OO JAVA
6주차 인터페이스 (1)

6.1. 인터페이스와 코드 재사용

6.1. 인터페이스와 코드 재사용

- 서점 프로그램 리뷰
  - Manageable 인터페이스를 통해 서점이 제공하는 기능을 Book이 아닌 다른 클래스에도 제공 가능해짐
  - 서점 코드를 그대로 사용하면서 여러 가지 클래스를 지원하는 효과

- Manageable 인터페이스를 학생에도 적용가능할까?
  - 학생도 read, print, matches를 가지므로 인터페이스 상속 가능

  - 그러면 서점 클래스와 학과 클래스가 하는 일이 중복됨

6.2. Manageable 인터페이스

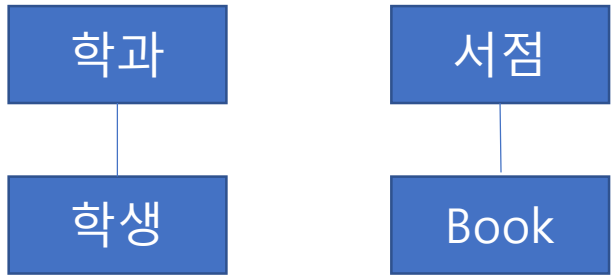
- Student 클래스에 적용
  - class Student implements Manageable { …
  - 인터페이스 구현 메소드를 pubic으로 바꿈

- 인터페이스 상속의 의미는?
  - 기본기능을 가진 인터페이스를 정의 (표준)
  - 그것을 구현하여 클래스를 작성

```
interface Manageable {
    void read(Scanner scan);
    void print();
    boolean matches(String kwd);
}
```

표준 인터페이스 사용

- 큰 프로젝트에서 코딩하는 방식을 통일
  - 클래스 종류가 많아질 때
  - 요소 클래스(데이터를 가지는 클래스)의 기본 인터페이스 사용
    - 기본적인 기능을 공통으로 사용: 입력, 출력, 비교, …
    - 클래스의 사용방식과 형식을 통일 – 달라지는 부분 최소화
    - => 코드재사용 가능성을 극대화
- 관리 클래스의 코드 재사용
  - 관리 클래스의 중복도 최소화



- 관리 클래스의 코드 재사용
  - 서점과 학과 클래스: readAll, printAll, search 등 하는 일이 비슷
  - 비슷한 일을 하는 관리 클래스들의 코드를 재사용하는 방법은?
  - 큰 프로젝트에서는 이런 관리 역할을 하는 클래스가 많아짐

- 관리클래스의 코드가 중복되지 않게 재사용
  - 중복되는 코드를 슈퍼 클래스로 만들고 그것을 상속

인터페이스와 조직관리 비유

- 대형 조직의 관리 기법
  - 표준화된 규칙: 업무시작과 끝, 휴식,
  - 업무 정의 표준: 물류, 매장관리, 계산 등
  - 업무 표준: 각 부문별 업무 방식과 내용의 표준화
  - 업무 표준화의 장점
    - 작업 방식에 대한 혼선 최소화
    - 공통으로 정해진 규칙에 따라 정보 소통 필요성 최소화
  - 관리의 표준화: 통일 서식 사용, 통일된 규칙과 방식 적용
  - 관리의 비용을 최소화
- 대형 소프트웨어에서 인터페이스의 도입
  - 클래스의 종류 별로 메소드 종류를 결정 (역할과 실행방식)
  - 각 클래스 작성 시 가이드라인의 역할
  - 관리 클래스 부분의 코드 재사용을 향상

6.3. Manager 클래스

- 학과와 서점 클래스 코드 중복
  - readAllStudents, printAllStudents, searchStudents, …
  - readAllBooks, printBooks, searchBooks, …

- 이들 코드의 재사용
  - 슈퍼 클래스를 만들어 공통 코드를 위로 올린다
  - 학과와 서점은 그 클래스를 상속, 달라지는 부분만 가짐

  - => Manager 클래스

- Manager 클래스
  - 공통되는 기능을 가짐
  - ArrayList를 Manageable로 선언 – 학생, Book, Pen등 모두 가능
  - 공통되는 코드들을 가짐

```
abstract public class Manager {
        ArrayList<Manageable> mList = new ArrayList<>();
        void readAll(String filename) { ...
        void printAll() { ...
        void search() { ...
        Manageable find(String kwd) { ...
}
```

Manager를 상속한 학과 클래스

- 학과 클래스는 run 함수만 가지면 됨
  - readAll, printAll, search를 모두 Manager가 제공

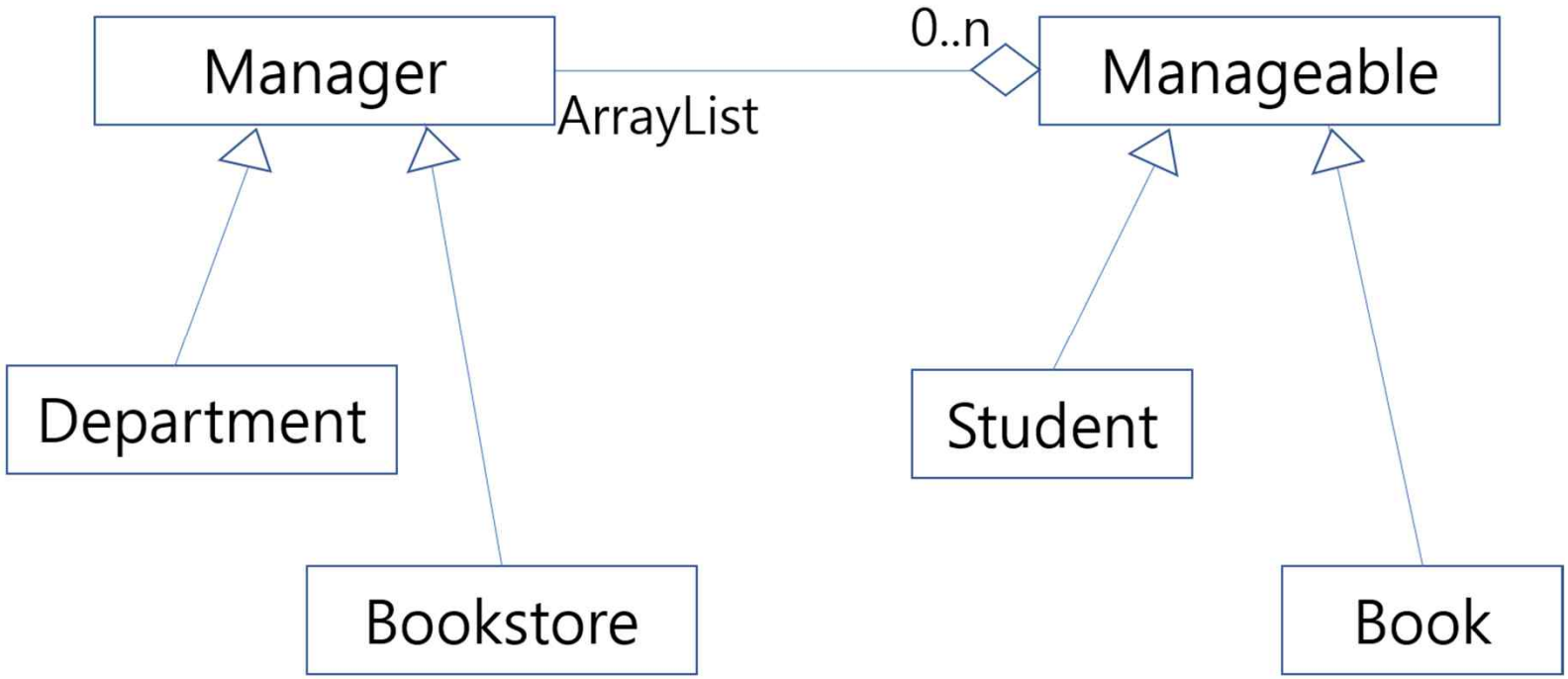
```
class Department extends Manager {
        public static void main(String args[]) {
                Department main = new Department();
                main.run();
        }
        void run() {
                readAll("student.txt");
                printAll();
                search();
        }
}
```

SRP
OCP
LSP
ISP
DIP

OO JAVA
6주차 인터페이스 (2)

6.2. Manager 클래스

클래스 관계도


Manager
0..n
Manageable
ArrayList
Department
Bookstore
Student
Book

printAll 메소드의 동작

- Manageable의 리스트로 출력
  - mList는 Manageable을 가진 리스트임
  - m.print() 에서 불려지는 print 메소드는 다형성에 의해 실제 객체의 출력 함수가 불려짐
- for (Manageable m: mList)
- m.print();

Manager의 search 함수

- Manager의 search 함수
  - Manageable의 요소에 대해 matches를 불러 비교
  - 매치되면 출력
  - 가상함수에 의해 실제 객체의 해당 메소드가 불려짐
- mList
  - 학과 클래스에서는 학생,
  - 서점 클래스에서는 Book 이 들어있음

```
void search() {
        String kwd = null;
        while (true) {
            System.out.print(“>> ”);
            kwd = scan.next();
            if (kwd.equals(“end”))
                break;
            for (Manageable m: mList) {
                if (m.matches(kwd))
                    m.print();
            }
```

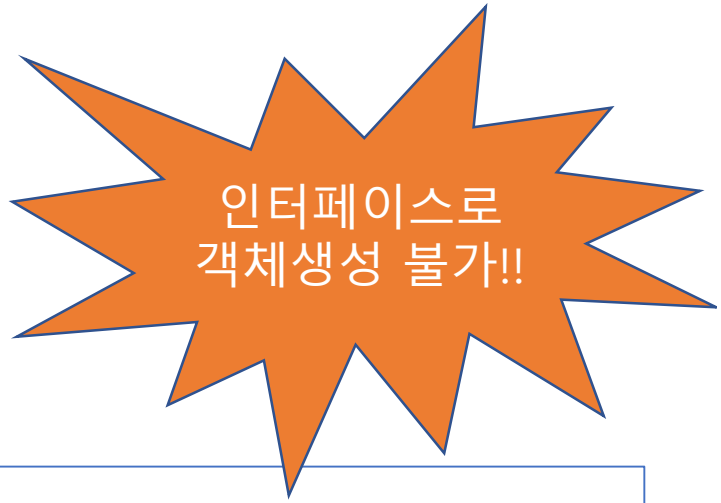
Manageable의 readAll

- 파일을 열어
  - 객체를 생성하고
  - Read 메소드로 해당 데이터를 읽게 함
  - mList에 추가

```
void readAll(String filename) {
        Scanner filein = openFile(filename);
        Manageable m = null;
        while (filein.hasNext()) {
                m = new Manageable();  // 컴파일 오류
                m.read(filein);
                mList.add(m);
        }
        filein.close();
}
```

Manageable의 readAll

- 파일을 열어
  - 객체를 생성하고
  - Read 메소드로 해당 데이터를 읽게 함
  - mList에 추가




```
void readAll(String filename) {
        Scanner filein = openFile(filename);
        Manageable m = null;
        while (filein.hasNext()) {
                m = new Manageable();  // 컴파일 오류
                m.read(filein);
                mList.add(m);
        }
        filein.close();
}
```

인터페이스와 객체 생성

- 객체 생성은 구체 클래스만 가능
  - Student인지, Book인지 정해 주어야 함
- readAll은 어떻게 객체를 생성해야 할까?
  - Student인지 Book인지는 누가 아는가?
  - 학과 또는 서점 클래스에서 판단해야 함
  - 해당 클래스에게 객체를 생성해 달라고 요청해야 함

확인학습문제

1. 프로젝트에서 사용하는 표준 인터페이스를 도입할 때 장점은 무엇인가?

2. 전혀 상관없는 클래스들을 인터페이스로 상속할 때 장점은 무엇인가?

3. 인터페이스를 상속하는 클래스에서 해주어야 하는 일은 무엇인가?

SRP
OCP
LSP
ISP
DIP

OO JAVA
6주차 인터페이스 (3)

6.4. Factory 인터페이스

6.4. Factory 인터페이스

- 팩토리 인터페이스
  - interface Factory {
  - public Manageable create();
  - }

- 팩토리 인터페이스의 필요성
  - void readAll(학과 department) {
  - Manageable m = department.create(); // 학생 생성

  - Void readAll(서점 bookstore) {
  - Manageable m = bookstore.create(); // Book 생성

- 학과와 서점의 공통 슈퍼 타입
  - Manager를 상속하므로
  - 추가 상속은 인터페이스여야 가능
  - Factory라는 인터페이스에서 create 제공

```
void readAll(String filename, Factory fac) {
        Scanner filein = openFile(filename);
        Manageable m = null;
        while (filein.hasNext()) {
                m = fac.create();
                m.read(filein);
                mList.add(m);
        }
        filein.close();
}
```

Factory 구현

- Department가 Factory로 전달됨

```
class Department extends Manager implements Factory {
        …
        public Student create() {
                return new Student();
        }
        void run() {
                readAll("student.txt", this);
                printAll();
                search();
        }
```

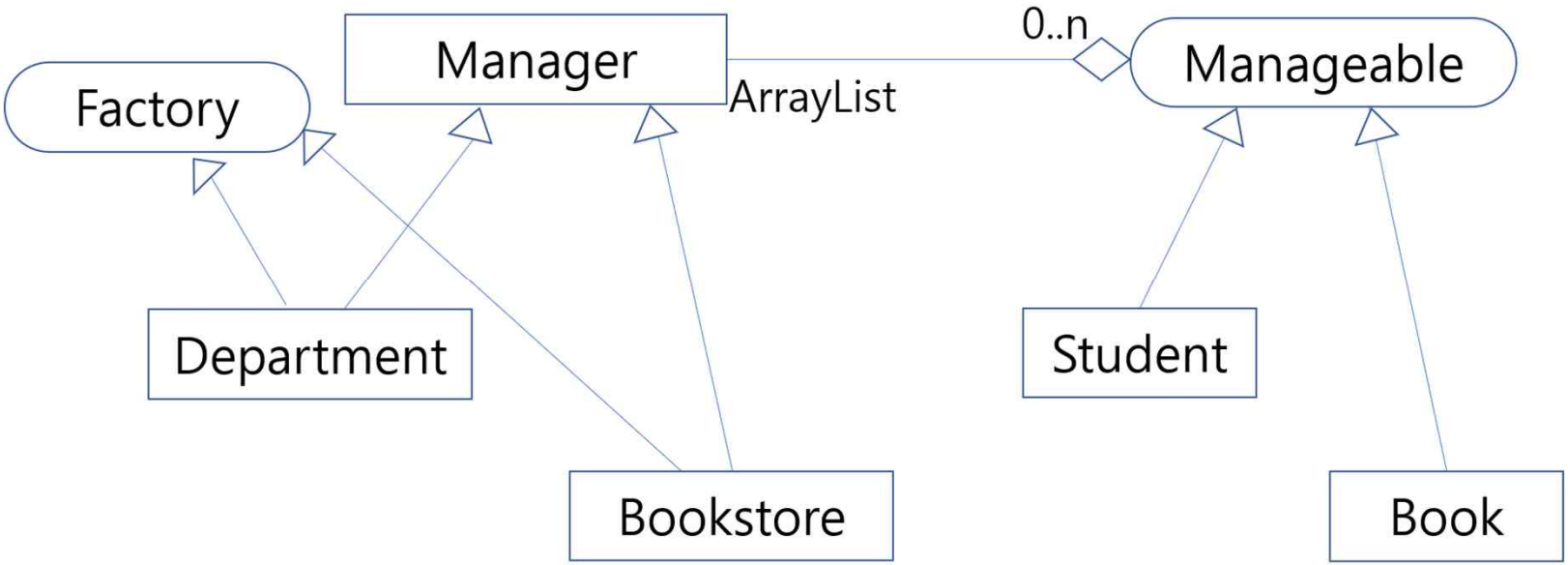
Factory 구현

- Bookstore가 Factory로 전달됨

```
class Bookstore extends Manager implements Factory {
        …
        public Book create() {
                return new Book();
        }
        void run() {
                readAll("book.txt", this);
                printAll();
                search();
        }
```

클래스 관계도

- 관리 클래스
  - Manager와 Factory를 공통으로 상속



3. 자바에서 여러 가지 인터페이스를 동시에 구현할 때 장점은?

4. 인터페이스 A와 B를 구현한 클래스 C가 A로 쓰일 수 있는 이유는?

5. C 객체가 A로 쓰일 때 호출할 수 있는 메소드는 무엇인가?

- 학과와 서점 클래스가 Factory 인터페이스를 구현하는 이유는 무엇인가?, Factory를 인터페이스로 하는 이유는?

SRP
OCP
LSP
ISP
DIP

OO JAVA
6주차 인터페이스 (4)

인터페이스 구현과 코드재사용

Manager 기반 구현의 한계

- Manageable과 다른 기능의 메소드 필요
  - void print(bLecture)
    - 학생에서 수강과목을 출력할지 말지 결정하는 매개변수?
    - 인터페이스 메소드를 기본으로 하고 추가 기능은 추가 메소드로 제공
  - void read(Scanner scan); 이외에 학과를 보내야 한다면?
  - Bookstore 의 create()에서 상속에 따라 다른 클래스를 돌려주려면?
    - 스캐너를 매개변수로 받아 create에서 switch 문을 사용
    - Factory 인터페이스를 바꾸어야 함

- 인터페이스란 간단, 명료, 통일 => 특수 상황을 인터페이스에 추가할 수는 없음

- 다른 기능을 추가하는 방법
  - 추가로 메소드를 만든다
  - 인터페이스의 메소드를 수정한다.
  - 인터페이스에 메소드를 추가한다

코드 재사용의 원칙

- 가능한 부분은 재사용하고
- 달려져야 하는 부분은 새로 작성
- 재사용할 코드를 건드리지 않고 할 수 있으면 더 좋음

학과 클래스에서 학생과 과목 관리

- Manager 클래스를 필드로 사용하는 방법
  - 상속하지 않고 Manager를 두 개 가지는 방법
  - Manager studentMgr = new Manager();
  - Manager lectureMgr = new Manager();
  - readAll, printAll, search, find 기능을 Manager가 제공
  - 이것을 이용하여 프로그램을 작성

- 학생이 수강과목을 찾기 위해서 lectureMgr.find를 필요로 함
  - read 메소드에 이것을 넣어서 보내면? 인터페이스 변경 필요 X
  - lectureMgr를 static으로 선언하고 그것을 직접 접근하는 방법

```
public class Department {
	Scanner scan = new Scanner(System.in);
	Manager studentMgr = new Manager();
	static Manager lectureMgr = new Manager();
	void run() {
		lectureMgr.readAll("lecture.txt", new Factory() {
			public Manageable create() {
				return new Lecture();
			}
		});
		lectureMgr.printAll();
		studentMgr.readAll("student.txt", new Factory() {
			public Manageable create() {
				return new Student();
			}
		});
		studentMgr.printAll();
		studentMgr.search(scan);
	}
```

```
public void read(Scanner scan) {  // Student
    …
    String code;
    Lecture lecture = null;
    while (true) {
        code = scan.next();
        if (code.equals("0"))
            break;
        lecture = (Lecture)Department.lectureMgr.find(code);
        if (lecture == null) {
            System.out.println("과목없음: " + code);
            continue;
        }
        registeredList.add(lecture);
    }
}
```

확인학습문제

4. Manageable이나 Manager 등의 방법으로 슈퍼 타입을 만드는 장점은 무엇인가?

5. Manager 클래스를 사용하기 위해 Factory 인터페이스가 필요한 이유는 무엇인가?

6. Manager를 상속한 클래스에서 Factory도 상속할 수 있는 이유는 무엇인가?

SRP
OCP
LSP
ISP
DIP

OO JAVA
6주차 인터페이스 (5)

6장 인터페이스 코딩