AMITY UNIVERSITY, PATNA

AMITY INSTITUTE OF INFORMATION TECHNOLOGY

Advanced Java Lab File
LAB - 3

BCA

Name : Nishant Kumar
Program/Semester : BCA— 6 'A'
Enroll. Number : A45304821038
Submitted to : Dr. Naveen Kumar Singh

HIBERNATE (THROUGH ANNOTATIONS)

Problem Statement :

Hibernate program to demonstrate crud operations with the help of annotations.

Introduction :

The objective of this proj ect is to develop a Hibernate program that performs CRUD (Create, Read, Update, Delete) operations on a table named "politics". The program will utilize two XML files for configuration and mapping purposes. The first XML file, hibernate.cfg.xml, will contain properties related to the database connection, including driver class name, URL username, password, dialect, and a mapping tag specifying the location of the second XMLT11e, leaders.hbm.xml. The second XML file, leaders.hbm.xml, will define the mapping between Java objects and the "politics" table, linking each attribute of the leader class to the corresponding columns in the table.

Problem Description:

The program will consist of the following functionalities:

1. Insert Record :

Users will have the option to insert a new record into the "politics" table. Upon selecting this option, the program will prompt the user to enter details about the political leader, including the leader's name, party affiliation, title, and a notable achievement. The entered information will then be added as a new record in the database.

2. Retrieve a Particular Record :

Users can retrieve a specific political leader by providing the leader's ID. If a leader with the provided ID exists in the "politics" table, the program will display the corresponding information, including the leader's name, party affiliation, title, and notable achievement. If the leader is not found, the program will display a message indicating that the leader with the entered ID is not found.

3. Retrieve All Records :

Users can retrieve all records from the "politics" table. The program will display information about all political leaders stored in the database, including their names, party affiliations, titles, and notable achievements.

4. Update Record :

Users will be able to update information about a particular political leader. Upon selecting this option, the program will prompt the user to enter the ID of the leader whose information they want to update. If the provided ID corresponds to a leader in the database, the program will allow the user to modify the leader's party affiliation. If the leader is not found, the program will display a message indicating that the leader with the entered ID is not found.

5. Delete Record :

Users can delete a record of a political leader from the "politics" table by providing the leader's ID. If the provided ID matches a leader in the database, the program will delete the corresponding record.

If the leader is not found, the program will display a message indicating that the leader with the entered ID is not found.

6. Exit Program :
Users will have the option to exit the program. Upon selecting this option, the program will close the Hibernate session and factory, allowing users to exit the program gracefully.

Implementation Approach:
The program will be implemented using Hibernate, a popular object-relational mapping (ORM) framework for Java. Hibernate will handle the mapping between Java objects and the database tables, utilizing a separate mapping file to define the mapping configuration. The program will provide a user-friendly menu interface for interacting with the database, ensuring ease of use and clarity for users. Each CRUD operation will be implemented as a separate method or class, following a modular and object-oriented approach.

Expected Input and Output:

1. Insert Record :
User inputs : Leader's name, party affiliation, title, notable achievement.

2. Retrieve a Particular Record :
User inputs : Leader's ID.
Output : Information about the specified leader if found, or a message indicating that the leader with the entered ID is not found.

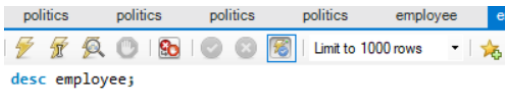
3. Retrieve All Records :
Output : Information about all political leaders stored in the database.

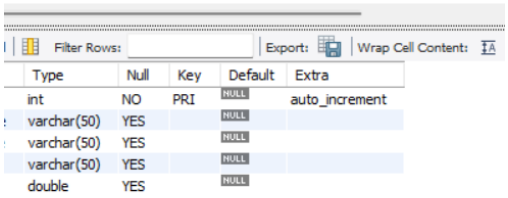
4. Update Record :
User inputs : Leader's ID and new party name.
Output : Confirmation message upon successful update, or a message indicating that the leader with the entered ID is not found.

5. Delete Record :
User inputs : Leader's ID.
Output : Confirmation message upon successful deletion, or a message indicating that the leader with the entered ID is not found.

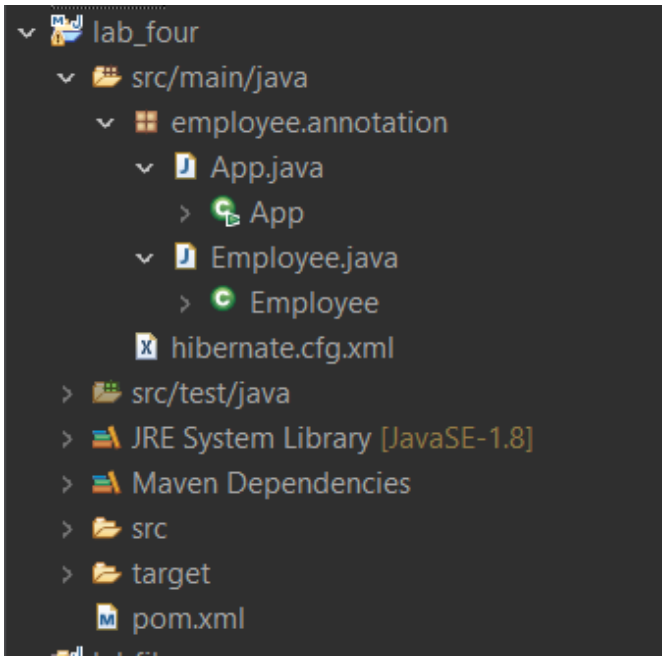
6. Exit Program :
Output : Termination of the program.

Tools and Technologies used:

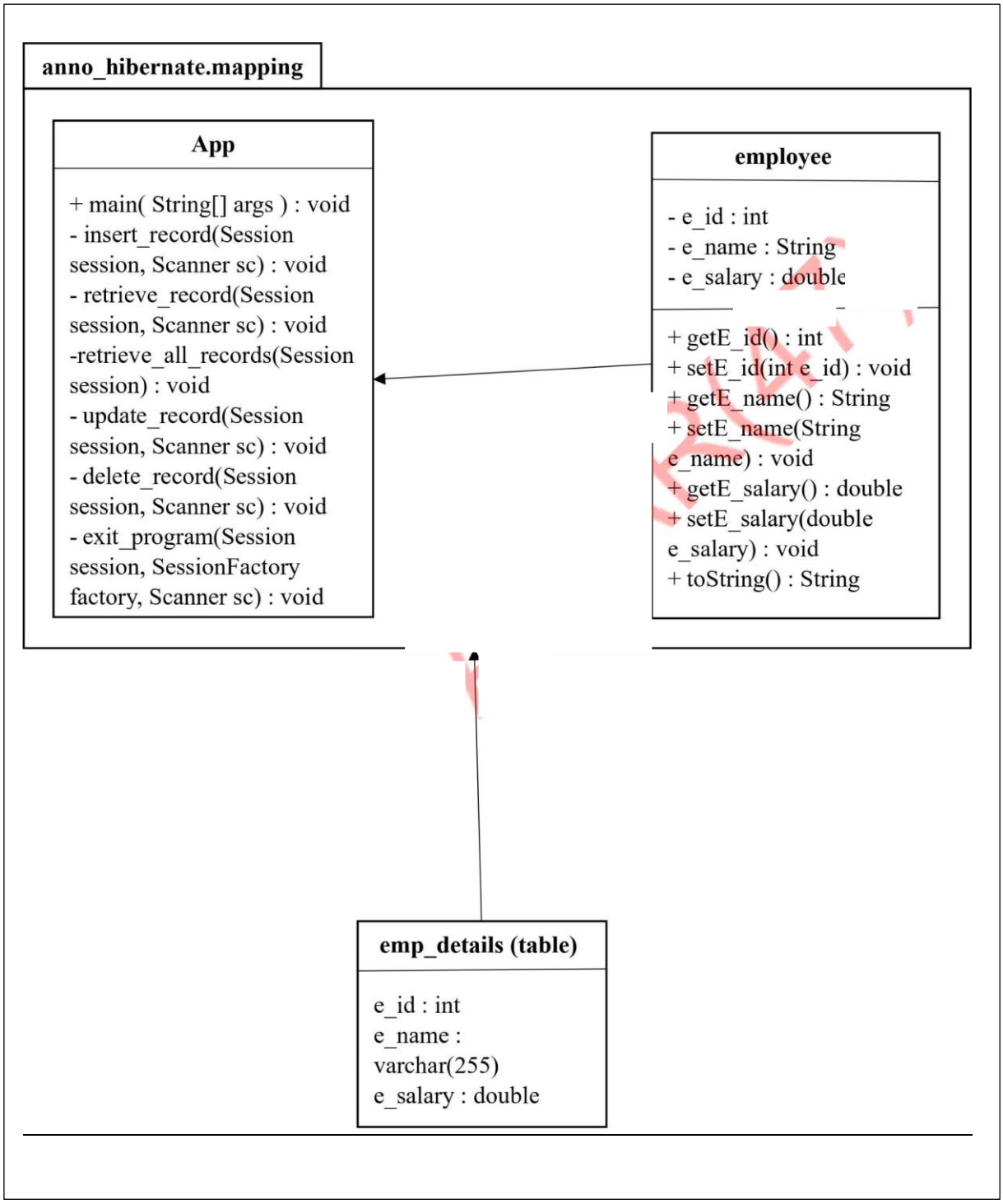




Folder Structure



Class Diagram — The class diagram for my project is as follows :



CODE

App . j ava

```
package employee.annotation;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import java.util.List;
import java.util.Scanner;
import org.hibernate.query.Query;

public class App
{
    public static void main( String[] args )
    {
      Configuration config = new Configuration();
            config.configure();
            SessionFactory factory = config.buildSessionFactory();
            Session session = factory.openSession();
            Scanner sc = new Scanner(System.in);
            while(true)
            {
                  System.out.println("\nChoose the operation you want
to perform :");
                  System.out.println("1. Insert a record into the
table");
                  System.out.println("2. Retrieve a particular record
from the table");
                  System.out.println("3. Retrieve all the records from
the table");
                  System.out.println("4. Update some record from the
table ");
                  System.out.println("5. Delete some record from the
table ");
                  System.out.println("7. Exit");
                  System.out.println("\nPlease enter your choice :");
                  int choice = sc.nextInt();
                  switch(choice)
                  {
                  case 1 : insert_record(session, sc);
                           break;
                  case 2 : retrieve_record(session, sc);
                        break;
                  case 3 : retrieve_all_records(session);
                        break;
```

CODE

```
                case 4 : update_record(session, sc);
                      break;
                case 5 : delete_record(session, sc);
                      break;
                case 6 : exit_program(session, factory, sc);
                      break;
                }
          }
    }
    private static void insert_record(Session session, Scanner sc)
    {
                Transaction transaction =
session.beginTransaction();
                System.out.println("\nPerforming INSERT
operation......");
                System.out.println("\nPlease enter the first name of
the employee -");
                String fname = sc.next();
                System.out.println("Please enter the last name of
the employee -");
                String lname = sc.next();
                System.out.println("Please enter the department of
the employee -");
                String dept = sc.next();
                System.out.println("Please enter the salary of the
employee -");
                double sal = sc.nextDouble();
                System.out.println();
                Employee emp_insert = new Employee();
                emp_insert.setF_name(fname);
                emp_insert.setL_name(lname);
                emp_insert.setE_dept(dept);
                emp_insert.setE_sal(sal);
                session.save(emp_insert);
                transaction.commit();
    }
    private static void retrieve_record(Session session, Scanner sc)
    {
       Transaction transaction = session.beginTransaction();
            System.out.println("\nRetrieving a particular employee
from the table based on his/her id......");
          System.out.println("\nEnter the id of the employee who's
information you want to retrieve - ");
            int retrieved_id = sc.nextInt();
            System.out.println();
            Employee E = session.get(Employee.class, retrieved_id);
            if (E != null)
            {
            System.out.println("Employee - " + E);
            }
          else
```

```
          {
          System.out.println("Employee with ID " + retrieved_id + "
doesn't exist.\n");
          }
           transaction.commit();
    }
    private static void retrieve_all_records(Session session)
    {
      Transaction transaction = session.beginTransaction();
          System.out.println("\nRetrieving everything from the
table......\n");
          Query<Employee> query = session.createQuery("FROM
leaders", Employee.class);
          List<Employee> employees_list = query.list();
          for (Employee emp : employees_list)
          {
          System.out.println("Employee - " + emp);
          }
          transaction.commit();
    }
    private static void update_record(Session session, Scanner sc)
    {
      Transaction transaction = session.beginTransaction();
          System.out.println("\nUpdating the department of the
employee......");
          System.out.println("\nEnter the id of the employee who's
department you want to update -");
          int id = sc.nextInt();
          Employee emp_update = session.get(Employee.class, id);
          if(emp_update != null)
          {
          System.out.println("\nEnter the name of the new
department for the employee -");
          String new_dept = sc.next();
          System.out.println();
          emp_update.setE_dept(new_dept);
          session.saveOrUpdate(emp_update);
          System.out.println("Department - " + emp_update + "
updated successfully.\n");
          }
          else
          {
          System.out.println("\nEmployee with ID " + id + " doesn't
exist.\n");
          }
          transaction.commit();
    }
    private static void delete_record(Session session, Scanner sc)
    {
      Transaction transaction = session.beginTransaction();
          System.out.println("\nDeleting some record from the
```

```
table......");
           System.out.println("\nEnter the id of the employee whose
information you want to delete -");
           int id = sc.nextInt();
           System.out.println();
           Employee emp_delete = session.get(Employee.class, id);
           if (emp_delete != null)
           {
           session.delete(emp_delete);
           System.out.println("Record - " + emp_delete + " deleted
successfully.\n");
           }
           else
           {
                 System.out.println("\nEmployee with ID " + id + " is
not found.\n");
           }
           transaction.commit();
    }
    private static void exit_program(Session session, SessionFactory
factory, Scanner sc)
    {
       System.out.println("\nExiting...\n");
           sc.close();
           session.close();
           factory.close();
           System.exit(0);
    }
}
```

Leaders.java

```
package employee.annotation;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "employee")
public class Employee {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int e_id;
private String F_name;
private String L_name;
private String e_dept;
private double e_sal;
```

CODE

```
public Employee() {
	super();
	// TODO Auto-generated constructor stub
}
public Employee(int e_id, String f_name, String l_name, String
e_dept, double e_sal) {
	super();
	this.e_id = e_id;
	F_name = f_name;
	L_name = l_name;
	this.e_dept = e_dept;
	this.e_sal = e_sal;
}
public int getE_id() {
	return e_id;
}
public void setE_id(int e_id) {
	this.e_id = e_id;
}
public String getF_name() {
	return F_name;
}
public void setF_name(String f_name) {
	F_name = f_name;
}
public String getL_name() {
	return L_name;
}
public void setL_name(String l_name) {
	L_name = l_name;
}
public String getE_dept() {
	return e_dept;
}
public void setE_dept(String e_dept) {
	this.e_dept = e_dept;
}
public double getE_sal() {
	return e_sal;
}
public void setE_sal(double e_sal) {
	this.e_sal = e_sal;
}
@Override
public String toString() {
	return "Employee [e_id=" + e_id + ", F_name=" + F_name + ",
L_name=" + L_name + ", e_dept=" + e_dept + ", e_sal="
			+ e_sal + "]";
}
```

CODE

hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>

        <property
name="connection.driver_class">com.mysql.cj.jdbc.Driver</proper
ty>
        <property
name="connection.url">jdbc:mysql://localhost:3306/student</prop
erty>
        <property name="connection.username">root</property>
        <property
name="connection.password">Mysql@2024</property>
        <property
name="dialect">org.hibernate.dialect.MySQLDialect</property>
        <property
name="current_session_context_class">thread</property>
        <property
name="cache.provider_class">org.hibernate.cache.internal.NoCach
eProvider</property>
        <property name="show_sql">true</property>
        <property name="hbm2ddl.auto">update</property>
    <mapping class="employee.annotation.Employee"/>


    </session-factory>
</hibernate-configuration>
```

pom. Xml

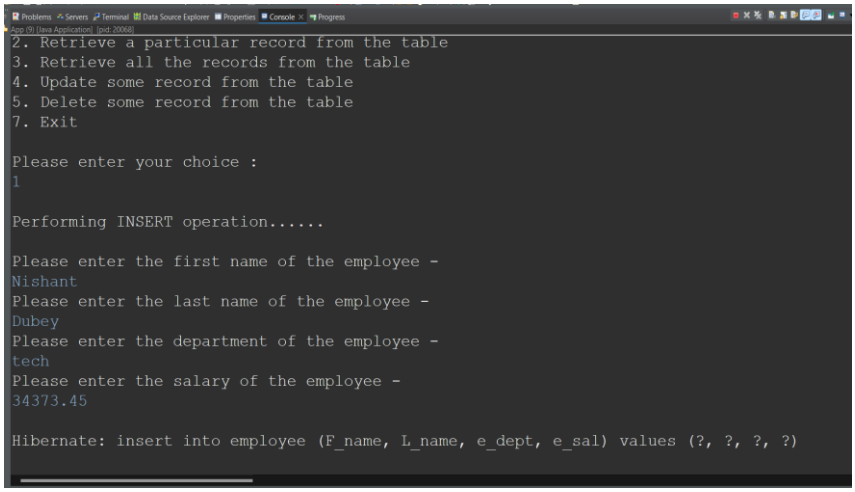
```
 <project xmlns="http://maven.apache.org/POM/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
     <modelVersion>4.0.0</modelVersion>

     <groupId>employee</groupId>
     <artifactId>annotation</artifactId>
     <version>0.0.1-SNAPSHOT</version>
     <packaging>jar</packaging>
```

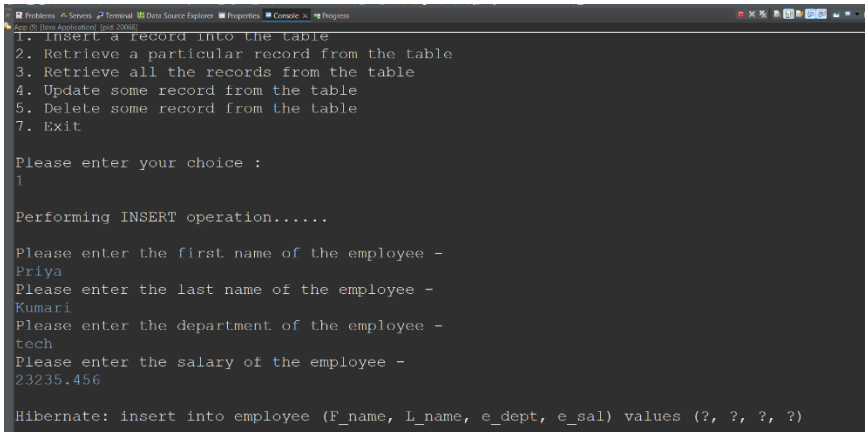
CODE

```
      <name>annotation</name>
      <url>http://maven.apache.org</url>

      <properties>
            <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
      </properties>

      <dependencies>
            <dependency>
                  <groupId>junit</groupId>
                  <artifactId>junit</artifactId>
                  <version>3.8.1</version>
                  <scope>test</scope>
            </dependency>

            <!-- https://mvnrepository.com/artifact/mysql/mysql-
connector-java -->
            <dependency>
                  <groupId>mysql</groupId>
                  <artifactId>mysql-connector-java</artifactId>
                  <version>8.0.33</version>
            </dependency>
            <!--
https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
            <dependency>
                  <groupId>org.hibernate</groupId>
                  <artifactId>hibernate-core</artifactId>
                  <version>5.4.5.Final</version>
            </dependency>
      </dependencies>
</project>
```

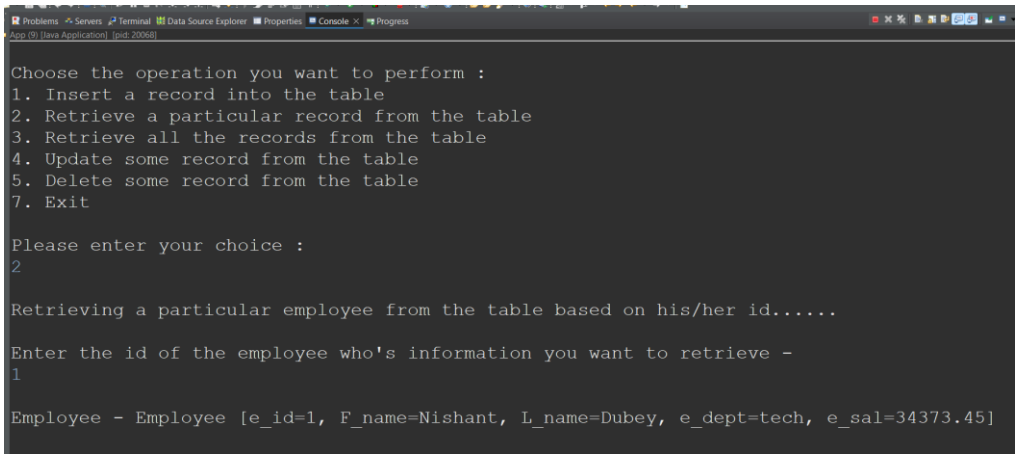
INPUT/OUTPUT

Inserting into database

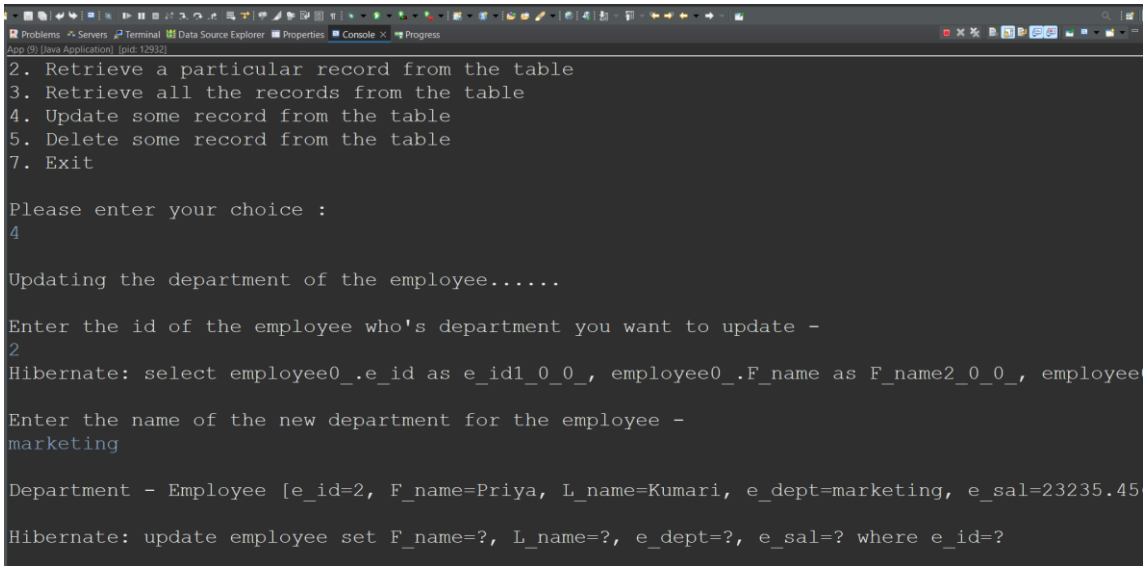




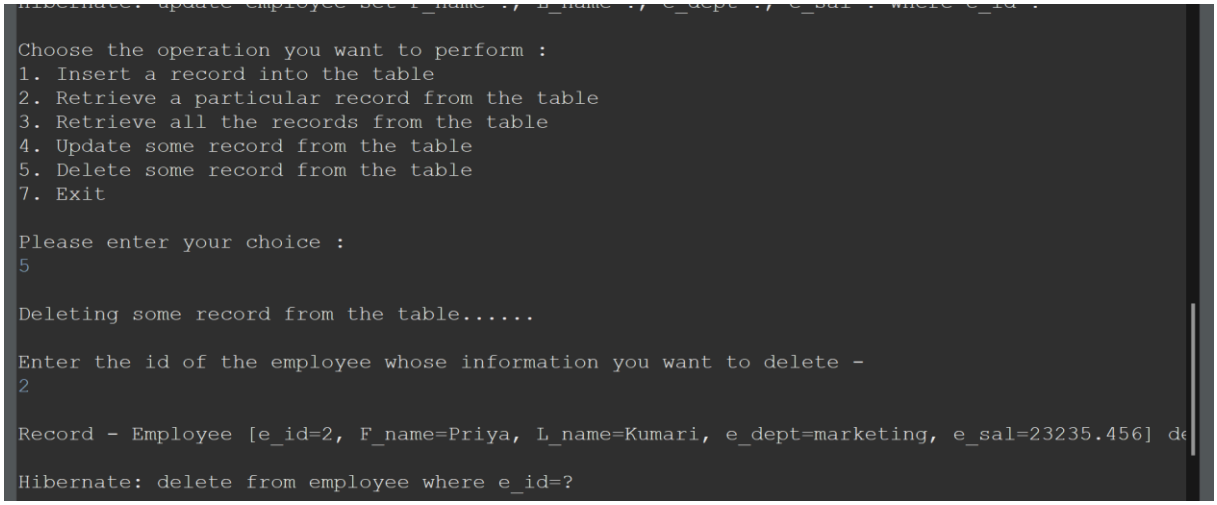
Retrieving a particular record from the table



Updating any record from the table



Deleting any record from the table



Exiting the program