**Hands On 1: Write queries on country table using Query Methods**   
  
**Country.java**

package com.cognizant.ormlearn.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
  
@Entity  
@Table(name = "country")  
public class Country {  
  
 @Id  
 private String code;  
 private String name;  
public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

**CountryRepository.java**

package com.cognizant.ormlearn.repository;  
  
import com.cognizant.ormlearn.entity.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
import java.util.List;  
  
public interface CountryRepository extends JpaRepository<Country, String> {  
 List<Country> findByNameContaining(String substring);  
 List<Country> findByNameContainingOrderByNameAsc(String substring);  
 List<Country> findByNameStartingWith(String prefix);  
}

**CountryService.java**

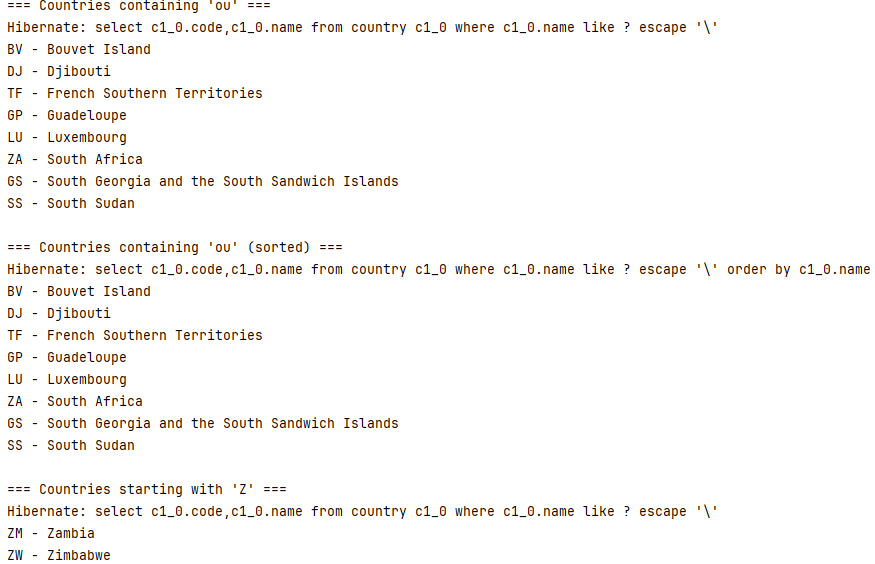
package com.cognizant.ormlearn.service;  
  
import com.cognizant.ormlearn.entity.Country;  
import com.cognizant.ormlearn.repository.CountryRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 public List<Country> searchByName(String substring) {  
 return countryRepository.findByNameContaining(substring);  
 }  
  
 public List<Country> searchByNameSorted(String substring) {  
 return countryRepository.findByNameContainingOrderByNameAsc(substring);  
 }  
  
 public List<Country> searchByStartingLetter(String prefix) {  
 return countryRepository.findByNameStartingWith(prefix);  
 }  
}

**OrmlearnApplication.java**

package com.cognizant.ormlearn;  
  
import com.cognizant.ormlearn.entity.Country;  
import com.cognizant.ormlearn.service.CountryService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
import java.util.List;  
  
@SpringBootApplication  
public class OrmlearnApplication implements CommandLineRunner {  
  
 @Autowired  
 private CountryService countryService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(OrmlearnApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 System.*out*.println("=== Countries containing 'ou' ===");  
 List<Country> list1 = countryService.searchByName("ou");  
 list1.forEach(c -> System.*out*.println(c.getCode() + " - " + c.getName()));  
  
 System.*out*.println("\n=== Countries containing 'ou' (sorted) ===");  
 List<Country> list2 = countryService.searchByNameSorted("ou");  
 list2.forEach(c -> System.*out*.println(c.getCode() + " - " + c.getName()));  
  
 System.*out*.println("\n=== Countries starting with 'Z' ===");  
 List<Country> list3 = countryService.searchByStartingLetter("Z");  
 list3.forEach(c -> System.*out*.println(c.getCode() + " - " + c.getName()));  
 }  
}

**application.properties**

spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
  
spring.jpa.show-sql=true  
spring.jpa.hibernate.ddl-auto=create  
spring.datasource.initialization-mode=always  
  
logging.level.com.example.country=DEBUG

**Output**

**Hands On 2: Write queries on stock table using Query Methods**   
  
**Stock.java**

package com.cognizant.ormlearn.model;  
  
import jakarta.persistence.\*;  
import java.math.BigDecimal;  
import java.util.Date;  
  
@Entity  
@Table(name = "stock")  
public class Stock {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int stId;  
  
 @Column(name = "st\_code")  
 private String stCode;  
  
 @Column(name = "st\_date")  
 @Temporal(TemporalType.*DATE*)  
 private Date stDate;  
  
 @Column(name = "st\_open")  
 private BigDecimal stOpen;  
  
 @Column(name = "st\_close")  
 private BigDecimal stClose;  
  
 @Column(name = "st\_volume")  
 private BigDecimal stVolume;  
  
 *// Getters and Setters* public int getStId() {  
 return stId;  
 }  
  
 public void setStId(int stId) {  
 this.stId = stId;  
 }  
  
 public String getStCode() {  
 return stCode;  
 }  
  
 public void setStCode(String stCode) {  
 this.stCode = stCode;  
 }  
  
 public Date getStDate() {  
 return stDate;  
 }  
  
 public void setStDate(Date stDate) {  
 this.stDate = stDate;  
 }  
  
 public BigDecimal getStOpen() {  
 return stOpen;  
 }  
  
 public void setStOpen(BigDecimal stOpen) {  
 this.stOpen = stOpen;  
 }  
  
 public BigDecimal getStClose() {  
 return stClose;  
 }  
  
 public void setStClose(BigDecimal stClose) {  
 this.stClose = stClose;  
 }  
  
 public BigDecimal getStVolume() {  
 return stVolume;  
 }  
  
 public void setStVolume(BigDecimal stVolume) {  
 this.stVolume = stVolume;  
 }  
  
 @Override  
 public String toString() {  
 return "Stock{" +  
 "stId=" + stId +  
 ", stCode='" + stCode + '\'' +  
 ", stDate=" + stDate +  
 ", stOpen=" + stOpen +  
 ", stClose=" + stClose +  
 ", stVolume=" + stVolume +  
 '}';  
 }  
}

**StockRepository.java**

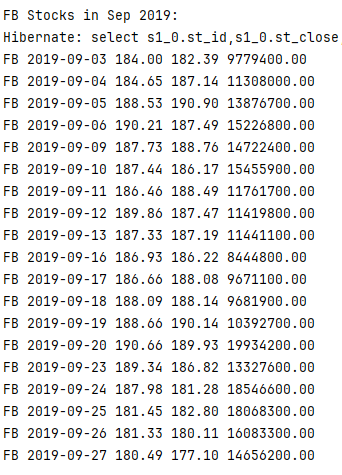
package com.cognizant.ormlearn.repository;  
  
import com.cognizant.ormlearn.model.Stock;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import java.math.BigDecimal;  
import java.time.LocalDate;  
import java.util.List;  
  
@Repository  
public interface StockRepository extends JpaRepository<Stock, Integer> {  
List<Stock> findByStCodeAndStDateBetween(String code, LocalDate start, LocalDate end);

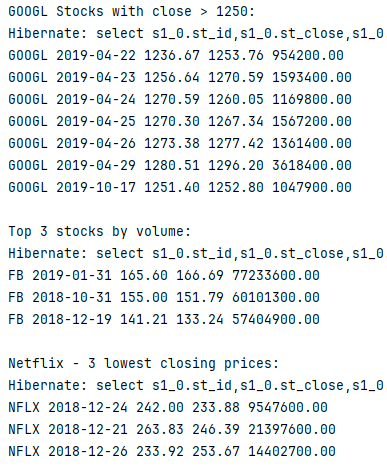
List<Stock> findByStCodeAndStCloseGreaterThan(String code, BigDecimal price);  
List<Stock> findTop3ByOrderByStVolumeDesc();  
List<Stock> findTop3ByStCodeOrderByStCloseAsc(String code);  
}

**OrmlearnApplication.java**

package com.cognizant.ormlearn;  
  
import com.cognizant.ormlearn.model.Stock;  
import com.cognizant.ormlearn.repository.StockRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
import java.math.BigDecimal;  
import java.time.LocalDate;  
import java.util.List;  
  
@SpringBootApplication  
public class OrmlearnApplication implements CommandLineRunner {  
  
 @Autowired  
 private StockRepository stockRepository;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(OrmlearnApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 getFacebookStocksInSep2019();  
 getGoogleStocksGreaterThan1250();  
 getTop3HighestVolume();  
 getNetflixLowest3Close();  
 }  
  
 private void getFacebookStocksInSep2019() {  
 System.*out*.println("\nFB Stocks in Sep 2019:");  
 List<Stock> stocks = stockRepository.findByStCodeAndStDateBetween(  
 "FB", LocalDate.*of*(2019, 9, 1), LocalDate.*of*(2019, 9, 30));  
 stocks.forEach(this::printStock);  
 }  
  
 private void getGoogleStocksGreaterThan1250() {  
 System.*out*.println("\nGOOGL Stocks with close > 1250:");  
 List<Stock> stocks = stockRepository.findByStCodeAndStCloseGreaterThan(  
 "GOOGL", new BigDecimal("1250"));  
 stocks.forEach(this::printStock);  
 }  
  
 private void getTop3HighestVolume() {  
 System.*out*.println("\nTop 3 stocks by volume:");  
 List<Stock> stocks = stockRepository.findTop3ByOrderByStVolumeDesc();  
 stocks.forEach(this::printStock);  
 }  
  
 private void getNetflixLowest3Close() {  
 System.*out*.println("\nNetflix - 3 lowest closing prices:");  
 List<Stock> stocks = stockRepository.findTop3ByStCodeOrderByStCloseAsc("NFLX");  
 stocks.forEach(this::printStock);  
 }  
  
 private void printStock(Stock s) {  
 System.*out*.println(s.getStCode() + " " + s.getStDate() + " " + s.getStOpen() + " " + s.getStClose() + " " + s.getStVolume());  
 }  
}

**Output**





**Hands On 3: Create payroll tables and bean mapping**   
  
**Department.java**

package com.cognizant.employee.model;  
  
import jakarta.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "department")  
public class Department {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 @Column(name = "dp\_name")  
 private String name;  
  
 @OneToMany(mappedBy = "department", fetch = FetchType.*LAZY*)  
 private List<Employee> employeeList;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 List<Employee> getEmployeeList() { return employeeList; }  
 public void setEmployeeList(List<Employee> employeeList) { this.employeeList = employeeList; }  
  
 @Override  
 public String toString() {  
 return "Department{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

**Employee.java**

package com.cognizant.employee.model;  
  
import jakarta.persistence.\*;  
import java.util.Date;  
import java.util.Set;  
  
@Entity  
@Table(name = "employee")  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 @Column(name = "em\_name")  
 private String name;  
  
 @Column(name = "em\_salary")  
 private double salary;  
  
 @Column(name = "em\_permanent")  
 private boolean permanent;  
  
 @Column(name = "em\_date\_of\_birth")  
 private Date dateOfBirth;  
  
 @ManyToOne  
 @JoinColumn(name = "em\_dp\_id")  
 private Department department;  
  
 @ManyToMany  
 @JoinTable(  
 name = "employee\_skill",  
 joinColumns = @JoinColumn(name = "es\_em\_id"),  
 inverseJoinColumns = @JoinColumn(name = "es\_sk\_id")  
 )  
 private Set<Skill> skillList;  
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 double getSalary() { return salary; }  
 public void setSalary(double salary) { this.salary = salary; }  
  
 public boolean isPermanent() { return permanent; }  
 public void setPermanent(boolean permanent) { this.permanent = permanent; }  
  
 public Date getDateOfBirth() { return dateOfBirth; }  
 public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }  
  
 public Department getDepartment() { return department; }  
 public void setDepartment(Department department) { this.department = department; }  
  
 public Set<Skill> getSkillList() { return skillList; }  
 public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }  
  
 @Override  
 public String toString() {  
 return "Employee{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", salary=" + salary +  
 ", permanent=" + permanent +  
 ", dateOfBirth=" + dateOfBirth +  
 ", department=" + department +  
 ", skills=" + skillList +  
 '}';  
 }  
}

**Skill.java**

package com.cognizant.employee.model;  
  
import jakarta.persistence.\*;  
import java.util.Set;  
  
@Entity  
@Table(name = "skill")  
public class Skill {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 @Column(name = "sk\_name")  
 private String name;  
  
 @ManyToMany(mappedBy = "skillList")  
 private Set<Employee> employeeList;  
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 Set<Employee> getEmployeeList() { return employeeList; }  
 public void setEmployeeList(Set<Employee> employeeList) { this.employeeList = employeeList; }  
  
 @Override  
 public String toString() {  
 return "Skill{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

**DepartmentRepository.java**

package com.cognizant.employee.repository;  
  
import com.cognizant.employee.model.Department;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface DepartmentRepository extends JpaRepository<Department, Integer> {  
}

**EmployeeRepository.java**

package com.cognizant.employee.repository;  
  
import com.cognizant.employee.model.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}

**SkillRepository.java**

package com.cognizant.employee.repository;  
  
import com.cognizant.employee.model.Skill;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface SkillRepository extends JpaRepository<Skill, Integer> {  
}

**EmployeeApplication.java**

package com.cognizant.employee;  
  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class EmployeeApplication implements CommandLineRunner {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(EmployeeApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) throws Exception {  
 System.*out*.println("Application started successfully!");  
 }  
}

**Output**



**Hands On 4: Implement many to one relationship between Employee and Department**   
  
**Department.java**

package com.cognizant.employee.model;  
  
import jakarta.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "department")  
public class Department {  
  
 @Id  
 @Column(name = "dp\_id")  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 @Column(name = "dp\_name")  
 private String name;  
  
 @OneToMany(mappedBy = "department", fetch = FetchType.*LAZY*)  
 private List<Employee> employeeList;  
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 List<Employee> getEmployeeList() { return employeeList; }  
 public void setEmployeeList(List<Employee> employeeList) { this.employeeList = employeeList; }  
  
 @Override  
 public String toString() {  
 return "Department{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

**Employee.java**

package com.cognizant.employee.model;  
  
import jakarta.persistence.\*;  
import java.util.Date;  
import java.util.Set;  
  
@Entity  
@Table(name = "employee")  
public class Employee {  
  
 @Id  
 @Column(name = "em\_id")  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 @Column(name = "em\_name")  
 private String name;  
  
 @Column(name = "em\_salary")  
 private double salary;  
  
 @Column(name = "em\_permanent")  
 private boolean permanent;  
  
 @Column(name = "em\_date\_of\_birth")  
 private Date dateOfBirth;  
  
 @ManyToOne  
 @JoinColumn(name = "em\_dp\_id")  
 private Department department;  
  
 @ManyToMany  
 @JoinTable(  
 name = "employee\_skill",  
 joinColumns = @JoinColumn(name = "es\_em\_id"),  
 inverseJoinColumns = @JoinColumn(name = "es\_sk\_id")  
 )  
 private Set<Skill> skillList;  
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 double getSalary() { return salary; }  
 public void setSalary(double salary) { this.salary = salary; }  
  
 public boolean isPermanent() { return permanent; }  
 public void setPermanent(boolean permanent) { this.permanent = permanent; }  
  
 public Date getDateOfBirth() { return dateOfBirth; }  
 public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }  
  
 public Department getDepartment() { return department; }  
 public void setDepartment(Department department) { this.department = department; }  
  
 public Set<Skill> getSkillList() { return skillList; }  
 public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }  
  
 @Override  
 public String toString() {  
 return "Employee{" +  
 "\nid=" + id +  
 ", \nname='" + name + '\'' +  
 ", \nsalary=" + salary +  
 ", \npermanent=" + permanent +  
 ", \ndateOfBirth=" + dateOfBirth +  
 ", \ndepartment=" + department +  
 ", \nskills=" + skillList +  
 '}';  
 }  
}

**Skill.java**

package com.cognizant.employee.model;  
  
import jakarta.persistence.\*;  
import java.util.Set;  
  
@Entity  
@Table(name = "skill")  
public class Skill {  
  
 @Id  
 @Column(name = "sk\_id")  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 @Column(name = "sk\_name")  
 private String name;  
  
 @ManyToMany(mappedBy = "skillList")  
 private Set<Employee> employeeList;  
  
 *// Getters and Setters* 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 Set<Employee> getEmployeeList() { return employeeList; }  
 public void setEmployeeList(Set<Employee> employeeList) { this.employeeList = employeeList; }  
  
 @Override  
 public String toString() {  
 return "\nSkill{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

**DepartmentRepository.java**

package com.cognizant.employee.repository;  
  
import com.cognizant.employee.model.Department;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface DepartmentRepository extends JpaRepository<Department, Integer> {  
}

**EmployeeRepository.java**

package com.cognizant.employee.repository;  
  
import com.cognizant.employee.model.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}

**SkillRepository.java**

package com.cognizant.employee.repository;  
  
import com.cognizant.employee.model.Skill;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface SkillRepository extends JpaRepository<Skill, Integer> {  
}

**EmployeeService.java**

package com.cognizant.employee.service;  
  
import com.cognizant.employee.model.Employee;  
import com.cognizant.employee.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import jakarta.transaction.Transactional;  
  
@Service  
public class EmployeeServiceImpl implements EmployeeService {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Override  
 public Employee get(int id) {  
 return employeeRepository.findById(id).orElse(null);  
 }  
  
 @Override  
 @Transactional  
 public void save(Employee employee) {  
 employeeRepository.save(employee);  
 }  
}

**DepartmentService.java**

package com.cognizant.employee.service;  
  
import com.cognizant.employee.model.Department;  
import com.cognizant.employee.repository.DepartmentRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
@Service  
public class DepartmentService {  
  
 @Autowired  
 private DepartmentRepository departmentRepository;  
  
 @Override  
 public Department get(int id) {  
 return departmentRepository.findById(id).orElse(null);  
 }  
}

**EmployeeApplication.java**package com.cognizant.employee;  
  
import com.cognizant.employee.model.Employee;  
import com.cognizant.employee.model.Department;  
import com.cognizant.employee.service.EmployeeService;  
import com.cognizant.employee.service.DepartmentService;  
import org.springframework.transaction.annotation.Transactional;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.boot.CommandLineRunner;  
  
import org.springframework.context.annotation.Bean;  
  
import java.util.Date;  
  
@SpringBootApplication  
public class EmployeeApplication {  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(EmployeeApplication.class);  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(EmployeeApplication.class, args);  
 }  
  
 @Bean  
 public CommandLineRunner run(EmployeeService employeeService, DepartmentService departmentService) {  
 return args -> {testGetEmployee(employeeService);  
 testAddEmployee(employeeService, departmentService);testUpdateEmployee(employeeService, departmentService);  
 };  
 }  
  
 @Transactional  
 private void testGetEmployee(EmployeeService employeeService) {  
 *LOGGER*.info("Start");  
  
 Employee employee = employeeService.get(1);  
  
 if (employee == null) {  
 *LOGGER*.warn("Employee with ID 1 not found.");  
 return;  
 }  
  
 *LOGGER*.info("Employee Details:");  
 *LOGGER*.info("ID : {}", employee.getId());  
 *LOGGER*.info("Name : {}", employee.getName());  
 *LOGGER*.info("Salary : {}", employee.getSalary());  
 *LOGGER*.info("Permanent : {}", employee.isPermanent());  
 *LOGGER*.info("Date of Birth: {}", employee.getDateOfBirth());  
  
 if (employee.getDepartment() != null) {  
 *LOGGER*.info("Department : {} (ID: {})",  
 employee.getDepartment().getName(),  
 employee.getDepartment().getId());  
 } else {  
 *LOGGER*.info("Department : Not Assigned");  
 }  
  
 if (employee.getSkillList() != null && !employee.getSkillList().isEmpty()) {  
 *LOGGER*.info("Skills:");  
 employee.getSkillList().forEach(skill ->  
 *LOGGER*.info(" - {} (ID: {})", skill.getName(), skill.getId())  
 );  
 } else {  
 *LOGGER*.info("Skills : None");  
 }  
  
 *LOGGER*.info("End");  
 }  
  
 @Transactional  
 private void testAddEmployee(EmployeeService employeeService, DepartmentService departmentService) {  
 *LOGGER*.info("Start");  
  
 Employee employee = new Employee();  
 employee.setName("John Doe");  
 employee.setSalary(50000.00);  
 employee.setPermanent(true);  
 employee.setDateOfBirth(new Date());  
  
 Department department = departmentService.get(1);  
 employee.setDepartment(department);  
  
 employeeService.save(employee);  
  
 *LOGGER*.info("New Employee Added:");  
 *LOGGER*.info("ID : {}", employee.getId());  
 *LOGGER*.info("Name : {}", employee.getName());  
 *LOGGER*.info("Salary : {}", employee.getSalary());  
 *LOGGER*.info("Permanent : {}", employee.isPermanent());  
 *LOGGER*.info("Date of Birth: {}", employee.getDateOfBirth());  
  
 if (employee.getDepartment() != null) {  
 *LOGGER*.info("Department : {} (ID: {})",  
 employee.getDepartment().getName(),  
 employee.getDepartment().getId());  
 } else {  
 *LOGGER*.info("Department : Not Assigned");  
 }  
  
 if (employee.getSkillList() != null && !employee.getSkillList().isEmpty()) {  
 *LOGGER*.info("Skills:");  
 employee.getSkillList().forEach(skill ->  
 *LOGGER*.info(" - {} (ID: {})", skill.getName(), skill.getId()));  
 } else {  
 *LOGGER*.info("Skills : None");  
 }  
  
 *LOGGER*.info("End");  
 }  
  
 @Transactional  
 private void testUpdateEmployee(EmployeeService employeeService, DepartmentService departmentService) {  
 *LOGGER*.info("Start");  
  
 Employee employee = employeeService.get(1);  
 Department newDepartment = departmentService.get(2);  
 employee.setDepartment(newDepartment);  
  
 employeeService.save(employee);  
  
 *LOGGER*.info("Employee Updated:");  
 *LOGGER*.info("ID : {}", employee.getId());  
 *LOGGER*.info("Name : {}", employee.getName());  
 *LOGGER*.info("Salary : {}", employee.getSalary());  
 *LOGGER*.info("Permanent : {}", employee.isPermanent());  
 *LOGGER*.info("Date of Birth: {}", employee.getDateOfBirth());  
  
 if (employee.getDepartment() != null) {  
 *LOGGER*.info("Department : {} (ID: {})",  
 employee.getDepartment().getName(),  
 employee.getDepartment().getId());  
 } else {  
 *LOGGER*.info("Department : Not Assigned");  
 }  
  
 if (employee.getSkillList() != null && !employee.getSkillList().isEmpty()) {  
 *LOGGER*.info("Skills:");  
 employee.getSkillList().forEach(skill ->  
 *LOGGER*.info(" - {} (ID: {})", skill.getName(), skill.getId()));  
 } else {  
 *LOGGER*.info("Skills : None");  
 }  
  
 *LOGGER*.info("End");  
 }  
  
}

**Output**

