

2023/11/03 자유주제 과제

# 사원관리프로그램

# CONTENTS



기능 요구사항

클래스 명세서

클래스 다이어그램

소스 코드 & 구현

# 기능 요구사항

## 주요 기능

1. 사원 추가
2. 사원 정보 수정
3. 사원 전체 보기
4. 부서 이동
5. 사원 삭제
6. 부서별 사원 정보 txt파일생성
7. 프로그램 종료

1

2

3

4

프로그램 실행 시  
주요 기능 메뉴가 자동  
출력된다

각 메뉴에 진입시  
요구사항에 맞는 작업을  
진행한다

현재 정보를 가지고  
txt파일을 저장하되  
부서별 처리하도록 한다

txt파일 생성 시 동일한 템  
플릿 기준으로 기재되도록  
한다

## 기능 요구사항

No	내용	담당자
1	프로그램이 실행되면 중요 메뉴로 이동한다	김가율
2	사원 등록 시 사원 번호는 중복 불가하다	김가율
3	사원 정보 수정 시 사원 번호로 조회한다	김가율
4	사원 정보 수정 기능에서 수정 가능한 항목은 직위와 부서만 가능하다	김가율
5	정보 수정 시 변경 전 후에 내용을 출력해준다	김가율
6	사원 전체 보기 진행 시 등록된 순서는 상관없이 출력한다	김가율
7	부서별 사원 정보 조회 시 각자 등록되어 있는 부서명이 일치한 정보만 출력된다	김가율
8	사원 삭제 시 정말 삭제할 것인지 1번 더 확인한다	김가율
9	사원 전체 조회 시 사원번호를 포함한 모든 내용을 출력한다	김가율
10	부서별 txt파일로 추출 진행 시 기존 내용을 유지 및 삭제 여부를 확인한다	김가율
11	기존 내용을 유지 하는 경우 기존 입력된 내용은 삭제 되지 않도록 한다	김가율
12	기존 내용을 유지 하지 않을 경우 현재 기준으로 내용을 추출한다	김가율
13	요청한 부서의 txt파일 추출 시 일정한 템플릿으로 저장하도록 한다	김가율
14	중요 메뉴에서 없는 메뉴 번호를 눌러도 프로그램은 종료되지 않는다	김가율

# 클래스 명세서

employees_program						설명	
파키지	접근제어자	static	자료형	이름	매개변수	리턴값	설명
클래스	public		Main				프로그램 시작
<b>employees_program</b>							
파키지	<b>employees_program.activeProgram</b>						
분류	접근제어자	static	자료형	이름	매개변수	리턴값	설명
클래스	public		AcitveEmployoeProgram				메인 메뉴 클래스
멤버변수	private	static	HashMap<Integer, Employees>	employessList			사원번호와 객체저장
멤버변수	private	static	String[]	deptName			회사내 부서리스트
멤버변수	private	static	String[]	jobTitle			회사내 직위리스트
메서드	private		void	info()			메뉴 출력 메서드
메서드	private		void	viewEmployees()			기능중 전체보기 메서드
getter	public	static	HashMap<Integer, Employees>	getEmployessList()	return	return	다른 클래스에서도 사용할 수 있도록 생성
getter	public	static	String[]	getDeptName()	return	return	다른 클래스에서도 사용할 수 있도록 생성
getter	public	static	String[]	getJobTitle()	return	return	다른 클래스에서도 사용할 수 있도록 생성
setter	public	static	void	setEmployessList			HashMap에 데이터 등록시 필요한 setter
<b>employees_program</b>							
파키지	<b>employees_program.activeProgram</b>						
분류	접근제어자	static	자료형	이름	매개변수	리턴값	설명
클래스	public		Employees				사원의 추상화 클래스
멤버변수	private		int	emNum			사원번호
멤버변수	private		String	emName			사원이름
멤버변수	private		int	emAge			사원나이
멤버변수	private		String	emDept			근무 부서
멤버변수	private		String	emJobTitle			직위
getter	public		int	getEmNum	return	return	다른 클래스에서도 사용할 수 있도록 생성
setter	public		void	setEmNum			다른 클래스에서도 사용할 수 있도록 생성
getter	public		String	getEmName	return	return	다른 클래스에서도 사용할 수 있도록 생성
setter	public		void	setEmName			다른 클래스에서도 사용할 수 있도록 생성
getter	public		int	getEmAge	return	return	다른 클래스에서도 사용할 수 있도록 생성
setter	public		void	setEmAge			다른 클래스에서도 사용할 수 있도록 생성
getter	public		String	getEmDept	return	return	다른 클래스에서도 사용할 수 있도록 생성
setter	public		void	setEmDept			다른 클래스에서도 사용할 수 있도록 생성
getter	public		String	getEmJobTitle	return	return	다른 클래스에서도 사용할 수 있도록 생성
setter	public		void	setEmJobTitle			다른 클래스에서도 사용할 수 있도록 생성

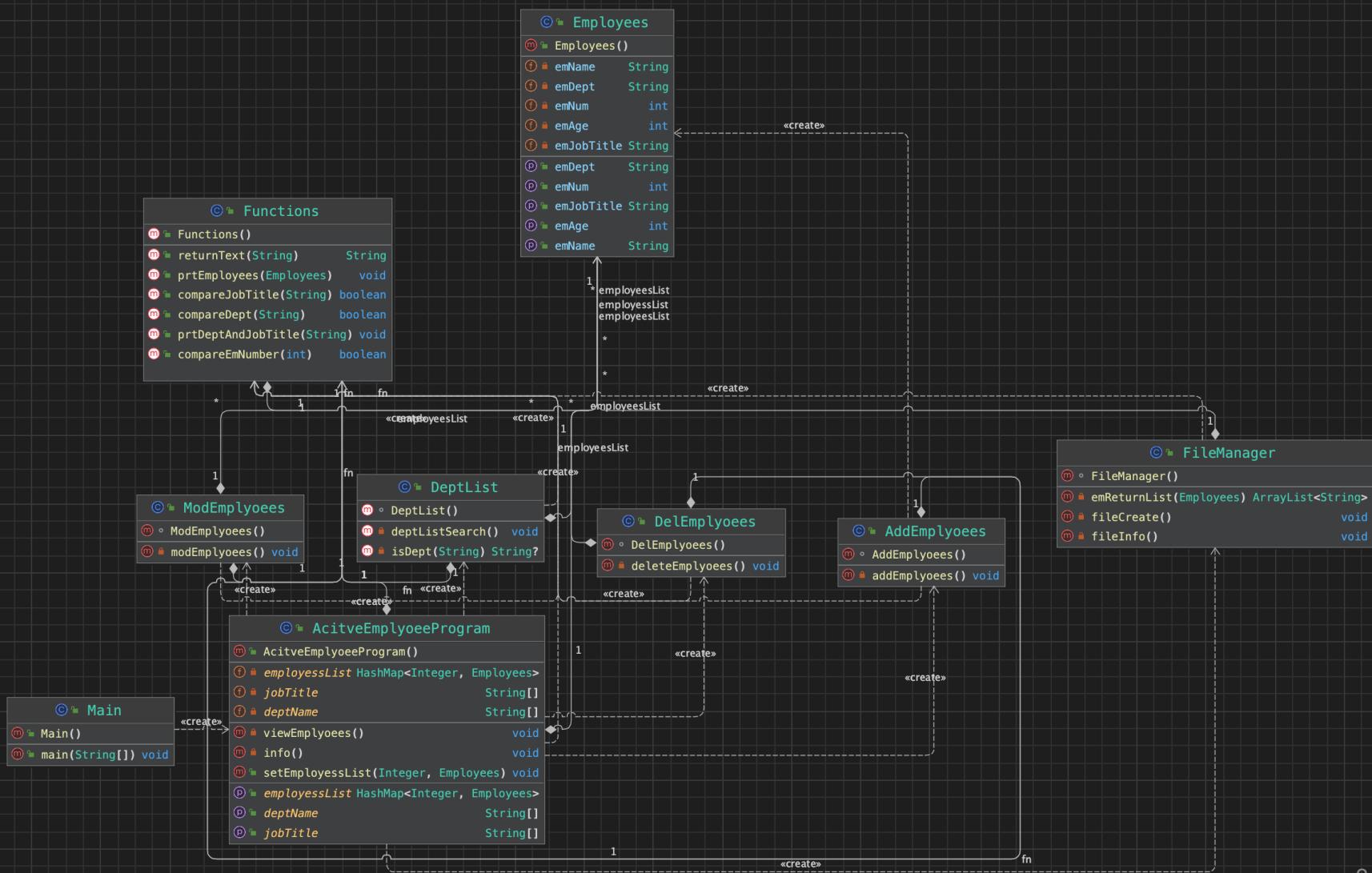
# 클래스 명세서

패키지		employees_program					
패키지		employees_program.activeProgram					
분류	접근제어자	static	자료형	이름	매개변수	리턴값	설명
클래스	public		AddEmployees	fn			사원 등록 클래스
멤버변수	private		Function				Function 클래스의 기능 사용하기 위함
생성자	default			AddEmployees			생성자 내부에 사원 등록 메서드 실행
메서드	private		void	addEmployees			사원 등록 메서드
패키지		employees_program					
패키지		employees_program.activeProgram					
분류	접근제어자	static	자료형	이름	매개변수	리턴값	설명
클래스	public		DelEmployees	fn			사원 삭제 클래스
멤버변수	private		Function				Function 클래스의 기능 사용하기 위함
멤버변수	private		HashMap<Integer, Employees>	employeesList			ArrayList의 주소를 getter호출받아 사용
생성자	default			DelEmployees			생성자 내부에 삭제메서드 실행
메서드	private		void	deleteEmployees			사원 삭제 메서드
패키지		employees_program					
패키지		employees_program.activeProgram					
분류	접근제어자	static	자료형	이름	매개변수	리턴값	설명
클래스	public		DeptList				부서별 사원 정보 검색 클래스
멤버변수	private		Function	fn			Function 클래스의 기능 사용하기 위함
멤버변수	private		HashMap<Integer, Employees>	employeesList			ArrayList의 주소를 getter호출받아 사용
멤버변수	private		Employees	em			null처리
멤버변수	private		String[]	deptName			null처리
생성자	default			DeptList			생성자 내부에 부서별 검색 메서드 실행
메서드	private		void	deptListSearch			부서별 검색을 처리하는 메서드
메서드	private		String	inDept	String	return	부서별 검색 시 value값을 이용하여 데이터검색
패키지		employees_program					
패키지		employees_program.activeProgram					
분류	접근제어자	static	자료형	이름	매개변수	리턴값	설명
클래스	public		ModEmployees	fn			사원 수정 클래스
멤버변수	private		Function				Function 클래스의 기능 사용하기 위함
멤버변수	private		HashMap<Integer, Employees>	employeesList			ArrayList의 주소를 getter호출받아 사용
생성자	default			ModEmployees			생성자 내부에 사원 수정 메서드 실행
메서드	private		void	modEmployees			사원 수정 메서드

# 클래스 명세서

패키지		employees_program						
매개변수		employees_program.activeProgram						
분류	접근제어자	static	자료형	이름	매개변수	리턴값	설명	
클래스	public			FileManager				
멤버변수	private			Employees	em			
멤버변수	private			ArrayList<String>	outEmInfo			
멤버변수	private			HashMap<Integer, Employees>	employeesList			
생성자	default			FileManager				생성자 내부에 fileCreate메서드 호출
메서드	private			void	fileCreate			
메서드	private			ArrayList<String>	emReturnList	Employees	return	write진행시 일정 템플릿으로 될 수 있도록함
메서드	private			void	fileInfo			
패키지		employees_program						
매개변수		employees_program.activeProgram						
분류	접근제어자	static	자료형	이름	매개변수	리턴값	설명	
클래스	public			Functions				
멤버변수	private			Employees	em			
멤버변수	private			HashMap<Integer, Employees>	employeesList			
멤버변수	private			String	jobTitle			
멤버변수	private			String[]	deptName			
메서드	public			void	prtDeptAndJobTitle	String		
메서드	public			boolean	compareDept	String	return	입력 부서가 목록에 있는지 여부 판단 후 결과 리턴
메서드	public			boolean	compareJobTitle	String	return	입력 직위가 목록에 있는지 여부 판단 후 결과 리턴
메서드	public			boolean	compareEmNumber	int	return	사원 번호가 동일한 사람이 있는지 확인 후 결과 리턴
메서드	public			void	prtEmployees	Employees		
메서드	public			String	returnText	String	return	사원 정보 출력 메서드
메서드	public							

# 주요 소스코드



# 소스 코드 & 구현 – 준비한 입력 내용

사원번호	1234	사원번호	1	사원번호	111	사원번호	10
이름	김또또	이름	김대표	이름	정회계	이름	조임원
나이	40	나이	59	나이	42	나이	50
부서	개발팀	부서	경영팀	부서	회계팀	부서	인사팀
직위	과장	직위	임원	직위	부장	직위	임원
사원번호	2331	사원번호	2	사원번호	1228	사원번호	99
이름	김개발	이름	박사모	이름	박과장	이름	홍부장
나이	31	나이	55	나이	38	나이	46
부서	개발팀	부서	경영팀	부서	회계팀	부서	인사팀
직위	사원	직위	임원	직위	과장	직위	부장
사원번호	1222	사원번호	2334	사원번호	2111	사원번호	808
이름	김대리	이름	김주임	이름	홍주임	이름	공과장
나이	35	나이	34	나이	31	나이	44
부서	개발팀	부서	경영팀	부서	회계팀	부서	인사팀
직위	대리	직위	주임	직위	주임	직위	과장
사원번호	1212	사원번호	2999	사원번호	2555	사원번호	3399
이름	김부장	이름	김경영	이름	김회계	이름	김인사
나이	45	나이	30	나이	26	나이	29
부서	개발팀	부서	경영팀	부서	회계팀	부서	인사팀
직위	부장	직위	사원	직위	사원	직위	사원

# 소스 코드 & 구현 – 사원 등록

```
1 usage new *
private void addEmployees() { Complexity is 5 Everything is cool!
Employees em = new Employees();
System.out.println("사원 등록 메뉴 입니다");
try {
    int emNum = Integer.parseInt(fn.returnText("사원 번호"));
    // 동일 사번이 있는지 판단 후 리턴
    if (!(fn.compareEmNumber(emNum))) {
        em.setEmNum(emNum);
        em.setEmName(fn.returnText("이름"));
        em.setEmAge(Integer.parseInt(fn.returnText("사원 나이")));
        fn.prtDeptAndJobTitle(text: "부서");
        em.setEmDept(fn.returnText("근무 부서"));
        // 입력된 부서가 내부 부서 목록과 일치 판단 후 리턴
        if (fn.compareDept(em.getEmDept())) {
            fn.prtDeptAndJobTitle(text: "직위");
            em.setEmJobTitle(fn.returnText("직위 등록"));
            // 입력된 직위가 내부 직위 목록과 일치 판단 후 리턴
            if (fn.compareJobTitle(em.getEmJobTitle()))
                setEmployessList(em.getEmNum(), em);
        }
    }
} catch (Exception e) {
    e.printStackTrace();
    System.out.println("Unable to process");
}
}
```

-----  
사원 관리 프로그램입니다

1. 사원 추가
2. 사원 수정
3. 사원 전체보기
4. 부서별 사원 정보 조회
5. 사원 삭제
6. 부서 별 사원 정보 추출
7. 프로그램 종료

메뉴 번호 입력 >> 1

사원 등록 메뉴 입니다

사원 번호 입력 >> 3399

동일한 사원 번호가 있습니다

이름 입력 >> 김인사

사원 나이 입력 >> 29

선택 가능한 부서 목록 : 인사팀 회계팀 개발팀 영업팀 인프라팀 경영팀

근무 부서 입력 >> 인사팀

선택 가능한 직위 목록 : 임원 부장 차장 과장 대리 주임 사원

직위 등록 입력 >> 사원

-----  
사원 등록시 중복된 사원 번호는 입력되지 않도록 하였습니다  
부서와 직위 목록 외에 다른 글이 적히면 등록되지 않습니다

# 소스 코드 & 구현 – 사원 삭제

```
Functions fn = new Functions();
1 usage
private HashMap<Integer,Employees> employeesList = getEmployeesList();

1 usage new *
DeleteEmployees() { deleteEmployees(); }

1 usage new *
private void deleteEmployees() { Complexity is 4 Everything is cool!
    System.out.println("사원 삭제 메뉴 입니다");
    try {
        int delEmNumber = Integer.parseInt(fn.returnText("삭제할 사원 번호"));
        if (fn.compareEmNumber(delEmNumber)) {
            String result = fn.returnText("정말 삭제하겠습니까 ? Y / N ").toLowerCase();
            if (result.equals("y")) {
                employeesList.remove(delEmNumber);
                System.out.println("사번 : " + delEmNumber + " 삭제 완료");
            } else System.out.println("초기 메뉴로 돌아갑니다");
        }
    } catch (Exception e) {
        e.printStackTrace();
        System.out.println("Unable to process");
    }
}
```

사원 관리 프로그램입니다

1. 사원 추가
2. 사원 수정
3. 사원 전체보기
4. 부서별 사원 정보 조회
5. 사원 삭제
6. 부서 별 사원 정보 추출
7. 프로그램 종료

메뉴 번호 입력 >> 5

사원 삭제 메뉴 입니다

삭제할 사원 번호 입력 >> 2

정말 삭제하겠습니까 ? Y / N 입력 >> y

사번 : 2 삭제 완료

사원 관리 프로그램입니다

사원 삭제 진행 시 한번 더 생각 할 수 있도록  
재차 확인하는 로직으로 처리하였습니다

# 소스 코드 & 구현 – 부서별조회

```
public class DeptList { Complexity is 5 Everything is cool!
    1 usage
    private HashMap<Integer, Employees> employeesList = getEmployeesList();
    3 usages
    private Employees em = null;
    1 usage
    private String[] deptName = null;
    4 usages
    private Functions fn = new Functions();

    1 usage new *
    DeptList() {deptListSearch();}

    1 usage new *
    private void deptListSearch() {
        deptName = ActiveEmployeeProgram.getDeptName();
        System.out.println("부서별 근무 사원 정보 메뉴입니다");
        // 블록을 출력 메서드
        fn.prtDeptAndJobTitle( text: "부서");
        String deptNames = fn.returnText("조회 부서 명");
        // 일치한 부서의 사원 정보를 출력해주는 메서드
        if (fn.compareDept(deptNames)) isDept(deptNames);
    }

    1 usage new *
    private| String isDept(String deptNames) { Complexity is 4 Everything is cool!
        for (Integer key : employeesList.keySet()) {
            this.em = getEmployeesList().get(key);
            if (em.getEmDept().equals(deptNames)) {
                fn.prtEmployees(em);
            }
        }
        return null;
    }
```

```
부서별 근무 사원 정보 메뉴입니다
선택 가능한 부서 목록 : 인사팀 회계팀 개발팀 영업팀 인프라팀 경영팀
조회 부서 명 입력 >> 인사팀
-----
직 위 : 부장
사원번호 : 100
근무부서 : 인사팀
이 름 : 홍부장
나 이 : 46
-----
직 위 : 사원
사원번호 : 3399
근무부서 : 인사팀
이 름 : 김인사
나 이 : 29
-----
직 위 : 과장
사원번호 : 808
근무부서 : 인사팀
이 름 : 공과장
나 이 : 46
-----
직 위 : 임원
사원번호 : 10
근무부서 : 인사팀
이 름 : 조임원
나 이 : 50
```

# 소스 코드 & 구현 – 사원 수정

```
1 usage: new <...>
private void modEmployees(){ Complexity is 7 It's time to do something...
    Employees em = null;
    System.out.println("사원 정보 수정입니다");
    try {
        int emNum = Integer.parseInt(fn.returnText("수정 할 사원 번호"));
        if (fn.compareEmNumber(emNum)) {
            em = employeesList.get(emNum);
            System.out.println("수정 가능 항목 : 직위 or 부서");
            String editStr = fn.returnText("수정 항목");
            if (editStr.equals("부서")) {
                fn.prtDeptAndJobTitle(text: "부서");
                String afterStr = fn.returnText("부서");
                if (fn.compareDept(afterStr)) {
                    System.out.println("변경 전 [ " + em.getEmDept() + " ] ");
                    em.setEmDept(afterStr);
                    System.out.println("변경 후 [ " + em.getEmDept() + " ] ");
                    System.out.println(" [ " + em.getEmName() + " ] 부서 변경 완료 ");
                }
            } else if (editStr.equals("직위")) {
                fn.prtDeptAndJobTitle(text: "직위");
                String afterStr = fn.returnText("직위");
                if (fn.compareJobTitle(afterStr)) {
                    System.out.println("변경 전 [ " + em.getEmJobTitle() + " ] ");
                    em.setEmJobTitle(afterStr);
                    System.out.println("변경 후 [ " + em.getEmJobTitle() + " ] ");
                    System.out.println(" [ " + em.getEmName() + " ] 님 부서 변경 완료 ");
                }
            } else {
                System.out.println("Unable to process");
            }
        } catch (Exception e){
            e.printStackTrace();
            System.out.println("Unable to process");
        }
    }
```

직 위 : 주임

사원번호 : 2111

근무부서 : 회계팀

이 름 : 홍주임

나 이 : 31

사원 정보 수정입니다

수정 할 사원 번호 입력 >> 2111

수정 가능 항목 : 직위 or 부서

수정 항목 입력 >> 부서

선택 가능한 부서 목록 : 인사팀 회계팀 개발팀 영업팀 인프라팀 경영팀

부서 입력 >> 영업팀

변경 전 [ 회계팀 ]

변경 후 [ 영업팀 ]

[ 홍주임 ] 부서 변경 완료

직위 변경이나 부서이동을 위해 변경하는 수정구현 입니다

# 소스 코드 & 구현 – txt write

```
fn.prtDeptAndJobTitle( text: "부서");
// 조회할 부서명 입력
String deptTitle = fn.returnText("부서");
if (fn.compareDept(deptTitle)) {
    File file = null;
    FileWriter fw = null;
    BufferedWriter bw = null;
    try {
        // File에 입력될 내용을 List형식으로 등록함으로 List 생성
        outEmInfo = new ArrayList<>();
        file = new File( pathname: deptTitle + ".txt");
        fw = new FileWriter(file, choic);
        bw = new BufferedWriter(fw);
        // employeesList <key 사번, value Employees> value값 가져오는 반복문
        for (Integer key : employeesList.keySet()) {
            this.em = employeesList.get(key);
            // 입력한 부서명과 등록된 사원들 중 부서명이 일치한 사원이 확인하는 판단문
            if (employeesList.get(key).getEmDept().equals(deptTitle)) {
                // 부서명 동일한 Employees 대상의 정보를 txt파일에 입력되기 전 작업
                emReturnList(em);
                // 파일에 저장하는 작업
                for (String writeText : outEmInfo) {
                    bw.write(writeText);
                    bw.write( str: "\n");
                }
            }
        }
        bw.flush();
        System.out.println(" [ " + deptTitle + " ] 추출 완료");
    } catch (IOException e) { throw new RuntimeException(e);
    } catch (Exception ee) {ee.printStackTrace();
    } finally {
        try { bw.close();
        } catch (IOException e) { throw new RuntimeException(e);
    }
}
```

-----  
사원 관리 프로그램입니다

1. 사원 추가
2. 사원 수정
3. 사원 전체보기
4. 부서별 사원 정보 조회
5. 사원 삭제
6. 부서 별 사원 정보 추출
7. 프로그램 종료

메뉴 번호 입력 >> 6

부서별 사원 정보를 txt 파일로 만들어 저장합니다.

부서명이 파일명으로 자동 생성 됩니다

- 1번 : 기존 txt 파일 내용 유지 및 현재 기록 추가
- 2번 : 기존 txt 파일 내용 삭제 후 현재 상태로 기록 작성

번호 입력 >> 1

선택 가능한 부서 목록 : 인사팀 회계팀 개발팀 영업팀 인프라팀 경영팀

부서 입력 >> 인사팀

[ 인사팀 ] 추출 완료

## 소스 코드 & 구현 – txt파일 write

```
선택 가능한 부서 목록 : 인사팀 회계팀 개발팀 영업팀 인프라팀 경영팀  
조회 부서 명 입력 >> 인사팀
```

```
직 위 : 부장
```

```
사원번호 : 100
```

```
근무부서 : 인사팀
```

```
이 름 : 홍부장
```

```
나 이 : 46
```

```
직 위 : 사원
```

```
사원번호 : 3399
```

```
근무부서 : 인사팀
```

```
이 름 : 김인사
```

```
나 이 : 29
```

```
직 위 : 과장
```

```
사원번호 : 808
```

```
근무부서 : 인사팀
```

```
이 름 : 공과장
```

```
나 이 : 46
```

```
직 위 : 임원
```

```
사원번호 : 10
```

```
근무부서 : 인사팀
```

```
이 름 : 조임원
```

```
나 이 : 50
```

```
1    사원 번호 : 100
```

```
2    이 름 : 홍부장
```

```
3    나 이 : 46
```

```
4    부 서 : 인사팀
```

```
5    직 위 : 부장
```

```
6    -----
```

```
7    사원 번호 : 3399
```

```
8    이 름 : 김인사
```

```
9    나 이 : 29
```

```
10   부 서 : 인사팀
```

```
11   직 위 : 사원
```

```
12   -----
```

```
13   사원 번호 : 808
```

```
14   이 름 : 김과장
```

```
15   나 이 : 46
```

```
16   부 서 : 인사팀
```

```
17   직 위 : 과장
```

```
18   -----
```

```
19   사원 번호 : 10
```

```
20   이 름 : 조임원
```

```
21   나 이 : 50
```

```
22   부 서 : 인사팀
```

```
23   직 위 : 임원
```

```
24   -----
```

실제 hashMap에 있는 데이터값 , txt파일로 write시 작성된 내용

# 감사합니다

 [source code gitHub url](#)

by. 김가율