3조

Java & SQL 프로젝트

직원정보 검색 프로그램 구현

JDBC를 이용한 JAVA 애플리케이션과 MySQL 간 연결 및 조작 실습

안병주, 이태민, 차민정

목차

CONTENTS

Java & SQL
프로젝트

1 프로젝트 개요

Java와 SQL을 기반으로 한 직원 정보 관리 시스템

프로젝트 목표

직원정보를 다양한 기준으로 검색 가능한 시스템 개발

테스트 및 안정성

정상적인 실행 여부 확인
예외 처리 및 갑작스러운 종료 방지 설계

기술 구현 방식

메서드 및 클래스 구조화
인터페이스 정의로
유지보수성과 확장성 강화

프로그래밍 언어

JAVA

데이터베이스

MySQL

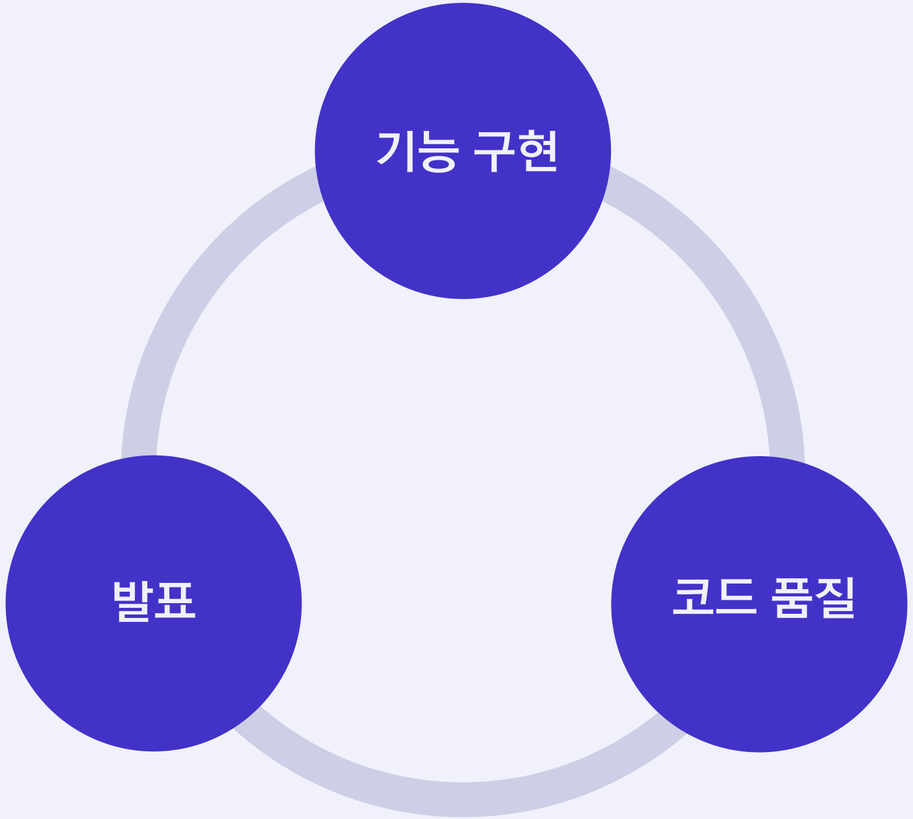
개발 환경

Eclipse &
MySQL workbench

1 프로젝트 평가방안

JDBC를 이용한 자바애플리케이션과
MySQL 간 연결 및 조작

1. 메서드화, 클래스화가 잘 되어 있는가?
2. 중복되는 코드를 제거했는가?
3. 변수명, 메서드명, 클래스명이 적절한가?
4. 애플리케이션이 갑자기 종료되지 않는가?



2 팀원 소개

안병주

1. 직원이름으로 검색 기능
2. 입사년도로 검색 기능

이태민

1. 직무 검색 기능
2. 도시 이름 검색 기능
3. 통계 자료 출력 기능(전체 , 세부 통계)

차민정

1. 부서번호로 검색 기능
2. 부서장 성으로 부서원 검색 기능
3. 나라 이름으로 검색 기능

3 작성 프로그램 구조

DAO(Data Access Object) 패턴을 사용한 직원 관리 시스템

메인 애플리케이션

Test.java
(Main Application)

사용자 입력 처리
DAO 호출
결과출력
검색/통계 기능

사용

EmpDao 인터페이스가 데이터 접근 계약을 정의

EmpDao.java
(Interface)

getByName()
getByHireYear()
getByCity()
getStatistics()...

구현

EmpDaoImpl.java
(Implementation)

MySQL 연결
PreparedStatement
SQL 쿼리 실행
ResultSet 처리
Connection 관리
8가지 검색 메서드

SQL 연결

MySQL DataBase

employees 테이블
departments 테이블
locations 테이블
countries 테이블

생성/
반환

Employee 클래스는
직원 정보를 담는 데이터 모델 역할

Employee.java
(Data Model)

필드
employeeId
fullName
email
phoneNumber
hireDateString
jobID
salary...

EmpDaoImpl 클래스가
MySQL DB와의 실제 연결 및 Read 작업을 구현

4 작성 프로그램 세부설명

Employee 클래스

11개 필드
Getter/Setter/toString

필드를 private로 → Getter/Setter 메서드로 접근

toString() : 객체 정보를 문자열로 변환하여 출력용으로 사용

소스코드

private 필드

```
// 필드들
private int employeeId;
private String fullName;
private String email;
private String phoneNumber;
private String hireDateString;
private String jobId;
private double salary;
private int managerId;
private int departmentId;
private String cityName;
private String countryName;
```

get{필드명}();

```
// Getter 메서드들
public int getEmployeeId() {
    return employeeId;
}

public String getFullName() {
    return fullName;
}

public String getEmail() {
    return email;
}

public String getPhoneNumber() {
    return phoneNumber;
}

public String getHireDateString() {
    return hireDateString;
}

public String getJobId() {
    return jobId;
}

public String getCityName() {
    return cityName;
}
```

set{필드명}();

```
// Setter 메서드들
public void setEmployeeId(int employeeId) {
    this.employeeId = employeeId;
}

public void setFullName(String fullName) {
    this.fullName = fullName;
}

public void setEmail(String email) {
    this.email = email;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}

public void setHireDateString(String hireDateString) {
    this.hireDateString = hireDateString;
}

public void setJobId(String jobId) {
    this.jobId = jobId;
}

public void setCityName(String cityName) {
    this.cityName = cityName;
}
```

toString(); → 출력양식 통일

```
public String toString() {
    return "Employee{" +
            "employeeId=" + employeeId +
            ", fullName='" + fullName + '\'' +
            ", email='" + email + '\'' +
            ", phoneNumber='" + phoneNumber + '\'' +
            ", hireDateString='" + hireDateString + '\'' +
            ", jobId='" + jobId + '\'' +
            ", salary='" + salary + '\'' +
            ", manager='" + managerId + '\'' +
            ", departmentId='" + departmentId + '\'' +
            ", cityName='" + cityName + '\'' +
            ", countryName='" + countryName + '\'' +
            "}";
}
```

EmpDao 인터페이스

8개 검색 메서드
계약 정의 인터페이스

- getByName(String name): 직원 이름으로 LIKE 검색하여 해당 직원 목록 반환
- getByHireYear(int year): 입사년도로 검색하여 해당 연도 입사자 목록 반환
- getByDepartmentId(int departmentId): 부서번호로 검색하여 해당 부서 직원 목록 반환
- getByJobTitle(String jobTitle): 직무ID로 검색하여 해당 직무 담당자 목록 반환
- getByCity(String city): 도시명으로 검색하여 해당 도시 근무자 목록 반환 (3테이블 JOIN)
- getByManagerLastName(String lastName): 부서장 성으로 검색하여 해당 부서장 산하 직원 목록 반환
- getByCountryName(String country): 국가명으로 검색하여 해당 국가 근무자 목록 반환 (4테이블 JOIN)
- getStatistics(): 전체 직원 통계와 부서별 상세 통계를 Map으로 반환

4 작성 프로그램 세부설명

Employee Impl 클래스

실제 DB 연동하는 핵심 구현체
(1개 생성자, 1개 연결메서드,
8개 검색메서드)

- 생성자: Properties 파일에서 DB 연결정보를 읽어와 필드에 저장
- getConnection(): DriverManager를 사용해 DB 연결객체 생성하여 반환
- 각 검색 메서드들: PreparedStatement로 SQL 실행 후 ResultSet을 Employee 객체 리스트로 변환

DB 정보를 별도의 파일로 관리

보안, 유지보수, 배포 편의성

dp-info.properties

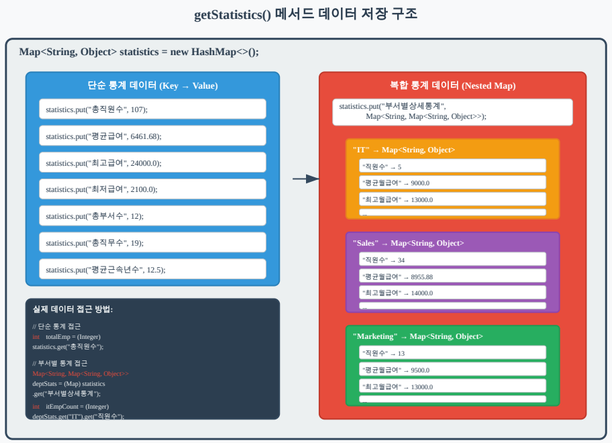
- driverClassName=com.mysql.cj.jdbc.Driver
- url=jdbc:mysql://localhost:3306/newhr
- userName=root
- password=rootroot

getStatistics() 데이터 구조 핵심 요약:

🔧 2단계 중첩 Map 구조:

1단계: Map<String, Object> – 전체 통계 + 부서별통계 Map 저장

2단계: Map<String, Map<String, Object>> – 부서명 → 각 부서 상세정보



5 프로그램 시연

Java 직원정보 검색시스템 테스트 시나리오

Scanner로 사용자 입력을 받아 8가지 검색기능을 순차적으로 실행하고 결과 출력

순서	메서드명	메서드 기능 설명	입력값
1	getByName	직원이름을 입력하여 직원 정보를 검색할 수 있는 기능	King
2	getByHireYear	입사년도를 입력하여 직원 정보를 검색할 수 있는 기능	2014
3	getByManagerLastName	부서장 성으로 부서원을 검색할 수 있는 기능	yang
4	getByCity	도시이름으로 해당 도시에서 근무중인 직원리스트를 검색하는 기능	London
5	getEmpListByCountryName	나라이름으로 해당 나라에 근무중인 직원 검색 기능	canada
6	getByJobTitle	직무명을 검색하여 해당 직무에 종사하고 있는 직원 리스트를 검색 기능	IT_PROG
7	getByDepartmentId	부서번호로 해당 부서에 근무중인 직원 정보를 검색할 수 있는 기능	20
8	getStatistics	전사 통계 및 부서별 통계자료 출력	엔터 입력

6 프로젝트 회고 – 안병주

노력했던 점	가장 어려웠던 부분은 **인터페이스**와 **예외 처리** 개념이었습니다. **인터페이스**의 개념은 "**무엇을 할지**" **인터페이스**에서 미리 정해두고, "**어떻게 할지**" 각 클래스에서 구현한다고 생각하니 점차 이해되었고, **구현 클래스**에서는 예외를 직접 처리하지 않고 **호출한 쪽에 넘기는 구조**라는 점을 이해하려고 노력했습니다.
인상적인 점	EmpDao **인터페이스**를 통해서 미리 **메서드를 정의**하고 팀원들이 각자 맡은 기능을 따로 개발하거나 **동시 작업**이 가능했는데, **프로젝트를 기획**할 때 이렇게도 접근 할 수 있다는 점이 인상적이었습니다.
새로 배우게 된 점	코딩할 때 **기획**하고 **설계**하는 연습을 많이 해봐야겠다고 느꼈고, 강의를 수강하면서 어려웠던 개념들을 **프로젝트**에 직접 적용하고 부족한 부분은 채워 갈 수 있었습니다. 다음 **프로젝트를** 진행할 때는 전체적으로 어떻게 코딩할지 **기획**하고 **설계**하는 과정을 통해 **기능 중심**이 아닌 **구조 중심**으로 생각해 봐야 할 것 같습니다.

6 프로젝트 회고 – 이태민

노력했던 점	평가 기준에 맞춰 **코드 반복**을 **최소화**하기 위해 노력했습니다. 수업시간에 배웠던 내용을 최대한 사용하기 위해 노력했습니다. 프로젝트 관리를 **체계화**하기 위해 **노션**을 이용해 협업했습니다.
인상적인 점	Python 프로젝트만 하다 Java 프로젝트를 하니 **Python**이 얼마나 **편리한 언어**였는지를 깨닫는 시간이었습니다.
새로 배우게 된 점	다른 사람에게 머릿속으로 생각한 구조를 말로 풀어서 설명하는 것이 어렵다는 걸 느꼈습니다. 설명을 하는 과정에서 **개념**을 **정확**하게 **숙지**를 하지 못하고 있다는 걸 알았습니다. **드라이버**는 **클래스 레벨**의 **리소스**이므로 **static 블록**에서 관리하는 것이 **최적**이라는 걸 배웠습니다.

6 프로젝트 회고 – 차민정

노력했던 점	SQL 쿼리 작성 중 **JOIN 순서**와 **기준**을 이해하고자 노력했습니다. 또한 SQL에서 **서브 쿼리**를 이해하여 기능 구현에 적용하고자 노력했습니다. 만들어진 **인터페이스**에 맞춰 기능을 구현하고자 노력했습니다.
인상적인 점	EmpDao 인터페이스와 이를 구현한 EmpDaoImpl 클래스를 만들고, DB 연결 과정에서 **중첩된 try-catch** 사용 방식이 인상적이었습니다.
새로 배우게 된 점	JOIN 순서와 기준을 알기 위해 검색하던 중 **PK, FK**를 알게되었습니다. 이를 바탕으로 직접 **테이블 간의 관계**를 정리했습니다. 또한 이번 프로젝트를 통해 **static 블록**은 클래스 로딩 시 한 번만 실행된다는 점도 새롭게 배웠습니다.

7 Q&A

3조

Java & SQL 프로젝트

JDBC를 이용한 JAVA 애플리케이션과 MySQL 간 연결 및 조작 실습

THANK YOU

Q&A