Spring Data JPA part 1 Exercise :

(1) Create an Employee Entity which contains following fields

Name

Id

Age

Location

```
//
package com.example.employee.entities;
import javax.persistence.*;
@Entity // mapping entity to database table
@Table(name = "employee") // table name in db
public class Employee {
    @Id // annotation for primary key
    @GeneratedValue(strategy = GenerationType.IDENTITY)// telling
that the id is auto incremented
    private int id;
    private String name;
    private int age;
    @Column(name = "location") // column name in the database is
location
    private String loc;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getLoc() {
        return loc;
    }
    public void setLoc(String loc) {
        this.loc = loc;
```

```
    }
    @Override
    public String toString() {
        return "Employee{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", age=" + age +
                ", loc='" + loc + '\'' +
                '}';
    }
}
```

(2) Set up EmployeeRepository with Spring Data JPA

```
package com.example.employee.repos;
import com.example.employee.entities.Employee;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository;
import java.util.List;
public interface EmployeeRepo extends
PagingAndSortingRepository<Employee,Integer> {
    // PagingAndSortingRepository extends CurdRepository
    List<Employee>findByName(String name); // finder for Q-8
    List<Employee>findByNameLike(String name);// finder for Q-9
    List<Employee>findByAgeBetween(int age1,int age2); // finder
for Q-10
}
```

(3) Perform Create Operation on Entity using Spring Data JPA

```
@Test
public void createEmployee() { //create Q-3
    Employee emp1 = new Employee();
    emp1.setName("jimmy");
    emp1.setAge(45);
    emp1.setLoc("berlin");
    employeeRepo.save(emp1);
    Employee emp2 = new Employee();
    emp2.setName("dom");
    emp2.setAge(16);
    emp2.setLoc("brimingham");
    employeeRepo.save(emp2);
}
```

Result Grid Filter Rows:

#	id	name	age	location
1	1	jimmy	45	berlin
2	2	dom	16	brimingham
*	NULL	NULL	NULL	NULL

(4) Perform Update Operation on Entity using Spring Data JPA

```
@Test
public void updateEmployee() { //update Q-4
    if (employeeRepo.existsById(2)) {
        System.out.println("------employee exist--------");
        Employee emp = employeeRepo.findById(2).get();
        emp.setLoc("london"); // updating employee location
        employeeRepo.save(emp);
    } else {
        System.out.println("------employee not exist--------");
    }
}
```

```
2020-04-09 13:27:37.588  INFO 4101 --- [    Test worker] c.e.employee.EmployeeApplicationTests    :
------employee exist--------
2020-04-09 13:27:38.026  INFO 4101 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  :
```

#	id	name	age	location
1	1	jimmy	45	berlin
2	2	dom	16	london
*	NULL	NULL	NULL	NULL

(5) Perform Read Operation on Entity using Spring Data JPA

```
@Test // Read Q-5
public void readEmployee() {
    Employee emp = employeeRepo.findById(1).get();
    assertNotNull(emp);
    assertEquals("jimmy", emp.getName());
    System.out.println("reading employee data");
    System.out.println(emp.getName() + "-------------->" +
emp.getLoc());// reading the data

System.out.println("-------------------------------------------")
;
}
```

```
2020-04-09 13:30:17.079  INFO 4238 --- [    Test worke
2020-04-09 13:30:17.500  INFO 4238 --- [    Test worke
reading employee data
jimmy-------------->berlin
---------------------------------------------
```

(6) Perform Delete Operation on Entity using Spring Data JPA

```java
@Test //Delete Q-6
public void deleteEmployee() {
    if (employeeRepo.existsById(6)) {
        System.out.println("------employee exist--------");
        employeeRepo.deleteById(6); // deleting the employee
    } else {
        System.out.println("------employee not exist--------");
    }
}
```

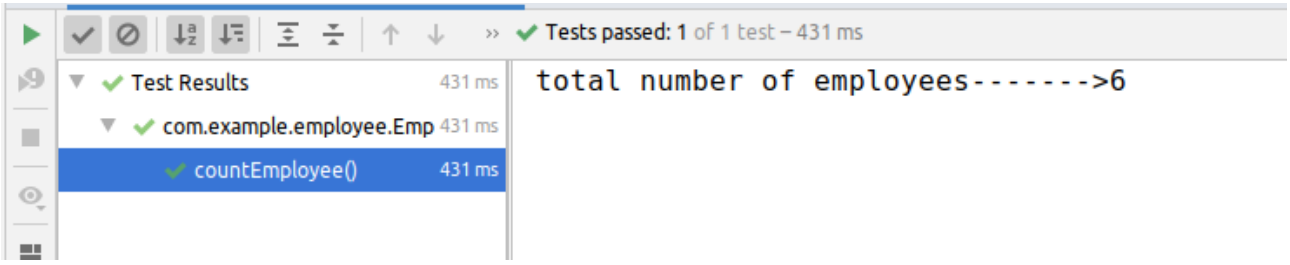
#	id	name	age	location
1	1	jimmy	45	berlin
2	2	dom	16	london
3	3	alpha	28	new jersey
4	4	omega	45	kuwait
5	5	alpha	28	new jersey
6	6	omega	45	kuwait
7	7	otis	24	england
8	8	eric	21	baker street
*	NULL	NULL	NULL	NULL

Result Grid Filter Rows:

#	id	name	age	location
1	1	jimmy	45	berlin
2	2	dom	16	london
3	3	alpha	28	new jersey
4	4	omega	45	kuwait
5	7	otis	24	england
6	8	eric	21	baker street
*	NULL	NULL	NULL	NULL

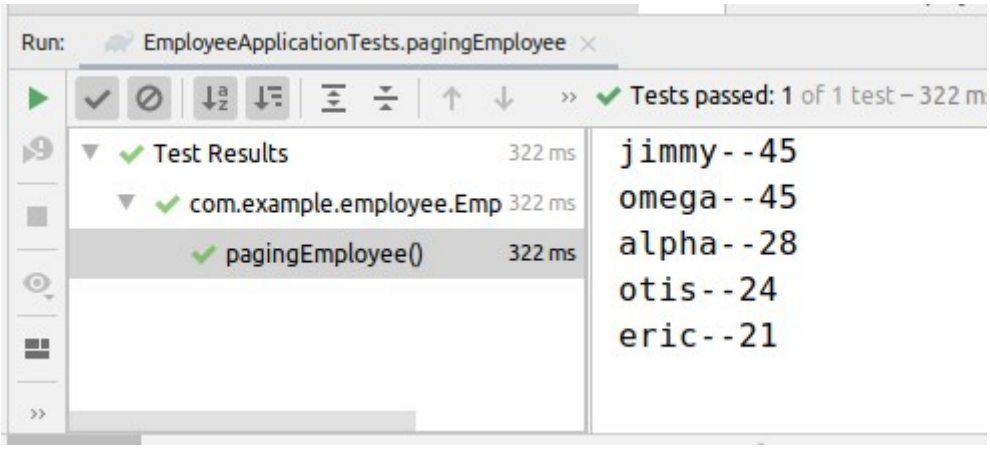
(7) Get the total count of the number of Employees

```
@Test //count employee Q-7
public void countEmployee() {
    System.out.println("total number of employees------->" +
employeeRepo.count());
}
```



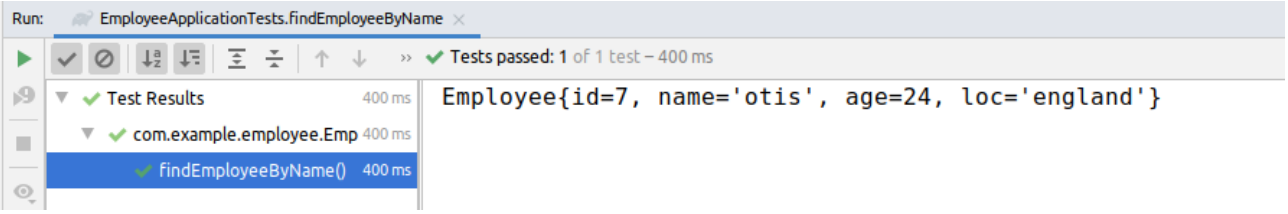
(8) Implement Pagination and Sorting on the bases of Employee Age

```
  @Test //(8) Implement Pagination and Sorting on the bases of
Employee Age
public void pagingEmployee(){
   Pageable pageable= PageRequest.of(0,5,
Sort.Direction.DESC,"age");
   Page<Employee> employees=employeeRepo.findAll(pageable);
   employees.forEach(e-> System.out.println(e.getName()
+"--"+e.getAge()));
}
```



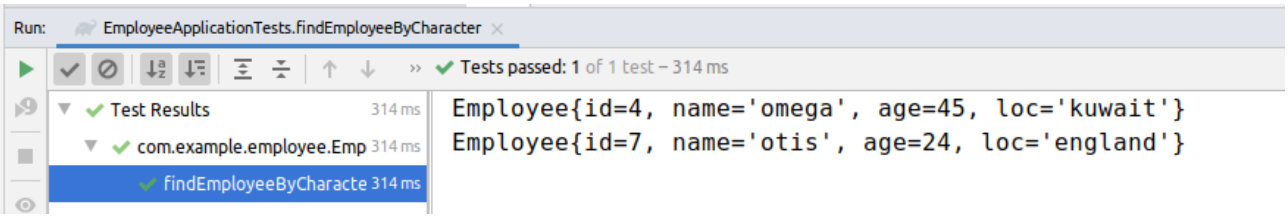
(9) Create and use finder to find Employee by Name

```
@Test //Q-9Create and use finder to find Employee by Name
public void findEmployeeByName() {
    List<Employee> employees = employeeRepo.findByName("otis");
    if(employees==null){
        System.out.println("employee with that name not found");
    }
    employees.forEach(p -> System.out.println(p.toString()));
}
```



(10) Create and use finder to find Employees starting with A character

```
@Test //(10) Create and use finder to find Employees starting with
'A' character
public void findEmployeeByCharacter() {
    List<Employee> employees = employeeRepo.findByNameLike("o%");
    employees.forEach(p -> System.out.println(p.toString()));
}
```



(11) Create and use finder to find Employees Between the age of 28 to 32

```
@Test //(11) Create and use finder to find Employees Between the
age of 21 to 28
public  void findEmployeeByAge(){
   List<Employee> employees = employeeRepo.findByAgeBetween(21,28);
   employees.forEach(e-> System.out.println(e.getName()
+"--"+e.getAge()));
}
```