is successful, the program should display the message "Connected to the database". If the username or password is incorrect, the program should display "Invalid username or password".

Procedure/Program:

```
import java.sql.*;
Public Class Test Connection {
    Public static void main (String() args) {
String url = "jdbc : postgresql://local
    host:5432/test";
String user = "your_username";
String password = "your_password";
```

Experiment#			
Date		Student ID	
		Student Name	

```
try (Connection conn = DriverManager.get
    Connection (url, user, password) ){
System.out.println ("connected to database"),
}catch (SQLException e) {
  if (e.getSQLState().equals ("28000")).
  System.out.println ("Invalid username or
                       password");
  else
  e.printStackTrace();
  }}
}
```

Experiment#			
Date		Student ID	
		Student Name	

✓ **Data and Results:**

Successfully established a connection to the PostgreSQL "test" database, with error handling for authentication issues

✓ **Analysis and Inferences:**

The program verified database connectivity, ensuring access control works as expected and handling valid and invalid login attempts.

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.