

# **JAVA SWINGS BASED- CARPOOLING**

**- SQL CONNECTIVITY USING JDBC**

**A**

*Report*

*Submitted in partial fulfillment of the  
Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

**IN**

**INFORMATION TECHNOLOGY**

**By**

**V Shwetha&lt1602-19-737-106>**

Under the guidance of Ms B. Leelavathy



Department of Information Technology  
Vasavi College of Engineering (Autonomous)  
(Affiliated to Osmania University)  
Ibrahimbagh, Hyderabad-31



## BONAFIDE CERTIFICATE

This is to certify that this project report titled '**CARPOOLING**' is a project work of Ms. V Shwetha bearing roll no.1602-19-737-106 who carried out the project under my supervision in the IV semester for the academic year 2020- 2021.

Signature  
Internal Examiner

Signature  
External Examiner

# DBMS ASSIGNMENT

## CAR POOLING



NAME: V Shwetha  
ROLL NUMBER: 1602-19-737-106  
CLASS: IT-B

## **ABSTRACT**

*“CAR POOLING” is a system that helps users who own a car to arrange a carpool and the other users to find a car to travel. It is a system used to find cars to travel in which the owners are traveling alone and are ready to share the journey by charging the customers some money. This can help users save their money and time both by sharing their riders. The database has the data of both customers and owners along with the location where the carpool is arranged. This project has a total of 10 tables. It shows which owner is ready to share their ride along with the provisions of the ride which helps customers to choose the car easily by their requirements.*

## **Requirement Analysis:**

*List Of Tables:*

- 
1. Owner
  2. Customer
  3. Car
  4. Pool
  5. Provisions
  6. Owns
  7. Arranges
  8. Registered
  9. Provides

*List of Attributes with their Domains:*

Owner:

- Owner\_ID
- Name
- Age
- Mobile Number
- Gender
- Address

Customer:

- Customer\_ID
- Name
- Age
- Mobile Number
- Gender
- Address

Car:

- Car No

- Model
- Color
- Licence No

Pool:

- Pool ID
- Start
- End
- Date
- Time

Provisions:

- Owner\_Id
- Luggage Size
- Occupancy

In Registered Table:

- Location

## **AIM AND PRIORITY OF THE PROJECT**

To create a Java GUI based information retrieval system of “*Car Pooling System*” which takes the values like: id, Name, Age, etc. from the person who owns the car and the person who wants to access the pool(customer) arranged by customer. These values are to be updated in the database using JDBC connectivity .

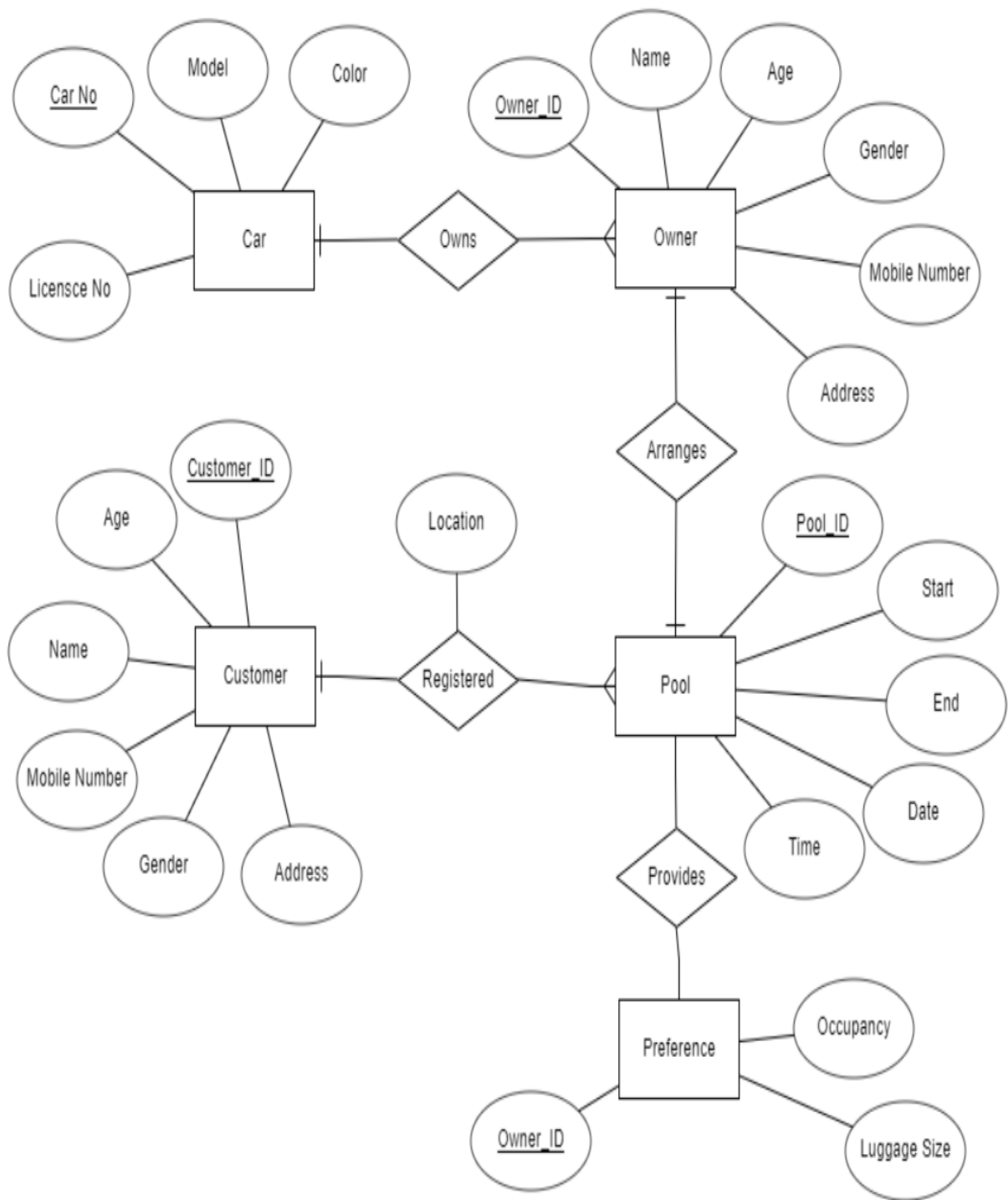
## ARCHITECTURE AND TECHNOLOGY

**Software used:** Java Eclipse, Oracle 11g Database, Java SE version 13, SQL\*Plus.

**Java SWINGS:** Java SWINGS is an API to develop GUI or window-based applications in java. Java SWING components are platform-independent. It is lightweight. The javax.swing package provides classes for SWING API such as JTextField, JLabel, JTextArea, JRadioButton, JCheckBox, JChoice, JList etc.

**SQL:** Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySQL, Oracle, Infomix, Sybase, MS Access) use SQL as the standard database query language. SQL is used to perform all types of data operations in RDBMS

## Entity Relationship Model:



## Mapping Constraints:





*Owner is connected to Car:* A single Owner can own multiple cars so it is one to many participation


*Owner is connected to Pool:* A single Owner can arrange many car pools at different times, so it is one to many.


*Customer is connected to Pool:* A Pool can be registered by more than one customer, so it is one to many.

*Pool is connected to Provisions:* A pool provides provisions. It is one to one.

## DDL Commands:

- SQL>create table car(  
2 car\_no varchar2(20) primary key,  
3 model varchar2(20),

- 
- 4 color varchar2(20),
  - 5 licence\_no varchar2(15));
  - SQL> create table owner(
    - 2 owner\_id varchar2(20) primary key,
    - 3 name varchar2(30),
    - 4 age number(2),
    - 5 mobile\_number number(10),
    - 6 gender char(5),
    - 7 dress varchar2(100));
  - SQL>create table customer(
    - 2 customer\_id varchar2(20) primary key,
    - 3 name varchar2(30),
    - 4 age number(2),
    - 5 mobile\_number number(10),
    - 6 gender char(5),
    - 7 address varchar2(100));
  - SQL> create table pool(
    - 2 pool\_id varchar2(20) primary key,
    - 3 start\_p varchar2(25),
    - 4 end varchar2(25),
    - 5 day varchar(20),
    - 6 timing varchar(20));
  - SQL> create table provisions(
    - 2 owner\_id varchar2(20) primary key,
    - 3 luggage\_size number(5),
    - 4 occupancy number(2));
  - SQL> create table owns(
    - 2 owner\_id varchar2(20),
    - 3 car\_no varchar2(20),
    - 4 foreign key(owner\_id) references owner(owner\_id),
    - 5 foreign key(car\_no) references car(car\_no),
    - 6 primary key(owner\_id, car\_no));
  - SQL> create table arranges(
    - 2 owner\_id varchar2(20),
    - 3 car\_no varchar2(20),
    - 4 foreign key(owner\_id) references owner(owner\_id),
    - 5 foreign key(car\_no) references car(car\_no),
    - 6 primary key(owner\_id, car\_no),
    - 7 date varchar2(10),
    - 8 time varchar2(10),
    - 9 status varchar2(10));

- 
- 2 owner\_id varchar2(20),
  - 3 pool\_id varchar2(20),
  - 4 foreign key(owner\_id) references owner(owner\_id),
  - 5 foreign key(pool\_id) references pool(pool\_id),
  - 6 primary key(owner\_id, pool\_id));
  - SQL> create table registered(
    - 2 customer\_id varchar2(20),
    - 3 pool\_id varchar2(20),
    - 4 location varchar2(100),
    - 5 foreign key(customer\_id) references customer(customer\_id),
    - 6 foreign key(pool\_id) references pool(pool\_id),
    - 7 primary key(customer\_id, pool\_id));
  - SQL> create table provides(
    - 2 pool\_id varchar2(20),
    - 3 owner\_id varchar2(20),
    - 4 foreign key(owner\_id) references owner(owner\_id),
    - 5 foreign key(pool\_id) references pool(pool\_id),
    - 6 primary key(owner\_id, pool\_id));

## DML Commands:

- SQL> insert into owner values('&owner\_id', '&name', &age, &mobile\_number, '&gender', '&address');
- SQL> insert into customer values('&customer\_id', '&name', &age, &mobile\_number, '&gender', '&address');
- SQL> insert into customer('&customer\_id', '&name', &age, &mobile\_number, '&gender', '&address');

- SQL> insert into provisions values('&owner\_id', &luggage\_size, &occupancy);
- SQL> insert into pool values('&pool\_id', '&start\_p', '&end', '&day', '&timing');
- SQL> insert into owns values('&owner\_id', '&car\_no');
- SQL> insert into provides values('&pool\_id', '&owner\_id');
- SQL> insert into arranges values('&owner\_id', '&pool\_id');
- SQL> insert into registered values('&customer\_id', '&pool\_id', '&location');

## OUTPUTS

```
SQL> select * from tab;
```

| TNAME      | TABTYPE | CLUSTERID |
|------------|---------|-----------|
| -----      | -----   | -----     |
| ARRANGES   | TABLE   |           |
| CAR        | TABLE   |           |
| CUSTOMER   | TABLE   |           |
| OWNER      | TABLE   |           |
| OWNS       | TABLE   |           |
| POOL       | TABLE   |           |
| PROVIDES   | TABLE   |           |
| PROVISIONS | TABLE   |           |
| REGISTERED | TABLE   |           |

```
9 rows selected.
```

```
SQL> desc owner;
```

| Name          | Null?    | Type          |
|---------------|----------|---------------|
| -----         | -----    | -----         |
| OWNER_ID      | NOT NULL | VARCHAR2(20)  |
| NAME          |          | VARCHAR2(30)  |
| AGE           |          | NUMBER(2)     |
| MOBILE_NUMBER |          | NUMBER(10)    |
| GENDER        |          | CHAR(5)       |
| ADDRESS       |          | VARCHAR2(100) |

```
SQL> insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address');
Enter value for owner_id: 1
Enter value for name: Ram
Enter value for age: 23
Enter value for mobile_number: 1234567890
Enter value for gender: male
Enter value for address: 1-13, Kukatpally, Hyderabad.
old 1: insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into owner values('1', 'Ram', 23, 1234567890, 'male', '1-13, Kukatpally, Hyderabad.')

1 row created.

SQL> insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address');
Enter value for owner_id: 2
Enter value for name: Shwetha
Enter value for age: 20
Enter value for mobile_number: 1523698455
Enter value for gender: female
Enter value for address: 1-15, Gandimaisamma, Medak.
old 1: insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into owner values('2', 'Shwetha', 20, 1523698455, 'female', '1-15, Gandimaisamma, Medak.')

1 row created.

SQL> insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address');
Enter value for owner_id: 3
Enter value for name: Ashwini
Enter value for age: 20
Enter value for mobile_number: 1485963271
Enter value for gender: female
Enter value for address: 1-14/25, Punjagutta, Hyderabad.
old 1: insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into owner values('3', 'Ashwini', 20, 1485963271, 'female', '1-14/25, Punjagutta, Hyderabad.')

1 row created.

SQL> insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address');
Enter value for owner_id: 4
Enter value for name: Shwetha
Enter value for age: 20
Enter value for mobile_number: 1478523690
Enter value for gender: female
Enter value for address: 1-20, Prashanth Nagar, Hyderabad.
old 1: insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into owner values('4', 'Shwetha', 20, 1478523690, 'female', '1-20, Prashanth Nagar, Hyderabad.')

1 row created.

SQL> insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address');
Enter value for owner_id: 5
Enter value for name: Abhi
Enter value for age: 22
Enter value for mobile_number: 5236987410
Enter value for gender: male
Enter value for address: 1-16, Shapur Nagar, Hyderabad.
old 1: insert into owner values('&owner_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into owner values('5', 'Abhi', 22, 5236987410, 'male', '1-16, Shapur Nagar, Hyderabad.')

1 row created.
```

```
SQL> select * from owner;
```

| OWNER_ID | NAME    | AGE | MOBILE_NUMBER |
|----------|---------|-----|---------------|
| 1        | Ram     | 23  | 1234567890    |
| 2        | Shwetha | 20  | 1523698455    |
| 3        | Ashwini | 20  | 1485963271    |
| 4        | Shwetha | 20  | 1478523690    |
| 5        | Abhi    | 22  | 5236987410    |

male  
1-13, Kukatpally, Hyderabad.

female  
1-15, Gandimaisamma, Medak.

| OWNER_ID | NAME    | AGE | MOBILE_NUMBER |
|----------|---------|-----|---------------|
| 3        | Ashwini | 20  | 1485963271    |
| 4        | Shwetha | 20  | 1478523690    |

female  
1-14/25, Punjagutta, Hyderabad.

female

| OWNER_ID                          | NAME | AGE | MOBILE_NUMBER |
|-----------------------------------|------|-----|---------------|
| 1-20, Prashanth Nagar, Hyderabad. |      |     |               |

1-20, Prashanth Nagar, Hyderabad.

male  
1-16, Shapur Nagar, Hyderabad.

```
SQL> desc customer;
```

| Name          | Null?    | Type          |
|---------------|----------|---------------|
| CUSTOMER_ID   | NOT NULL | VARCHAR2(20)  |
| NAME          |          | VARCHAR2(30)  |
| AGE           |          | NUMBER(2)     |
| MOBILE_NUMBER |          | NUMBER(10)    |
| GENDER        |          | CHAR(5)       |
| ADDRESS       |          | VARCHAR2(100) |

```
SQL> insert into customer values('&customer_id', '&name', &age, &mobile_number, '&gender', '&address');
Enter value for customer_id: 1001
Enter value for name: Harini
Enter value for age: 21
Enter value for mobile_number: 4789635210
Enter value for gender: female
Enter value for address: 1-30, Chintal, Hyderabad.
old 1: insert into customer values('&customer_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into customer values('1001', 'Harini', 21, 4789635210, 'female', '1-30, Chintal, Hyderabad.')

1 row created.
```

```
SQL> /
Enter value for customer_id: 1002
Enter value for name: Arjun
Enter value for age: 35
Enter value for mobile_number: 2369514780
Enter value for gender: Male
Enter value for address: 1-11, Dundigal, Medak.
old 1: insert into customer values('&customer_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into customer values('1002', 'Arjun', 35, 2369514780, 'Male', '1-11, Dundigal, Medak.')

1 row created.
```

```
SQL> /
Enter value for customer_id: 1003
Enter value for name: Raja
Enter value for age: 40
Enter value for mobile_number: 4785123609
Enter value for gender: male
Enter value for address: 1-12, Balanagar, Hyderabad.
old 1: insert into customer values('&customer_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into customer values('1003', 'Raja', 40, 4785123609, 'male', '1-12, Balanagar, Hyderabad.')

1 row created.
```

```
SQL> /
Enter value for customer_id: 1004
Enter value for name: Sona
Enter value for age: 20
Enter value for mobile_number: 2014563987
Enter value for gender: female
Enter value for address: 1-22, Miyapur, Hyderabad.
old 1: insert into customer values('&customer_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into customer values('1004', 'Sona', 20, 2014563987, 'female', '1-22, Miyapur, Hyderabad.')

1 row created.
```

```
SQL> /
Enter value for customer_id: 1005
Enter value for name: Ashwin
Enter value for age: 30
Enter value for mobile_number: 3012547896
Enter value for gender: male
Enter value for address: 1-16, JNTU, Hyderabad.
old 1: insert into customer values('&customer_id', '&name', &age, &mobile_number, '&gender', '&address')
new 1: insert into customer values('1005', 'Ashwin', 30, 3012547896, 'male', '1-16, JNTU, Hyderabad.')

1 row created.
```



```
SQL> select * from customer;
```

| CUSTOMER_ID | NAME | AGE | MOBILE_NUMBER |
|-------------|------|-----|---------------|
|-------------|------|-----|---------------|

GENDER

ADDRESS

|      |        |    |            |
|------|--------|----|------------|
| 1001 | Harini | 21 | 4789635210 |
|------|--------|----|------------|

female

1-30, Chintal, Hyderabad.

|      |       |    |            |
|------|-------|----|------------|
| 1002 | Anjun | 35 | 2369514780 |
|------|-------|----|------------|

Male

1-11, Dundigal, Medak.

| CUSTOMER_ID | NAME | AGE | MOBILE_NUMBER |
|-------------|------|-----|---------------|
|-------------|------|-----|---------------|

GENDER

ADDRESS

|      |      |    |            |
|------|------|----|------------|
| 1003 | Raja | 40 | 4785123609 |
|------|------|----|------------|

male

1-12, Balanagar, Hyderabad.

|      |      |    |            |
|------|------|----|------------|
| 1004 | Sona | 20 | 2014563987 |
|------|------|----|------------|

female

| CUSTOMER_ID | NAME | AGE | MOBILE_NUMBER |
|-------------|------|-----|---------------|
|-------------|------|-----|---------------|

GENDER

ADDRESS

1-22, Miyapur, Hyderabad.

|      |        |    |            |
|------|--------|----|------------|
| 1005 | Ashwin | 30 | 3012547896 |
|------|--------|----|------------|

male

1-16, JNTU, Hyderabad.

```
SQL> desc car;
```

| Name       | Null?    | Type         |
|------------|----------|--------------|
| CAR_NO     | NOT NULL | VARCHAR2(20) |
| MODEL      |          | VARCHAR2(20) |
| COLOR      |          | VARCHAR2(20) |
| LICENCE_NO |          | VARCHAR2(15) |

```

SQL> insert into car values('&car_no', '&model', '&color', '&licence_no');
Enter value for car_no: AP123 21
Enter value for model: Breeza
Enter value for color: Black
Enter value for licence_no: TS34567891011224
old 1: insert into car values('&car_no', '&model', '&color', '&licence_no')
new 1: insert into car values('AP123 21', 'Breeza', 'Black', 'TS34567891011224')

1 row created.

SQL> /
Enter value for car_no: Ap456 78
Enter value for model: Creta
Enter value for color: Grey
Enter value for licence_no: TS12345678911123
old 1: insert into car values('&car_no', '&model', '&color', '&licence_no')
new 1: insert into car values('Ap456 78', 'Creta', 'Grey', 'TS12345678911123')

1 row created.

SQL> /
Enter value for car_no: Ap567 89
Enter value for model: Aulto
Enter value for color: Blue
Enter value for licence_no: TS23456789101112
old 1: insert into car values('&car_no', '&model', '&color', '&licence_no')
new 1: insert into car values('Ap567 89', 'Aulto', 'Blue', 'TS23456789101112')

1 row created.

SQL> /
Enter value for car_no: Ap123 56
Enter value for model: Swift
Enter value for color: White
Enter value for licence_no: TS56789101112131
old 1: insert into car values('&car_no', '&model', '&color', '&licence_no')
new 1: insert into car values('Ap123 56', 'Swift', 'White', 'TS56789101112131')

1 row created.

SQL> /
Enter value for car_no: Ap567 12
Enter value for model: Swift
Enter value for color: Black
Enter value for licence_no: TS23789101110134
old 1: insert into car values('&car_no', '&model', '&color', '&licence_no')
new 1: insert into car values('Ap567 12', 'Swift', 'Black', 'TS23789101110134')

1 row created.

```

```
SQL> select * from car;
```

| CAR_NO   | MODEL  | COLOR | LICENCE_NO       |
|----------|--------|-------|------------------|
| AP123 21 | Breeza | Black | TS34567891011224 |
| Ap456 78 | Creta  | Grey  | TS12345678911123 |
| Ap567 89 | Aulto  | Blue  | TS23456789101112 |
| Ap123 56 | Swift  | White | TS56789101112131 |
| Ap567 12 | Swift  | Black | TS23789101110134 |

```
SQL> desc provisions;
```

| Name         | Null?    | Type         |
|--------------|----------|--------------|
| OWNER_ID     | NOT NULL | VARCHAR2(20) |
| LUGGAGE_SIZE |          | NUMBER(5)    |
| OCCUPANCY    |          | NUMBER(2)    |

```
SQL> insert into provisions values('&owner_id', &luggage_size, &occupancy);
```

```
Enter value for owner_id: 1
```

```
Enter value for luggage_size: 10
```

```
Enter value for occupancy: 3
```

```
old 1: insert into provisions values('&owner_id', &luggage_size, &occupancy)
```

```
new 1: insert into provisions values('1', 10, 3)
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for owner_id: 2
```

```
Enter value for luggage_size: 10
```

```
Enter value for occupancy: 4
```

```
old 1: insert into provisions values('&owner_id', &luggage_size, &occupancy)
```

```
new 1: insert into provisions values('2', 10, 4)
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for owner_id: 3
```

```
Enter value for luggage_size: 12
```

```
Enter value for occupancy: 2
```

```
old 1: insert into provisions values('&owner_id', &luggage_size, &occupancy)
```

```
new 1: insert into provisions values('3', 12, 2)
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for owner_id: 4
```

```
Enter value for luggage_size: 5
```

```
Enter value for occupancy: 1
```

```
old 1: insert into provisions values('&owner_id', &luggage_size, &occupancy)
```

```
new 1: insert into provisions values('4', 5, 1)
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for owner_id: 5
```

```
Enter value for luggage_size: 5
```

```
Enter value for occupancy: 2
```

```
old 1: insert into provisions values('&owner_id', &luggage_size, &occupancy)
```

```
new 1: insert into provisions values('5', 5, 2)
```

```
SQL> select * from provisions;
```

| OWNER_ID | LUGGAGE_SIZE | OCCUPANCY |
|----------|--------------|-----------|
| 1        | 10           | 3         |
| 2        | 10           | 4         |
| 3        | 12           | 2         |
| 4        | 5            | 1         |
| 5        | 5            | 2         |

```
SQL> desc pool;
```

| Name    | Null?    | Type         |
|---------|----------|--------------|
| POOL_ID | NOT NULL | VARCHAR2(20) |
| START_P |          | VARCHAR2(25) |
| END     |          | VARCHAR2(25) |
| DAY     |          | VARCHAR2(20) |
| TIMING  |          | VARCHAR2(20) |

```
SQL> insert into pool values('&pool_id', '&start_p', '&end', '&day', '&timing');
Enter value for pool_id: TS1001
Enter value for start_p: Punjagutta
Enter value for end: Balanagar
Enter value for day: 01-May-2021
Enter value for timing: 5:00PM
old 1: insert into pool values('&pool_id', '&start_p', '&end', '&day', '&timing')
new 1: insert into pool values('TS1001', 'Punjagutta', 'Balanagar', '01-May-2021', '5:00PM')

1 row created.

SQL> /
Enter value for pool_id: TS1002
Enter value for start_p: Balanagar
Enter value for end: Kukatpally
Enter value for day: 02-MAY-2021
Enter value for timing: !2:00pm
old 1: insert into pool values('&pool_id', '&start_p', '&end', '&day', '&timing')
new 1: insert into pool values('TS1002', 'Balanagar', 'Kukatpally', '02-MAY-2021', '!2:00pm')

1 row created.

SQL> /
Enter value for pool_id: TS1003
Enter value for start_p: Miyapur
Enter value for end: chinthal
Enter value for day: 01-MAY-2021
Enter value for timing: 7:00AM
old 1: insert into pool values('&pool_id', '&start_p', '&end', '&day', '&timing')
new 1: insert into pool values('TS1003', 'Miyapur', 'chinthal', '01-MAY-2021', '7:00AM')

1 row created.

SQL> /
Enter value for pool_id: TS1004
Enter value for start_p: Gandimaisamma
Enter value for end: Begumpet
Enter value for day: 03-MAY-2021
Enter value for timing: 9:00PM
old 1: insert into pool values('&pool_id', '&start_p', '&end', '&day', '&timing')
new 1: insert into pool values('TS1004', 'Gandimaisamma', 'Begumpet', '03-MAY-2021', '9:00PM')

1 row created.
```

```

SQL> /
Enter value for pool_id: TS1005
Enter value for start_p: GAndimaisamma
Enter value for end: Balanagar
Enter value for day: 01-MAY-2021
Enter value for timing: 10:30AM
old 1: insert into pool values('&pool_id', '&start_p', '&end', '&day', '&timing')
new 1: insert into pool values('TS1005', 'GAndimaisamma', 'Balanagar', '01-MAY-2021', '10:30AM')

1 row created.

SQL> select * from pool;

POOL_ID          START_P          END
-----
DAY            TIMING
-----
TS1001          Punjagutta      Balanagar
01-May-2021     5:00PM
TS1002          Balanagar       Kukatpally
02-MAY-2021     12:00pm
TS1003          Miyapur         chinthal
01-MAY-2021     7:00AM

POOL_ID          START_P          END
-----
DAY            TIMING
-----
TS1004          Gandimaisamma   Begumpet
03-MAY-2021     9:00PM
TS1005          GAndimaisamma   Balanagar
01-MAY-2021     10:30AM

```

```
SQL> desc owns;
Name                                     Null?   Type
-----
OWNER_ID                               NOT NULL VARCHAR2(20)
CAR_NO                                 NOT NULL VARCHAR2(20)
```

```
SQL> insert into owns values('&owner_id', '&car_no');
Enter value for owner_id: 1
Enter value for car_no: AP123 21
old 1: insert into owns values('&owner_id', '&car_no')
new 1: insert into owns values('1', 'AP123 21')
```

1 row created.

```
SQL> /
Enter value for owner_id: 2
Enter value for car_no: Ap567 12
old 1: insert into owns values('&owner_id', '&car_no')
new 1: insert into owns values('2', 'Ap567 12')
```

1 row created.

```
SQL> /
Enter value for owner_id: 3
Enter value for car_no: Ap123 56
old 1: insert into owns values('&owner_id', '&car_no')
new 1: insert into owns values('3', 'Ap123 56')
```

1 row created.

```
SQL> /
Enter value for owner_id: 4
Enter value for car_no: Ap567 89
old 1: insert into owns values('&owner_id', '&car_no')
new 1: insert into owns values('4', 'Ap567 89')
```

1 row created.

```
SQL> /
Enter value for owner_id: 5
Enter value for car_no: Ap456 78
old 1: insert into owns values('&owner_id', '&car_no')
new 1: insert into owns values('5', 'Ap456 78')
```

1 row created.

```
SQL> select * from owns;
```

| OWNER_ID | CAR_NO   |
|----------|----------|
| 1        | AP123 21 |
| 2        | Ap567 12 |
| 3        | Ap123 56 |
| 4        | Ap567 89 |
| 5        | Ap456 78 |

```
SQL> desc provides;
Name                                     Null?   Type
-----
POOL_ID                                NOT NULL VARCHAR2(20)
OWNER_ID                               NOT NULL VARCHAR2(20)
```

```
SQL> insert into provides values('&pool_id', '&owner_id');
Enter value for pool_id: TS1001
Enter value for owner_id: 5
old 1: insert into provides values('&pool_id', '&owner_id')
new 1: insert into provides values('TS1001', '5')
```

1 row created.

```
SQL> /
Enter value for pool_id: TS1002
Enter value for owner_id: 4
old 1: insert into provides values('&pool_id', '&owner_id')
new 1: insert into provides values('TS1002', '4')
```

1 row created.

```
SQL> /
Enter value for pool_id: TS1003
Enter value for owner_id: 3
old 1: insert into provides values('&pool_id', '&owner_id')
new 1: insert into provides values('TS1003', '3')
```

1 row created.

```
SQL> /
Enter value for pool_id: TS1004
Enter value for owner_id: 2
old 1: insert into provides values('&pool_id', '&owner_id')
new 1: insert into provides values('TS1004', '2')
```

1 row created.

```
SQL> /
Enter value for pool_id: TS1005
Enter value for owner_id: 1
old 1: insert into provides values('&pool_id', '&owner_id')
new 1: insert into provides values('TS1005', '1')
```

1 row created.

```
SQL> select * from provides;
```

| POOL_ID | OWNER_ID |
|---------|----------|
| TS1005  | 1        |
| TS1004  | 2        |
| TS1003  | 3        |
| TS1002  | 4        |
| TS1001  | 5        |



```
SQL> desc arranges;
```

| Name     | Null?    | Type         |
|----------|----------|--------------|
| OWNER_ID | NOT NULL | VARCHAR2(20) |
| POOL_ID  | NOT NULL | VARCHAR2(20) |

```
SQL> insert into arranges values('&owner_id', '&pool_id');
```

```
Enter value for owner_id: 3
```

```
Enter value for pool_id: TS1003
```

```
old 1: insert into arranges values('&owner_id', '&pool_id')
```

```
new 1: insert into arranges values('3', 'TS1003')
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for owner_id: 5
```

```
Enter value for pool_id: TS1001
```

```
old 1: insert into arranges values('&owner_id', '&pool_id')
```

```
new 1: insert into arranges values('5', 'TS1001')
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for owner_id: 1
```

```
Enter value for pool_id: TS1005
```

```
old 1: insert into arranges values('&owner_id', '&pool_id')
```

```
new 1: insert into arranges values('1', 'TS1005')
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for owner_id: 4
```

```
Enter value for pool_id: TS1002
```

```
old 1: insert into arranges values('&owner_id', '&pool_id')
```

```
new 1: insert into arranges values('4', 'TS1002')
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for owner_id: 2
```

```
Enter value for pool_id: TS1004
```

```
old 1: insert into arranges values('&owner_id', '&pool_id')
```

```
new 1: insert into arranges values('2', 'TS1004')
```

```
1 row created.
```

```
SQL> select * from arranges;
```

| OWNER_ID | POOL_ID |
|----------|---------|
| 1        | TS1005  |
| 2        | TS1004  |
| 3        | TS1003  |
| 4        | TS1002  |
| 5        | TS1001  |

```
SQL> desc registered;
```

| Name        | Null?    | Type          |
|-------------|----------|---------------|
| CUSTOMER_ID | NOT NULL | VARCHAR2(20)  |
| POOL_ID     | NOT NULL | VARCHAR2(20)  |
| LOCATION    |          | VARCHAR2(100) |

```
SQL> insert into registered values('&customer_id', '&pool_id', '&location');
```

```
Enter value for customer_id: 1001
```

```
Enter value for pool_id: TS1004
```

```
Enter value for location: Gandimaisamma signal
```

```
old 1: insert into registered values('&customer_id', '&pool_id', '&location')
```

```
new 1: insert into registered values('1001', 'TS1004', 'Gandimaisamma signal')
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for customer_id: 1002
```

```
Enter value for pool_id: TS1005
```

```
Enter value for location: Shapur Nagar Bustop
```

```
old 1: insert into registered values('&customer_id', '&pool_id', '&location')
```

```
new 1: insert into registered values('1002', 'TS1005', 'Shapur Nagar Bustop')
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for customer_id: 1003
```

```
Enter value for pool_id: TS1001
```

```
Enter value for location: SR nagar metro station towards Balanagar
```

```
old 1: insert into registered values('&customer_id', '&pool_id', '&location')
```

```
new 1: insert into registered values('1003', 'TS1001', 'SR nagar metro station towards Balanagar')
```

```
1 row created.
```

```
SQL> /
```

```
Enter value for customer_id: 1004
```

```
Enter value for pool_id: TS1002
```

```
Enter value for location: Prashanth Nagar, Near Y-Junction
```

```
old 1: insert into registered values('&customer_id', '&pool_id', '&location')
```

```
new 1: insert into registered values('1004', 'TS1002', 'Prashanth Nagar, Near Y-Junction')
```

```
1 row created.
```

```

SQL> /
Enter value for customer_id: 1005
Enter value for pool_id: TS1003
Enter value for location: JNTU college
old 1: insert into registered values('&customer_id', '&pool_id', '&location')
new 1: insert into registered values('1005', 'TS1003', 'JNTU college')

1 row created.

SQL> select * from registered;

CUSTOMER_ID          POOL_ID
-----
LOCATION
-----
1001                TS1004
Gandimaisamma signal

1002                TS1005
Shapur Nagar Bustop

1003                TS1001
SR nagar metro station towards Balanagar

CUSTOMER_ID          POOL_ID
-----
LOCATION
-----
1004                TS1002
Prashanth Nagar, Near Y-Junction

1005                TS1003
JNTU college

```

## IMPLEMENTATION

## Front end programs and its connectivity:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

```
public void connectToDB()
{
    try
    {
        Connection
        con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1
        521:xe","Shwetha","vasavi");
        statement=con.createStatement();
        statement.executeUpdate("commit");
    }
    catch (SQLException connectException)
    {
        System.out.println(connectException.getMessage());
        System.out.println(connectException.getSQLState());
        System.out.println(connectException.getErrorCode());
        System.exit(1);
    }
}
```

Thus, the connection from Java to Oracle database is performed and therefore, can be used for updating tables in the database directly.

## Java Code:

```
package DBMS;
import java.awt.*;
```

```

import java.awt.event.*;
import java.sql.*;
@SuppressWarnings("serial")
public class InsertTables extends Frame implements ActionListener
{
    MenuBar mb;
    MenuItem
m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16,m17,m18,m19;
    Menu owner,customer,pool,arranges,registered;
    Button insertButton,submit;
    TextField owner_idText, nameText, ageText, mobile_numberText, genderText,
addressText;
    TextField customer_idText, NameText, AgeText, Mobile_NumberText,
GenderText, AddressText;
    TextField pool_idText, start_pText, endText, dayText, timingText;
    TextField Owner_idText, Pool_idText;
    TextField Customer_IdText, Pool_IdText, locationText;
    TextArea errorText;
    Connection connection;
    Statement statement;
    Button modify;
    List ownerList,customerList,poolList,arrangesList,registeredList;
    ResultSet rs;
    Button deleteRowButton;
    public InsertTables()
    {
        try
        {
            Class.forName ("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Unable to find and load driver");
            System.exit(1);
        }
        connectToDB ();
    }
    public void connectToDB()
    {

```

```

        try
        {
connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","Shwetha","v
asavi");

            statement = connection.createStatement();

        }
        catch (SQLException connectException)
        {
            System.out.println(connectException.getMessage());
            System.out.println(connectException.getSQLState());
            System.out.println(connectException.getErrorCode());
            System.exit(1);
        }
    }

    public void buildFrame()
    {
        //Basic Frame Properties
        setTitle("Car Pooling");
        setSize(500, 600);
        setVisible(true);

        //menubar
        mb = new MenuBar();
        setMenuBar(mb);
        setSize(550,500);
        setLayout(null);
        setVisible(true);
        owner=new Menu("Owner");
        m1=new MenuItem("Insert Owner");
        m2=new MenuItem("Update Owner");
        m3=new MenuItem("Delete Owner");
        m4=new MenuItem("View Owner");
        owner.add(m1);
        owner.add(m2);
        owner.add(m3);
        owner.add(m4);
        mb.add(owner);
    }
}

```

```
customer=new Menu("Customer");
m5=new MenuItem("Insert Customer");
m6=new MenuItem("Update Customer");
m7=new MenuItem("Delete Customer");
m8=new MenuItem("View Customer");
customer.add(m5);
customer.add(m6);
customer.add(m7);
customer.add(m8);
mb.add(customer);
pool=new Menu("Pool");
m9=new MenuItem("Insert Pool");
m10=new MenuItem("Update Pool");
m11=new MenuItem("Delete Pool");
m12=new MenuItem("View Pool");
pool.add(m9);
pool.add(m10);
pool.add(m11);
pool.add(m12);
mb.add(pool);
arranges=new Menu("Arranges");
m13=new MenuItem("Insert Arranges");
m14=new MenuItem("Delete Arranges");
m15=new MenuItem("View Arranges");
arranges.add(m13);
arranges.add(m14);
arranges.add(m15);
mb.add(arranges);
registered=new Menu("Registered");
m16=new MenuItem("Insert Registerd");
m17=new MenuItem("Update Registered");
m18=new MenuItem("Delete Registered");
m19=new MenuItem("View Registered");
registered.add(m16);
registered.add(m17);
registered.add(m18);
registered.add(m19);
mb.add(registered);
```

```
m1.addActionListener(this);
m2.addActionListener(this);
m3.addActionListener(this);
m4.addActionListener(this);
m5.addActionListener(this);
m6.addActionListener(this);
m7.addActionListener(this);
m8.addActionListener(this);
m9.addActionListener(this);
m10.addActionListener(this);
m11.addActionListener(this);
m12.addActionListener(this);
m13.addActionListener(this);
m14.addActionListener(this);
m15.addActionListener(this);
m16.addActionListener(this);
m17.addActionListener(this);
m18.addActionListener(this);
m19.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae)
    {
        String arg = ae.getActionCommand();
        if(arg.equals("Insert Owner"))
            this.buildGUIOwner();
        if(arg.equals("Update Owner"))
            this.updateGUIOwner();
        if(arg.equals("Delete Owner"))
            this.deleteGUIOwner();
        if(arg.equals("View Owner"))
            this.viewGUIOwner();
        if(arg.equals("Insert Customer"))
            this.buildGUICustomer();
        if(arg.equals("Update Customer "))
            this.updateGUICustomer();
        if(arg.equals("Delete Customer"))
            this.deleteGUICustomer();
        if(arg.equals("View Customer"))
```



```

        this.viewGUICustomer();
    if(arg.equals("Insert Pool"))
        this.buildGUIPool();
    if(arg.equals("Update Pool"))
        this.updateGUIPool();
    if(arg.equals("Delete Pool"))
        this.deleteGUIPool();
    if(arg.equals("View Pool"))
        this.viewGUIPool();
    if(arg.equals("Insert Arranges"))
        this.buildGUIArranges();
    if(arg.equals("Delete Arranges"))
        this.deleteGUIArranges();
    if(arg.equals("View Arranges"))
        this.viewGUIArranges();
    if(arg.equals("Insert Registered"))
        this.buildGUIRegistered();
    if(arg.equals("Delete Registered"))
        this.deleteGUIRegistered();
    if(arg.equals("Update Registered"))
        this.updateGUIRegistered();
    if(arg.equals("View Registered"))
        this.viewGUIRegistered();
}
public void buildGUIOwner()
{
    removeAll();
    //Handle Insert Account Button
    insertButton = new Button("Submit");
    insertButton.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
                String query= "INSERT INTO owner VALUES('"+
owner_idText.getText() + "', " + "'" + nameText.getText() + "', " + "'" + ageText.getText() + "', " +
"'" + mobile_numberText.getText() + "', " + "'" + genderText.getText() + "', " + "'" +
addressText.getText() + "'" + ")";

```

```

        int i = statement.executeUpdate(query);
        errorText.append("\nInserted " + i + " rows successfully");
    }
    catch (SQLException insertException)
    {
        displaySQLErrors(insertException);
    }
}

});
owner_idText = new TextField(15);
nameText = new TextField(15);
ageText = new TextField(15);
mobile_numberText= new TextField(15);
genderText = new TextField(15);
addressText = new TextField(15);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Owner ID:"));
first.add(owner_idText);
first.add(new Label("Owner Name:"));
first.add(nameText);
first.add(new Label("Age:"));
first.add(ageText);
first.add(new Label("Mobile_Number"));
first.add(mobile_numberText);
first.add(new Label("Gender:"));
first.add(genderText);
first.add(new Label("Address:"));
first.add(addressText);
first.setBounds(125,90,200,100);
Panel second = new Panel(new GridLayout(4, 1));
second.add(insertButton);
second.setBounds(125,220,150,100);
Panel third = new Panel();
third.add(errorText);
third.setBounds(125,320,300,200);

```

```

        add(first);
        add(second);
        add(third);setLayout(new FlowLayout());
        setVisible(true);
    }
    public void buildGUICustomer()
    {
        removeAll();
        //Handle Insert Account Button
        insertButton = new Button("Submit");
        insertButton.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                try
                {
                    String query= "INSERT INTO customer VALUES("+
customer_idText.getText() + "," + "" + NameText.getText() + "," + "" + AgeText.getText() +
", " + "" + Mobile_NumberText.getText() + "," + "" + GenderText.getText() + "," + "" +
AddressText.getText() + "" + ")";

                    int i = statement.executeUpdate(query);
                    errorText.append("\nInserted " + i + " rows successfully");
                }
                catch (SQLException insertException)
                {
                    displaySQLErrors(insertException);
                }
            }
        });
        customer_idText = new TextField(15);
        NameText = new TextField(15);
        AgeText = new TextField(15);
        Mobile_NumberText= new TextField(15);
        GenderText = new TextField(15);
        AddressText = new TextField(15);
        errorText = new TextArea(10, 40);
        errorText.setEditable(false);
        Panel first = new Panel();
        first.setLayout(new GridLayout(4, 2));
    }

```

```

first.add(new Label("Customer ID:"));
first.add(customer_idText);
first.add(new Label("Customer Name:"));
first.add(NameText);
first.add(new Label("Age:"));
first.add(AgeText);
first.add(new Label("Mobile_Number"));
first.add(Mobile_NumberText);
first.add(new Label("Gender:"));
first.add(GenderText);
first.add(new Label("Address:"));
first.add(AddressText);
first.setBounds(125,90,200,100);
Panel second = new Panel(new GridLayout(4, 1));
second.add(insertButton);
second.setBounds(125,220,150,100);
Panel third = new Panel();
third.add(errorText);
third.setBounds(125,320,300,200);
add(first);
add(second);
add(third);setLayout(new FlowLayout());
setVisible(true);
}
public void buildGUIPool()
{
    removeAll();
    //Handle Insert Account Button
    insertButton = new Button("Submit");
    insertButton.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
                String query= "INSERT INTO pool VALUES('"+
pool_idText.getText() + "', " + "'" + start_pText.getText() + "'," + "'" + endText.getText() + "'," +
"'" + dayText.getText() + "'," + "'" + timingText.getText() + "'"");
                int i = statement.executeUpdate(query);
            }
            catch (SQLException ex)
            {
                JOptionPane.showMessageDialog(this, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    });
}
}

```

```

        errorText.append("\nInserted " + i + " rows successfully");
    }
    catch (SQLException insertException)
    {
        displaySQLErrors(insertException);
    }
}

});
pool_idText = new TextField(15);
start_pText = new TextField(15);
endText = new TextField(15);
dayText= new TextField(15);
timingText = new TextField(15);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Pool ID:"));
first.add(pool_idText);
first.add(new Label("Start Point:"));
first.add(start_pText);
first.add(new Label("End Point:"));
first.add(endText);
first.add(new Label("Day:"));
first.add(dayText);
first.add(new Label("Timings:"));
first.add(timingText);
first.setBounds(125,90,200,100);
Panel second = new Panel(new GridLayout(4, 1));
second.add(insertButton);
second.setBounds(125,220,150,100);
Panel third = new Panel();
third.add(errorText);
third.setBounds(125,320,300,200);
add(first);
add(second);
add(third);setLayout(new FlowLayout());
setVisible(true);

```

```

    }
    public void buildGUIArranges()
    {
        removeAll();
        //Handle Insert Account Button
        insertButton = new Button("Submit");
        insertButton.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                try
                {
                    String query= "INSERT INTO arranges VALUES('"+
Owner_idText.getText() + "', '"+Pool_idText.getText() + "')";
                    int i = statement.executeUpdate(query);
                    errorText.append("\nInserted " + i + " rows successfully");
                }
                catch (SQLException insertException)
                {
                    displaySQLErrors(insertException);
                }
            }
        });
        Owner_idText = new TextField(15);
        Pool_idText=new TextField(15);
        errorText = new TextArea(10, 40);
        errorText.setEditable(false);
        Panel first = new Panel();
        first.setLayout(new GridLayout(4, 2));
        first.add(new Label("Owner ID:"));
        first.add(Owner_idText);
        first.add(new Label("Pool ID:"));
        first.add(Pool_idText);
        first.setBounds(125,90,200,100);
        Panel second = new Panel(new GridLayout(4, 1));
        second.add(insertButton);
        second.setBounds(125,220,150,100);
        Panel third = new Panel();
        third.add(errorText);
    }
}

```

```

third.setBounds(125,320,300,200);
add(first);
add(second);
add(third);setLayout(new FlowLayout());
setVisible(true);
}
public void buildGUIRegistered()
{
    removeAll();
    registeredList = new List(6);
    loadRegistered();
    add(registeredList);

    //When a list item is selected populate the text fields
    registeredList.addItemListener(new ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {
            try
            {
                rs = statement.executeQuery("SELECT * FROM
registered");
                while (rs.next())
                {
                    if
(rs.getString("customer_id").equals(registeredList.getSelectedItem()))
                        break;
                }
                if (!rs.isAfterLast())
                {
                    Customer_IdText.setText(rs.getString("customer_id"));
                }
            }
            catch (SQLException selectException)
            {
                displaySQLErrors(selectException);
            }
        }
    }
}

```

```

});
//Handle Insert Account Button
insertButton = new Button("Submit");
insertButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            String query= "INSERT INTO registered VALUES('"+
Customer_IdText.getText() + "', '"+Pool_IdText.getText() + "', '"+locationText.getText() + "')";
            int i = statement.executeUpdate(query);
            errorText.append("\nInserted " + i + " rows successfully");
        }
        catch (SQLException insertException)
        {
            displaySQLErrors(insertException);
        }
    }
});

Customer_IdText = new TextField(15);
Customer_IdText.setEditable(false);
Pool_IdText=new TextField(15);
locationText=new TextField(15);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Customer ID:"));
first.add(Customer_IdText);
first.add(new Label("Pool ID:"));
first.add(Pool_IdText);
first.add(new Label("Location:"));
first.add(locationText);
first.setBounds(125,90,200,100);
Panel second = new Panel(new GridLayout(4, 1));
second.add(insertButton);
second.setBounds(125,220,150,100);
Panel third = new Panel();

```



```
third.add(errorText);
third.setBounds(125,320,300,200);
add(first);
add(second);
add(third);
setLayout(new FlowLayout());
setVisible(true);
}
private void loadMenu()
{
    try
    {
        rs = statement.executeQuery("SELECT * FROM owner");
        while (rs.next())
        {
            ownerList.add(rs.getString("owner_id"));
        }
    }
    catch (SQLException e)
    {
        displaySQLErrors(e);
    }
}
public void updateGUIOwner()
{
    removeAll();
    ownerList = new List(6);
    loadMenu();
    add(ownerList);
    ownerList.addItemListener(new ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {
            try
            {
                rs = statement.executeQuery("SELECT * FROM owner");
                while (rs.next())
                {
```

```

        if
(rs.getString("owner_id").equals(ownerList.getSelectedItem()))
            break;
    }
    if (!rs.isAfterLast())
    {
        owner_idText.setText(rs.getString("owner_id"));
        nameText.setText(rs.getString("name"));
        ageText.setText(rs.getString("age"));

        mobile_numberText.setText(rs.getString("mobile_number"));
        genderText.setText(rs.getString("gender"));
        addressText.setText(rs.getString("address"));
    }
}
catch (SQLException selectException)
{
    displaySQLErrors(selectException);
}
});
modify = new Button("Modify");
modify.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("UPDATE owner "
            + "SET name=" + "" + nameText.getText()
+ "" + "gender=" + genderText.getText() + "" + "mobile_number=" + mobile_numberText.getText()
+ "" + "age=" + ageText.getText() + "" + "address=" + addressText.getText() + ""
            + " WHERE owner_id = " + ownerList.getSelectedItem() +
            "");
            errorText.append("\nUpdated " + i + " rows successfully");
            ownerList.removeAll();
            loadMenu();
        }
        catch (SQLException insertException)

```

```
        {
            displaySQLErrors(insertException);
        }
    }

});
owner_idText = new TextField(15);
owner_idText.setEditable(false);
nameText = new TextField(15);
ageText = new TextField(15);
//ageText.setEditable(false);
mobile_numberText = new TextField(15);
//mobile_numberText.setEditable(false);
genderText = new TextField(15);
//genderText.setEditable(false);
addressText = new TextField(15);
//addressText.setEditable(false);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Owner ID:"));
first.add(owner_idText);
first.add(new Label("Name:"));
first.add(nameText);
first.add(new Label("Age"));
first.add(ageText);
first.add(new Label("Mobile Number:"));
first.add(mobile_numberText);
first.add(new Label("Gender:"));
first.add(genderText);
first.add(new Label("Address:"));
first.add(addressText);
Panel second = new Panel(new GridLayout(4, 1));
second.add(modify);

Panel third = new Panel();
third.add(errorText);
```

```

        add(first);

        add(second);
        add(third);

        //setTitle("Update ....");
        //setSize(500, 600);
        setLayout(new FlowLayout());
        setVisible(true);
    }
    public void deleteGUIOwner()
    {
        removeAll();
        ownerList = new List(10);
        loadMenu();
        add(ownerList);

        //When a list item is selected populate the text fields
        ownerList.addItemListener(new ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                try
                {
                    rs = statement.executeQuery("SELECT * FROM owner");
                    while (rs.next())
                    {
                        if
(rs.getString("owner_id").equals(ownerList.getSelectedItem()))
                            break;
                    }
                    if (!rs.isAfterLast())
                    {
                        owner_idText.setText(rs.getString("owner_id"));
                        nameText.setText(rs.getString("name"));
                        ageText.setText(rs.getString("age"));

mobile_numberText.setText(rs.getString("mobile_number"));
                        genderText.setText(rs.getString("gender"));
                    }
                }
            }
        });
    }
}

```

```

        addressText.setText(rs.getString("address"));
    }
}
catch (SQLException selectException)
{
    displaySQLErrors(selectException);
}
}
});
deleteRowButton = new Button("Delete Row");
deleteRowButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("DELETE FROM owner
WHERE owner_id = '" + ownerList.getSelectedItem()+"'");
            errorText.append("\nDeleted " + i + " rows successfully");
            owner_idText.setText(null);
            nameText.setText(null);
            ageText.setText(null);
            mobile_numberText.setText(null);
            genderText.setText(null);
            addressText.setText(null);
            ownerList.removeAll();
            loadMenu();
        }
        catch (SQLException deleteException)
        {
            displaySQLErrors(deleteException);
        }
    }
});
owner_idText = new TextField(15);
nameText = new TextField(15);
ageText = new TextField(15);
mobile_numberText = new TextField(15);

```

```
genderText = new TextField(15);
addressText = new TextField(15);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
owner_idText.setEditable(false);
nameText.setEditable(false);
ageText.setEditable(false);
mobile_numberText.setEditable(false);
genderText.setEditable(false);
addressText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Owner ID:"));
first.add(owner_idText);
first.add(new Label("Name:"));
first.add(nameText);
first.add(new Label("Age:"));
first.add(ageText);
first.add(new Label("Mobile Number:"));
first.add(mobile_numberText);
first.add(new Label("Gender:"));
first.add(genderText);
first.add(new Label("Address:"));
first.add(addressText);
Panel second = new Panel(new GridLayout(4, 1));
second.add(deleteRowButton);
Panel third = new Panel();
third.add(errorText);
add(first);
add(second);
add(third);

setLayout(new FlowLayout());
setVisible(true);

}
public void viewGUIOwner()
```

```

{
    removeAll();
    ownerList = new List(6);
    loadMenu();
    add(ownerList);

    //When a list item is selected populate the text fields
    ownerList.addItemListener(new ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {
            try
            {
                rs = statement.executeQuery("SELECT * FROM owner");
                while (rs.next())
                {
                    if
(rs.getString("owner_id").equals(ownerList.getSelectedItem()))
                        break;
                }
                if (!rs.isAfterLast())
                {
                    owner_idText.setText(rs.getString("owner_id"));
                    nameText.setText(rs.getString("name"));
                    ageText.setText(rs.getString("age"));

                    mobile_numberText.setText(rs.getString("mobile_number"));
                    genderText.setText(rs.getString("gender"));
                    addressText.setText(rs.getString("address"));
                }
            }
            catch (SQLException selectException)
            {
                displaySQLErrors(selectException);
            }
        }
    });
    //Handle Update Menu Button
    modify = new Button("Update Owner");

```

```

modify.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("UPDATE owner "
            + "SET name=" + nameText.getText()
            + " WHERE owner_id = " + ownerList.getSelectedItemId() +
            "");

            errorText.append("\nUpdated " + i + " rows successfully");
            ownerList.removeAll();
            loadMenu();
        }
        catch (SQLException insertException)
        {
            displaySQLErrors(insertException);
        }
    }
});

owner_idText = new TextField(15);
owner_idText.setEditable(false);
nameText = new TextField(15);
nameText.setEditable(false);
ageText = new TextField(15);
ageText.setEditable(false);
mobile_numberText = new TextField(15);
mobile_numberText.setEditable(false);
genderText = new TextField(15);
genderText.setEditable(false);
addressText = new TextField(15);
addressText.setEditable(false);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Owner ID:"));
first.add(owner_idText);

```



```

first.add(new Label("Name:"));
first.add(nameText);
first.add(new Label("Age:"));
first.add(ageText);
first.add(new Label("Mobile Number:"));
first.add(mobile_numberText);
first.add(new Label("Gender:"));
first.add(genderText);
first.add(new Label("Address:"));
first.add(addressText);

Panel second = new Panel(new GridLayout(4, 1));
//second.add(modify);

Panel third = new Panel();
third.add(errorText);

add(first);

add(second);
add(third);

//setTitle("Update ....");
//setSize(500, 600);
setLayout(new FlowLayout());
setVisible(true);

}
private void loadCustomer()
{
    try
    {
        rs = statement.executeQuery("SELECT * FROM customer");
        while (rs.next())
        {
            customerList.add(rs.getString("customer_id"));
        }
    }
}

```

```

        catch (SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void updateGUICustomer()
    {
        removeAll();
        customerList = new List(6);
        loadCustomer();
        add(customerList);
        customerList.addItemListener(new ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                try
                {
                    rs = statement.executeQuery("SELECT * FROM
customer");
                    while (rs.next())
                    {
                        if
(rs.getString("customer_id").equals(customerList.getSelectedItem()))
                        break;
                    }
                    if (!rs.isAfterLast())
                    {
                        customer_idText.setText(rs.getString("customer_id"));
                        NameText.setText(rs.getString("name"));
                        AgeText.setText(rs.getString("age"));

                        Mobile_NumberText.setText(rs.getString("mobile_number"));
                        GenderText.setText(rs.getString("gender"));
                        AddressText.setText(rs.getString("address"));
                    }
                }
            }
        });
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }

```

```

        }
    }
});
modify = new Button("Modify");
modify.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("UPDATE customer "
            + "SET name=" + "" + NameText.getText()
            + "" + "gender=" + GenderText.getText() + "" + "mobile_number=" + Mobile_NumberText.getText(
            + "" + "age=" + AgeText.getText() + "" + "address=" + AddressText.getText() + ""
            + " WHERE customer_id = " +
            customerList.getSelectedItem() + "");
            errorText.append("\nUpdated " + i + " rows successfully");
            customerList.removeAll();
            loadCustomer();
        }
        catch (SQLException insertException)
        {
            displaySQLErrors(insertException);
        }
    }
});
customer_idText = new TextField(15);
customer_idText.setEditable(false);
NameText = new TextField(15);
AgeText = new TextField(15);
//ageText.setEditable(false);
Mobile_NumberText = new TextField(15);
//mobile_numberText.setEditable(false);
GenderText = new TextField(15);
//genderText.setEditable(false);
AddressText = new TextField(15);
//addressText.setEditable(false);
errorText = new TextArea(10, 40);

```

```
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Customer ID:"));
first.add(customer_idText);
first.add(new Label("Name:"));
first.add(NameText);
first.add(new Label("Age"));
first.add(AgeText);
first.add(new Label("Mobile Number:"));
first.add(Mobile_NumberText);
first.add(new Label("Gender:"));
first.add(GenderText);
first.add(new Label("Address:"));
first.add(AddressText);
Panel second = new Panel(new GridLayout(4, 1));
second.add(modify);

Panel third = new Panel();
third.add(errorText);

add(first);

add(second);
add(third);

//setTitle("Update ....");
//setSize(500, 600);
setLayout(new FlowLayout());
setVisible(true);
}
public void deleteGUICustomer()
{
    removeAll();
    customerList = new List(10);
    loadCustomer();
    add(customerList);
}
```

```

//When a list item is selected populate the text fields
customerList.addItemListener(new ItemListener()
{
    public void itemStateChanged(ItemEvent e)
    {
        try
        {
            rs = statement.executeQuery("SELECT * FROM
customer");
            while (rs.next())
            {
                if
(rs.getString("customer_id").equals(customerList.getSelectedItem()))
                break;
            }
            if (!rs.isAfterLast())
            {
                customer_idText.setText(rs.getString("customer_id"));
                NameText.setText(rs.getString("name"));
                AgeText.setText(rs.getString("age"));

                Mobile_NumberText.setText(rs.getString("mobile_number"));
                GenderText.setText(rs.getString("gender"));
                AddressText.setText(rs.getString("address"));
            }
        }
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }
});
deleteRowButton = new Button("Delete Row");
deleteRowButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {

```

```

        Statement statement = connection.createStatement();
        int i = statement.executeUpdate("DELETE FROM
customer WHERE customer_id = " + customerList.getSelectedItem()+"");
        errorText.append("\nDeleted " + i + " rows successfully");
        customer_idText.setText(null);
        NameText.setText(null);
        AgeText.setText(null);
        Mobile_NumberText.setText(null);
        GenderText.setText(null);
        AddressText.setText(null);
        customerList.removeAll();
        loadCustomer();
    }
    catch (SQLException deleteException)
    {
        displaySQLErrors(deleteException);
    }
}

});
customer_idText = new TextField(15);
NameText = new TextField(15);
AgeText = new TextField(15);
Mobile_NumberText = new TextField(15);
GenderText = new TextField(15);
AddressText = new TextField(15);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
customer_idText.setEditable(false);
NameText.setEditable(false);
AgeText.setEditable(false);
Mobile_NumberText.setEditable(false);
GenderText.setEditable(false);
AddressText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Customer ID:"));
first.add(customer_idText);
first.add(new Label("Name:"));
first.add(NameText);

```

```

first.add(new Label("Age:"));
first.add(AgeText);
first.add(new Label("Mobile Number:"));
first.add(Mobile_NumberText);
first.add(new Label("Gender:"));
first.add(GenderText);
first.add(new Label("Address:"));
first.add(AddressText);
Panel second = new Panel(new GridLayout(4, 1));
second.add(deleteRowButton);
Panel third = new Panel();
third.add(errorText);
add(first);
add(second);
add(third);

setLayout(new FlowLayout());
setVisible(true);

}
public void viewGUICustomer()
{
    removeAll();
    customerList = new List(6);
    loadCustomer();
    add(customerList);

    //When a list item is selected populate the text fields
    customerList.addItemListener(new ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {
            try
            {
                rs = statement.executeQuery("SELECT * FROM
customer");

                while (rs.next())
                {

```

```

                                if
(rs.getString("customer_id").equals(customerList.getSelectedItem()))
                                break;
                                }
                                if (!rs.isAfterLast())
                                {

customer_idText.setText(rs.getString("customer_id"));
                                NameText.setText(rs.getString("name"));
                                AgeText.setText(rs.getString("age"));

Mobile_NumberText.setText(rs.getString("mobile_number"));
                                GenderText.setText(rs.getString("gender"));
                                AddressText.setText(rs.getString("address"));

                                }
                                }
                                catch (SQLException selectException)
                                {

                                displaySQLErrors(selectException);

                                }

});
//Handle Update Menu Button
modify = new Button("Update Customer");
modify.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("UPDATE customer "
            + "SET name=" + nameText.getText()
            + " WHERE customer_id = " +
customerList.getSelectedItem() + "");
            errorText.append("\nUpdated " + i + " rows successfully");
            customerList.removeAll();
            loadCustomer();

        }
        catch (SQLException insertException)

```



```
        {
            displaySQLErrors(insertException);
        }
    }

});
customer_idText = new TextField(15);
customer_idText.setEditable(false);
NameText = new TextField(15);
NameText.setEditable(false);
AgeText = new TextField(15);
AgeText.setEditable(false);
Mobile_NumberText = new TextField(15);
Mobile_NumberText.setEditable(false);
GenderText = new TextField(15);
GenderText.setEditable(false);
AddressText = new TextField(15);
AddressText.setEditable(false);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Customer ID:"));
first.add(customer_idText);
first.add(new Label("Name:"));
first.add(NameText);
first.add(new Label("Age:"));
first.add(AgeText);
first.add(new Label("Mobile Number:"));
first.add(Mobile_NumberText);
first.add(new Label("Gender:"));
first.add(GenderText);
first.add(new Label("Address:"));
first.add(AddressText);

Panel second = new Panel(new GridLayout(4, 1));
//second.add(modify);

Panel third = new Panel();
```

```
third.add(errorText);

add(first);

add(second);
add(third);

//setTitle("Update ....");
//setSize(500, 600);
setLayout(new FlowLayout());
setVisible(true);

}
private void loadPool()
{
    try
    {
        rs = statement.executeQuery("SELECT * FROM pool");
        while (rs.next())
        {
            poolList.add(rs.getString("pool_id"));
        }
    }
    catch (SQLException e)
    {
        displaySQLErrors(e);
    }
}
public void updateGUIPool()
{
    removeAll();
    poolList = new List(6);
    loadPool();
    add(poolList);
    poolList.addItemListener(new ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {

```

```

        try
        {
            rs = statement.executeQuery("SELECT * FROM pool");
            while (rs.next())
            {
                if
(rs.getString("pool_id").equals(poolList.getSelectedItem()))
                break;
            }
            if (!rs.isAfterLast())
            {
                pool_idText.setText(rs.getString("pool_id"));
                start_pText.setText(rs.getString("start_p"));
                endText.setText(rs.getString("end"));
                dayText.setText(rs.getString("day"));
                timingText.setText(rs.getString("timing"));
            }
        }
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }

});
modify = new Button("Modify");
modify.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("UPDATE pool "
            + "SET start_p=" + "" + start_pText.getText()
            + "," + "end=" + endText.getText() + "," + "day=" + dayText.getText() + "," + "timing=" + timingText.g
            etText() + ""
            + " WHERE pool_id = " + poolList.getSelectedItem() +
            """);

            errorText.append("\nUpdated " + i + " rows successfully");
            poolList.removeAll();

```

```

        loadPool();
    }
    catch (SQLException insertException)
    {
        displaySQLErrors(insertException);
    }
}

});
pool_idText = new TextField(15);
pool_idText.setEditable(false);
start_pText = new TextField(15);
endText = new TextField(15);
//ageText.setEditable(false);
dayText = new TextField(15);
//mobile_numberText.setEditable(false);
timingText = new TextField(15);
//genderText.setEditable(false);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Pool ID:"));
first.add(pool_idText);
first.add(new Label("Start Point:"));
first.add(start_pText);
first.add(new Label("End Point"));
first.add(endText);
first.add(new Label("Day:"));
first.add(dayText);
first.add(new Label("Timings:"));
first.add(timingText);
Panel second = new Panel(new GridLayout(4, 1));
second.add(modify);

Panel third = new Panel();
third.add(errorText);

add(first);

```

```

        add(second);
        add(third);

        //setTitle("Update ....");
        //setSize(500, 600);
        setLayout(new FlowLayout());
        setVisible(true);
    }
    public void deleteGUIPool()
    {
        removeAll();
        poolList = new List(10);
        loadPool();
        add(poolList);

        //When a list item is selected populate the text fields
        poolList.addItemListener(new ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                try
                {
                    rs = statement.executeQuery("SELECT * FROM pool");
                    while (rs.next())
                    {
                        if
(rs.getString("pool_id").equals(poolList.getSelectedItem()))
                            break;
                    }
                    if (!rs.isAfterLast())
                    {
                        pool_idText.setText(rs.getString("pool_id"));
                        start_pText.setText(rs.getString("start_p"));
                        endText.setText(rs.getString("end"));
                        dayText.setText(rs.getString("day"));
                        timingText.setText(rs.getString("timing"));
                        //addressText.setText(rs.getString("address"));
                    }
                }
            }
        });
    }

```

```

        }
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }
});
deleteRowButton = new Button("Delete Row");
deleteRowButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("DELETE FROM pool
WHERE pool_id = '" + poolList.getSelectedItem()+"'");
            errorText.append("\nDeleted " + i + " rows successfully");
            pool_idText.setText(null);
            start_pText.setText(null);
            endText.setText(null);
            dayText.setText(null);
            timingText.setText(null);
            poolList.removeAll();
            loadPool();
        }
        catch (SQLException deleteException)
        {
            displaySQLErrors(deleteException);
        }
    }
});
pool_idText = new TextField(15);
start_pText = new TextField(15);
endText = new TextField(15);
dayText = new TextField(15);
timingText = new TextField(15);
//addressText = new TextField(15);
errorText = new TextArea(10, 40);

```

```

        errorText.setEditable(false);
        pool_idText.setEditable(false);
        start_pText.setEditable(false);
        endText.setEditable(false);
        dayText.setEditable(false);
        timingText.setEditable(false);
        //addressText.setEditable(false);
        Panel first = new Panel();
        first.setLayout(new GridLayout(4, 2));
        first.add(new Label("Pool ID:"));
        first.add(pool_idText);
        first.add(new Label("Starting Point:"));
        first.add(start_pText);
        first.add(new Label("End Point:"));
        first.add(endText);
        first.add(new Label("Day:"));
        first.add(dayText);
        first.add(new Label("Timings:"));
        first.add(timingText);
        Panel second = new Panel(new GridLayout(4, 1));
        second.add(deleteRowButton);
        Panel third = new Panel();
        third.add(errorText);
        add(first);
        add(second);
        add(third);

        setLayout(new FlowLayout());
        setVisible(true);
    }
    public void viewGUIPool()
    {
        removeAll();
        poolList = new List(6);
        loadPool();
        add(poolList);
    }

```

```

//When a list item is selected populate the text fields
poolList.addItemListener(new ItemListener()
{
    public void itemStateChanged(ItemEvent e)
    {
        try
        {
            rs = statement.executeQuery("SELECT * FROM pool");
            while (rs.next())
            {
                if
(rs.getString("pool_id").equals(poolList.getSelectedItem()))
                    break;
            }
            if (!rs.isAfterLast())
            {
                pool_idText.setText(rs.getString("pool_id"));
                start_pText.setText(rs.getString("start_p"));
                endText.setText(rs.getString("end"));
                dayText.setText(rs.getString("day"));
                timingText.setText(rs.getString("timing"));
            }
        }
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }
});

//Handle Update Menu Button
modify = new Button("Update Pool");
modify.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();

```



```

        int i = statement.executeUpdate("UPDATE pool "
        + "SET start_p=" + start_pText.getText()
        + " WHERE pool_id = " + poolList.getSelectedItem() +
        "");

        errorText.append("\nUpdated " + i + " rows successfully");
        poolList.removeAll();
        loadPool();
    }
    catch (SQLException insertException)
    {
        displaySQLErrors(insertException);
    }
}

});
pool_idText = new TextField(15);
pool_idText.setEditable(false);
start_pText = new TextField(15);
start_pText.setEditable(false);
endText = new TextField(15);
endText.setEditable(false);
dayText = new TextField(15);
dayText.setEditable(false);
timingText = new TextField(15);
timingText.setEditable(false);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Pool ID:"));
first.add(pool_idText);
first.add(new Label("Starting Point:"));
first.add(start_pText);
first.add(new Label("End Point:"));
first.add(endText);
first.add(new Label("Day:"));
first.add(dayText);
first.add(new Label("Timings:"));
first.add(timingText);

```

```
Panel second = new Panel(new GridLayout(4, 1));
//second.add(modify);

Panel third = new Panel();
third.add(errorText);

add(first);

add(second);
add(third);

//setTitle("Update ....");
//setSize(500, 600);
setLayout(new FlowLayout());
setVisible(true);
}
private void loadArranges()
{
    try
    {
        rs = statement.executeQuery("SELECT * FROM arranges");
        while (rs.next())
        {
            arrangesList.add(rs.getString("Owner_id"));
        }
    }
    catch (SQLException e)
    {
        displaySQLErrors(e);
    }
}
public void deleteGUIArranges()
{
    removeAll();
    arrangesList = new List(10);
    loadArranges();
    add(arrangesList);
}
```

```

//When a list item is selected populate the text fields
arrangesList.addItemListener(new ItemListener()
{
    public void itemStateChanged(ItemEvent e)
    {
        try
        {
            rs = statement.executeQuery("SELECT * FROM
arranges");
            while (rs.next())
            {
                if
(rs.getString("Owner_id").equals(arrangesList.getSelectedItem()))
                    break;
            }
            if (!rs.isAfterLast())
            {
                Owner_idText.setText(rs.getString("Owner_id"));
                Pool_idText.setText(rs.getString("Pool_id"));
            }
        }
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }
});
deleteRowButton = new Button("Delete Row");
deleteRowButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("DELETE FROM arranges
WHERE Owner_id = '" + arrangesList.getSelectedItem()+"'");
            errorText.append("\nDeleted " + i + " rows successfully");
        }
        catch (SQLException deleteException)
        {
            displaySQLErrors(deleteException);
        }
    }
});

```

```

        Owner_idText.setText(null);
        Pool_idText.setText(null);
        arrangesList.removeAll();
        loadArranges();
    }
    catch (SQLException deleteException)
    {
        displaySQLErrors(deleteException);
    }
}

});
Owner_idText = new TextField(15);
Pool_idText = new TextField(15);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Owner_idText.setEditable(false);
Pool_idText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Owner ID:"));
first.add(Owner_idText);
first.add(new Label("Pool ID:"));
first.add(Pool_idText);
Panel second = new Panel(new GridLayout(4, 1));
second.add(deleteRowButton);
Panel third = new Panel();
third.add(errorText);
add(first);
add(second);
add(third);

setLayout(new FlowLayout());
setVisible(true);

}
public void viewGUIArranges()
{

```

```

removeAll();
arrangesList = new List(6);
loadArranges();
add(arrangesList);

//When a list item is selected populate the text fields
arrangesList.addItemListener(new ItemListener()
{
    public void itemStateChanged(ItemEvent e)
    {
        try
        {
            rs = statement.executeQuery("SELECT * FROM
arranges");

            while (rs.next())
            {
                if
(rs.getString("Owner_id").equals(arrangesList.getSelectedItem()))
                break;
            }
            if (!rs.isAfterLast())
            {
                Owner_idText.setText(rs.getString("Owner_id"));
                Pool_idText.setText(rs.getString("pool_id"));
            }
        }
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }
});

//Handle Update Menu Button
/*modify = new Button("Update Arranges");
modify.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try

```

```

        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("UPDATE arranges "
            + "SET name=" + nameText.getText()
            + " WHERE owner_id = " + ownerList.getSelectedItem() +
            "");

            errorText.append("\nUpdated " + i + " rows successfully");
            ownerList.removeAll();
            loadMenu();
        }
        catch (SQLException insertException)
        {
            displaySQLErrors(insertException);
        }
    }
});*/
Owner_idText = new TextField(15);
Owner_idText.setEditable(false);
Pool_idText = new TextField(15);
Pool_idText.setEditable(false);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Owner ID:"));
first.add(Owner_idText);
first.add(new Label("Pool ID:"));
first.add(Pool_idText);

Panel second = new Panel(new GridLayout(4, 1));
//second.add(modify);

Panel third = new Panel();
third.add(errorText);

add(first);

add(second);
add(third);

```

```

        //setTitle("Update ....");
        //setSize(500, 600);
        setLayout(new FlowLayout());
        setVisible(true);
    }
    private void loadRegistered()
    {
        try
        {
            rs = statement.executeQuery("SELECT * FROM registered");
            while (rs.next())
            {
                registeredList.add(rs.getString("customer_id"));
            }
        }
        catch (SQLException e)
        {
            displaySQLErrors(e);
        }
    }
    public void updateGUIRegistered()
    {
        removeAll();
        registeredList = new List(6);
        loadRegistered();
        add(registeredList);
        registeredList.addItemListener(new ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                try
                {
                    rs = statement.executeQuery("SELECT * FROM
registered");
                    while (rs.next())
                    {

```

```

        if
(rs.getString("Customer_Id").equals(registeredList.getSelectedItem()))
            break;
        }
        if (!rs.isAfterLast())
        {
            Customer_IdText.setText(rs.getString("Customer_Id"));
            Pool_IdText.setText(rs.getString("Pool_Id"));
            locationText.setText(rs.getString("location"));
        }
    }
    catch (SQLException selectException)
    {
        displaySQLErrors(selectException);
    }
});
modify = new Button("Modify");
modify.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();
            int i = statement.executeUpdate("UPDATE registered "
            + "SET location=" + "" + locationText.getText() + ""
            + " WHERE Customer_Id = '" +
registeredList.getSelectedItem() + "'");
            errorText.append("\nUpdated " + i + " rows successfully");
            registeredList.removeAll();
            loadRegistered();
        }
        catch (SQLException insertException)
        {
            displaySQLErrors(insertException);
        }
    }
}

```



```

    });
    Customer_IdText = new TextField(15);
    Customer_IdText.setEditable(false);
    Pool_IdText = new TextField(15);
    Pool_IdText.setEditable(false);
    locationText = new TextField(15);
    errorText = new TextArea(10, 40);
    errorText.setEditable(false);
    Panel first = new Panel();
    first.setLayout(new GridLayout(4, 2));
    first.add(new Label("Customer ID:"));
    first.add(Customer_IdText);
    first.add(new Label("Pool ID:"));
    first.add(Pool_IdText);
    first.add(new Label("Location"));
    first.add(locationText);

    Panel second = new Panel(new GridLayout(4, 1));
    second.add(modify);

    Panel third = new Panel();
    third.add(errorText);

    add(first);

    add(second);
    add(third);

    //setTitle("Update ....");
    //setSize(500, 600);
    setLayout(new FlowLayout());
    setVisible(true);
}
public void deleteGUIRegistered()
{
    removeAll();
    registeredList = new List(10);
    loadRegistered();

```

```

add(registeredList);

//When a list item is selected populate the text fields
registeredList.addItemListener(new ItemListener()
{
    public void itemStateChanged(ItemEvent e)
    {
        try
        {
            rs = statement.executeQuery("SELECT * FROM
registered");

            while (rs.next())
            {
                if
(rs.getString("Customer_Id").equals(registeredList.getSelectedItem()))
                    break;
            }
            if (!rs.isAfterLast())
            {
                Customer_IdText.setText(rs.getString("customer_id"));
                Pool_IdText.setText(rs.getString("Pool_id"));
                locationText.setText(rs.getString("location"));
            }
        }
        catch (SQLException selectException)
        {
            displaySQLErrors(selectException);
        }
    }
});
deleteRowButton = new Button("Delete Row");
deleteRowButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            Statement statement = connection.createStatement();

```

```

        int i = statement.executeUpdate("DELETE FROM
registered WHERE Customer_id = " + registeredList.getSelectedItem()+"");
        errorText.append("\nDeleted " + i + " rows successfully");
        Customer_IdText.setText(null);
        Pool_IdText.setText(null);
        locationText.setText(null);
        registeredList.removeAll();
        loadRegistered();
    }
    catch (SQLException deleteException)
    {
        displaySQLErrors(deleteException);
    }
}

});
Customer_IdText = new TextField(15);
Pool_IdText = new TextField(15);
locationText = new TextField(15);
errorText = new TextArea(10, 40);
errorText.setEditable(false);
Customer_IdText.setEditable(false);
Pool_IdText.setEditable(false);
locationText.setEditable(false);
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Customer ID:"));
first.add(Customer_IdText);
first.add(new Label("Pool ID:"));
first.add(Pool_IdText);
first.add(new Label("Location:"));
first.add(locationText);
Panel second = new Panel(new GridLayout(4, 1));
second.add(deleteRowButton);
Panel third = new Panel();
third.add(errorText);
add(first);
add(second);
add(third);

```

```

        setLayout(new FlowLayout());
        setVisible(true);

    }

    public void viewGUIRegistered()
    {
        removeAll();
        registeredList = new List(6);
        loadRegistered();
        add(registeredList);

        //When a list item is selected populate the text fields
        registeredList.addItemListener(new ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                try
                {
                    rs = statement.executeQuery("SELECT * FROM
registered");

                    while (rs.next())
                    {
                        if
(rs.getString("customer_id").equals(registeredList.getSelectedItem()))
                            break;
                    }
                    if (!rs.isAfterLast())
                    {
                        Customer_IdText.setText(rs.getString("Customer_id"));
                        Pool_IdText.setText(rs.getString("pool_id"));
                        locationText.setText(rs.getString("location"));
                    }
                }
                catch (SQLException selectException)
                {
                    displaySQLErrors(selectException);
                }
            }
        });
    }

```

```

        }
    });
    //Handle Update Menu Button
    modify = new Button("Update Registered");
    modify.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
                Statement statement = connection.createStatement();
                int i = statement.executeUpdate("UPDATE registered "
                + "SET location=" + locationText.getText()
                + " WHERE Customer_Id = '" +
registeredList.getSelectedItemAt() + "'");
                errorText.append("\nUpdated " + i + " rows successfully");
                registeredList.removeAll();
                loadRegistered();
            }
            catch (SQLException insertException)
            {
                displaySQLErrors(insertException);
            }
        }
    });
    Customer_IdText = new TextField(15);
    Customer_IdText.setEditable(false);
    Pool_IdText = new TextField(15);
    Pool_IdText.setEditable(false);
    locationText = new TextField(15);
    locationText.setEditable(false);
    errorText = new TextArea(10, 40);
    errorText.setEditable(false);
    Panel first = new Panel();
    first.setLayout(new GridLayout(4, 2));
    first.add(new Label("Customer ID:"));
    first.add(Customer_IdText);
    first.add(new Label("Pool ID:"));
    first.add(Pool_IdText);

```

```

        first.add(new Label("Location:"));
        first.add(locationText);
        Panel second = new Panel(new GridLayout(4, 1));
        //second.add(modify);

        Panel third = new Panel();
        third.add(errorText);

        add(first);

        add(second);
        add(third);

        //setTitle("Update ....");
        //setSize(500, 600);
        setLayout(new FlowLayout());
        setVisible(true);
    }
    public void displaySQLExceptions(SQLException e)
    {
        errorText.append("\nSQLException: " + e.getMessage() + "\n");
        errorText.append("SQLState:    " + e.getSQLState() + "\n");
        errorText.append("VendorError: " + e.getErrorCode() + "\n");
    }

    public static void main(String[] args)
    {
        InsertTables it = new InsertTables();
        it.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        });
        it.buildFrame();
    }
}

```

## FOLDER STRUCTURE:

This PC > OS (C:) > Users > SHWETHA > eclipse-workspace > DbmsAssignment

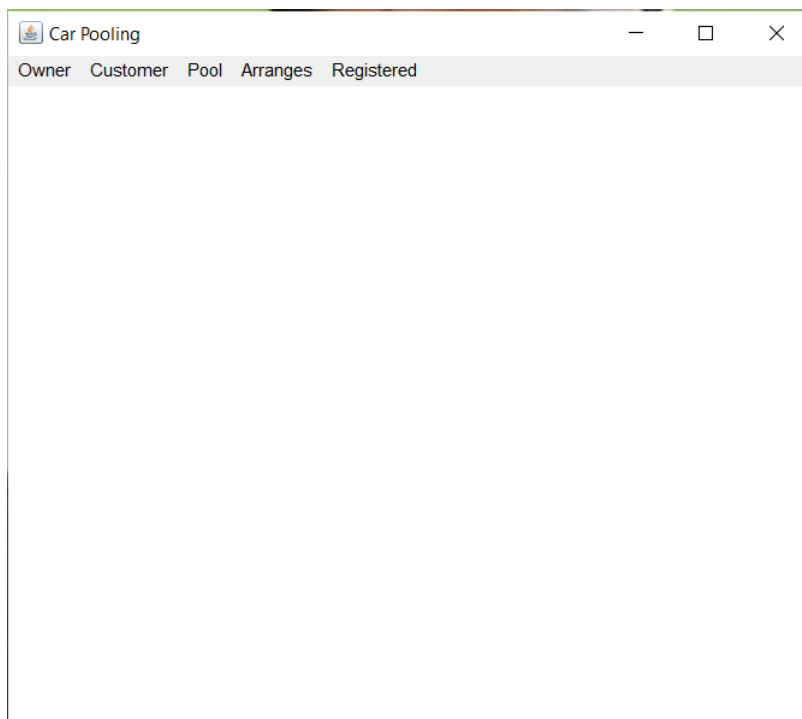
| Name       | Date modified      | Type           | Size |
|------------|--------------------|----------------|------|
| .settings  | 6/23/2021 9:49 PM  | File folder    |      |
| bin        | 6/23/2021 10:01 PM | File folder    |      |
| src        | 6/23/2021 10:01 PM | File folder    |      |
| .classpath | 6/23/2021 9:55 PM  | CLASSPATH File | 1 KB |
| .project   | 6/23/2021 9:49 PM  | PROJECT File   | 1 KB |

> This PC > OS (C:) > Users > SHWETHA > eclipse-workspace > DbmsAssignment > src > DBMS


| Name              | Date modified      | Type      | Size  |
|-------------------|--------------------|-----------|-------|
| CreateTables.java | 6/23/2021 10:00 PM | JAVA File | 1 KB  |
| InsertTables.java | 6/25/2021 10:42 PM | JAVA File | 53 KB |

## TESTING

## **OUTPUTS:**



**OWNER TABLE:**


 Car Pooling — □ ×

Owner Customer Pool Arranges Registered

|               |           |             |                                       |
|---------------|-----------|-------------|---------------------------------------|
| Owner ID:     | 101       | Owner Name: | <input type="button" value="Submit"/> |
| Shwetha       | Age:      | 20          |                                       |
| Mobile_Number | 123456789 | Gender:     |                                       |
| Female        | Address:  | Miyapur     |                                       |

Inserted 1 rows successfully



 Car Pooling

Owner Customer Pool Arranges Registered

101  
1  
2

Owner ID: 101 Name:


Age: 19

Mobile Number: 123546789 Gender:

Address:

Updated 1 rows successfully

Modify

 Car Pooling

Owner Customer Pool Arranges Registered

101  
1  
2

Owner ID: 2 Name:

Age: 20

Mobile Number: 1234567890 Gender:

Address:

Car Pooling
 —
□
×

Owner Customer Pool Arranges Registered

1

2

Owner ID:

Name:

Delete Row

Age:

Mobile Number:

Gender:

Address:

Deleted 1 rows successfully

## POOL:

Car Pooling
 —
□
×

Owner Customer Pool Arranges Registered

Pool ID:

1C

Start Point:

Submit

Miyapur

End Point:

Ameerpet

Day:

26-06-2021

Timings:

10:00Am

Inserted 1 rows successfully

Car Pooling

Owner Customer Pool Arranges Registered

1A  
1C

Pool ID: 1C Start Point: Modify

Miyapur End Point: Moosapet

Day: 26-06-2021 Timings:

10:30Am

Updated 1 rows successfully

Car Pooling

Owner Customer Pool Arranges Registered

1A  
1C

Pool ID: 1A Starting Point:

shapur End Point: ameerpeta

Day: today Timings:

timings

Car Pooling

Owner Customer Pool Arranges Registered

1A

Pool ID: Starting Point: Delete Row

End Point:

Day: Timings:

Deleted 1 rows successfully

## ARRANGES:

Car Pooling

Owner Customer Pool Arranges Registered


Owner ID: Submit

3

Pool ID:

100

SQLException: ORA-02291: integrity constraint (SHWETHA.S  
SQLState: 23000  
VendorError: 2291


 Car Pooling

Owner Customer Pool Arranges Registered

Owner ID:

Pool ID:

Inserted 1 rows successfully

 Car Pooling

Owner Customer Pool Arranges Registered

|   |                                 |
|---|---------------------------------|
| 1 | Owner ID:                       |
|   | <input type="text" value="1"/>  |
|   | Pool ID:                        |
|   | <input type="text" value="1A"/> |

Car Pooling

Owner

Customer

Pool

Arranges

Registered

Owner ID:

Pool ID:

Delete Row

Deleted 1 rows successfully

### ***REGISTERED TABLE:***

Car Pooling

Owner

Customer

Pool

Arranges

Registered

1006

Customer ID:


Pool ID:

Location:

1006

1A

hyderabad

 Car Pooling

Owner

Customer

Pool

Arranges


Registered

1006

Customer ID: 1006  
Pool ID: 1A  
Location: Secunderabad

Modify

Updated 1 rows successfully

 Car Pooling

Owner

Customer

Pool

Arranges

Registered

Customer ID:  
Pool ID:  
Location:

Delete Row

Deleted 1 rows successfully