

به نام خداوند بخشنده مهربان

محمد رضا مسیب زاده

تمرین جاوا :

https://github.com/M-Mosaiebzadeh/Maktab_W16.git

تمرین شماره ۱

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "footballGame". The `src/main/java` package contains the `App.java` file and several DAO classes (`AbstractDao`, `CityDao`, `CoachDao`, `CompetitionDao`, `ContractDao`, `DataAccess`, `MatchEventDao`, `PlayerDao`, `StadiumDao`, `TeamDao`, `UserDao`). It also includes an `entities` package with `enums` and various entity classes (`MatchEventType`, `City`, `Coach`, `Competition`, `Contract`, `IEntity`, `MatchEvent`, `Player`, `Stadium`, `Team`, `User`), and utility classes (`EntityManagerFactoryUtil`, `App`).
- Code Editor:** The `App.java` file is open, showing Java code for initializing DAOs and EntityManagers.
- Toolbars and Status Bar:** The top bar shows standard menu items like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help, and the current file `footballGame - App.java`. The bottom status bar shows the date and time (20:32, 12/30/2020), and the bottom right corner has an "Event Log" button.

The screenshot shows the Java code for the `App` class in the `footballGame` project. The code performs various database operations using Entity Manager and DAO classes.

```
public static void main(String[] args) {
    EntityManager entityManager = createEntityManagerFactory().createEntityManager();
    daoInitialization(entityManager);
    entityManager.getTransaction().begin();

    //create data
    createCompetition();

    //update match event
    MatchEvent matchEvent = matchEventDao.load(1);
    matchEvent.setType(RED_CARD);
    matchEventDao.update(matchEvent);

    //delete matchEvent <> not dependent > to other entities
    matchEventDao.delete(matchEventDao.load(1));

    //delete competition <> dependent > to match events
    competitionDao.delete(competitionDao.load(1));

    //delete stadium <> dependent > to competition
    stadiumDao.delete(stadiumDao.load(1));

    //delete contract <> not dependent > to other entities
    contractDao.delete(contractDao.load(1));
}
```

The screenshot shows the Java code for the `App` class in the `footballGame` project. The code performs various database operations using Entity Manager and DAO classes.

```
// contractDao.delete(contractDao.load(1));

//delete team <> dependent > to competition and contract
teamDao.delete(teamDao.load(1));

//delete city <> dependent > to team and stadium
cityDao.delete(cityDao.load(1));

//delete user directly <> dependent > to contract & player <> dependent > to competition and match
userDao.delete(userDao.load(1));
userDao.delete(userDao.load(3));

entityManager.getTransaction().commit();
entityManager.close();
shutdown();
}

public static void daoInitialization(EntityManager entityManager) {
    userDao = new UserDao(entityManager);
    playerDao = new PlayerDao(entityManager);
    coachDao = new CoachDao(entityManager);
    competitionDao = new CompetitionDao(entityManager);
    cityDao = new CityDao(entityManager);
    contractDao = new ContractDao(entityManager);
    matchEventDao = new MatchEventDao(entityManager);
    stadiumDao = new StadiumDao(entityManager);
    teamDao = new TeamDao(entityManager);
}
```

The screenshot shows a Java application named 'footballGame' with a main class 'App.java'. The code creates a player, coach, city, and team, and saves them to their respective DAOs. It then creates contracts for the player, coach, and stadium, and saves them to the contract DAO.

```
    }
    public static void createCompetition() {
        //team 1
        Player player = new Player();
        player.setFirstname(faker.name().firstName());
        player.setLastname(faker.name().lastName());
        player.setPosition("AB");
        playerDao.save(player);

        Player player2 = new Player();
        player2.setFirstname(faker.name().firstName());
        player2.setLastname(faker.name().lastName());
        player2.setPosition("CD");
        playerDao.save(player2);

        Coach coach = new Coach();
        coach.setFirstname(faker.name().firstName());
        coach.setLastname(faker.name().lastName());
        coachDao.save(coach);

        City city = new City();
        city.setCityName(faker.address().cityName());
        cityDao.save(city);

        Team team = new Team();
        team.setTeamName(faker.team().name());
        team.setCity(city);
        teamDao.save(team);

        Contract playerContract = new Contract();
        playerContract.setUser(player);
        playerContract.setTeam(team);
        playerContract.setSalary(1_000_000_000D);
        playerContract.setSeason(new Date(2020L));
        contractDao.save(playerContract);

        Contract player2Contract = new Contract();
        player2Contract.setUser(player2);
        player2Contract.setTeam(team);
        player2Contract.setSalary(1_200_000_000D);
        player2Contract.setSeason(new Date(2020L));
        contractDao.save(player2Contract);

        Contract coachContract = new Contract();
        coachContract.setUser(coach);
        coachContract.setTeam(team);
        coachContract.setSalary(2_200_000_000D);
        coachContract.setSeason(new Date(2020L));
        contractDao.save(coachContract);

        //match playing in stadium
        //=====
        Stadium stadium = new Stadium();
        stadium.setStadiumName("Azadi");
        stadium.setCapacity(65000L);
        stadium.setCity(city);
        stadiumDao.save(stadium);
        //=====
```

This screenshot is identical to the one above, showing the same Java code for creating a competition, player, coach, city, team, and contracts.

```
    }
    public static void createCompetition() {
        //team 1
        Player player = new Player();
        player.setFirstname(faker.name().firstName());
        player.setLastname(faker.name().lastName());
        player.setPosition("AB");
        playerDao.save(player);

        Player player2 = new Player();
        player2.setFirstname(faker.name().firstName());
        player2.setLastname(faker.name().lastName());
        player2.setPosition("CD");
        playerDao.save(player2);

        Coach coach = new Coach();
        coach.setFirstname(faker.name().firstName());
        coach.setLastname(faker.name().lastName());
        coachDao.save(coach);

        City city = new City();
        city.setCityName(faker.address().cityName());
        cityDao.save(city);

        Team team = new Team();
        team.setTeamName(faker.team().name());
        team.setCity(city);
        teamDao.save(team);

        Contract playerContract = new Contract();
        playerContract.setUser(player);
        playerContract.setTeam(team);
        playerContract.setSalary(1_000_000_000D);
        playerContract.setSeason(new Date(2020L));
        contractDao.save(playerContract);

        Contract player2Contract = new Contract();
        player2Contract.setUser(player2);
        player2Contract.setTeam(team);
        player2Contract.setSalary(1_200_000_000D);
        player2Contract.setSeason(new Date(2020L));
        contractDao.save(player2Contract);

        Contract coachContract = new Contract();
        coachContract.setUser(coach);
        coachContract.setTeam(team);
        coachContract.setSalary(2_200_000_000D);
        coachContract.setSeason(new Date(2020L));
        contractDao.save(coachContract);

        //match playing in stadium
        //=====
        Stadium stadium = new Stadium();
        stadium.setStadiumName("Azadi");
        stadium.setCapacity(65000L);
        stadium.setCity(city);
        stadiumDao.save(stadium);
        //=====
```

The screenshot shows the IntelliJ IDEA interface with the footballGame project open. The Project tool window on the left displays the directory structure, including src/main/java/App.java. The Editor tab bar shows 'App.java'. The code in App.java is as follows:

```
//team 2
Player player21 = new Player();
player21.setFirstname(faker.name().firstName());
player21.setLastname(faker.name().lastName());
player21.setPosition("AB");
playerDao.save(player21);

Player player22 = new Player();
player22.setFirstname(faker.name().firstName());
player22.setLastname(faker.name().lastName());
player22.setPosition("CD");
playerDao.save(player22);

Coach coach2 = new Coach();
coach2.setFirstname(faker.name().firstName());
coach2.setLastname(faker.name().lastName());
coachDao.save(coach2);

City city2 = new City();
city2.setCityName(faker.address().cityName());
cityDao.save(city2);

Team team2 = new Team();
team2.setTeamName(faker.team().name());
team2.setCity(city2);
teamDao.save(team2);

Contract player21Contract = new Contract();
player21Contract.setUser(player21);
player21Contract.setTeam(team2);
player21Contract.setSalary(new BigDecimal("1000000000"));

Contract player22Contract = new Contract();
player22Contract.setUser(player22);
player22Contract.setTeam(team2);
player22Contract.setSalary(new BigDecimal("1500000000"));
player22Contract.setSeason(new Date(2020L));
contractDao.save(player22Contract);

Contract coach2Contract = new Contract();
coach2Contract.setUser(coach2);
coach2Contract.setTeam(team2);
coach2Contract.setSalary(new BigDecimal("1900000000"));
coach2Contract.setSeason(new Date(2020L));
contractDao.save(coach2Contract);

//=====
//competition team1 => home && team2 => away
Competition competition = new Competition();
```

The screenshot shows the IntelliJ IDEA interface with the footballGame project open. The Project tool window on the left displays the directory structure, including src/main/java/App.java. The Editor tab bar shows 'App.java'. The code in App.java is as follows:

```
Team team2 = new Team();
team2.setTeamName(faker.team().name());
team2.setCity(city2);
teamDao.save(team2);

Contract player21Contract = new Contract();
player21Contract.setUser(player21);
player21Contract.setTeam(team2);
player21Contract.setSalary(new BigDecimal("900_000_0000"));
player21Contract.setSeason(new Date(2020L));
contractDao.save(player21Contract);

Contract player22Contract = new Contract();
player22Contract.setUser(player22);
player22Contract.setTeam(team2);
player22Contract.setSalary(new BigDecimal("1_500_000_0000"));
player22Contract.setSeason(new Date(2020L));
contractDao.save(player22Contract);

Contract coach2Contract = new Contract();
coach2Contract.setUser(coach2);
coach2Contract.setTeam(team2);
coach2Contract.setSalary(new BigDecimal("1_900_000_0000"));
coach2Contract.setSeason(new Date(2020L));
contractDao.save(coach2Contract);

//=====
//competition team1 => home && team2 => away
Competition competition = new Competition();
```

The screenshot shows the Java code for `App.java` in the `footballGame` project. The code handles player registration and competition setup.

```
Set<Player> homePlayers = new HashSet<>();
homePlayers.add(player);
homePlayers.add(player2);

Set<Player> awayPlayers = new HashSet<>();
awayPlayers.add(player21);
awayPlayers.add(player22);

competition.setAwayPlayers(awayPlayers);
competition.setHomePlayers(homePlayers);
competitionDao.save(competition);

//=====

//Events in match
MatchEvent matchEvent1 = new MatchEvent();
matchEvent1.setPlayer(player);
matchEvent1.setType(YELLOW_CARD);
matchEvent1.setTime(new Date());
matchEvent1.setCompetition(competition);
matchEventDao.save(matchEvent1);

MatchEvent matchEvent2 = new MatchEvent();
matchEvent2.setPlayer(player22);
matchEvent2.setType(INJURY);
matchEvent2.setTime(new Date());
matchEvent2.setCompetition(competition);
matchEventDao.save(matchEvent2);
```

The screenshot shows the run output for `App.java`. It displays a series of Hibernate SQL insert statements for various entities like User, Coach, City, Contract, and Stadium, followed by connection cleanup logs.

```
Hibernate: insert into user (first_name, last_name, position, DTYPE) values (?, ?, ?, 'Player')
Hibernate: insert into user (first_name, last_name, position, DTYPE) values (?, ?, ?, 'Player')
Hibernate: insert into user (first_name, last_name, DTYPE) values (?, ?, 'Coach')
Hibernate: insert into city (city_name) values (?)
Hibernate: insert into team (fk_city, team_name) values (?, ?)
Hibernate: insert into contract (salary, season, fk_team, fk_user) values (?, ?, ?, ?)
Hibernate: insert into contract (salary, season, fk_team, fk_user) values (?, ?, ?, ?)
Hibernate: insert into contract (salary, season, fk_team, fk_user) values (?, ?, ?, ?)
Hibernate: insert into stadium (capacity, fk_city, stadium_name) values (?, ?, ?)
Hibernate: insert into user (first_name, last_name, position, DTYPE) values (?, ?, ?, 'Player')
Hibernate: insert into user (first_name, last_name, position, DTYPE) values (?, ?, ?, 'Player')
Hibernate: insert into user (first_name, last_name, DTYPE) values (?, ?, 'Coach')
Hibernate: insert into city (city_name) values (?)
Hibernate: insert into team (fk_city, team_name) values (?, ?)
Hibernate: insert into contract (salary, season, fk_team, fk_user) values (?, ?, ?, ?)
Hibernate: insert into contract (salary, season, fk_team, fk_user) values (?, ?, ?, ?)
Hibernate: insert into contract (salary, season, fk_team, fk_user) values (?, ?, ?, ?)
Hibernate: insert into competition (fk_away_team, fk_home_team, season, fk_stadium) values (?, ?, ?, ?)
Hibernate: insert into match_event (fk_competition, fk_player, time, type) values (?, ?, ?, ?)
Hibernate: insert into match_event (fk_competition, fk_player, time, type) values (?, ?, ?, ?)
Hibernate: insert into competition_away_player (competition, away_player) values (?, ?)
Hibernate: insert into competition_away_player (competition, away_player) values (?, ?)
Hibernate: insert into competition_home_player (competition, home_player) values (?, ?)
Hibernate: insert into competition_home_player (competition, home_player) values (?, ?)
Dec 30, 2020 11:36:15 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/football2]

Process finished with exit code 0
```

دیتابیس این سوال به همراه جدول یوزر ها:

MySQL Workbench

Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator team user ×

1 • SELECT * FROM football2.user;

Result Grid | Filter Rows | Edit: | Export/Import: | Wrap Cell Content: |

DTYPE	user_id	first_name	last_name	position
Player	1	Cherlyn	Stracke	AB
Player	2	Edmund	McDermott	CD
Coach	3	Cherryl	Hettinger	HULL
Player	4	Frederic	Cremin	AB
Player	5	Alfredo	Kilback	CD
Coach	6	Florencio	Luettgen	HULL
HULL	HULL	HULL	HULL	HULL

user 1 ×

Output

Action Output

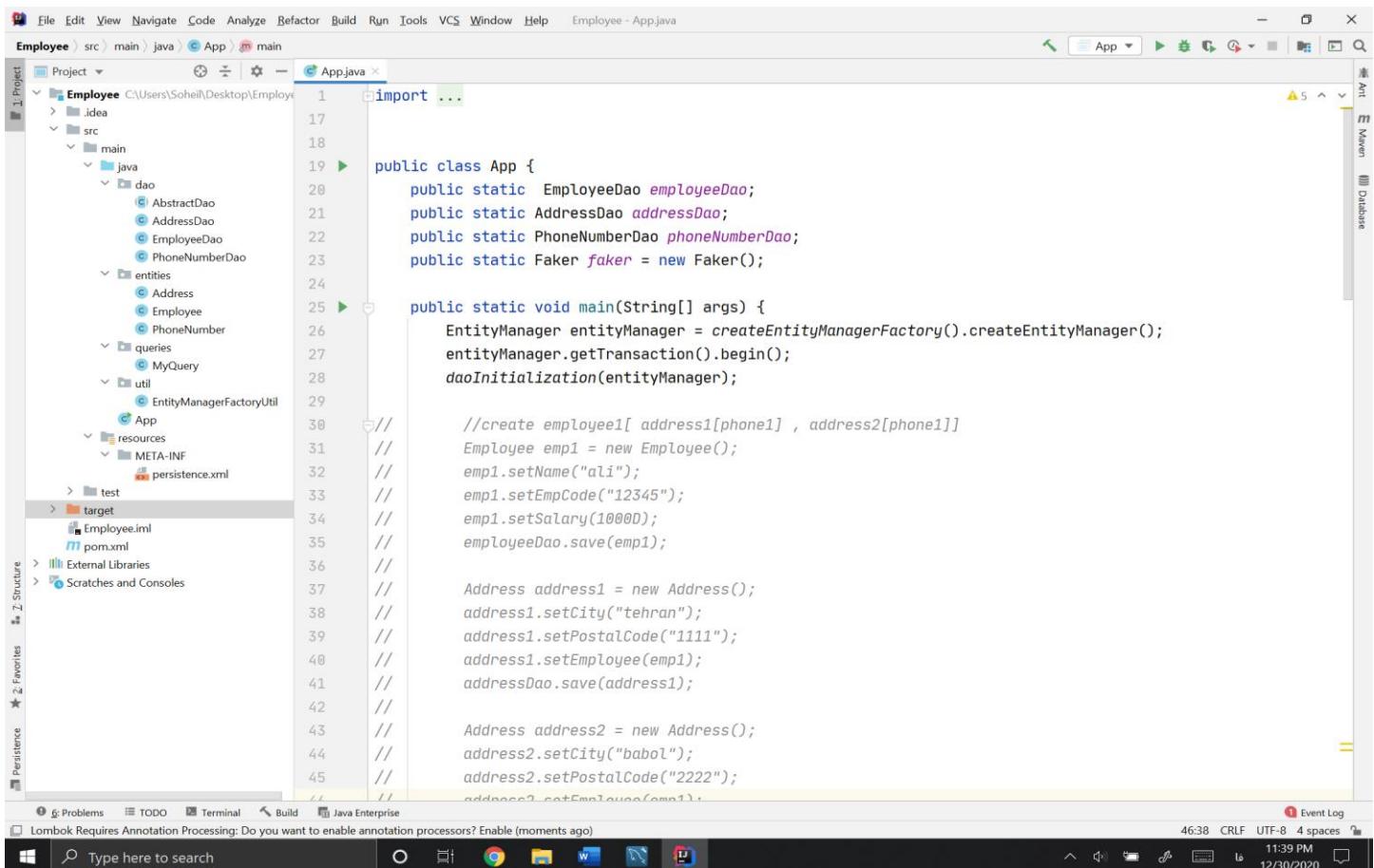
#	Time	Action	Message	Duration / Fetch
1	23:37:16	SELECT * FROM football2.user LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

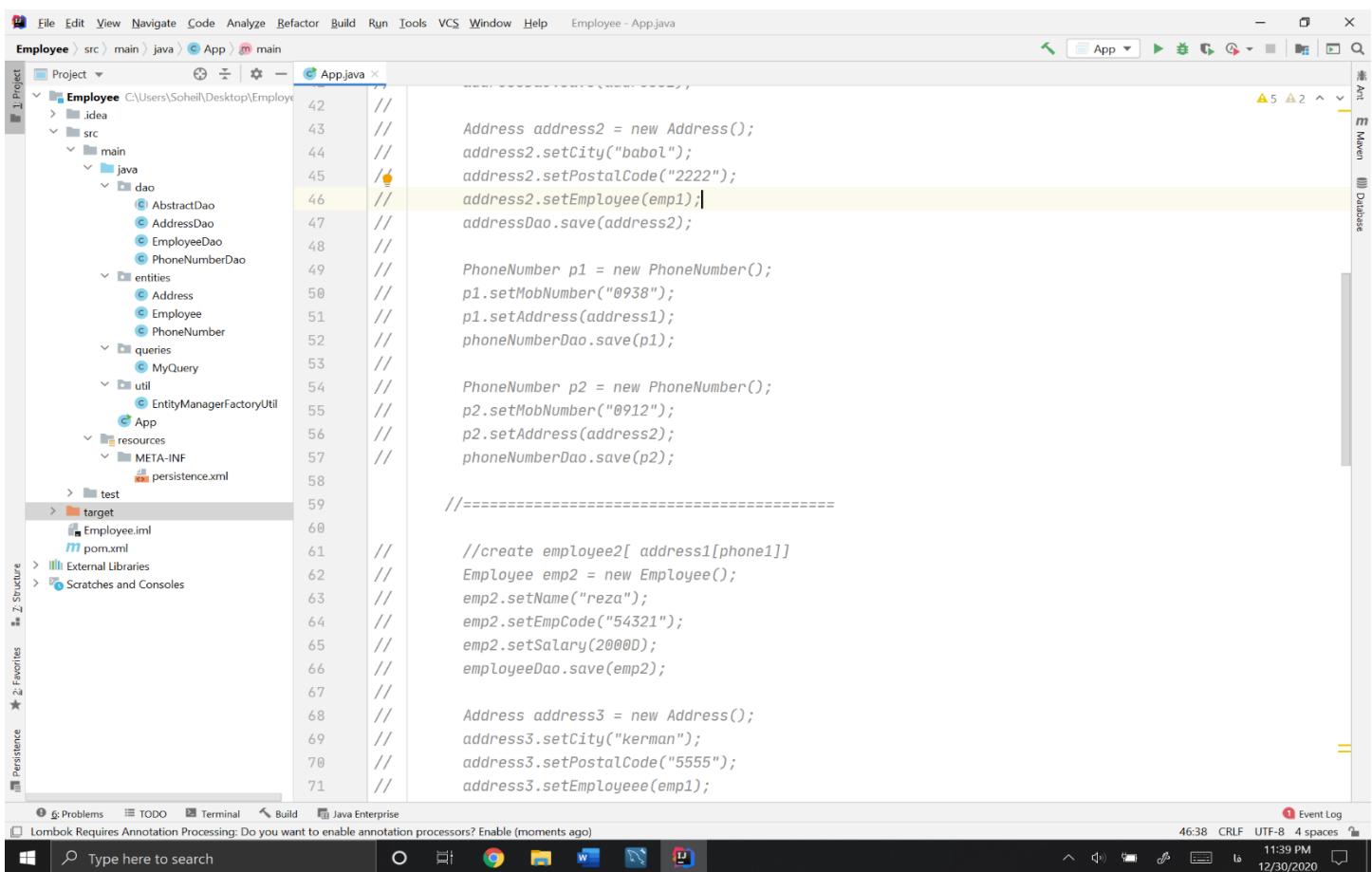
Type here to search

11:37 PM
12/30/2020

تمرین دوم :



```
import ...  
  
public class App {  
    public static EmployeeDao employeeDao;  
    public static AddressDao addressDao;  
    public static PhoneNumberDao phoneNumberDao;  
    public static Faker faker = new Faker();  
  
    public static void main(String[] args) {  
        EntityManager entityManager = createEntityManagerFactory().createEntityManager();  
        entityManager.getTransaction().begin();  
        daoInitialization(entityManager);  
  
        //create employee1[ address1[phone1] , address2[phone1]  
        Employee emp1 = new Employee();  
        emp1.setName("ali");  
        emp1.setEmpCode("12345");  
        emp1.setSalary(10000);  
        employeeDao.save(emp1);  
  
        Address address1 = new Address();  
        address1.setCity("tehran");  
        address1.setPostalCode("1111");  
        address1.setEmployee(emp1);  
        addressDao.save(address1);  
  
        Address address2 = new Address();  
        address2.setCity("babol");  
        address2.setPostalCode("2222");  
        address2.setEmployee(emp1);  
        addressDao.save(address2);  
  
        //=====  
        //create employee2[ address1[phone1]  
        Employee emp2 = new Employee();  
        emp2.setName("reza");  
        emp2.setEmpCode("54321");  
        emp2.setSalary(20000);  
        employeeDao.save(emp2);  
  
        Address address3 = new Address();  
        address3.setCity("kerman");  
        address3.setPostalCode("5555");  
        address3.setEmployee(emp1);  
        addressDao.save(address3);  
    }  
}
```



```
//  
//  
Address address2 = new Address();  
address2.setCity("babol");  
address2.setPostalCode("2222");  
address2.setEmployee(emp1);  
addressDao.save(address2);  
  
PhoneNumber p1 = new PhoneNumber();  
p1.setMobNumber("0938");  
p1.setAddress(address1);  
phoneNumberDao.save(p1);  
  
PhoneNumber p2 = new PhoneNumber();  
p2.setMobNumber("0912");  
p2.setAddress(address2);  
phoneNumberDao.save(p2);  
  
//=====  
//create employee2[ address1[phone1]  
Employee emp2 = new Employee();  
emp2.setName("reza");  
emp2.setEmpCode("54321");  
emp2.setSalary(20000);  
employeeDao.save(emp2);  
  
Address address3 = new Address();  
address3.setCity("kerman");  
address3.setPostalCode("5555");  
address3.setEmployee(emp1);  
addressDao.save(address3);
```

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar displays the project structure under "Employee".
- Code Editor:** The main window shows the content of `App.java`. The code implements a DAO pattern for managing employees and their addresses and phone numbers.
- Toolbars and Status Bar:** The top bar includes standard file operations like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help, and a specific tab for "Employee - App.java". The bottom status bar shows the current time as 11:39 PM and the date as 12/30/2020.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Employee - App.java
Employee > src > main > java > App > main
Project Employee C:\Users\Soheil\Desktop\Employee
> .idea
src
  > main
    > java
      > dao
        AbstractDao
        AddressDao
        EmployeeDao
        PhoneNumberDao
      entities
        Address
        Employee
        PhoneNumber
      queries
        MyQuery
      util
        EntityManagerFactoryUtil
      App
      resources
        META-INF
          persistence.xml
      test
      target
        EmployeeImpl
        pom.xml
External Libraries
Scratches and Consoles

Employee.java
67 // Address address3 = new Address();
68 // address3.setCity("kerman");
69 // address3.setPostalCode("5555");
70 // address3.setEmployee(emp1);
71 // address3.setEmployee(emp2);
72 // addressDao.save(address3);

73 // PhoneNumber p3 = new PhoneNumber();
74 // p3.setMobNumber("0902");
75 // p3.setAddress(address3);
76 // phoneNumberDao.save(p3);

77 // =====
78 // =====
79 // =====
80 // =====
81 // =====
82 // =====
83 // MyQuery myQuery = new MyQuery(entityManager);
84 // System.out.println(myQuery.mostEmployeeSalaryNative());
85 // System.out.println(myQuery.mostEmployeeSalaryJPQL());
86 // System.out.println(myQuery.mostSalaryJPQL());
87 // System.out.println(myQuery.mostSalaryNative());
88 // =====
89 // =====
90 // =====
91 // // delete phone number if exist
92 // System.out.println(employeeDao.load(1));
93 // if (phoneNumberDao.load(1) != null){
94 //   phoneNumberDao.delete(phoneNumberDao.load(1));
95 //   System.out.println("phone number is deleted");
96 // }
97 // 
```

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Bar:** Shows the project structure under "Employee".
- Code Editor:** Displays the content of `App.java`. The code handles the deletion of various entities (Employee, Address, PhoneNumber) and their associated DAOs. It also includes static methods for creating employees.
- Toolbars and Status Bar:** Standard IntelliJ IDEA toolbars and status bar at the bottom.
- Event Log:** Shows a message about Lombok annotation processing.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Employee - App.java

Employee > src > main > java > App > main

Project Employee C:\Users\Soheil\Desktop\Employee 93 // if (phoneNumberDao.load(1) != null){ 94 // phoneNumberDao.delete(phoneNumberDao.load(1)); 95 // System.out.println("phone number is deleted"); 96 // } 97 // 98 // //delete address if exist 99 // if (addressDao.load(3) != null){ 100 // addressDao.delete(addressDao.load(3)); 101 // System.out.println("address is deleted"); 102 // } 103 // 104 // //delete employee if exist 105 // if (employeeDao.load(1) != null){ 106 // employeeDao.delete(employeeDao.load(1)); 107 // System.out.println("employee is deleted"); 108 // } 109 // 110 // entityManager.getTransaction().commit(); 111 entityManager.close(); 112 shutdown(); 113 114 } 115 public static void daoInitialization(EntityManager entityManager) { 116 employeeDao = new EmployeeDao(entityManager); 117 addressDao = new AddressDao(entityManager); 118 phoneNumberDao = new PhoneNumberDao(entityManager); 119 } 120 @ 121 public static Employee createEmployee1() {...} 122 public static void createEmployee2(Employee emp) {...} 123 }
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Employee - App.java

Employee > src > main > java > App > main

Project Run: App

```

Dec 30, 2020 11:42:15 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=root, password=****}
Dec 30, 2020 11:42:15 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Dec 30, 2020 11:42:15 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
Dec 30, 2020 11:42:16 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
Dec 30, 2020 11:42:19 PM org.hibernate.resource.transaction.backend.jdbc.internal.JdbcTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionPr
Hibernate: create table address (address_id integer not null auto_increment, city varchar(255), postal_address varchar(255), postal_code varc
Hibernate: create table employee (employee_id integer not null auto_increment, emp_code varchar(255), name varchar(255), salary double precisi
Hibernate: create table phone_number (phone_number_id integer not null auto_increment, mob_name varchar(255), tel_name varchar(255), fk_addres
Hibernate: alter table address add constraint FKmdvad9l0j5ex1khs2l45fxooq foreign key (fk_employee) references employee (employee_id)
Hibernate: alter table address add constraint FK738dow27btb4b145x8ri3ww5f foreign key (employee) references employee (employee_id)
Hibernate: alter table phone_number add constraint FKleddji44bt9vl5lh0niti1d6y0 foreign key (fk_address) references address (address_id)
Hibernate: insert into employee (emp_code, name, salary) values (?, ?, ?)
Hibernate: insert into address (city, fk_employee, employee, postal_address, postal_code) values (?, ?, ?, ?, ?)
Hibernate: insert into address (city, fk_employee, employee, postal_address, postal_code) values (?, ?, ?, ?, ?)
Hibernate: insert into phone_number (fk_address, mob_name, tel_name) values (?, ?, ?)
Hibernate: insert into phone_number (fk_address, mob_name, tel_name) values (?, ?, ?)
Hibernate: insert into employee (emp_code, name, salary) values (?, ?, ?)
Hibernate: insert into address (city, fk_employee, employee, postal_address, postal_code) values (?, ?, ?, ?, ?)
Hibernate: insert into phone_number (fk_address, mob_name, tel_name) values (?, ?, ?)
Dec 30, 2020 11:42:20 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/employee]

Process finished with exit code 0

```

Build completed successfully in 7 s 926 ms (moments ago)

Event Log 44:1 CRLF UTF-8 4 spaces

11:42 PM ENG 12/30/2020

دیتابیس employee به همراه تیبل address

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

SCHEMAS

- book
- employee**
 - Tables: address, employee, phone_number
 - Views
 - Stored Procedures
 - Functions
- football
- football2
- football_league
- maktab
- sakila
- school
- student
- sys

Administration Schemas

No object selected

address 1

Output:

address_id	city	postal_address	postal_code	fk_employee	employee
1	tehran	HULL	1111	1	HULL
2	babol	HULL	2222	1	HULL
3	kerman	HULL	5555	2	1

Action Output:

#	Time	Action	Message	Duration / Fetch
2	23:41:16	DROP DATABASE 'employee'	3 row(s) affected	0.391 sec
3	23:41:28	Apply changes to employee	Changes applied	
4	23:42:34	SELECT * FROM employee.employee LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
5	23:42:37	SELECT * FROM employee.address LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

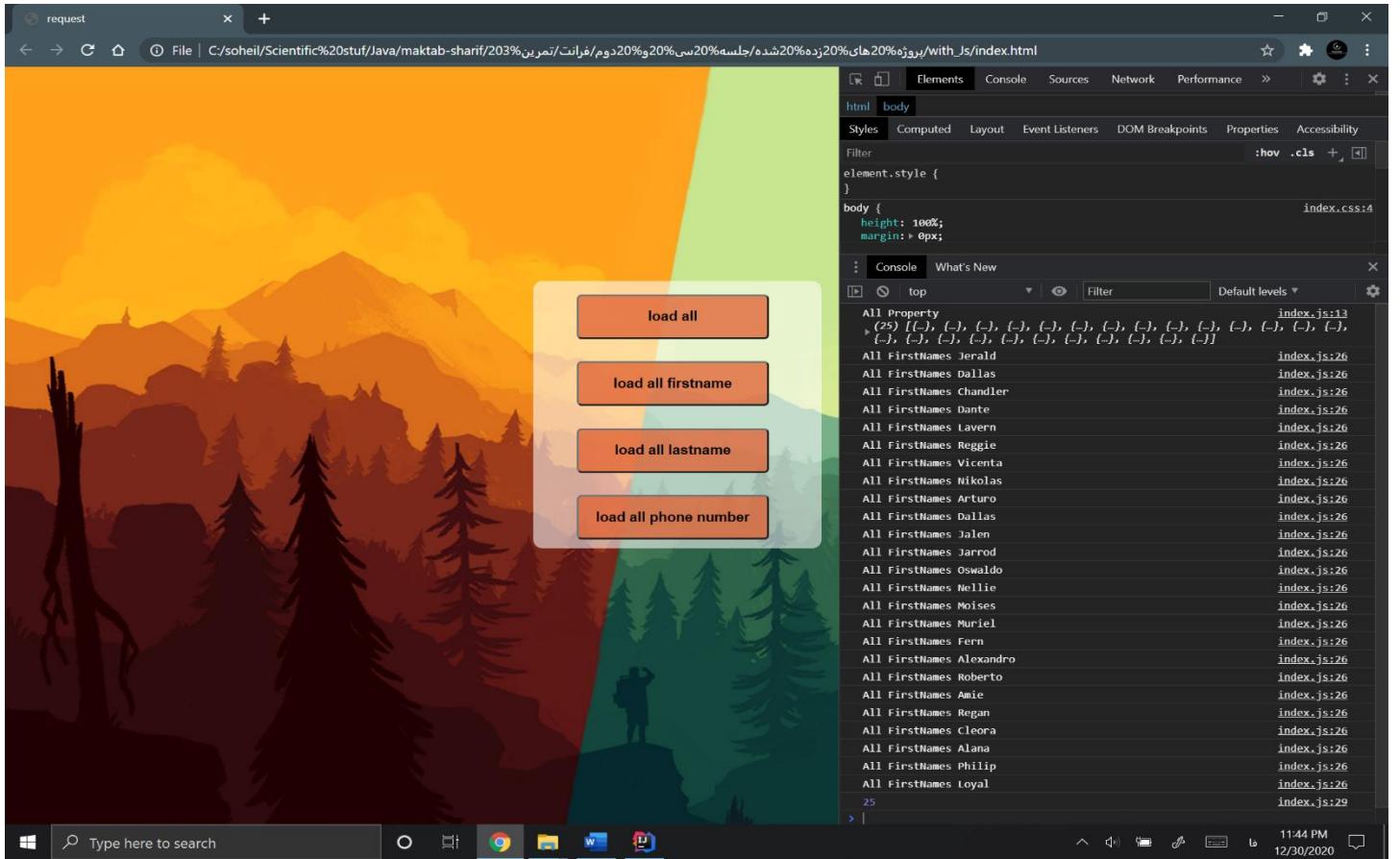
Object Info Session

Type here to search

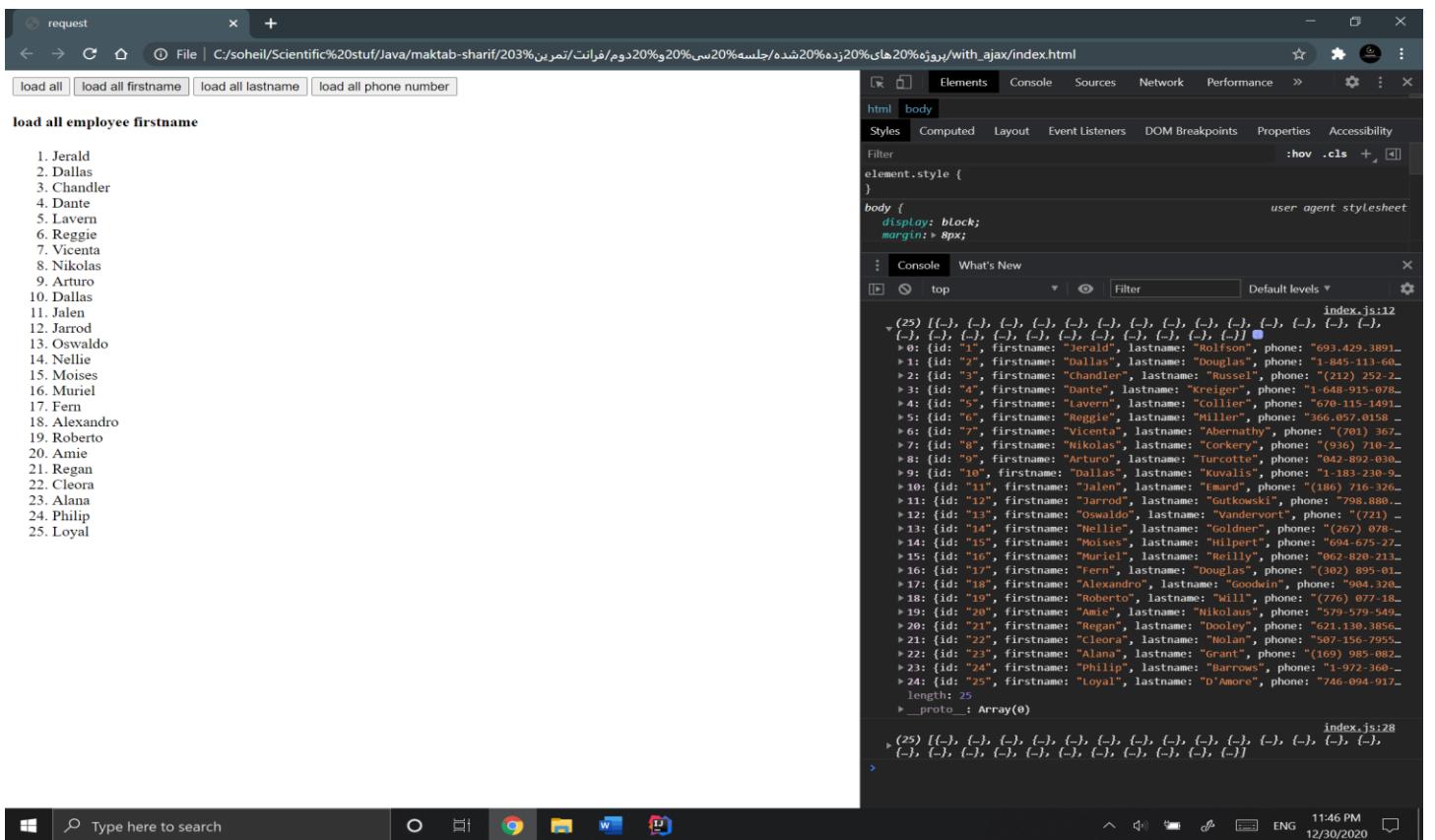
Event Log 44:1 CRLF UTF-8 4 spaces

11:42 PM ENG 12/30/2020

تمرین سوم روش اول js:



تمرین سوم روش اول ajax:



تمرین چهارم:

