Demonstrate implementation of O/R Mapping

1. One-to-Many / Many-to-One Mapping

Employee.java

```
@Entity
@Table(name = "employee")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    private double salary;

    @ManyToOne
    @JoinColumn(name = "em_dp_id")  // FK column in employee table
    private Department department;

    // Getters, setters, toString
}
```

Department.java

```
@Entity
@Table(name = "department")
public class Department {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;


    private String name;


    @OneToMany(mappedBy = "department", fetch = FetchType.EAGER)
    private Set<Employee> employeeList;


    // Getters, setters, toString
}
```

2. Many-to-Many Mapping

Employee.java

```
@ManyToMany(fetch = FetchType.EAGER)
@JoinTable(
    name = "employee_skill",
    joinColumns = @JoinColumn(name = "es_em_id"),
    inverseJoinColumns = @JoinColumn(name = "es_sk_id")
)
```

```
private Set<Skill> skillList;
```

Skill.java

```
@ManyToMany(mappedBy = "skillList")
private Set<Employee> employeeList;
```

3. Test in OrmLearnApplication.java

Fetching an Employee with Department & Skills

```
private static void testGetEmployee() {
   Employee employee = employeeService.get(1);
   LOGGER.debug("Employee: {}", employee);
   LOGGER.debug("Department: {}", employee.getDepartment());
   LOGGER.debug("Skills: {}", employee.getSkillList());
}
```

Add Skill to Employee

```
private static void testAddSkillToEmployee() {
   Employee employee = employeeService.get(2);
   Skill skill = skillService.get(3);
   employee.getSkillList().add(skill);
   employeeService.save(employee);
}
```

1.Many-to-One: Employee → Department

```
Employee employee = employeeService.get(1);
LOGGER.debug("Employee: {}", employee);
```

LOGGER.debug("Department: {}", employee.getDepartment());

Output

```
Employee: Employee [id=1, name=John, salary=50000.0, permanent=true, dateOfBirth=1990-01-15]
Department: Department [id=1, name=HR]
```

2. One-to-Many: Department → List of Employees

Department dept = departmentService.get(1);

LOGGER.debug("Department: {}", dept);

LOGGER.debug("Employees: {}", dept.getEmployeeList());

Output

```
Department: Department [id=1, name=HR]
Employees: [
  Employee [id=1, name=John],
  Employee [id=2, name=Jane]
]
```

3. Many-to-Many: Employee ↔ Skills

Employee emp = employeeService.get(1);

LOGGER.debug("Employee: {}", emp);

LOGGER.debug("Skills: {}", emp.getSkillList());

output

```
Employee: Employee [id=1, name=John]
Skills: [
  Skill [id=1, name=Java],
  Skill [id=2, name=Python]
]
```

4. **Add Skill to Employee**

Employee emp = employeeService.get(1);

Skill skill = skillService.get(3); // Let's say: Angular

emp.getSkillList().add(skill);

employeeService.save(emp);

output

```
Start
Employee before: Employee [id=1, name=John, skills=[Java, Python]]
Employee after adding new skill: [Java, Python, Angular]
End
```

Demonstrate implementation of Query Methods feature of Spring Data JPA

Use Case: Country Entity

@Entity

@Table(name = "country")

```
public class Country {
    @Id
    @Column(name = "co_code")
    private String code;

    @Column(name = "co_name")
    private String name;

    // Getters, setters, toString()
}
```

CountryRepository With Query Methods

```
@Repository
public interface CountryRepository extends JpaRepository<Country, String> {

    // 1. Find countries that contain a substring in name
    List<Country> findByNameContaining(String keyword);

    // 2. Find countries that contain substring and sort by name ASC
    List<Country> findByNameContainingOrderByNameAsc(String keyword);

    // 3. Find countries whose name starts with a specific letter
    List<Country> findByNameStartingWith(String prefix);
}
```

Code in OrmLearnApplication.java

```
@Autowired
private static CountryRepository countryRepository;

private static void testQueryMethods() {
   LOGGER.info("Start");

   // Contains "ou"
   List<Country> containOu = countryRepository.findByNameContaining("ou");
   LOGGER.debug("Countries with 'ou': {}", containOu);

   // Contains "ou" ordered
   List<Country> containOuAsc =
countryRepository.findByNameContainingOrderByNameAsc("ou");
   LOGGER.debug("Countries with 'ou' sorted: {}", containOuAsc);

   // Starts with 'Z'
   List<Country> startsWithZ = countryRepository.findByNameStartingWith("Z");
   LOGGER.debug("Countries starting with Z: {}", startsWithZ);

   LOGGER.info("End");
}
```

OUTPUT:

Countries with 'ou': [South Africa, Luxembourg, Bouvet Island, Djibouti, Guadeloupe, ...]

Countries with 'ou' sorted: [Bouvet Island, Djibouti, French Southern Territories, ...]

Countries starting with Z: [Zambia, Zimbabwe]

SUPERSET ID: 6407636

KANMANI MURUGHAIYAN