M T W T F S S
Page No.:
Date:
BEST SPIRAL

* What is JDBC and JDBC Drivers.

→ JDBC (Java database connectivity) is a Java API for connecting and interacting with database. JDBC drivers are software components that provide the necessary functionality to connect Java applications to different type of databases.

There are four types of JDBC drivers:

1. Type 1: JDBC - ODBC Bridge Driver (अभी चलता)
2. Type 2: Native - API partly Java Driver (database vendor provide)
3. Type 3: Network protocol Pure Java Driver
4. Type 4: Thin Driver (also known as the Direct to Database pure Java Driver)

→ Each type of driver has its own advantages and is suitable for different scenarios.

* JDBC Components

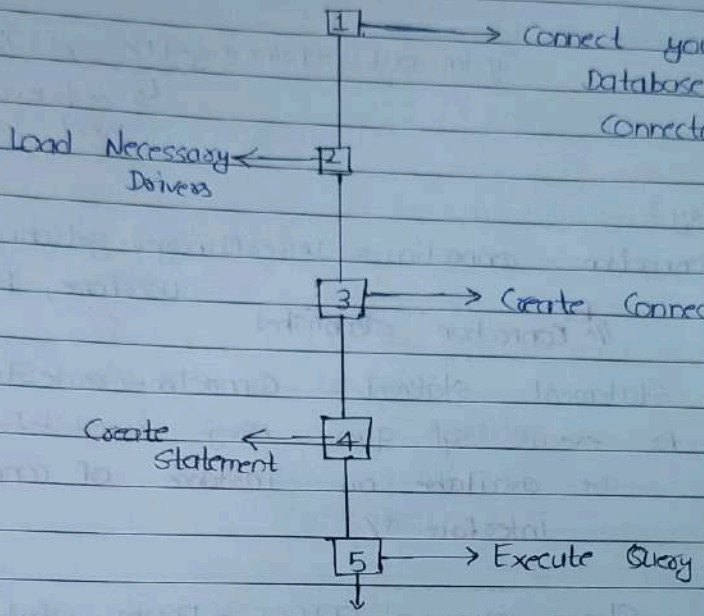
→ In addition to the JDBC drivers, there are several other components that make up the JDBC API, including:

- DriverManager class
- Connection interface
- Statement and PreparedStatement interface
- ResultSet interface

→ These components work together to provide a powerful and flexible API for working with

database in Java.

Program Flow



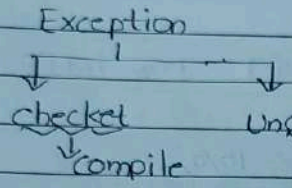

```
import java.sql.*;
public class Main {
    private static final String url = "jdbc:mysql://localhost:3306/mydb";
    private static final String username = "root";
    private static final String password = "admin12";

    psvm (String[] args) {
```

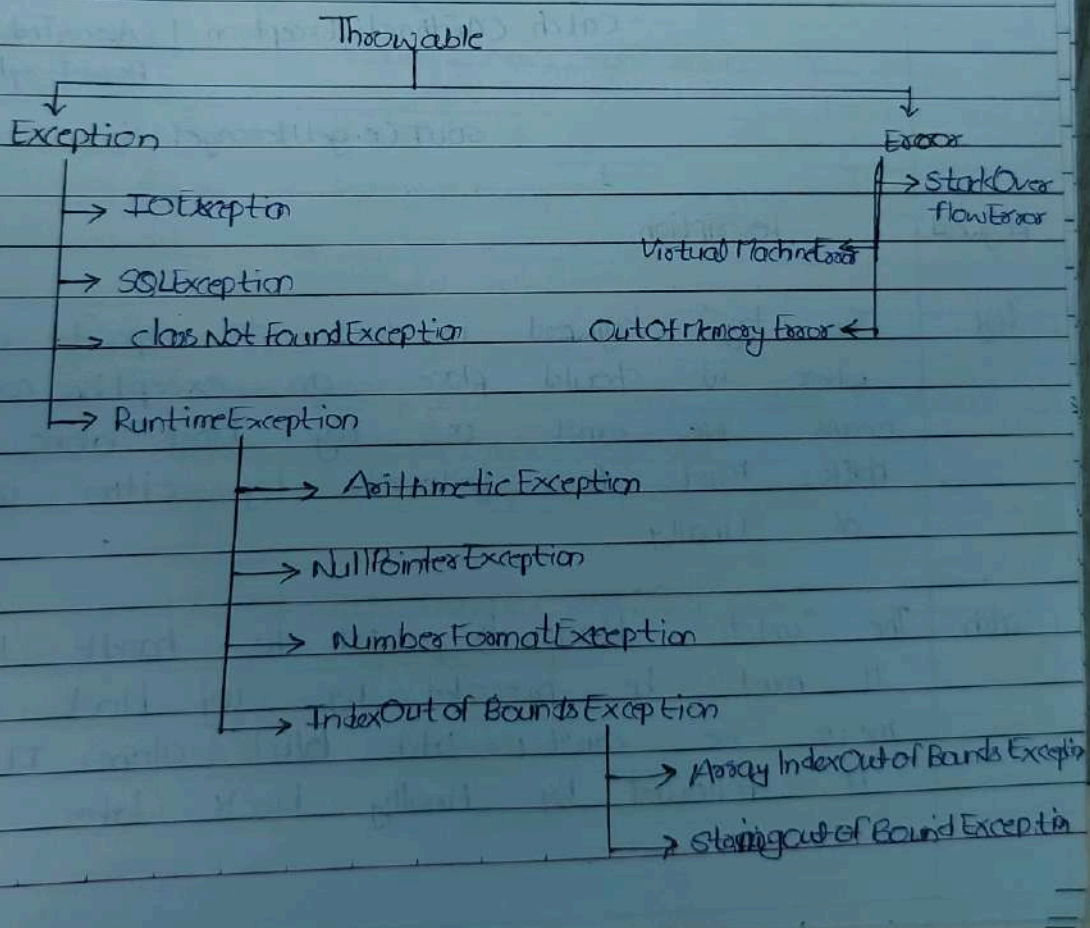
for security pov — for access in main method — so no change — database name

```
try {
    class.forName("com.mysql.cj.jdbc.Driver");
                    // for load drivers
} catch(ClassNotFoundException e) {

        System.out.println(e.getMessage());
                                ↳ retrieve what
                                  went wrong
}
try {
Connection connection = DriverManager.getConnection(Url,
                                    Username, password);
  //^ for connection stablished

  Statement statement = Connection.createStatement();
/* for execute sql query using .creatStat— that
   are availabe on instance of connection
      interface */

  String query = "SELECT * FROM student";
// this query execute by the help of statement interface
// this query is used for retrieve data

  statement.executeQuery(query);
ResultSet resultSet = statement.executeQuery(query);

While(resultSet.nxt()) {
  // Specify the table has remaining row or not

  int id = resultSet.getInt("id");
String name = resultSet.getString("name");
  int age = resultSet.getInt("age");
double marks = resultSet.getDouble("marks");
```

```
SOUT ("ID: " + id);
SOUT ("Name: " + name);
SOUT ("Age: " + age);
SOUT ("Marks: " + marks);
    resultSet.close(); statement.close(); connection.close();
} catch (SQLException e) {
        SOUT (e.getMessage());
    }
}
```

=





M T W T F S S
Page No.
Date
BEST SPIRAL

Eg-

```
public class ExceptionDemo {
    psvm(String[] args) {

        int a = 10;
        int b = 0;
        try {
            int c = a/b;
        } catch (ArithmeticException e) {
            SOUT(e.getMessage());
                ↳ print anything you want
        }

        int arr[] = new int[5];
        try {
            arr[6] = 10/0;
        }
        catch (ArithmeticException | ArrayIndexOutOfBoundsException e) {

            SOUT(e.getMessage());
        }
```

Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
Catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.

finally: The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is used to throw an exception.

throw: The "throw" keyword is used to throw an exception.

throws: The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception. It is always used with method signature.

=

```
int age = 12;
if (age < 18) {
    throw new RuntimeException("Sorry, You Can't vote!!");
} else {
    SOUT("You are eligible to vote!!");
}
```

=

```
public class ExceptionDemo {
    psvm( divisiondemo(int dividend, int divisor) throws
            ArithmeticException {
        SOUT( dividen/divisor);
    }
    psvm(String[] args) {
        divisiondemo(10, 0);
    }
}
```

=

```
import java.sql.*;

public class Tutorial5
private static String url = "jdbc:mysql://localhost:3306/mydb";
    private static String username = "root";
    private static String password = "Suraj321.S";
    public static void main(String[] args){

for insert //    String query = "INSERT INTO employer(id, name, salary)
                                VALUES (3, 'Sohan', 80000)";
for delete //  String query = "DELETE FROM employer WHERE id=4";
for update //  String query = "UPDATE employer\n" +
                            "SET salary = 90000\n" +
                            "WHERE id = 3";

        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
        }
        catch(ClassNotFoundException e){
            System.out.println(e.getMessage());
        }
        try{
            Connection con = DriverManager.getConnection(
                                url, username, password);
            Statement st = con.createStatement();

            int rowsaffected = st.executeUpdate(query);
```

```
if (rowsaffected >0) {
    // SOUT ("Insert Successfull");
    // SOUT ("Deletion Succesfull");
    // SOUT ("Updation Successfull");
} else {
    SOUT ("Unsuccessfull");
}
st.close();
con.close();
} catch (SQLException e){
    SOUT(e.getMessage());
}
}
```

Interface

PREPARED STATEMENTS :-

→ Prepared statements are a feature in database programming, commonly used in JDBC and other data access libraries.

→ They are used to execute SQL queries with placeholders for parameter.

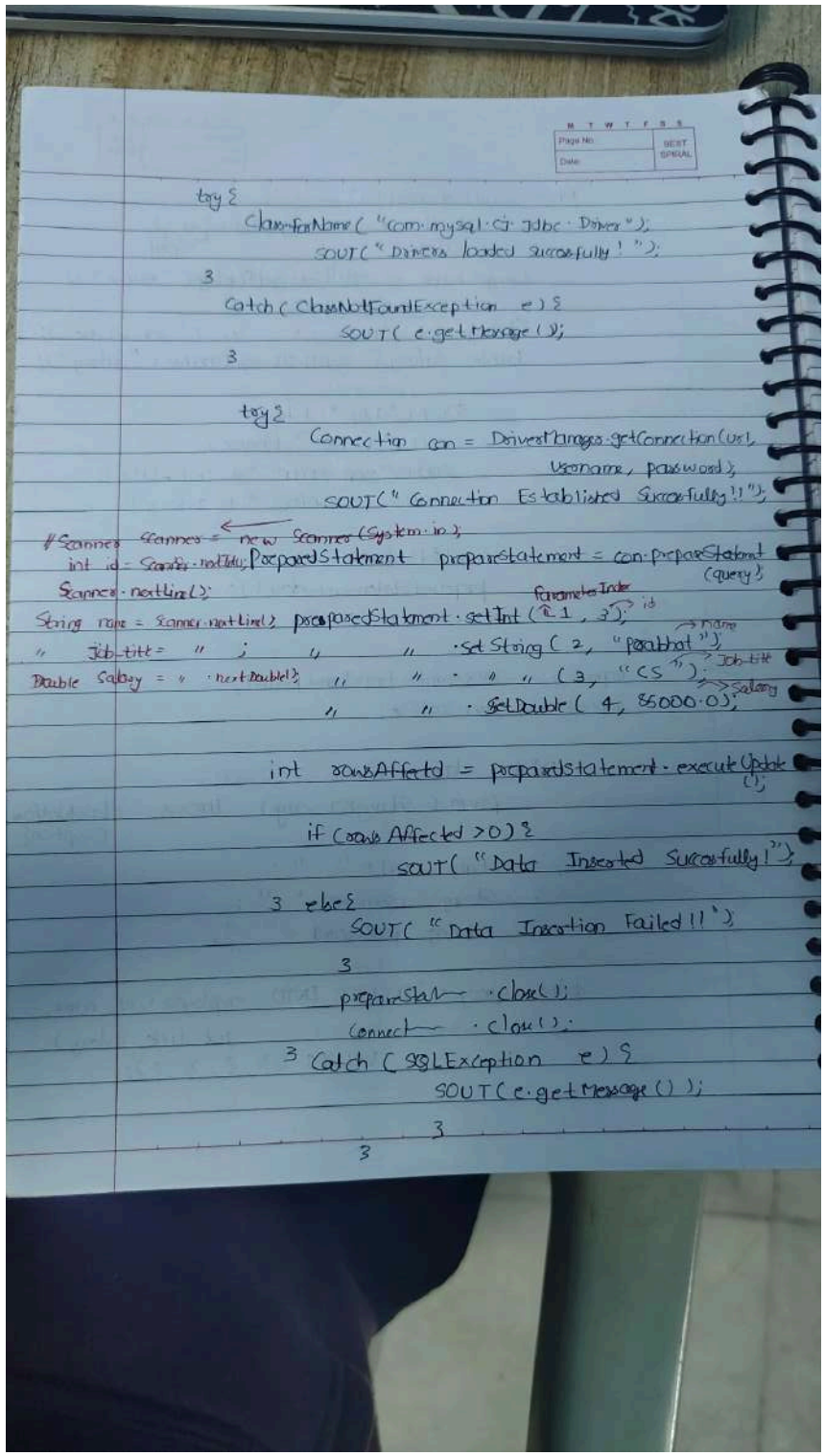
→ These placeholders are then filled with specific values when the query is executed.

→ They offer several advantages -

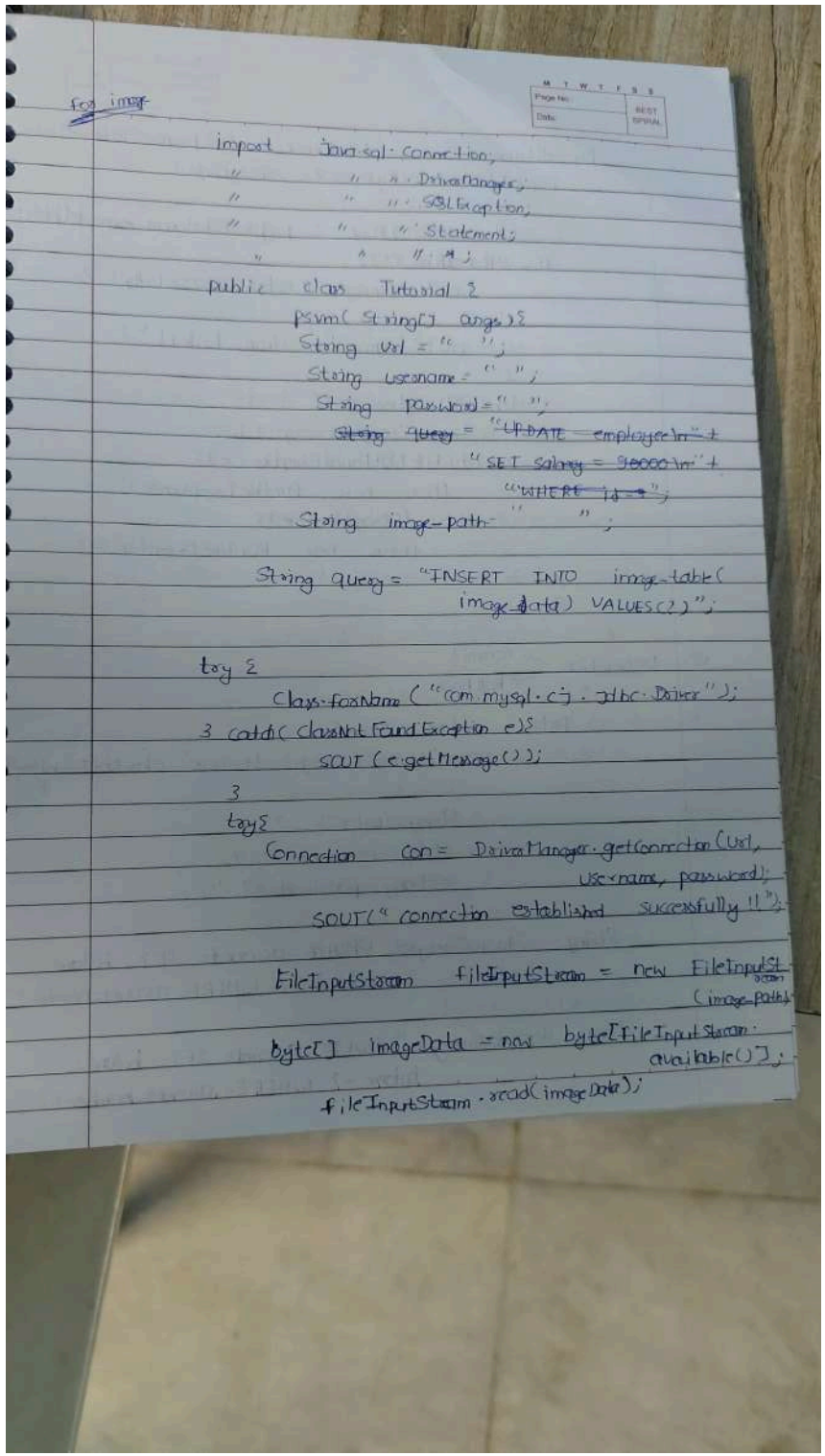
1. Protection against SQL Injection.
2. Improved Performance.
3. Code Readability and Maintainability.
4. Automatic DataType Handling
5. Portability. etc. ..

```
import javax.xml.transform.Result;
import java.sql.*;

Public class Main {
    psvm (String[] args) throws classNot Found
                                Exception {

    String url = " ";
    String username = " ";
    String password = " ";

String query = "Select * from employees where
                    name = ? " AND
                    job-title = ?";
try {
    Class.forName ("com.mysql.cj.jdbc.Driver");

    System.out.println ("Driver loaded Successfully!");
}
catch (ClassNotFoundException e) {
        SOUT (e.getMessage());
}

try {
    Connection con = DriverManager.getConnection(
                    url, username, password);

    SOUT ("Connection Establish Successfully!");

PreparedStatement preparedStatement = con.preparedSta
                                    (query);
preparedStatement.setString (1, "Hemant");
//p  "   "   "   (2, "DevOps Engr");
ResultSet resultset = prepared Statement.
                        executeQuery();
```

```
while(resultSet.next()){
    int id = resultSet.getInt("id");
    (java)                    (mysql)
    String name = resultSet.getString("name");

    String job_title = resultSet.getString("job_title");
    double salary = resultSet.getDouble("salary");

        SOUT("ID: " + id);
        SOUT("NAME: " + name);
        SOUT("JOB TITLE: " + job_title);
        SOUT("SALARY: " + salary);
}
    resultSet.close();
    preparedStatement.close();
}

import javax.xml.transform.Result;
import java.sql.*;

Public class Main{
    psvm(String[] args) throws classNotFound
                               Exception{

    String url = "";
    String username = "";
    String password = "";

String query = "INSERT INTO employees (id, name,
                job_title, salary)
               VALUES (?, ?, ?, ?);
```

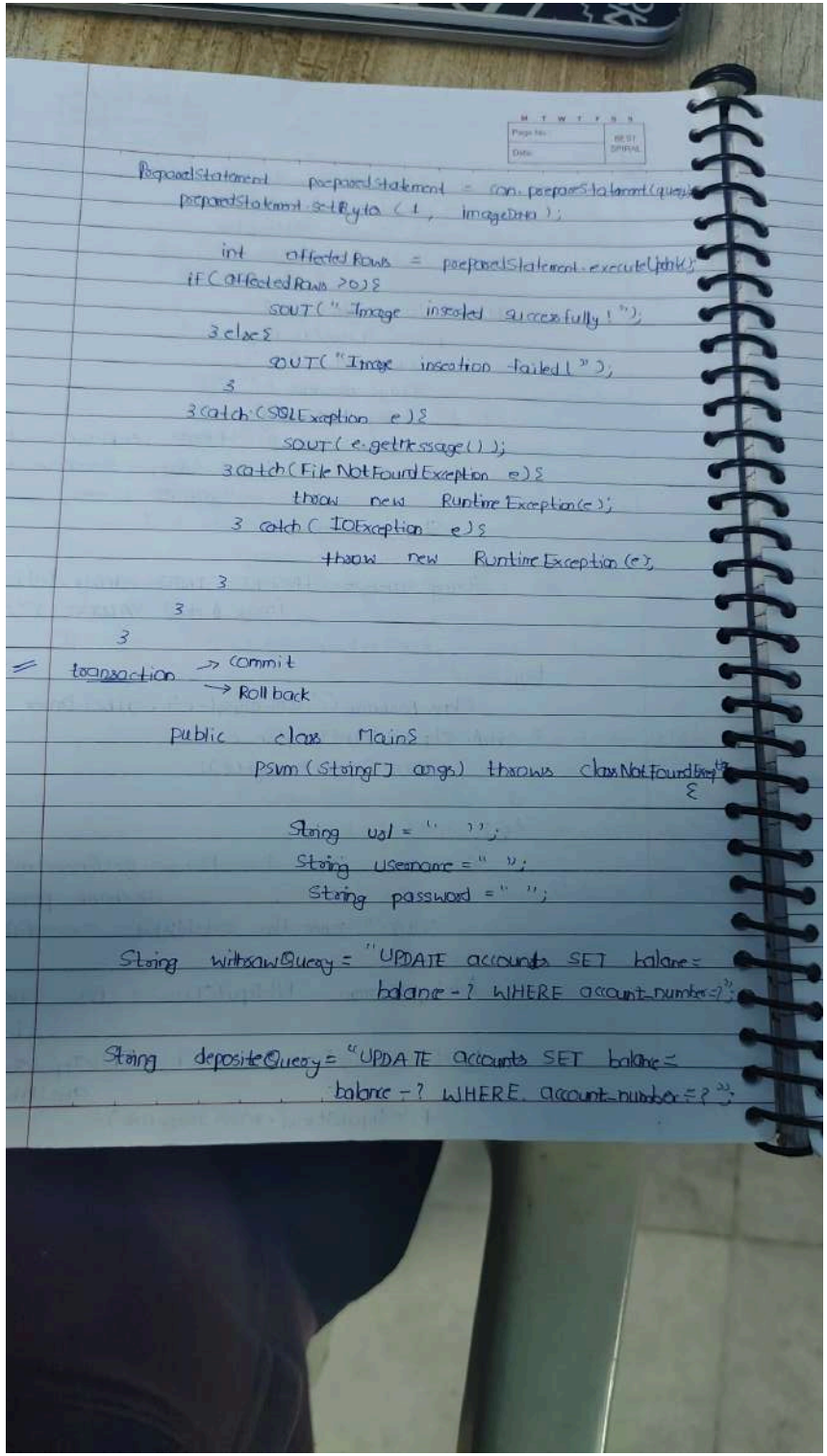


```
try {
    Class.forName("com.mysql.cj.jdbc.Driver");
        SOUT("Drivers loaded successfully!");
}
    Catch(ClassNotFoundException e) {
        SOUT(e.getMessage());
}

    try {
        Connection con = DriverManager.getConnection(url,
                                    username, password);
        SOUT("Connection Established Successfully!!");

// Scanner sc = new Scanner(System.in);
    int id = sc.nextInt(); PreparedStatement preparestatement = con.prepareStatement
                                                    (query);
    Scanner.nextLine();
String name = Scanner.nextLine(); preparedStatement.setInt(1, 2); id
                                  Parameter Index
"  Job_title = " ;  "  "  .setString(2, "prabhat"); name
Double Salary = " .nextDouble(); "  "  .  "  "  (3, "CS"); Job title
                                "  "  .setDouble(4, 85000.0); Salary

        int rowsAffected = preparedstatement.executeUpdate
                                                    ();

        if (rowsAffected > 0) {
            SOUT("Data Inserted Successfully!");
} else {
            SOUT("Data Insertion Failed!!");
}
        preparestat.close();
        connect.close();
} Catch (SQLException e) {
            SOUT(e.getMessage());
}
}
```



for image

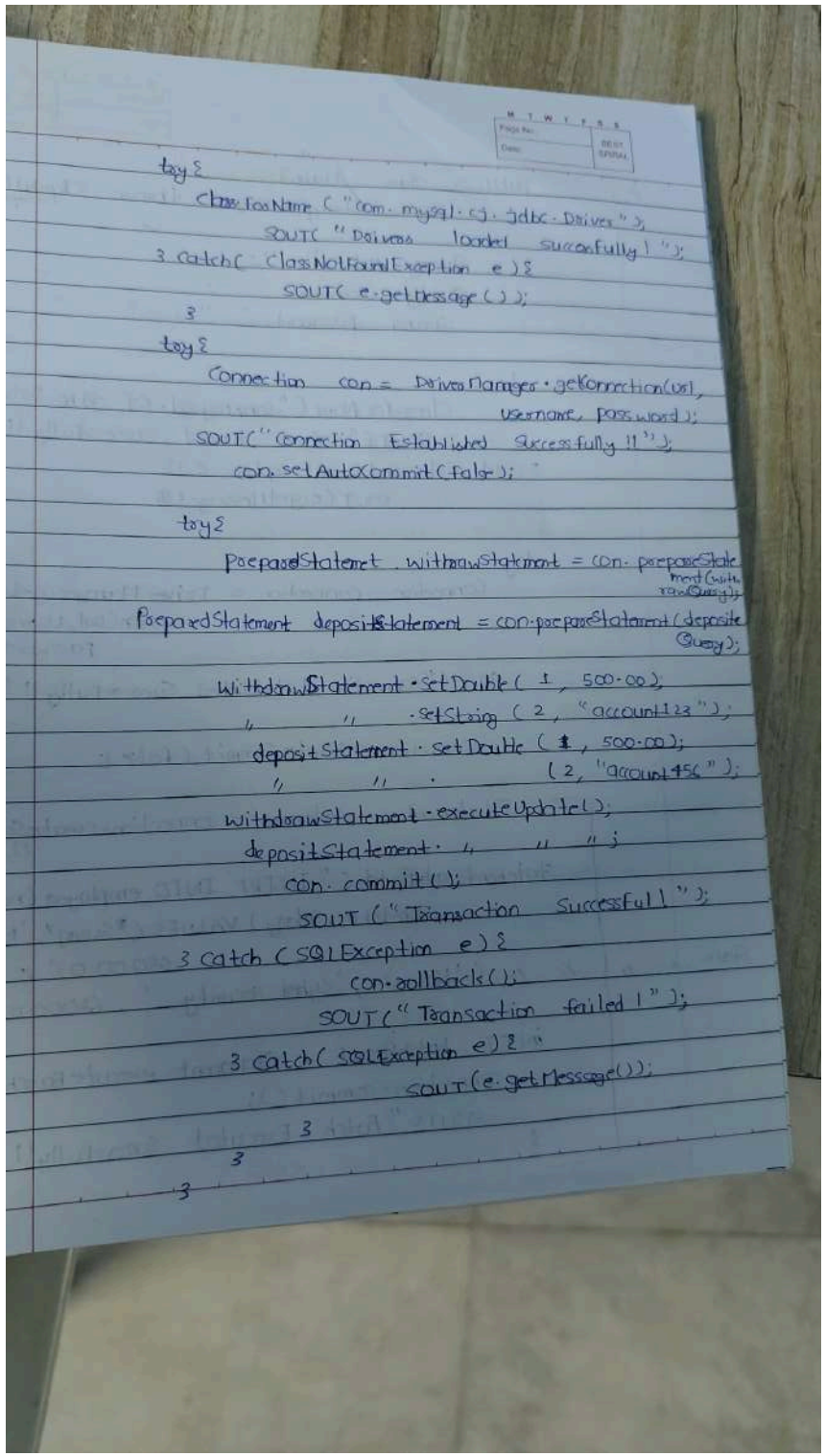
```
import java.sql.Connection;
//          //  .DriverManager;
//          //  .SQLException;
//          //  .Statement;
//          //  .*;

public class Tutorial 2
    psvm(String[] args){
    String url = " ";
    String username = " ";
    String password = " ";
    String query = "UPDATE employee1m" +
                   "SET salary = 90000 1m" +
                   "WHERE id = ";
    String image_path = " ";

    String query = "INSERT INTO image_table(
                    image_data) VALUES(?)";

try{
    Class.forName("com.mysql.cj.jdbc.Driver");
} catch(ClassNotFoundException e){
    SOUT(e.getMessage());
}
try{
    Connection con = DriverManager.getConnection(Url,
                     username, password);
    SOUT("connection established successfully !!");

    FileInputStream fileInputStream = new FileInputSt
                                      (image_path);

    byte[] imageData = new byte[fileInputStream.
                       available()];
    fileInputStream.read(imageData);
```



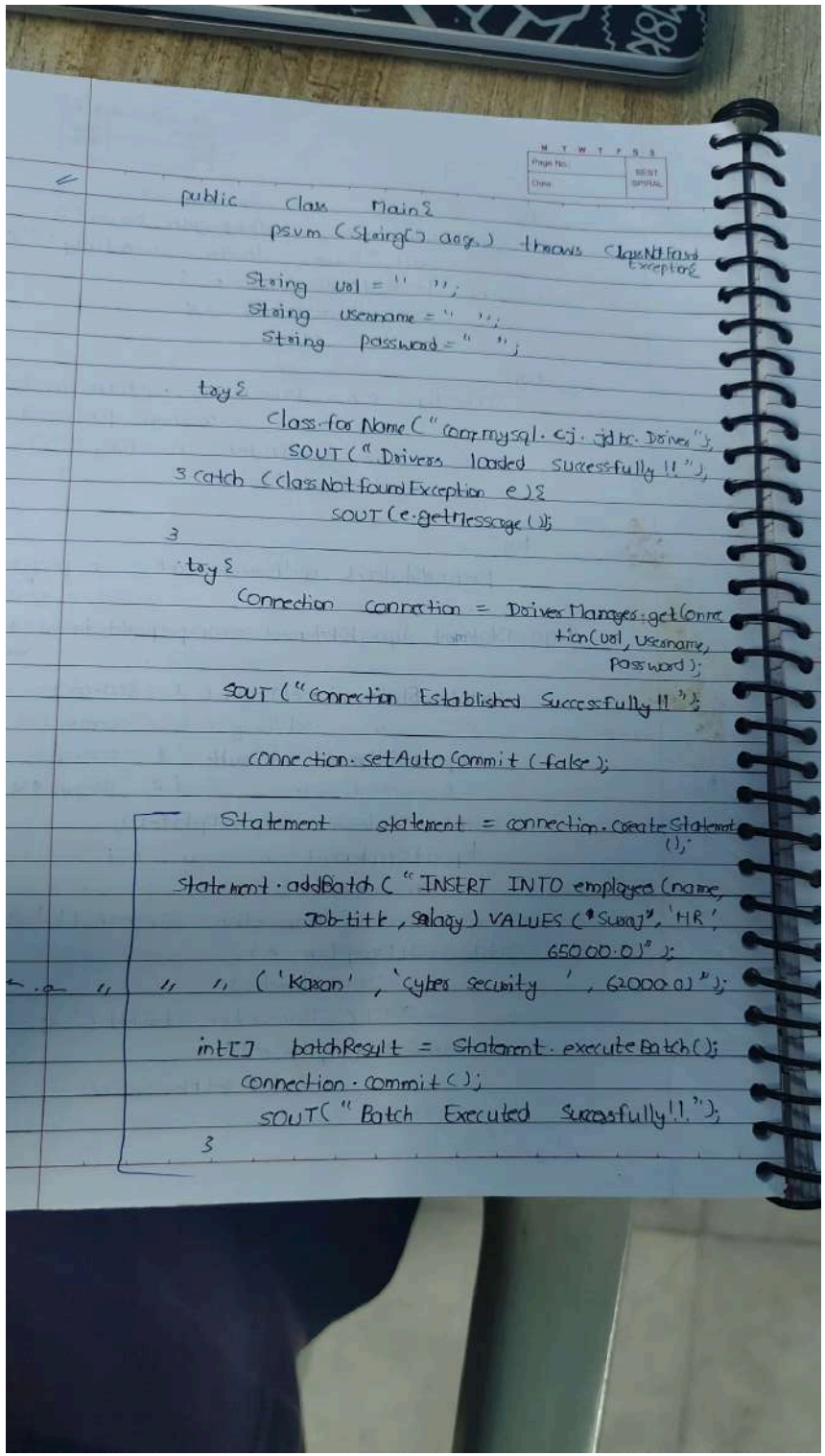
```
PreparedStatement preparedStatement = con.prepareStatement(query);
preparedStatement.setBytes(1, imageData);

    int affectedRows = preparedStatement.executeUpdate();
    if(affectedRows > 0){
        SOUT("Image inserted successfully!");
    }else{
        SOUT("Image insertion failed!");
    }
}catch(SQLException e){
    SOUT(e.getMessage());
}catch(FileNotFoundException e){
    throw new RuntimeException(e);
}catch(IOException e){
    throw new RuntimeException(e);
}
}
}
```

transaction → commit
→ Roll back

```
public class Main{
    psvm(String[] args) throws ClassNotFoundException{

        String url = "";
        String username = "";
        String password = "";

String withdrawQuery = "UPDATE accounts SET balance =
                balance - ? WHERE account_number = ?";

String depositeQuery = "UPDATE accounts SET balance =
                balance - ? WHERE account_number = ?";
```



```
try {
    Class.forName ("com.mysql.cj.jdbc.Driver");
        SOUT ("Drivers loaded successfully!");
} catch ( ClassNotFoundException e ) {
        SOUT (e.getMessage());
}
try {
    Connection con = DriverManager.getConnection(url,
                                    username, password);
    SOUT("Connection Established Successfully!!");
        con.setAutoCommit(false);

    try {
        PreparedStatement withdrawStatement = con.prepareStatement(withdrawQuery);
PreparedStatement depositStatement = con.prepareStatement(depositQuery);

        withdrawStatement.setDouble(1, 500.00);
        "       "   .setString(2, "account123");
        depositStatement.setDouble(1, 500.00);
        "       "   .          (2, "account456");
        withdrawStatement.executeUpdate();
        depositStatement.  "       "  ;
        con.commit();
            SOUT ("Transaction Successful!");
    } catch (SQLException e) {
            con.rollback();
            SOUT("Transaction failed!");
    } catch( SQLException e) {
            SOUT(e.getMessage());
    }
}
```



```
public class Main{
    psvm (String[] args) throws ClassNotFound
                                Exception{
        String url = " ";
        String username = " ";
        String password = " ";

    try{
        Class.forName("com.mysql.cj.jdbc.Driver");
        SOUT("Drivers loaded successfully!!");
    } catch (ClassNotFoundException e){
        SOUT(e.getMessage());
    }
    try{
        Connection connection = DriverManager.getConne
                                tion(url, username,
                                password);
        SOUT("Connection Established Successfully!!");

        connection.setAutoCommit(false);

        Statement statement = connection.createStatement
                              ();

        statement.addBatch("INSERT INTO employees(name,
                            job_title, salary) VALUES ('Suraj', 'HR',
                                                       65000.0)");
        // // // ('Karan', 'cyber security', 62000.0)");

        int[] batchResult = statement.executeBatch();
        connection.commit();
        SOUT("Batch Executed Successfully!!");
    }
```

```
        catch (SQLException e){
            SOUT(e.getMessage());
        }
    }

    PreparedStatement preparedStatement = connection.prepareStatement(query);

    Scanner scanner = new Scanner(System.in);
    While (true){
        SOUT("Name: ");
        String name = Scanner.nextLine();
        SOUT("Job Title: ");
        String job_titt = Scanner.nextLine();
        SOUT("Salary: ");
        double salary = Scanner.nextDouble();

    preparedStatement.setString(1, name);
    "       "   "   (2, job_title);
    "       "   "   (3, salary);
    preparedStatement.addBatch();
        SOUT("Add more values Y/N: ");
        String desion = scanner.nextLine();

    if (decision.toUpperCase().equal("N")){
            break;
        }
    }

        int[] batchResult = preparedStatement.
                                executeBatch();

        connection.commit();
        SOUT("Batch Execute Successfully.");
    } catch(SQLException e){
        SOUT(e.getMessage());
    }
}
```