**3 Methods in Java code**

import java.util.Scanner;

class Bill{

protected String customerName;

protected int roomNumber;

protected double fee;

protected String dateCredited;

Scanner in = new Scanner(System.in);

Bill() {

}

//Structure

protected void setBill( String customerName, int roomNumber,double fee, String dateCredited) {

this.customerName= customerName;

this.roomNumber= roomNumber;

this.fee= fee;

this.dateCredited= dateCredited;

}

public void printPayment() {

System.out.println("\r");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Customer "+this.customerName+"'s payment details\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("\r");

System.out.println("\t\t Customer name is "+ this.customerName);

System.out.println("\t\t Room Number is "+ this.roomNumber);

System.out.println("\t\t Amount is "+ this.fee+" dollors");

System.out.println("\t\t Date Credited is "+ this.dateCredited);

System.out.println("\r");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Your payment details is showing above\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

public void paymentMethod(int methodNumber) {

if (methodNumber == 1) {

System.out.println("Please Pay "+ this.fee+ " dollars"+" to the clerk");

}

else if (methodNumber == 2) {

System.out.println("Please tap or insert your credit card.");

System.out.println("Plese enter your pin.");

int p= in.nextInt();

System.out.println("Transaction Approved!");

}

}

public static void main(String[] args) {

Bill customer1= new Bill();

@SuppressWarnings("resource")

Scanner in = new Scanner(System.in);

System.out.println("Please input Customer's name");

String n = in.next();

System.out.println("Please input Room Number");

int r=in.nextInt();

System.out.println("Please input total fee");

int f= in.nextInt();

System.out.println("Please input dateCredited");

String d=in.next();

customer1.setBill(n,r,f,d);

customer1.printPayment();

int userChoice;

System.out.println("Please enter a choice for payment method: ");

System.out.println("1. Cash");

System.out.println("2. Credit");

userChoice= in.nextInt();

switch(userChoice) {

case 1:

customer1.paymentMethod(1);

break;

case 2:

customer1.paymentMethod(2);

break;

default:

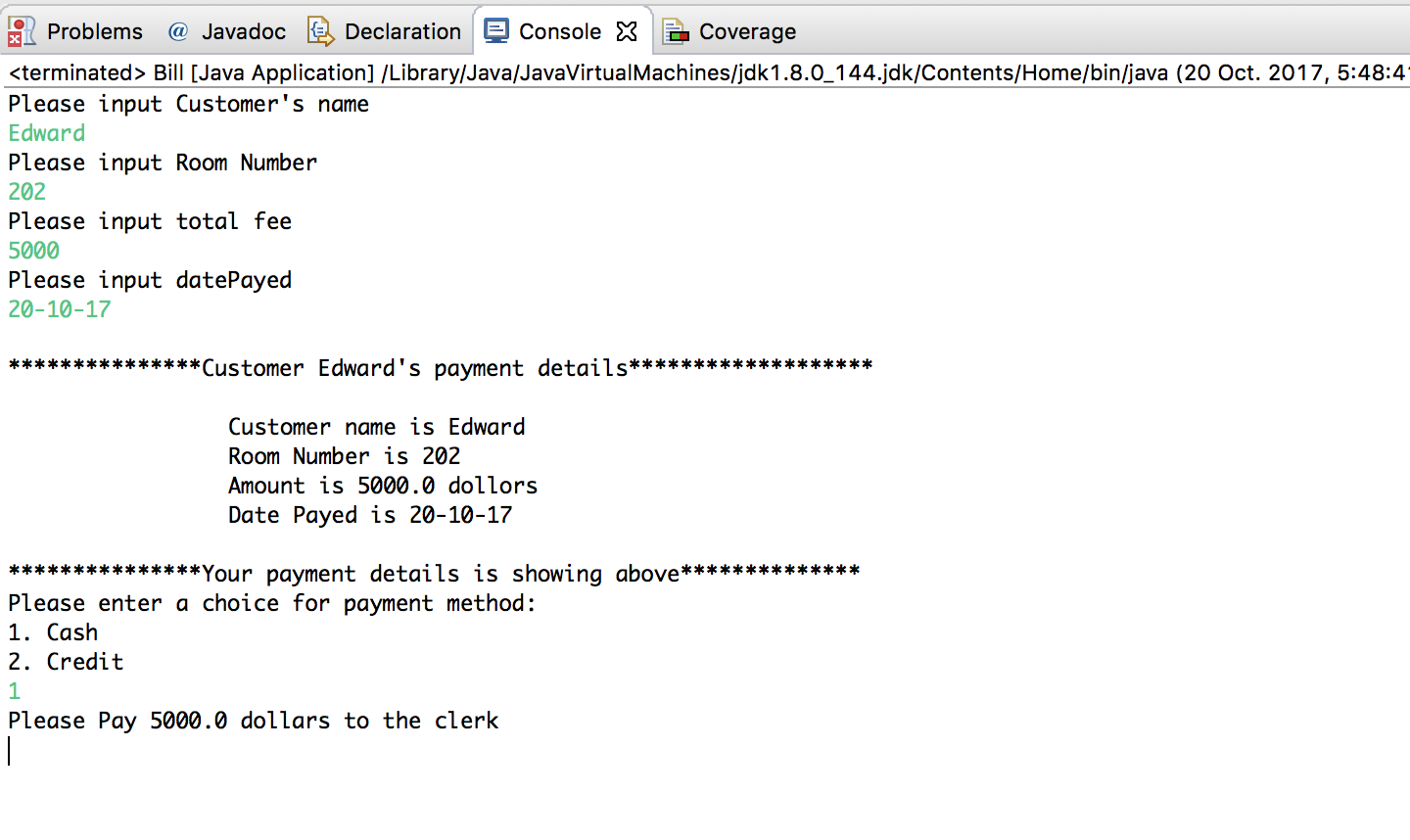
System.out.println("Invaild Choice!");

}

}

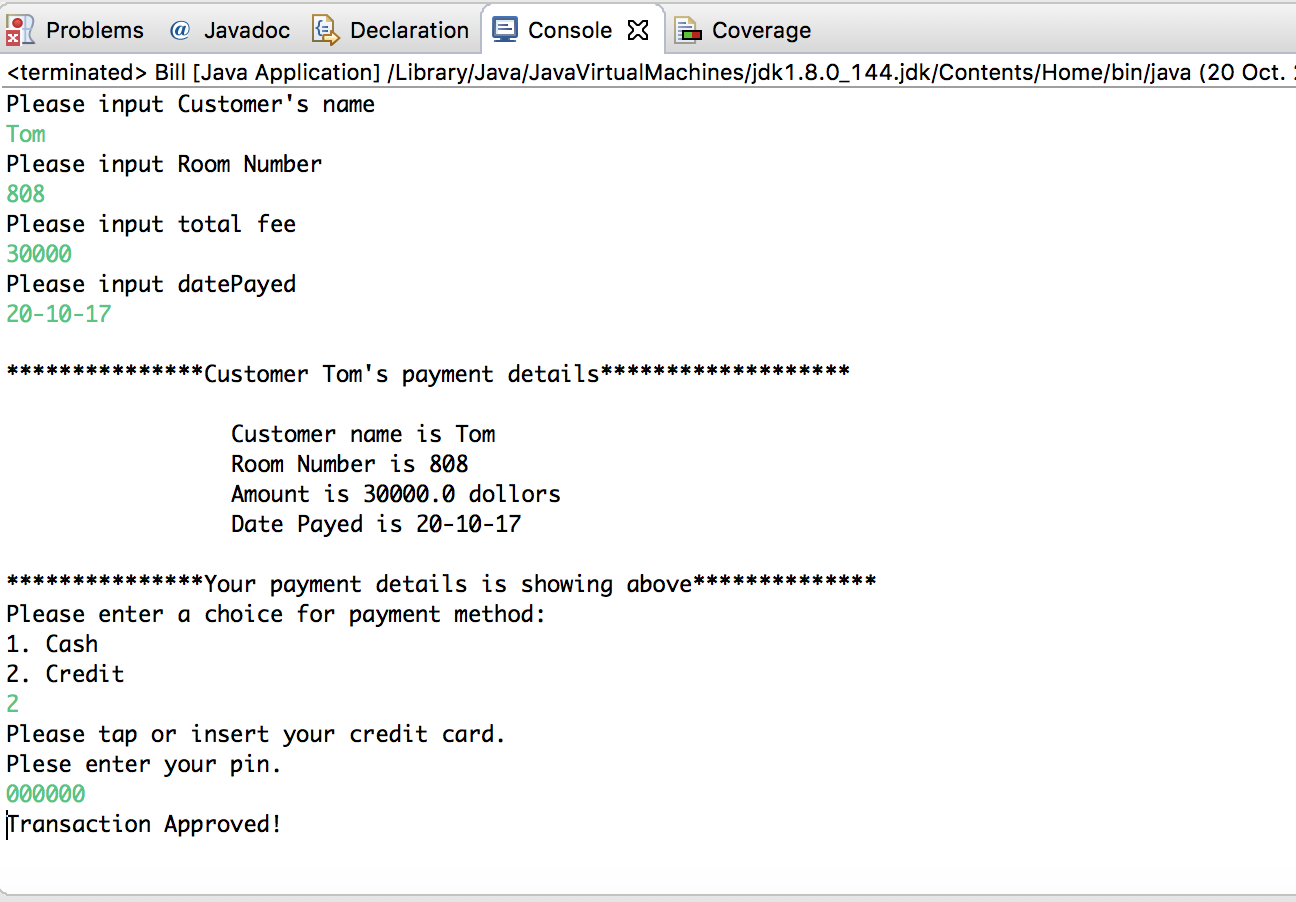
}

**Input values and output values: Test 1**



**Figure 1.1 Pay by cash**

**Input values and output values: Test 2**



**Figure 1.2 Pay by cash**

**Brief introductions for Java code**

1. The reason why we wrote this code is to help the clerk to generate the bill.
2. There are 3 methods in the code. The first one is to collect the customers’ information for the bill. The second one is to print the payment out for customer’s receipt. The third one enables customer to select the method they want to pay.
3. The difference between Test 1 and Test 2 is that we selected different number in the last method, which will cause different results as shown in the figure 1.1 and figure 1.2.

**User Interface Design**

Part1：input.java

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import java.awt.GridLayout;

public class Input {

Input(){

}

public static void Inputcall()

{

String name;

String room;

String startDate;

String endDate;

name = JOptionPane.showInputDialog(null,"What is the Customer name? \n");

JOptionPane.showMessageDialog(null, "Is the Customer " + name + " Right?");

room = (String) JOptionPane.showInputDialog(null, "Please select the room type (Double, Single, Luxury) ");

startDate = JOptionPane.showInputDialog(null, "When do you want to check in?");

endDate = JOptionPane.showInputDialog(null, "When do you want to check out?");

JOptionPane.showMessageDialog(null, "Hello " +name+ "! \n You have successfully booked the room");

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Object row[][] = { {name,room,startDate,endDate}};

Object column[] = { "Customer", "Room", "StartDate", "EndDate" };

JTable table = new JTable(row,column);

frame.add(new JScrollPane(table));

frame.setTitle("Confirmation");

frame.setSize(500, 500);

frame.setLocation(1000, 500);

frame.setVisible(true);

}

}

Part2：login.java

import javax.swing.\*;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import java.awt.Button;

import java.awt.GridLayout;

import java.awt.BorderLayout;

import java.awt.GridBagConstraints;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Login extends JFrame {

public static void Loginsession()

{

JFrame frame = new JFrame("Login Session");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(500, 300);

frame.setLocation(1000, 500);

JButton login = new JButton("login");

JButton exit\_btn = new JButton("exit");

JPanel login\_pan = new JPanel();

JPanel main\_window = new JPanel();

JTextField id = new JTextField(10);

JPasswordField password = new JPasswordField(10);

frame.setLayout(new GridLayout(3,2));

frame.add(new JLabel(" Administrator ID :"));

frame.add(id);

frame.add(new JLabel(" Administrator PW :"));

frame.add(password);

frame.add(login);

frame.add(exit\_btn);

frame.setVisible(true);

login.addActionListener(new ActionListener(){

@SuppressWarnings("deprecation")

@Override

public void actionPerformed(ActionEvent e){

String id\_tmp;

String pass\_tmp;

id\_tmp = id.getText();

pass\_tmp = password.getText();

if(id\_tmp.equals("admin") && pass\_tmp.equals("qwer"))

{

login\_pan.setVisible(false);

Input.Inputcall();

}

else

{

JOptionPane.showMessageDialog(null, "Check Admin ID and Password agian","Login\_Error",JOptionPane.ERROR\_MESSAGE);

}

}

});

}

}

part3: ui.java

import javax.swing.\*;

public class Ui {

public static void main(String[] args)

{

Login.Loginsession();

}

}

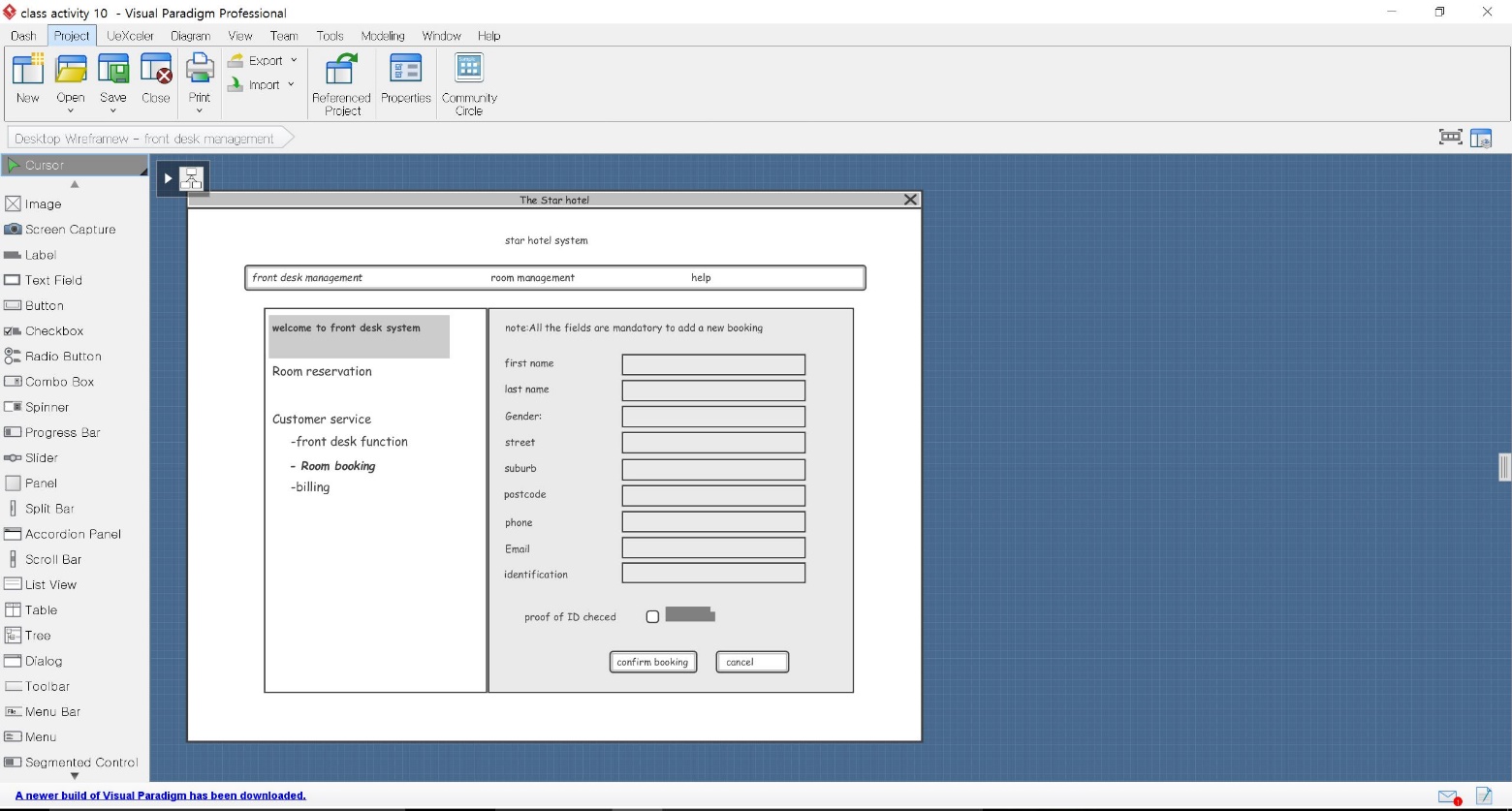


Figure 2.1 Hotel Management

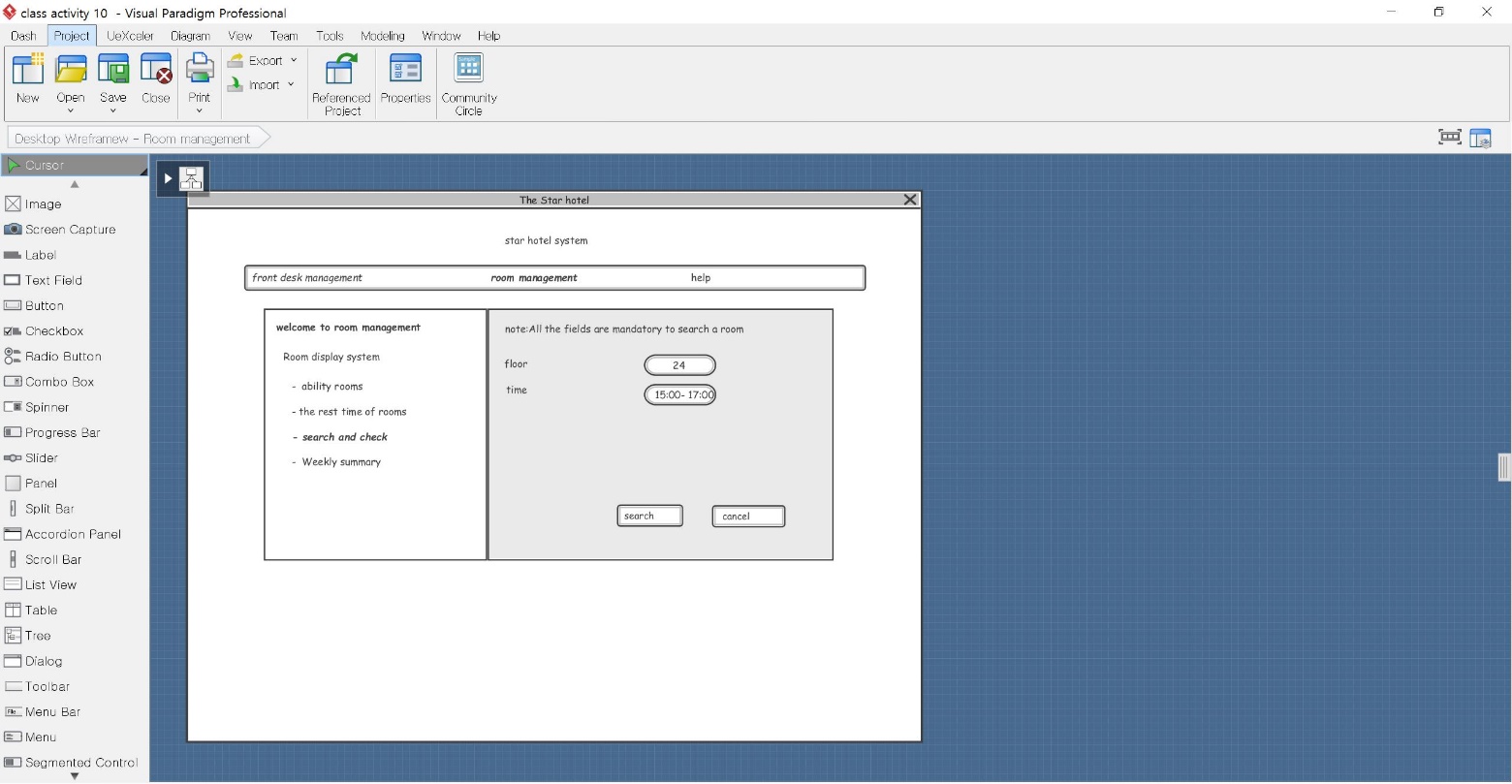


Figure 2.2 Room management(time)

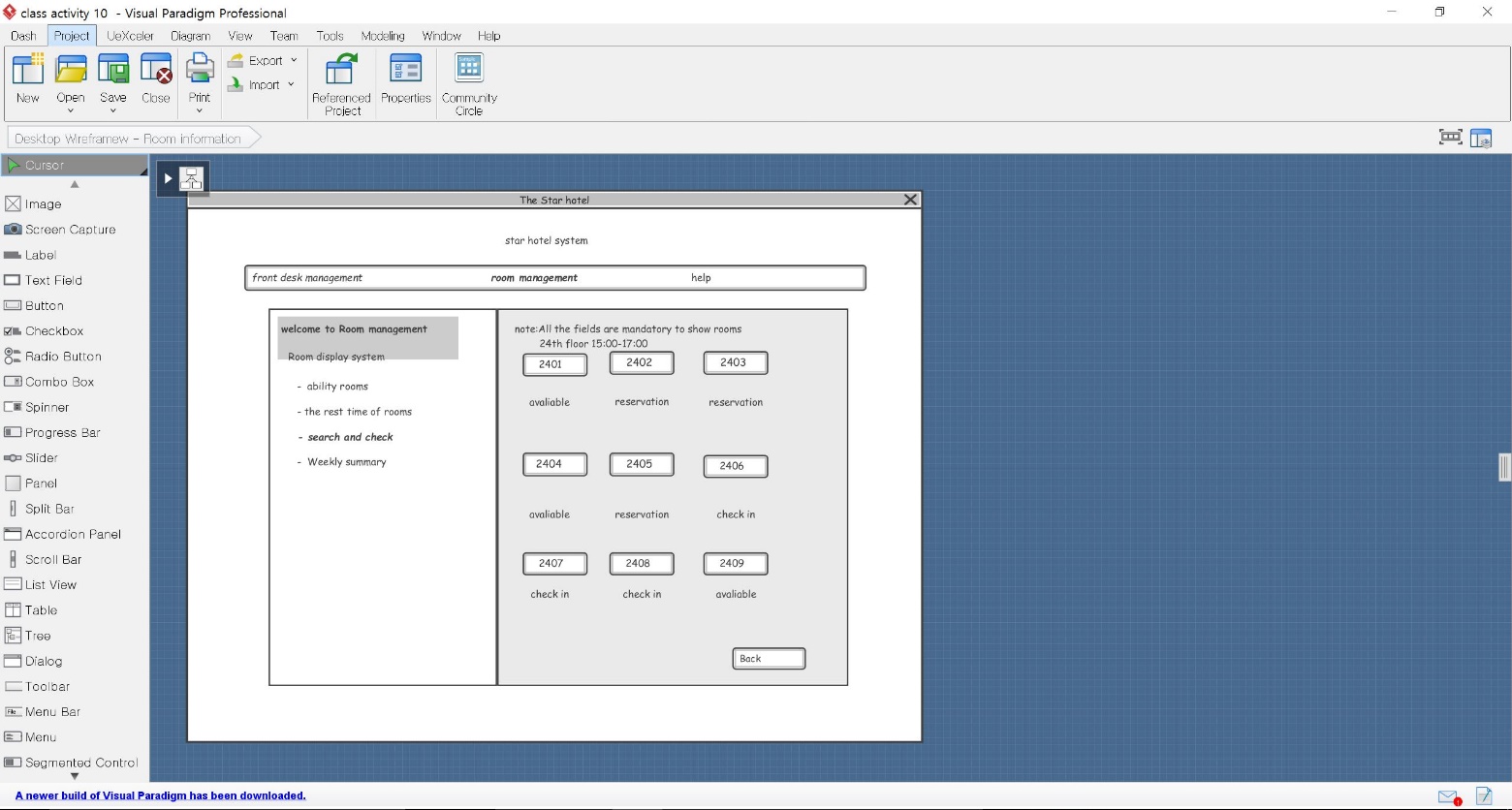


Figure 2.3 Room management(Room number)

**Database Connections**

Part 1 ConnectionConfiguration.java

import java.sql.Connection;

import java.sql.DriverManager;

/\*\*

\*

\*/

/\*\*

\*

\*

\*/

public class ConnectionConfiguration {

//Connection String for Database Connection.

public static final String ConnectionString= "jdbc:mysql://localhost:3306/world?autoReconnect=true&useSSL=false";

//User name for database connection.

public static final String user="root";

//Password for database connection.

public static final String password="cisco";

public static Connection getConnection()

{

Connection connection=null;

try{

connection = DriverManager.getConnection(ConnectionString,user,password);

}catch(Exception e){

e.printStackTrace();

}

return connection;

}

}

Part 2 HotelEmployee.java

/\*\*

\*

\*/

/\*\*

\*

\*

\*/

public class HotelEmployee {

private int EmployeeID;

private String EmployeeName;

private String Address;

private String ContactNo;

private String Email;

private String Gender;

//Default Constructor

public HotelEmployee()

{

}

//Parametrized Constructor to initialize values

public HotelEmployee(String EmployeeName,String Address,String ContactNo, String Email,String Gender)

{

this.EmployeeName=EmployeeName;

this.Address=Address;

this.ContactNo=ContactNo;

this.Email=Email;

this.Gender=Gender;

}

//Getter and Setter Methods to set and get values from user.

public int getID(){

return EmployeeID;

}

public void setID(int iniStaffID)

{

this.EmployeeID=iniStaffID;

}

public String getName()

{

return EmployeeName;

}

public void setName(String iniEmployeeName)

{

this.EmployeeName=iniEmployeeName;

}

public String getAddress()

{

return Address;

}

public void setAddress(String iniAddress)

{

this.Address=iniAddress;

}

public String getContactNo()

{

return ContactNo;

}

public void setContactNo(String iniPhoneNo)

{

this.ContactNo=iniPhoneNo;

}

public String getEmail()

{

return Email;

}

public void setEmail(String iniEmailAddress)

{

this.Email=iniEmailAddress;

}

public String getGender()

{

return Gender;

}

public void setGender(String iniSex)

{

this.Gender=iniSex;

}

}

Part 3 HotelEmployeeCRUD.java

import java.util.List;

import java.util.Scanner;

/\*\*

\*

\*/

/\*\*

\*

\*

\*/

public class HotelEmployeeCRUD {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@");

System.out.println("@@@@@@@@@@@@@@@@@@@@@ HotelDatabaseSystem @@@@@@@@@@@@@@@@@@@@@@@");

System.out.println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@");

init();

}

public static void init()

{

Scanner input =new Scanner(System.in);

HotelStaffOperationsImplementation hotel=new HotelStaffOperationsImplementation();

int userChoice;

System.out.println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@");

System.out.println("1. Insert the information about new Employee into database");

System.out.println("2. Select all saved records from database");

System.out.println("3. Update the information about an existed Employee into database");

System.out.println("4. Delete the information about Employees into database");

System.out.println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@");

System.out.println(" ");

System.out.println("Please select the number (1,2,3,4): ");

userChoice = input.nextInt();

switch(userChoice)

{

case 1:

System.out.println("Type employee information ");

String name,address,contactno,email,gender;

System.out.println("-> Employee Name: ");

name = input.next();

System.out.println("-> Address: ");

address = input.next();

System.out.println("-> Phone No: ");

contactno = input.next();

System.out.println("-> Email Address : ");

email = input.next();

System.out.println("-> Gender(M/F): ");

gender = input.next();

HotelEmployee emp =new HotelEmployee(name,address,contactno,email,gender);

hotel.insert(emp);

init();

break;

case 2:

List<HotelEmployee> selectEmployee=hotel.selectAll();

for(HotelEmployee h: selectEmployee)

{

System.out.println("\n");

System.out.println("ID : " + h.getID());

System.out.println("Employee Name : " + h.getName());

System.out.println("Address : " + h.getAddress());

System.out.println("Phone No : " + h.getContactNo());

System.out.println("Email Address ID : " + h.getEmail());

System.out.println("Gender(M/F) : " + h.getGender());

}

init();

break;

case 3:

System.out.println("Type employee ID that you would like to update: ");

int id=input.nextInt();

HotelEmployee employeeSelect =hotel.selectById(id);

System.out.println("Information below that you selected ");

System.out.println("Employee ID: " +employeeSelect.getID());

System.out.println("Employee Name: " +employeeSelect.getName());

System.out.println("Address: " +employeeSelect.getAddress());

System.out.println("Phone No: " +employeeSelect.getContactNo());

System.out.println("Email Address : " +employeeSelect.getEmail());

System.out.println("Gender: " +employeeSelect.getGender());

String uname,uaddress,ucontactno,uemail,ugender;

System.out.println("------------------------Type new information to update in database-----------------------");

System.out.println("Employee Name: ");

uname = input.next();

System.out.println("Address: ");

uaddress = input.next();

System.out.println("Phone No: ");

ucontactno = input.next();

System.out.println("Email Address : ");

uemail = input.next();

System.out.println("Gender(M/F): ");

ugender = input.next();

HotelEmployee employeeUpdate =new HotelEmployee(uname,uaddress,ucontactno,uemail,ugender);

hotel.update(employeeUpdate,id);

init();

break;

case 4:

int idDelete;

System.out.println("Type employee ID that you would like to delete: ");

idDelete=input.nextInt();

hotel.delete(idDelete);

init();

break;

default:

System.out.println("You have a wrong choice. Please check the ID again");

init();

break;

}

}

}

Part 4 HotelStaffOperationsImplementation.java

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

/\*\*

\*

\*/

/\*\*

\*

\*

\*/

public class HotelStaffOperationsImplementation {

public void insert(HotelEmployee emp)

{

Connection cnt = null;

PreparedStatement pps = null;

try {

cnt = ConnectionConfiguration.getConnection();

pps = cnt.prepareStatement("INSERT INTO HotelEmployee (EmployeeName,Address,ContactNo,Email,Gender)" +

"VALUES (?,?,?,?,?)");

pps.setString(1, emp.getName());

pps.setString(2, emp.getAddress());

pps.setString(3, emp.getContactNo());

pps.setString(4, emp.getEmail());

pps.setString(5, emp.getGender());

pps.executeUpdate();

System.out.println("\n@@@@@@@@@@@@@@@@@@@@Record has been successfully added to the database!!@@@@@@@@@@@@@@@@@@@@");

} catch (Exception e) {

e.printStackTrace();

} finally {

if (pps != null) {

try {

pps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if (cnt != null) {

try {

cnt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

public HotelEmployee selectById(int id)

{

HotelEmployee emp = new HotelEmployee();

Connection cnt = null;

PreparedStatement pps = null;

ResultSet resultSet = null;

try {

cnt = ConnectionConfiguration.getConnection();

pps = cnt.prepareStatement("SELECT \* FROM HotelEmployee WHERE EmployeeID = ?");

pps.setInt(1, id);

resultSet = pps.executeQuery();

while (resultSet.next()) {

emp.setID(resultSet.getInt("EmployeeID"));

emp.setName(resultSet.getString("EmployeeName"));

emp.setAddress(resultSet.getString("Address"));

emp.setContactNo(resultSet.getString("ContactNo"));

emp.setEmail(resultSet.getString("Email"));

emp.setGender(resultSet.getString("Gender"));

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if (resultSet != null) {

try {

resultSet.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if (pps != null) {

try {

pps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if (cnt != null) {

try {

cnt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

return emp;

}

public List<HotelEmployee> selectAll()

{

List<HotelEmployee> employees = new ArrayList<HotelEmployee>();

Connection cnt = null;

Statement st = null;

ResultSet rs = null;

try {

cnt = ConnectionConfiguration.getConnection();

st = cnt.createStatement();

rs = st.executeQuery("SELECT \* FROM HotelEmployee");

while (rs.next()) {

HotelEmployee emp = new HotelEmployee();

emp.setID(rs.getInt("EmployeeID"));

emp.setName(rs.getString("EmployeeName"));

emp.setAddress(rs.getString("Address"));

emp.setContactNo(rs.getString("ContactNo"));

emp.setEmail(rs.getString("Email"));

emp.setGender(rs.getString("Gender"));

employees.add(emp);

}

} catch (Exception e) {

e.printStackTrace();

} finally {

if (rs != null) {

try {

rs.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if (st != null) {

try {

st.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if (cnt != null) {

try {

cnt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

return employees;

}

public void delete(int id)

{

Connection cnt = null;

PreparedStatement pst = null;

try {

cnt = ConnectionConfiguration.getConnection();

pst = cnt.prepareStatement("DELETE FROM HotelEmployee WHERE EmployeeID = ?");

pst.setInt(1, id);

pst.executeUpdate();

System.out.println("\n@@@@@@@@@@@@@@@@@@Record has been successfully deleted from the database!!@@@@@@@@@@@@@@@@@@@@@@");

} catch (Exception e) {

e.printStackTrace();

} finally {

if (pst != null) {

try {

pst.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if (cnt != null) {

try {

cnt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

public void update(HotelEmployee emp,int id)

{

Connection cnt = null;

PreparedStatement pst = null;

try {

cnt = ConnectionConfiguration.getConnection();

pst = cnt.prepareStatement("UPDATE HotelEmployee SET " +

"EmployeeName = ?, Address = ?, ContactNo = ?, Email = ?, Gender = ? WHERE EmployeeID = ?");

pst.setString(1, emp.getName());

pst.setString(2, emp.getAddress());

pst.setString(3, emp.getContactNo());

pst.setString(4, emp.getEmail());

pst.setString(5, emp.getGender());

pst.setInt(6, id);

pst.executeUpdate();

System.out.println("@@@@@@@@@@@@@@@@@@Row updated successfully!!!!!!!@@@@@@@@@@@@@@@@@@@@@@@@@@@");

} catch (Exception e) {

e.printStackTrace();

} finally {

if (pst != null) {

try {

pst.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

if (cnt != null) {

try {

cnt.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

}

2.1 Running database connection code

We have made four functions.

1. Insert employee information
2. Select all saved record in the database
3. Update the information in database
4. Delete the information.

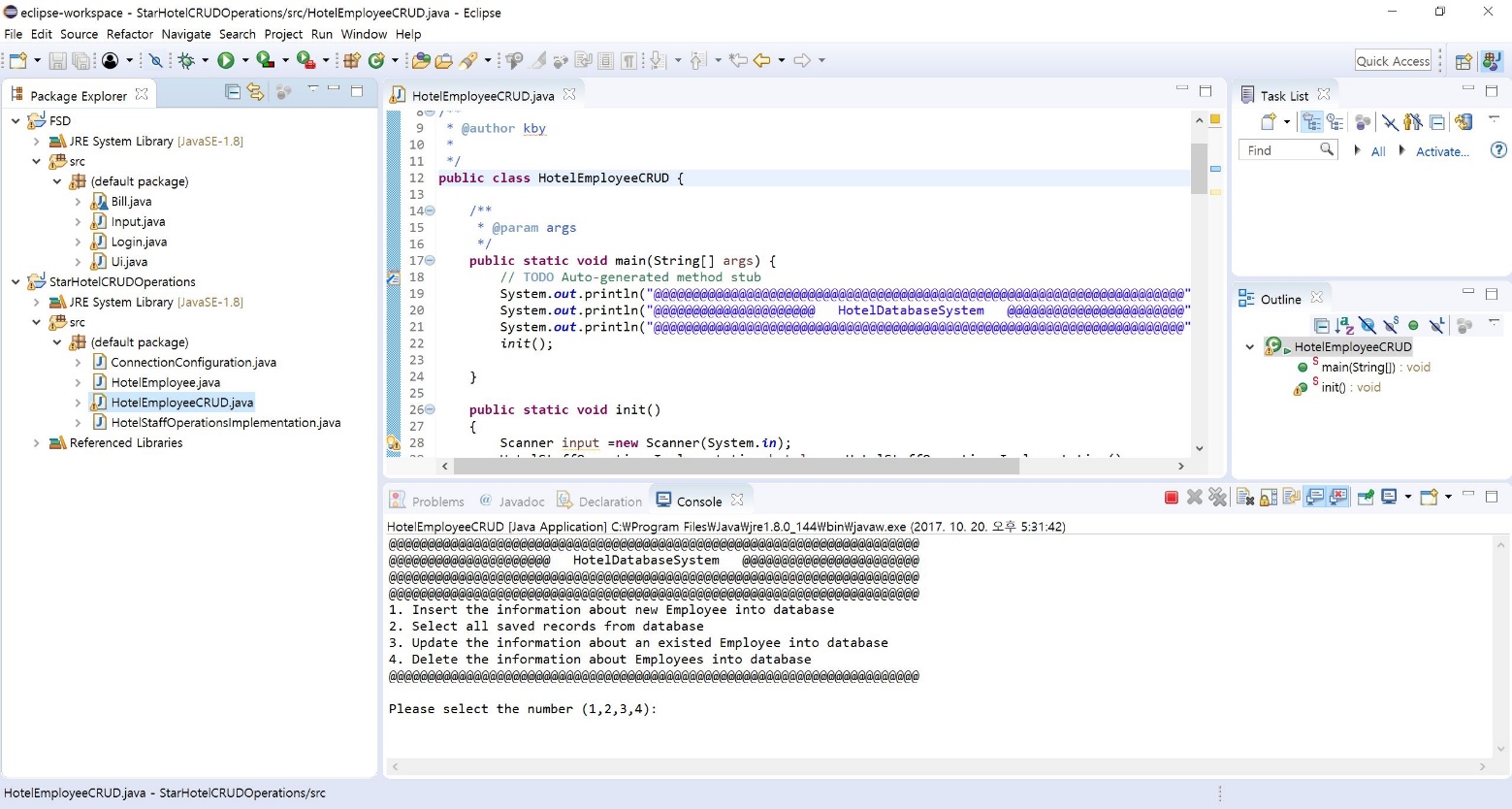


Figure2.1 Running HotelEmployeeCRUD.java code

To add data into database, I issued number 1 fucntion

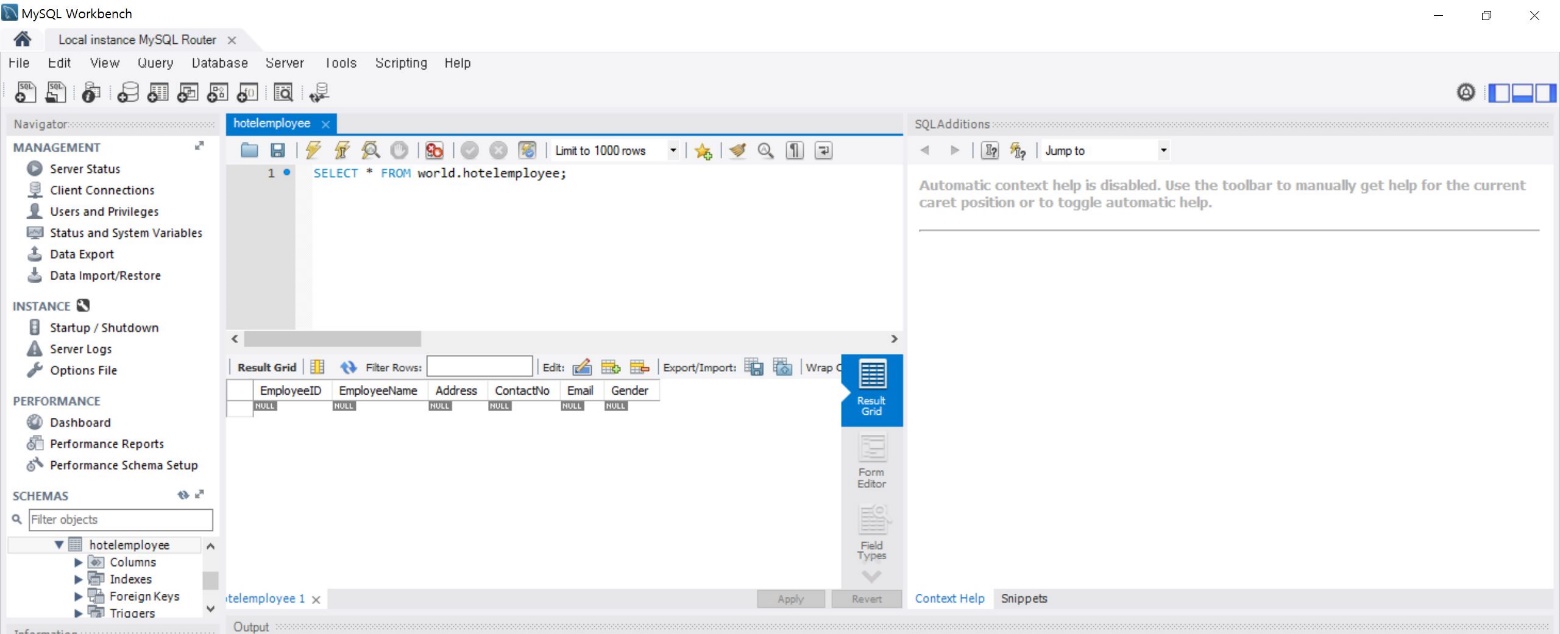


Figure 2.3 No data screen

At first, there is no information in database.

Next, I typed Employee’s name, Address, Phone number, Email address and Gender.

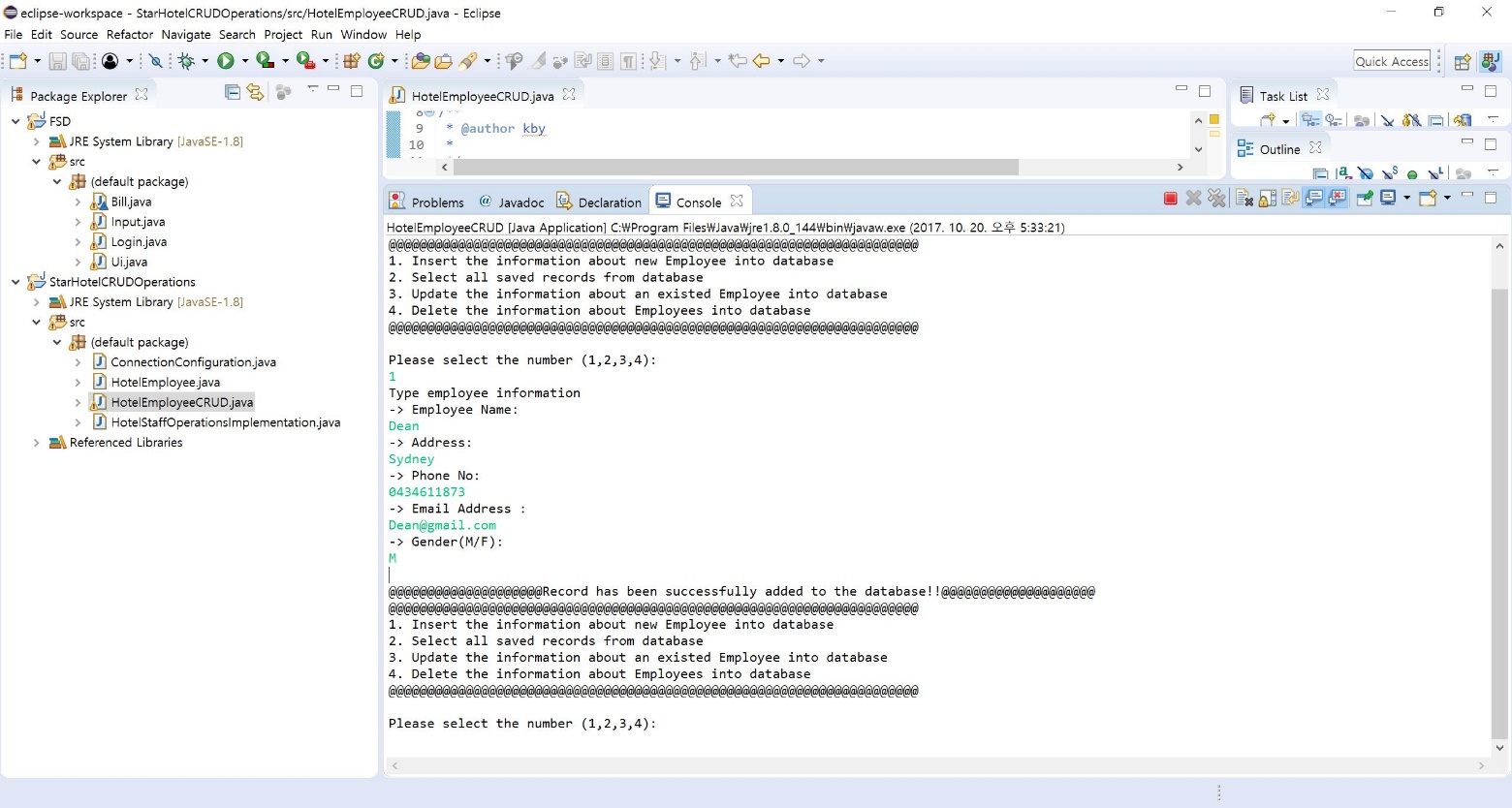


Figure 2.3 Typing employee information

Next, I typed Employee’s name, Address, Phone number, Email address and Gender.

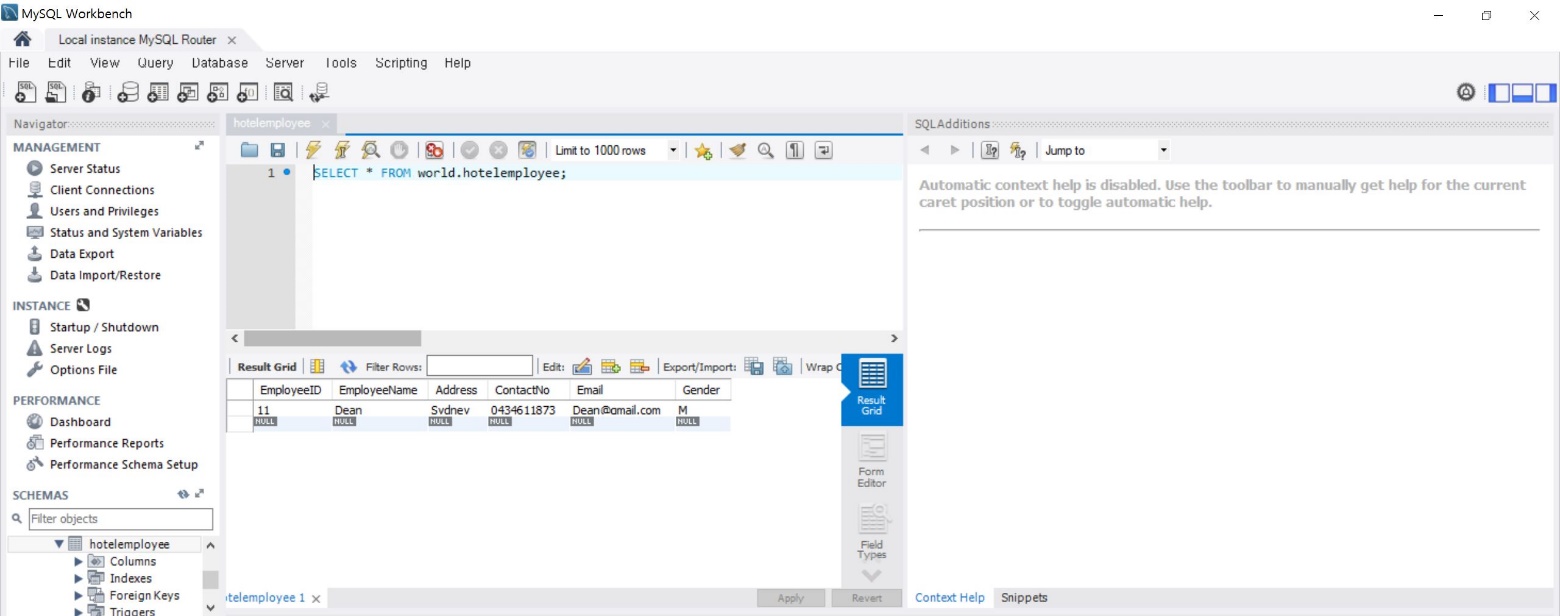




Figure 2.4 saved information from Java

In order to check the data whether it is saved well or not, I run red box above.

The data was successfully saved in the database.

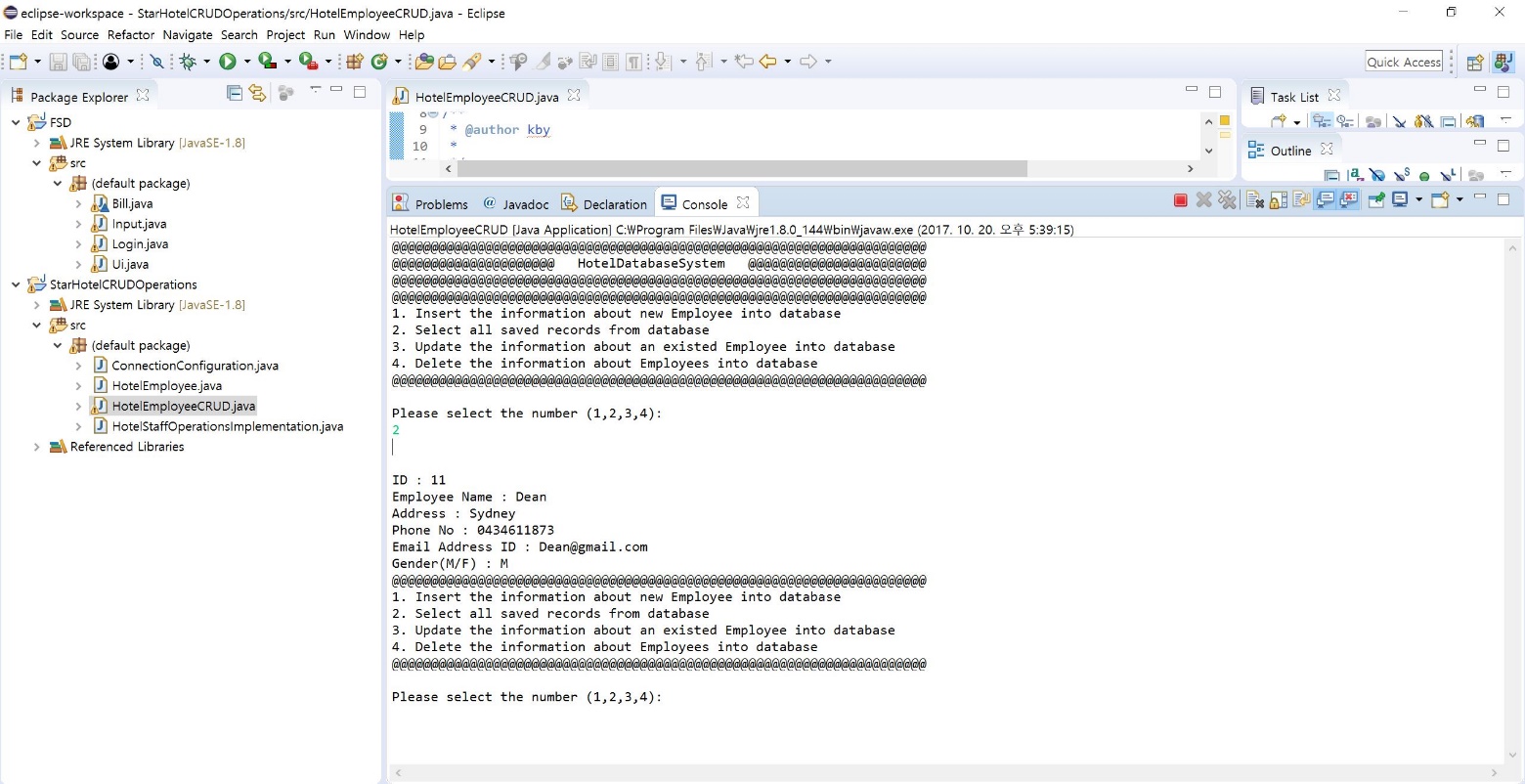


Figure 2.5 Option 2: Select all saved records from database

Administrator can check all saved information in the database.

This photo above is the result of the option 2.

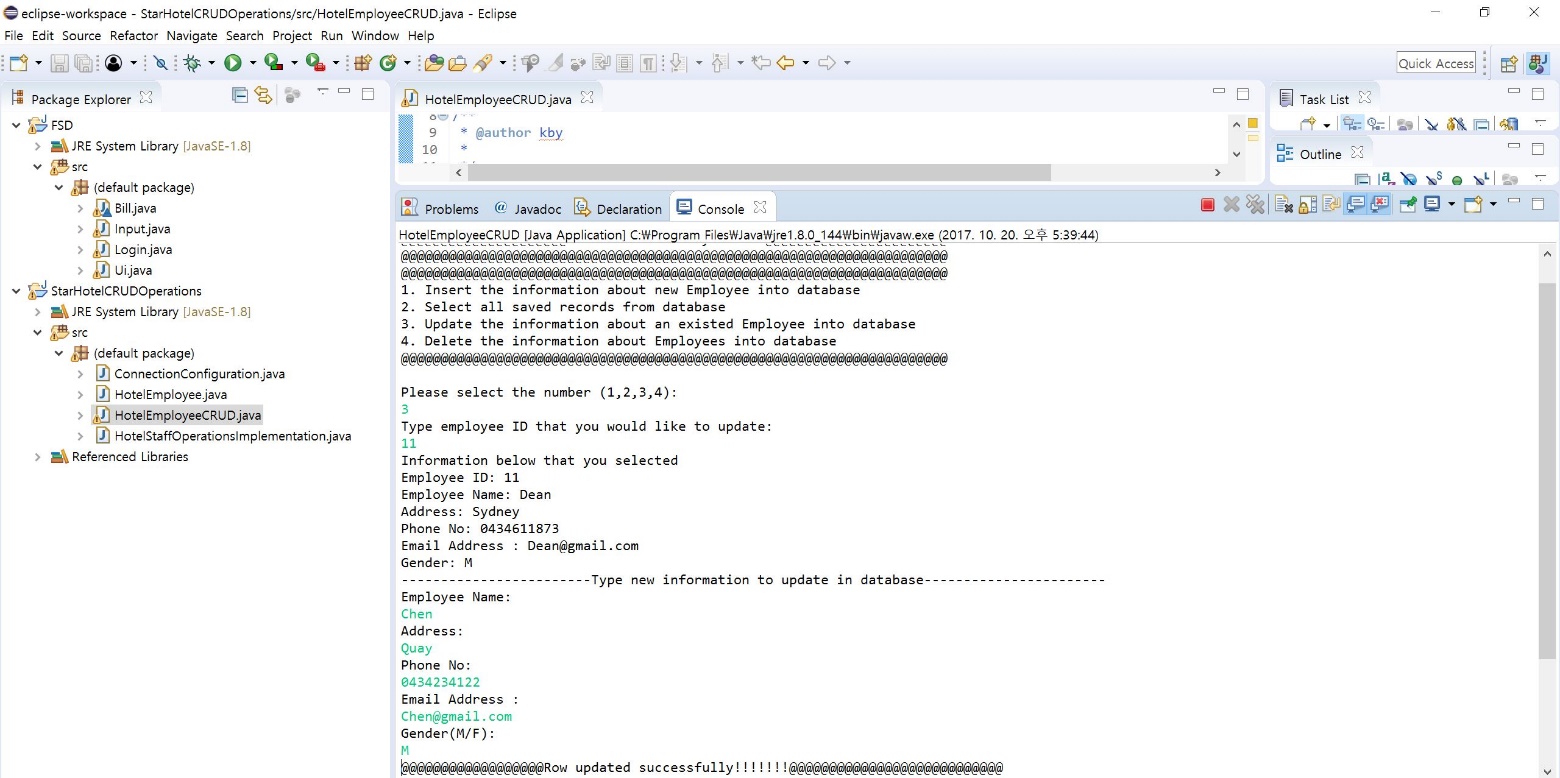


Figure 2.6 Option 3: Update the information about an existed Employee into database

An administrator can update information of Employee to use option 3.

I updated the 11st employee’s information.

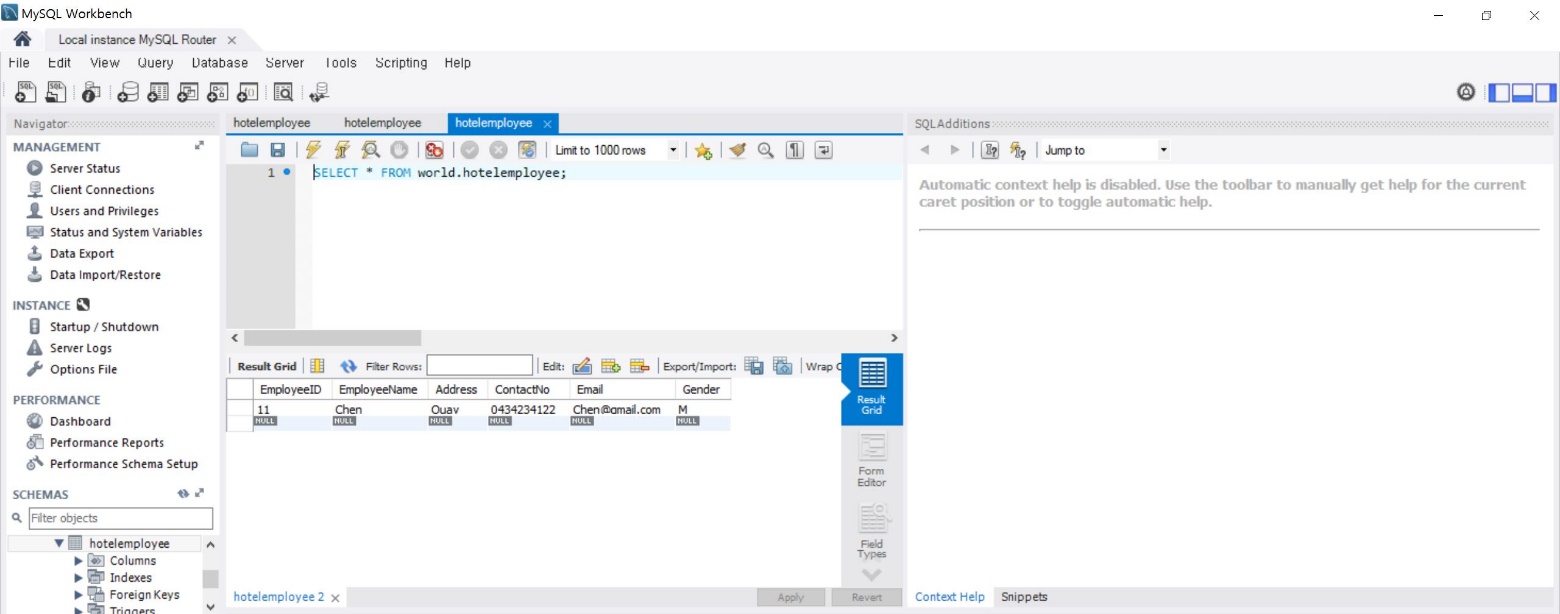


Figure 2.7 Updated information screen

To check the updated information, I run the program.

The data was successfully updated.

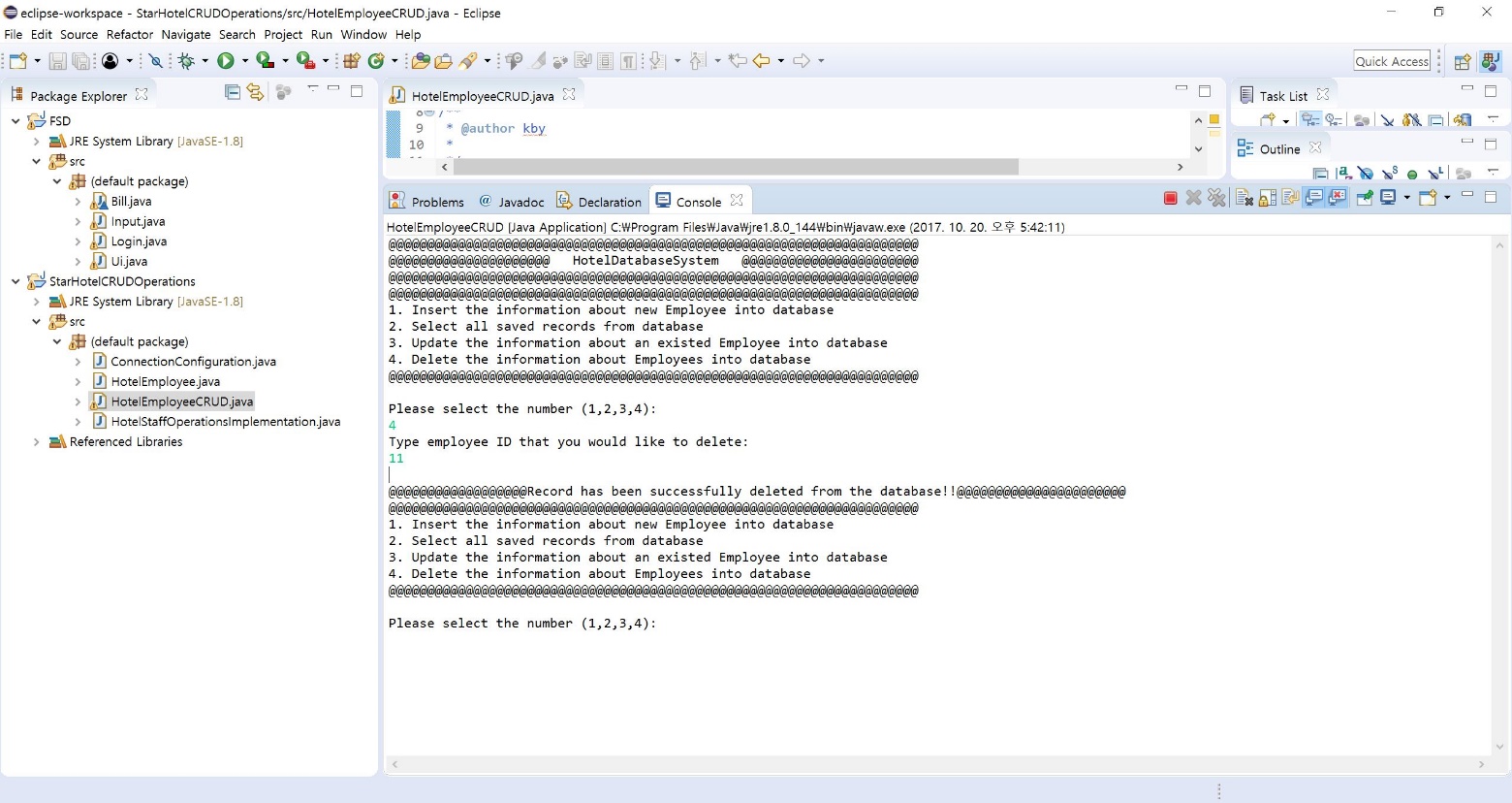


Figure 2.8 Option4: Delete employee information

Last option 4 is to delete employee information from the database.

I typed employee ID(11).

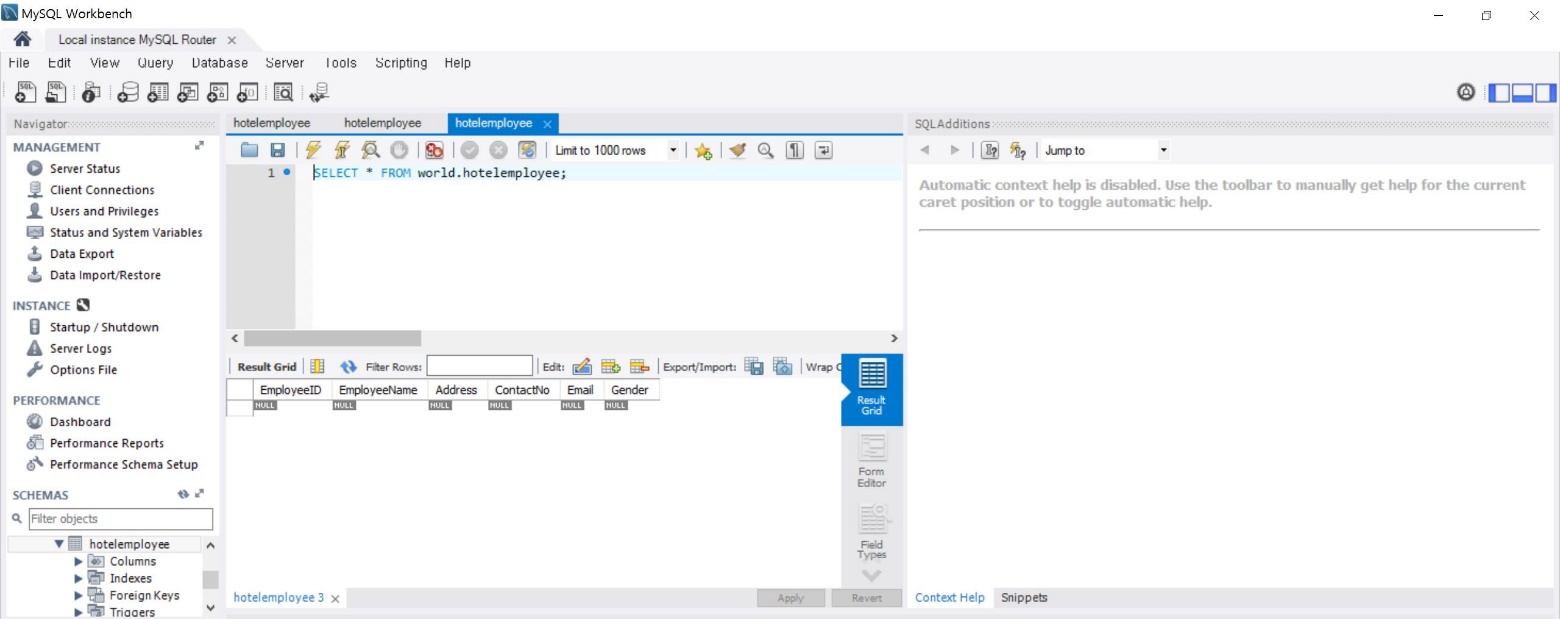


Figure 2.9

To see the result of deleting option, I run the program.

The data was successfully removed.