VISHWAKARMA UNIVERSITY

Maximising Human Potential

Group No.	Roll No.	SRN No.	Name	Div
1	01	202201722	Riya Dhadiwal	E
2	02	202201727	Atharva shevate	E
3	05	202201733	Carmen Maria Isaac	E

Mini Project Report on

TSDL-3-Java

Submitted to Vishwakarma University, Pune

By

Second Year Engineering

Department of Computer Engineering

Faculty of Science and Technology

Academic Year

2023-2024

CONTENTS

STUDENT ATTENDANCE TRACKER (LAB)

INTRODUCTION

This Java program serves as a student attendance tracker, offering functionality to record and manage attendance data efficiently. Leveraging a MySQL database, it enables users to mark attendance, view attendance records, monitor the total number of students present on a given day, and retrieve the total attendance count for individual students.

The program employs JDBC (Java Database Connectivity) to establish a connection with the local MySQL database, allowing seamless interaction with the 'attendance' table. Through a user-friendly interface, it prompts the user to choose from various options, facilitating easy navigation through different functionalities.

Key features of the program include: • Creation of an 'attendance' table to store student attendance records.

- Dynamic marking of attendance, including toggling present status and updating attendance counts.
- Displaying comprehensive attendance records, including student names, dates, present status, and total attendance counts.
- Providing real-time updates on the total number of students present on the current day.
- Offering the flexibility to retrieve the total attendance count for any student registered in the system.

With its intuitive design and robust functionality, this student attendance tracker simplifies the process of managing attendance data, making it an indispensable tool for educational institutions and organizations striving for efficient attendance monitoring.

METHODOLOGY

• **Java Database Connectivity (JDBC):**

- JDBC is a Java API that allows Java applications to interact with databases, enabling CRUD operations (Create, Read, Update,
Delete).

- In this code, JDBC is used to establish a connection with a MySQL database (`getConnection()` method) and execute SQL queries to manipulate data (`createTable()`, `markAttendance()`, `displayAttendance()`, `displayTotalPresent()`).

• **SQL (Structured Query Language):**

- SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS).
- The code utilizes SQL queries to create tables (`createTable()`), insert/update records (`markAttendance()`), and retrieve data (`displayAttendance()`, `displayTotalPresent()`).

• **HashMap:**

- HashMap is a data structure in Java that stores key-value pairs.
- In this code, a HashMap (`attendanceCounters`) is used to efficiently store and update the attendance counts for individual students.

• **Scanner Class:**

- The Scanner class in Java is used for obtaining the input of the primitive types like int, double, etc., and strings.
- In this code, Scanner is used to accept user inputs for student names, present status, and menu options.

• **Control Flow (if-else, switch):**

- The code employs various control flow statements such as if-else and switch to make decisions based on user inputs and execute appropriate actions.
- For example, in the `markAttendance()` method, if-else statements are used to handle cases where the student exists or is new, and switch-case is used in the `main()` method to execute different functionalities based on user input.

• **Date Handling:**

- The code utilizes SQL's `CURDATE()` function to handle dates and track attendance on the current date (`markAttendance()`, `displayTotalPresent()`).

• **Exception Handling:**

- Exception handling is used to manage errors that may occur during database operations or user input.

- In the `main()` method, a try-catch block is used to catch SQLExceptions that may occur during database operations and handle them appropriately.

•

String Manipulation:

- String manipulation techniques are employed to process and validate user inputs, such as converting input to lowercase (`input.equalsIgnoreCase("Y")`) and extracting strings from ResultSet (`resultSet.getString("name")`).

• Encapsulation:

- Encapsulation involves bundling the data (variables) and methods (functions) that operate on the data into a single unit, known as a class.
- In the `attendance1` class, methods like `markAttendance()`, `displayAttendance()`, etc., encapsulate functionalities related to attendance tracking.
- Data such as JDBC connection details (`JDBC_URL`, `USERNAME`, `PASSWORD`) is encapsulated within the class.

•

Abstraction:

- Abstraction hides the complex implementation details and provides a simplified view of the functionality to the user.
- The `attendance1` class serves as an abstraction layer for attendance tracking, exposing methods that allow users to interact with the system without needing to understand the underlying database operations or SQL queries.

• Polymorphism (Method Overloading):

- Polymorphism allows methods to be defined in multiple forms, depending on the number or type of parameters.
- The `markAttendance()` method demonstrates method overloading, as it handles both new and existing students by providing different sets of parameters.

• Collections (HashMap):

- Collections are used to store and manipulate groups of objects efficiently.

•

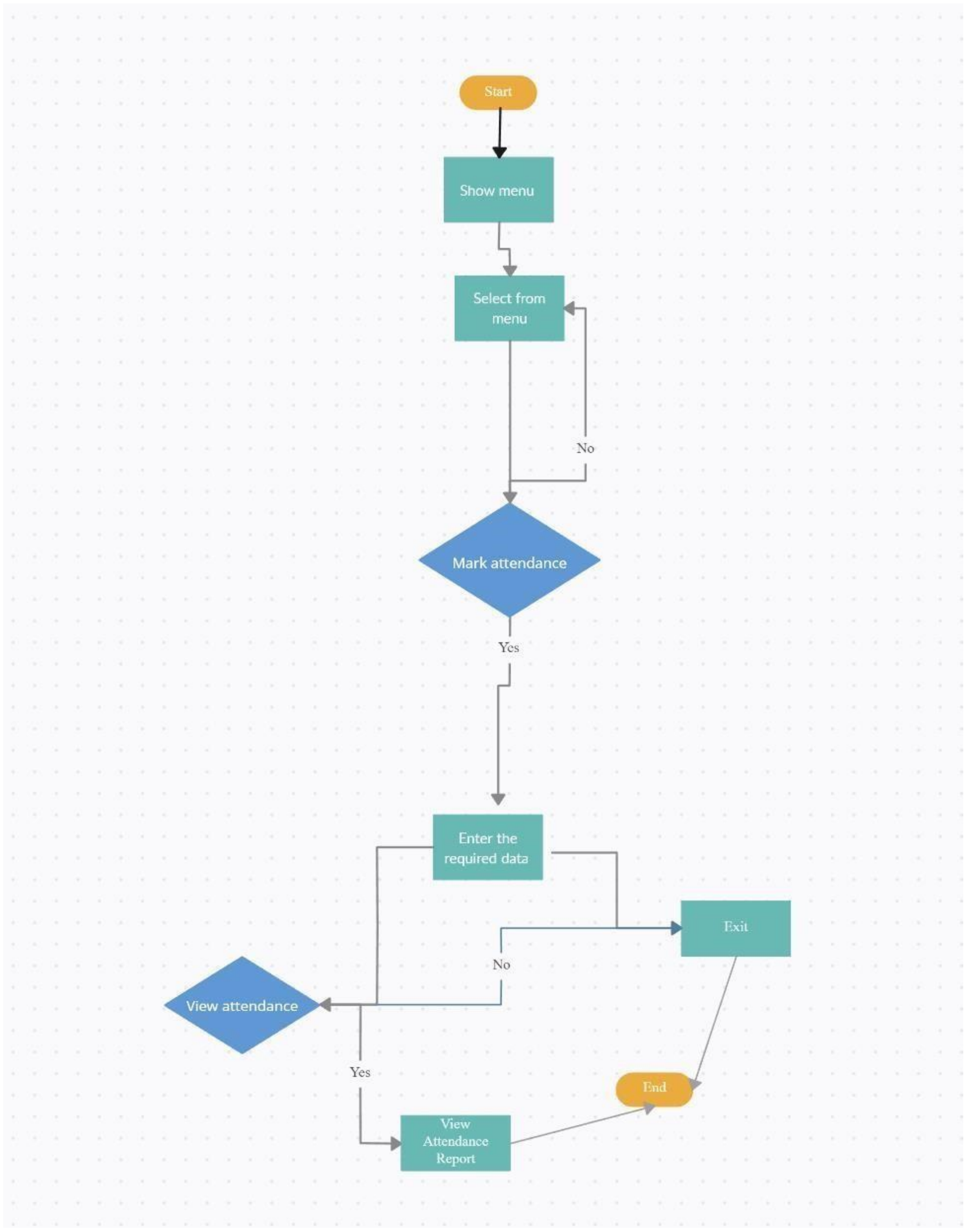
•

•

- The `attendanceCounters` variable, which is a `HashMap<String, Integer>`, stores student names as keys and their corresponding attendance counts as values. Database Connectivity (JDBC):
- JDBC (Java Database Connectivity) is used to connect Java applications with databases.
- The `getConnection()` method establishes a connection with a MySQL database, allowing the application to perform database operations such as table creation and data manipulation. Input/Output (I/O):
- Input/Output operations are performed using the `Scanner` class to accept user inputs from the console (`Scanner scan = new Scanner(System.in)`).
- The code prompts users to enter data such as student names and attendance status, which are then processed by the program.

These concepts collectively contribute to the functionality, organization, and readability of the code, making it easier to understand, extend, and maintain. They are fundamental principles in Java programming for building robust and scalable software systems.

FLOWCHART


Start
Show menu
Select from menu
No
Mark attendance
Yes
Enter the required data
Exit
No
View attendance
Yes
End
View Attendance Report

IMPLEMENTATION

Key Components:

Database Schema: Design tables for storing student information, courses, classes, and attendance records.

JDBC Setup: Establish JDBC connectivity to the database for CRUD operations.

User Interface: Develop a user-friendly interface for tasks like marking attendance, viewing records, and managing student data.

Java Classes: Implement classes to interact with the database, perform CRUD operations, and manage attendance logic.

Attendance Tracking Logic: Create logic to mark attendance, calculate percentages, and generate reports.

Security: Ensure secure access to the system, allowing only authorized users to modify attendance records.

1. Imports:

- `java.sql.*`: Importing classes required for JDBC (Java Database Connectivity) operations.
- `java.util.HashMap`: Importing the `HashMap` class for storing attendance counters.
- `java.util.Map`: Importing the `Map` interface to handle key-value pairs.
- `java.util.Scanner`: Importing the `Scanner` class to read user input from the console.

2. Class Declaration:

- Declaring a public class named `attendance1`. **3. Constants:**

- Declaring constants for the JDBC URL, database username, and password.

4. getConnection() Method:

- This method establishes a connection to the MySQL database using the JDBC URL, username, and password constants.

5. createTable() Method:

- This method creates the 'attendance' table in the database if it doesn't already exist.

6. markAttendance() Method:

- This method allows marking attendance for students. It checks if the student already exists in the database and updates their attendance status accordingly.

7. displayAttendance() Method:

- This method retrieves and displays all attendance records from the 'attendance' table.

8. displayTotalPresent() Method:

- This method calculates and displays the total number of students present today. **9. displayTotalAttendanceForStudent() Method:**

- This method allows the user to view the total attendance for a specific student.

10. Main Method:

- The main method establishes a connection to the database, creates the 'attendance' table if necessary, and presents a menu to the user to perform various operations like marking attendance, viewing attendance records, etc.

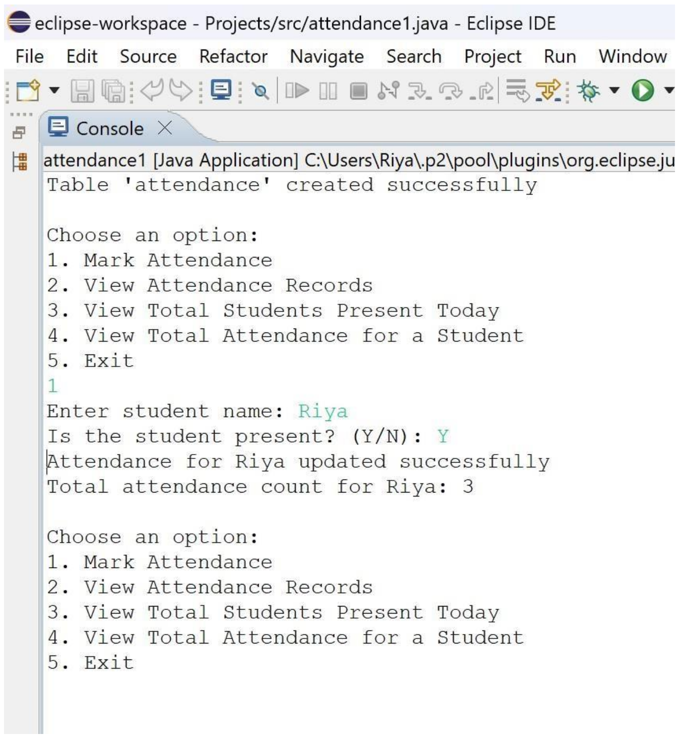
This code demonstrates the use of JDBC for database operations, including creating tables, inserting/updating records, and querying data. It also utilizes concepts like `Map` for storing attendance counters and `Scanner` for user input.

CODE:

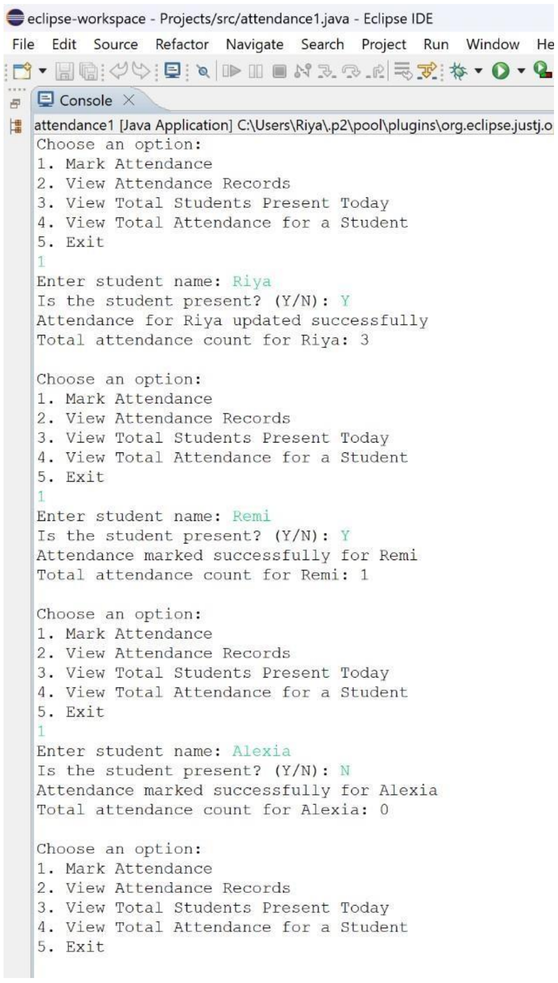
```
import java.sql.*; import
java.util.HashMap; import
java.util.Map;  import
java.util.Scanner;  public
class attendance1 {

   private static final String JDBC_URL = "jdbc:mysql://localhost:3306/attend";
private static final String USERNAME = "root";
   private static final String PASSWORD = "@Root310304R";

   private static Connection getConnection() throws SQLException {         return
DriverManager.getConnection(JDBC_URL, USERNAME,  PASSWORD);
   }

   private static void createTable(Connection connection) throws  SQLException
{
       String createTableSQL = "CREATE TABLE IF NOT EXISTS attendance
(id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(255), date
DATE, present BOOLEAN, attendance_count INT DEFAULT 0)";         try
(Statement statement = connection.createStatement()) {
statement.executeUpdate(createTableSQL);
           System.out.println("Table 'attendance' created successfully");
       }
   }

   private static void markAttendance(Connection connection, Map<String,
Integer> attendanceCounters) throws SQLException {
       Scanner scan = new Scanner(System.in);

       System.out.print("Enter student name: ");
       String name = scan.nextLine();

       // Check if the student already exists in the database
       String selectDataSQL = "SELECT * FROM attendance WHERE name =
?";        try (PreparedStatement selectStatement
=
connection.prepareStatement(selectDataSQL)) {
selectStatement.setString(1, name);
            ResultSet resultSet = selectStatement.executeQuery();             if
(resultSet.next()) {

                // If the student exists, get the existing attendance count
boolean present = resultSet.getBoolean("present");                present =
!present; // Toggle present status

                // Get the existing attendance count from the database                int
counter = resultSet.getInt("attendance_count");

                // Ask for attendance status even if student exists
                System.out.print("Is the student present? (Y/N): ");
String input = scan.nextLine();                boolean newPresent =
input.equalsIgnoreCase("Y");

                String updateDataSQL = "UPDATE attendance SET present = ?, date
= CURDATE(), attendance_count = ? WHERE name = ?";                try
(PreparedStatement updateStatement =
connection.prepareStatement(updateDataSQL)) {
updateStatement.setBoolean(1, newPresent);
                    updateStatement.setInt(2, newPresent ? counter + 1 : counter); //
Increment the counter only if present
updateStatement.setString(3, name);
updateStatement.executeUpdate();
                    System.out.println("Attendance for " + name + " updated
successfully");

                    // Update attendance counter
attendanceCounters.put(name, newPresent ? counter + 1 : counter);
                    System.out.println("Total attendance count for " + name + ": " +
(newPresent ? counter + 1 : counter));
                }
            } else {
                // If the student doesn't exist, insert a new attendance record
                System.out.print("Is the student present? (Y/N): ");
String input = scan.nextLine();                 boolean present =
input.equalsIgnoreCase("Y");

                String insertDataSQL = "INSERT INTO attendance (name, date,
present, attendance_count) VALUES (?, CURDATE(), ?, ?)";
                try (PreparedStatement insertStatement =
connection.prepareStatement(insertDataSQL)) {
insertStatement.setString(1, name);                 insertStatement.setBoolean(2,
present);
                    insertStatement.setInt(3, present ? 1 : 0); // Initialize the counter
accordingly
                    insertStatement.executeUpdate();
                    System.out.println("Attendance marked successfully for " + name);

                    // Update attendance counter
                    attendanceCounters.put(name, present ? 1 : 0);
System.out.println("Total attendance count for " + name + ": " + (present ? 1 : 0));
                }
            }
        }
    }

    private static void displayAttendance(Connection connection) throws
SQLException {
        String selectDataSQL = "SELECT * FROM attendance";         try
(Statement statement = connection.createStatement();
             ResultSet resultSet = statement.executeQuery(selectDataSQL)) {
            System.out.println("Student Attendance Records:");
            System.out.println("ID\tName\tDate\t\tPresent\tAttendance Count");
while (resultSet.next()) {                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");                Date date
= resultSet.getDate("date");                boolean present =
resultSet.getBoolean("present");                int attendanceCount =
resultSet.getInt("attendance_count");                System.out.println(id +
"\t" + name + "\t" + date + "\t" + (present ?
"Y" : "N") + "\t" + attendanceCount);
```

```
            }
        }
    }

    private static void displayTotalPresent(Connection connection) throws
SQLException {
        String selectDataSQL = "SELECT COUNT(*) AS total_present FROM
attendance WHERE present = true AND date = CURDATE()";        try
(Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(selectDataSQL)) {            if
(resultSet.next()) {
                int totalPresent = resultSet.getInt("total_present");
                System.out.println("Total students present today: " + totalPresent);
            }
        }
    }

    private static void displayTotalAttendanceForStudent(Map<String, Integer>
attendanceCounters) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter student name: ");
        String studentName = scan.nextLine();        int totalAttendance =
attendanceCounters.getOrDefault(studentName, 0);        System.out.println("Total
attendance for " + studentName + ": " + totalAttendance);
    }

    public static void main(String[] args) {        try (Connection
connection = getConnection()) {
createTable(connection);

            Map<String, Integer> attendanceCounters = new HashMap<>();

            Scanner scan = new Scanner(System.in);            while
(true) {
                System.out.println("\nChoose an option:");
                System.out.println("1. Mark Attendance");
                System.out.println("2. View Attendance Records");
                System.out.println("3. View Total Students Present Today");
                System.out.println("4. View Total Attendance for a Student");
System.out.println("5. Exit");

                int option = scan.nextInt();
                scan.nextLine(); // Consume newline character
                 switch (option)
{
case 1:
                    markAttendance(connection, attendanceCounters);
break;              case 2:
                    displayAttendance(connection);                break;
case 3:
                    displayTotalPresent(connection);
break;              case 4:
                    displayTotalAttendanceForStudent(attendanceCounters);
break;              case 5:
                    System.out.println("Exiting...");             return;
default:
                    System.out.println("Invalid option");
                }
            }
        }       catch       (SQLException       e)      {
e.printStackTrace();
        }
    }
}
```

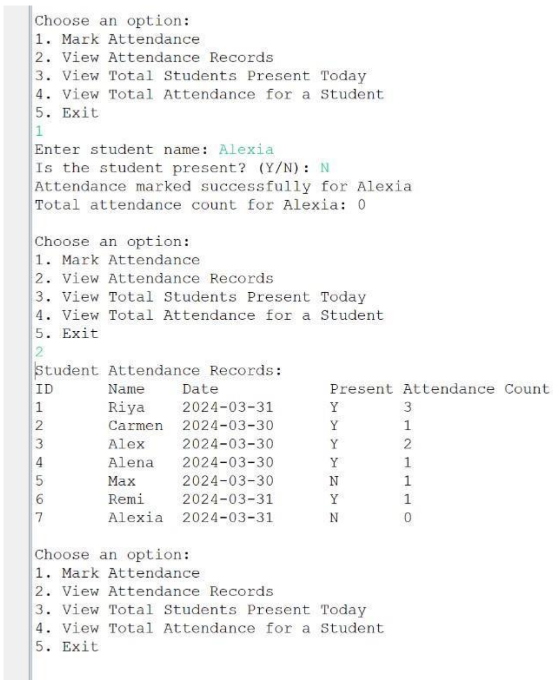
RESULTS


eclipse-workspace - Projects/src/attendance1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window
Console
attendance1 [Java Application] C:\Users\Riya\.p2\pool\plugins\org.eclipse.ju

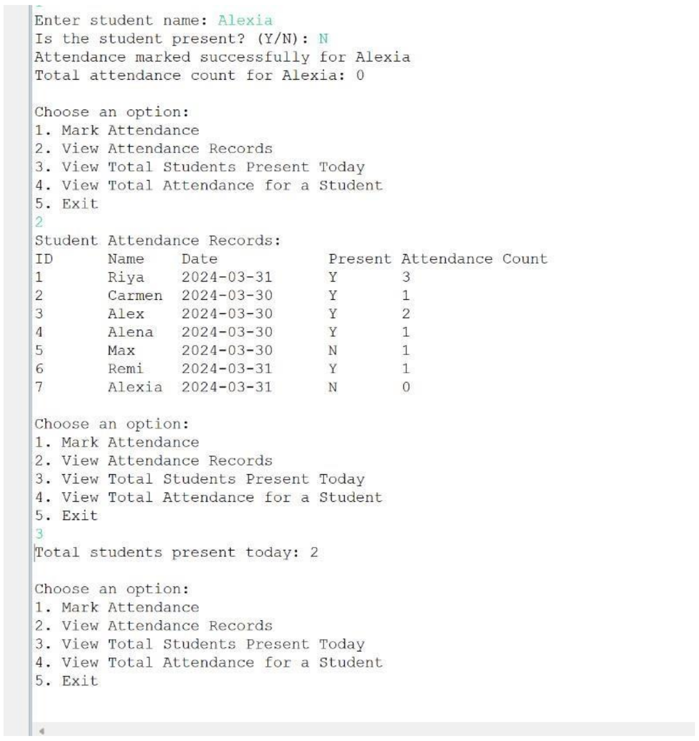
```
Table 'attendance' created successfully

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
1
Enter student name: Riya
Is the student present? (Y/N): Y
Attendance for Riya updated successfully
Total attendance count for Riya: 3

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
```


eclipse-workspace - Projects/src/attendance1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window He
Console

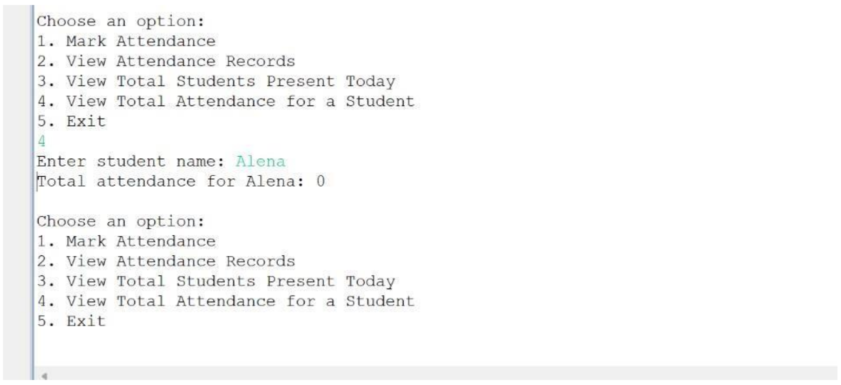
```
attendance1 [Java Application] C:\Users\Riya\.p2\pool\plugins\org.eclipse.justj.o
Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
1
Enter student name: Riya
Is the student present? (Y/N): Y
Attendance for Riya updated successfully
Total attendance count for Riya: 3

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
1
Enter student name: Remi
Is the student present? (Y/N): Y
Attendance marked successfully for Remi
Total attendance count for Remi: 1

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
1
Enter student name: Alexia
Is the student present? (Y/N): N
Attendance marked successfully for Alexia
Total attendance count for Alexia: 0

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
```



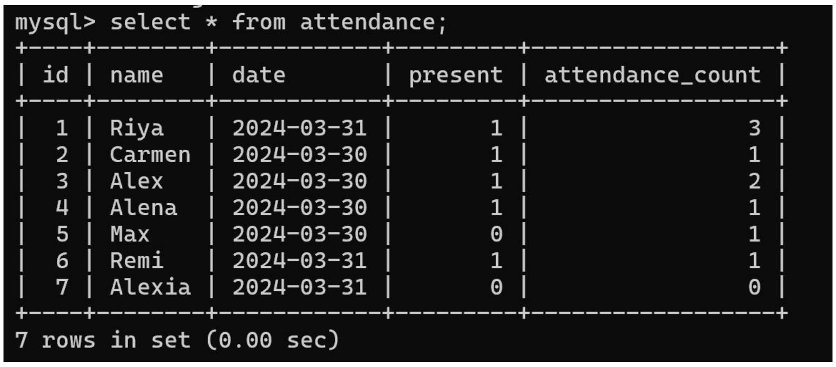
```
Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
1
Enter student name: Alexia
Is the student present? (Y/N): N
Attendance marked successfully for Alexia
Total attendance count for Alexia: 0

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
2
Student Attendance Records:
ID      Name    Date            Present Attendance Count
1       Riya    2024-03-31      Y       3
2       Carmen  2024-03-30      Y       1
3       Alex    2024-03-30      Y       2
4       Alena   2024-03-30      Y       1
5       Max     2024-03-30      N       1
6       Remi    2024-03-31      Y       1
7       Alexia  2024-03-31      N       0

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
```



```
Enter student name: Alexia
Is the student present? (Y/N): N
Attendance marked successfully for Alexia
Total attendance count for Alexia: 0

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
2
Student Attendance Records:
ID      Name    Date            Present Attendance Count
1       Riya    2024-03-31      Y       3
2       Carmen  2024-03-30      Y       1
3       Alex    2024-03-30      Y       2
4       Alena   2024-03-30      Y       1
5       Max     2024-03-30      N       1
6       Remi    2024-03-31      Y       1
7       Alexia  2024-03-31      N       0

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
3
Total students present today: 2

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
```



```
Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
4
Enter student name: Alena
Total attendance for Alena: 0

Choose an option:
1. Mark Attendance
2. View Attendance Records
3. View Total Students Present Today
4. View Total Attendance for a Student
5. Exit
```



```
mysql> select * from attendance;
+----+--------+------------+---------+------------------+
| id | name   | date       | present | attendance_count |
+----+--------+------------+---------+------------------+
|  1 | Riya   | 2024-03-31 |       1 |                3 |
|  2 | Carmen | 2024-03-30 |       1 |                1 |
|  3 | Alex   | 2024-03-30 |       1 |                2 |
|  4 | Alena  | 2024-03-30 |       1 |                1 |
|  5 | Max    | 2024-03-30 |       0 |                1 |
|  6 | Remi   | 2024-03-31 |       1 |                1 |
|  7 | Alexia | 2024-03-31 |       0 |                0 |
+----+--------+------------+---------+------------------+
7 rows in set (0.00 sec)
```

CONCLUSION

In summary, the provided Java code presents a basic yet functional attendance management system. It utilizes JDBC for database connectivity and implements essential features such as marking attendance, viewing records, and calculating total present students. Through user-friendly console interactions, it offers a straightforward interface for users to manage attendance efficiently. With error handling in place, it ensures reliability in database operations. Overall, the code serves as a foundation for building more sophisticated attendance management systems with additional features and scalability.