

소프트웨어공학개론

오유수, Ph.D.

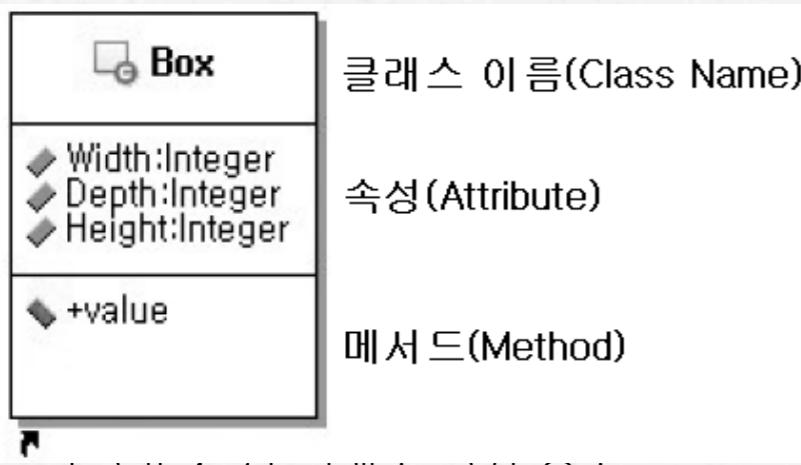
[Ref] “소프트웨어 공학의 소개”, 한혁수, 홍릉과학출판사

“성공적인 소프트웨어 개발 방법론”, 윤청, 생능출판사

오늘의 강의 목차

- 오늘의 강의 내용
 - 클래스 다이어그램
 - 클래스와 객체 개념
 - 클래스 추출
 - 클래스간 관계
 - 오늘의 강의 요약
 - 요약, Homework, 다음 강의 소개

클래스의 구성 요소



[그림 4-1] 클래스 구성 요소

클래스 구성 요소 : _____.

[표 4-1] 메서드의 종류와 기호

메서드의 종류	부호	내용
public	+	자신의 속성이나 동작을 외부에 공개하는 접근 제어
private	-	상속된 파생 클래스만이 엑세스할 수 있는 접근 제어
protected	#	구조체의 멤버 함수만 접근할 수 있으며 외부에서 엑세스할 수 없는 접근 제어

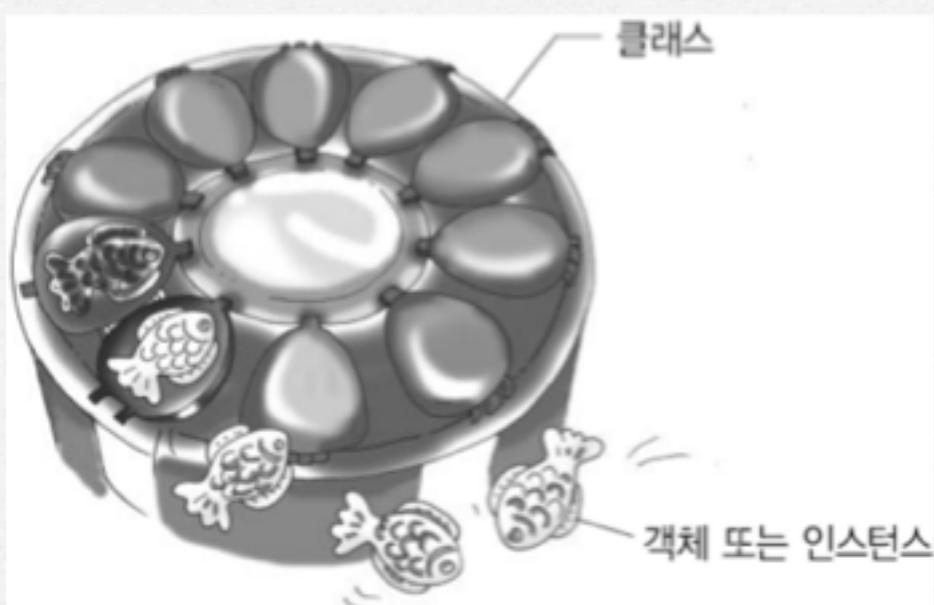
[코드 4-1] [그림 4-1]에 대한 자바 코드

```
01 // class name
02 Box{
03     // attribute
04     private int width;
05     private int depth;
06     private int hight;
07     // method
08     void value() {
09     }
10 }
```

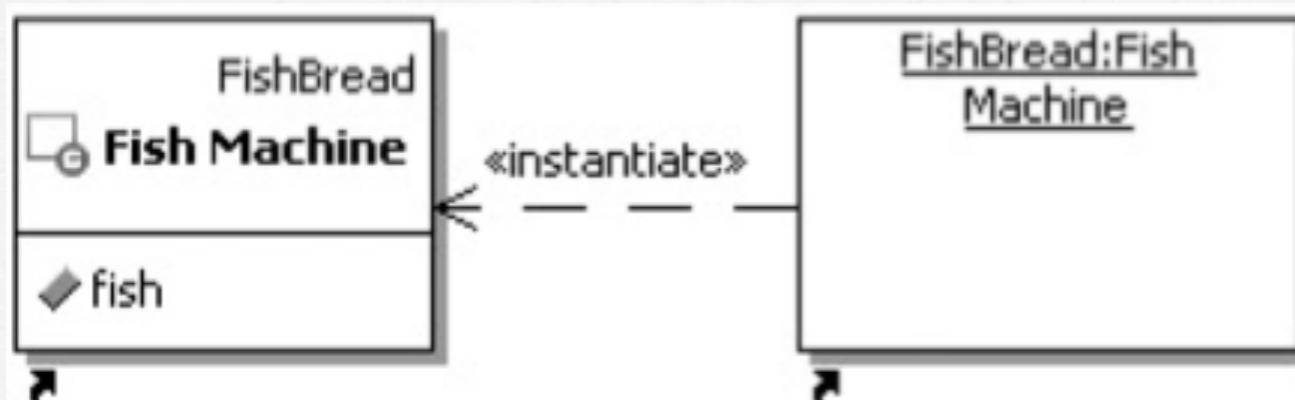
클래스의 구성 요소

- .
- 공통의 속성, 메서드(오퍼레이션), 관계, 의미를 공유하는 객체들의 집합
- 속성 (*attribute*)
 - 클래스의 구조적 특성에 이름을 붙인 것으로 특성에 해당하는 인스턴스가 보유할 수 있는 값의 범위를 기술
 - 속성은 영문자 소문자로 시작
- 메서드 (*method*)
 - 오퍼레이션이라고도 하며,
 - 이름, 타입, 매개변수들과 연관된 행위를 호출하는데 요구되는 제약사항들을 명세하는 클래스의 행위적 특징

- 클래스: 실제 현실에서 존재하는 사물을 의미
- 객체: 객체들을 추상화한 개념



[그림 4-2] 봉어빵 기계와 봉어빵(클래스와 객체의 관계)



[그림 4-3] 클래스와 객체와의 관계

- 예제 : 다음의 다이어그램은 하나의 계좌에 입금되는 클래스 (*Account*)와 객체를 생성하여 실행하는 메인 메서드를 포함하는 클래스 (*Application*)로 구성

[코드 4-2] 객체 생성의 자바 코드

```

01 class Account
02 {
03     int balance;
04     int deposit(int amount){
05         balance = balance + amount;
06         return (balance);
07     }
08 }
09
10 class Application{
11
12     public static void main(String args[])
13     {
14         Account account1;
15         // 객체 생성
16         account1 = new Account();
17         account1.balance = 5000;
18         account1.deposit(5000);
19         System.out.println("Balance = " +account1.balance);
20     }
21 }
```



[그림 4-4] 연관관계의 종류



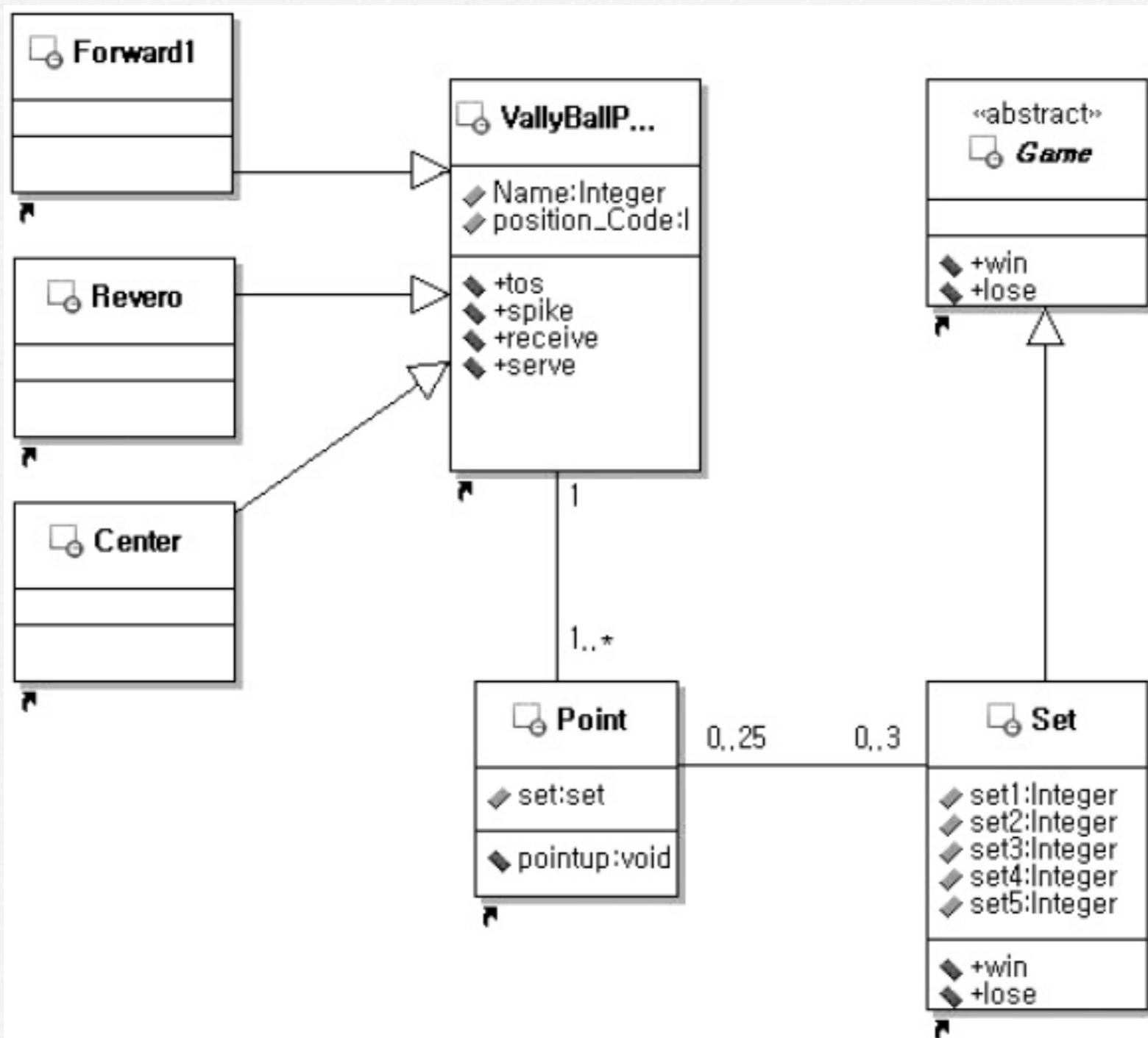
[그림 4-5] 연관관계의 종류

명세서에서 클래스 추출하기

배구 선수는 1명의 리베로와 1명의 센터, 2명의 후위, 공격수 2명으로 구성되어 있다. 각 선수는 리시브, 토스와 스파이크를 이용하여 상대편이 공을 받아낼 수 없도록 하여 코트에 공이 달도록 하는 게임이다. 배구 코트는 남녀 모두 길이 18m, 너비 9m의 직사각형 구획선으로 둘러싸였으며, 구획선으로부터 최소한 3m의 프리존이 있어야 하고 코트 면 위 7m 까지에는 어떠한 장애물이 있어서는 안 된다. 리베로가 받으면 센터가 토스를 올리고 공격수가 스파이크를 친다. 그렇게 하여 상대편 코트에 공이 닿으면 점수를 획득한다. 경기는 한 세트당 25점이며, 3세트를 먼저 획득한 팀이 승리를 하게 된다.

- 명세서를 통한 클래스 추출 : 배구선수, 점수, 세트, 경기
- 명세서를 통한 객체 추출 : 리베로, 센터, 후위, 공격수
- 명세서를 통한 메서드 추출 : 토스, 스파이크, 리시브, 이긴다, 진다, 올라간다.

- 명세서(시나리오) 상에서 클래스나 속성 또는 메서드의 추출은
 - 명사 : 클래스나 속성 (Attribute)
 - 동사 : 메서드
- 어떤 것이 클래스가 되고, 어떤 것이 속성이 되느냐는
 - DB 설계시에 엔티티와 필드의 관계처럼 설계자가 판단해서 어떻게 구분해서 사용하는게 더 나은지에 따라 '클래스'와 '속성'으로 추출



[그림 4-6] 배구에 관한 클래스 다이어그램

연관관계

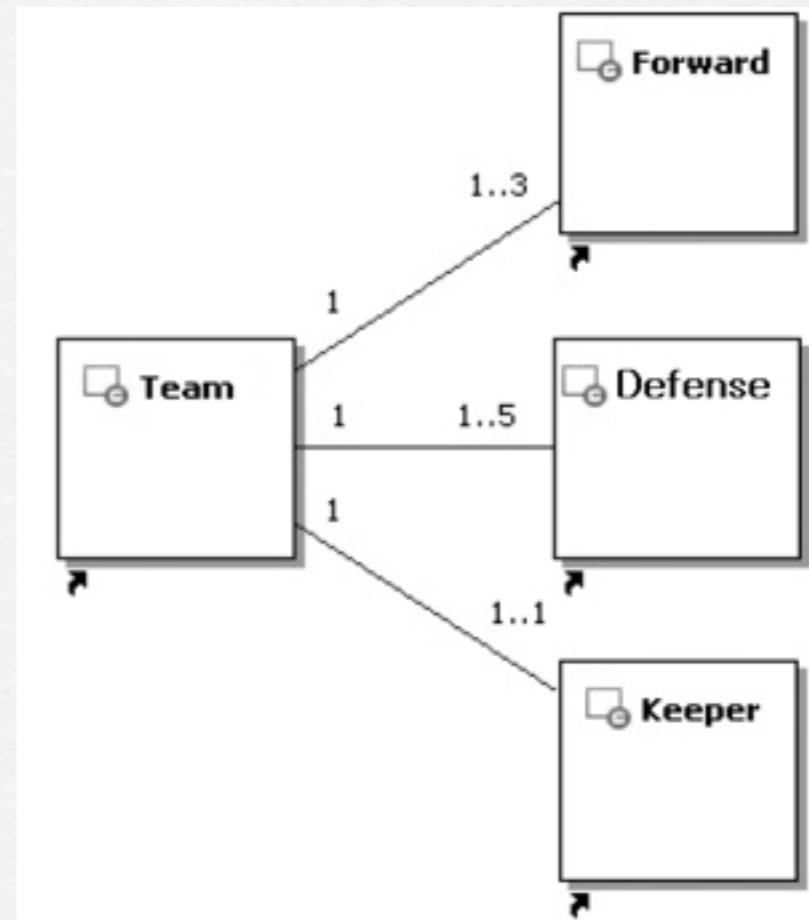
- 연관관계는 _____을 의미
- 클래스 다이어그램에서 연관관계는
 - 서로 개념적으로 연관없는 클래스는 없으므로 의존, 상속, 집합, 복합 등의 관계와 같이 표시하는 건 별로 중요하지 않다.
- 예제 : 축구팀과 선수와의 관계



[그림 4-7] 팀과 선수의 연관관계

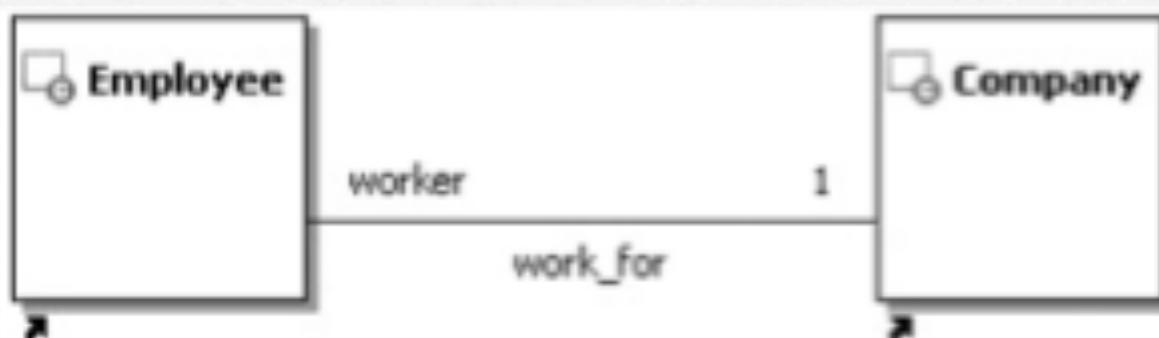


[그림 4-8] 선수와 구단의 연관관계

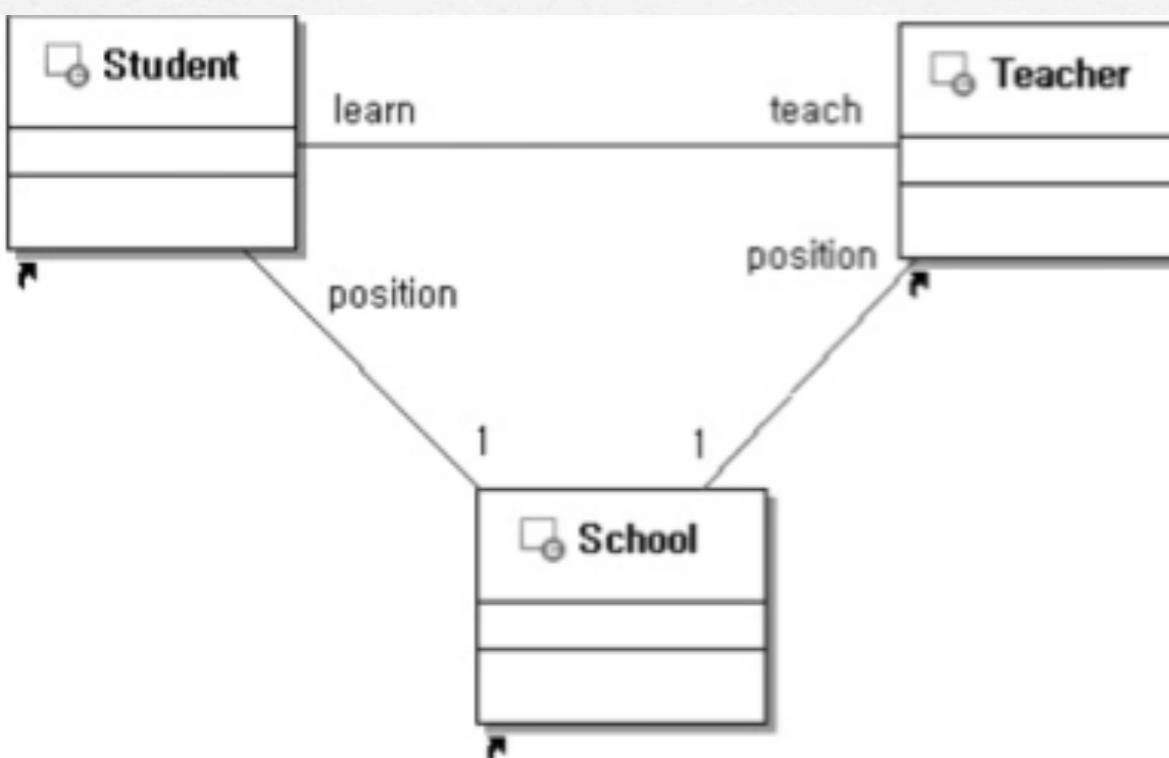


[그림 4-9] 하나의 클래스와 여러 클래스와의 연관관계

연관관계의 예들



[그림 4-10] 사원과 회사 사이의 연관관계



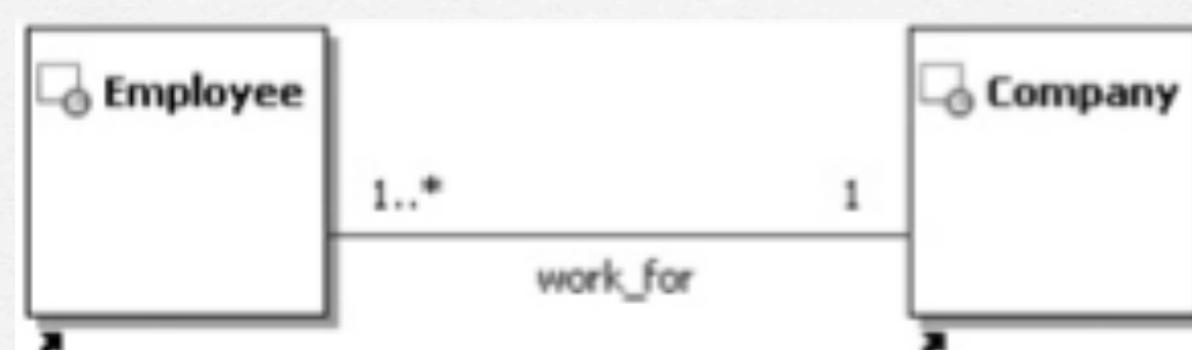
[그림 4-11] 학생, 교사, 학교 사이의 연간관계

연관관계의 다중성

- 다중성 : 두 클래스의 연관관계에서 실제로 연관을 가지는 객체의 수를 나타낸 것



[그림 4-12] 선수와 팀의 다중성



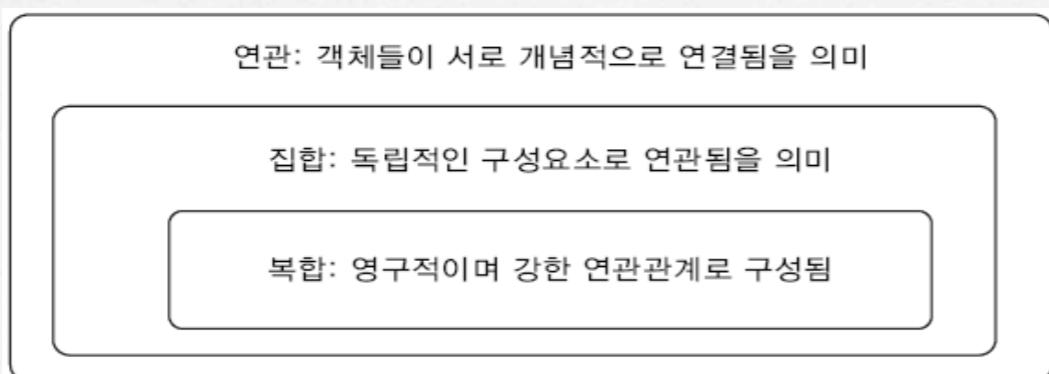
[그림 4-13] 회사와 사원의 다중성 관계

다중성 표현	의 미
1	한 객체와 연관된다. 표시하지 않아도 되는 기본값이다.
0..1	0개 또는 1개의 객체와 연관된다.
0..*	0개 또는 많은 수의 객체가 연관됨을 나타낸다.
*	0..*와 동일하다
1..*	1개 이상의 객체와 연관된다.
1..12	1개에서 12개까지의 객체가 연관됨을 나타낸다.
1..2, 4, 11	1개에서 2개까지 또는 4개 또는 11개의 객체가 연관됨을 나타낸다.

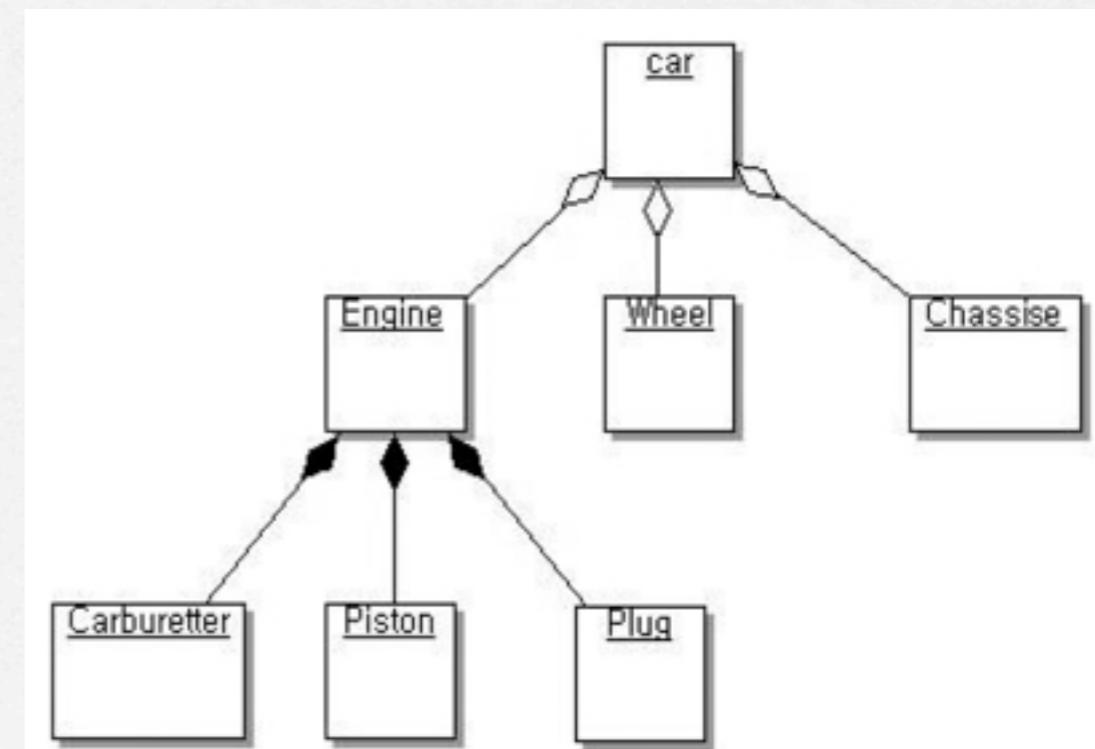
[표 4-2] 다중성의 표현

집합과 복합관계

- 집합(Aggregation)관계와 복합(Composition)관계 모두 연관관계에 포함되는 개념
- 집합관계
 - 하나의 객체에 여러 개의 독립적인 객체들이 구성되는 경우
- 복합관계
 - 더 강한 관계로 구성
 - 엔진은 카뷰레터, 피스톤, 플러그로 구성
 - 엔진의 구성 요소는 더 강한 관계



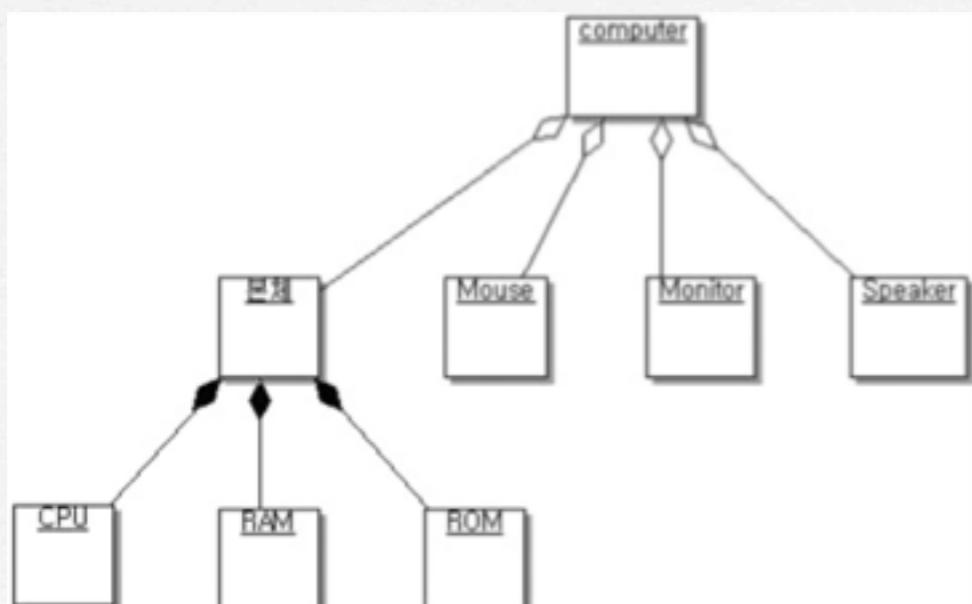
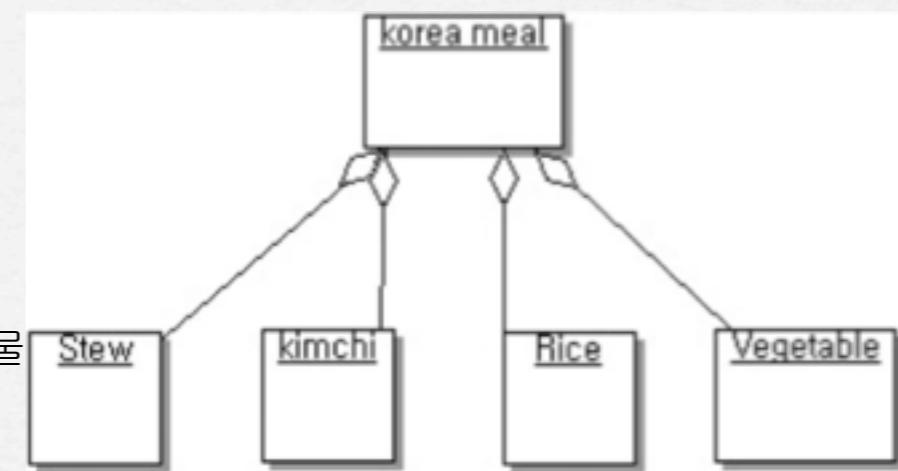
[그림 4-15] 차와 엔진, 바퀴, 차체(집합관계)
엔진과 카뷰레터, 피스톤, 플러그(복합관계)



집합과 복합 예제

- 식사가 밥,찌개,김치,나물 등으로 구성되어 있을 경우, 이들은 식사에 대한 구성 요소이기 때문에 집합관계
- 컴퓨터도 마찬가지로 그 구성 요소로 이루어지면 집합관계
- 컴퓨터 본체는 여러 가지의 구성 요소가 존재하는데 이는 영구적인 요소이기 때문에 복합관계

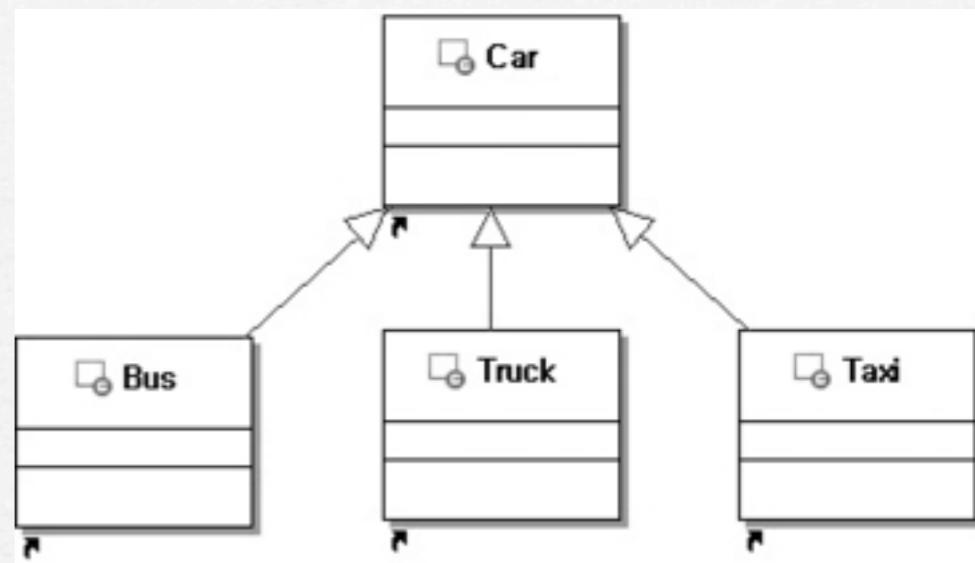
[그림 4-16] 식사 : 밥,찌개,김치,나물



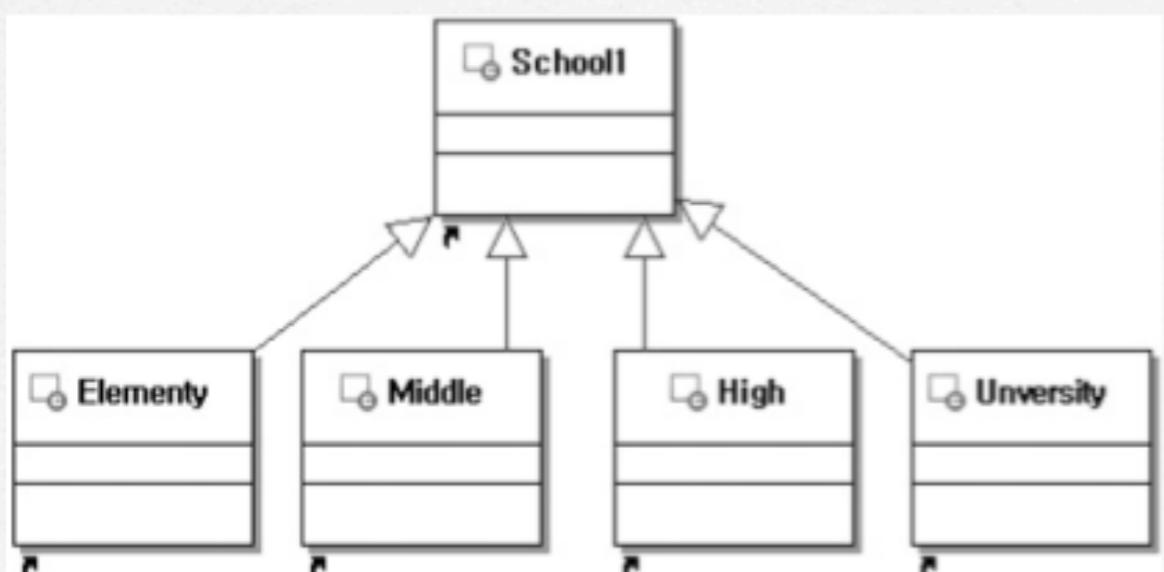
[그림 4-17] 컴퓨터와 모니터, 마우스, 키보드, 스피커(집합관계).
본체와 CPU, ROM, RAM(복합관계)

일반화관계

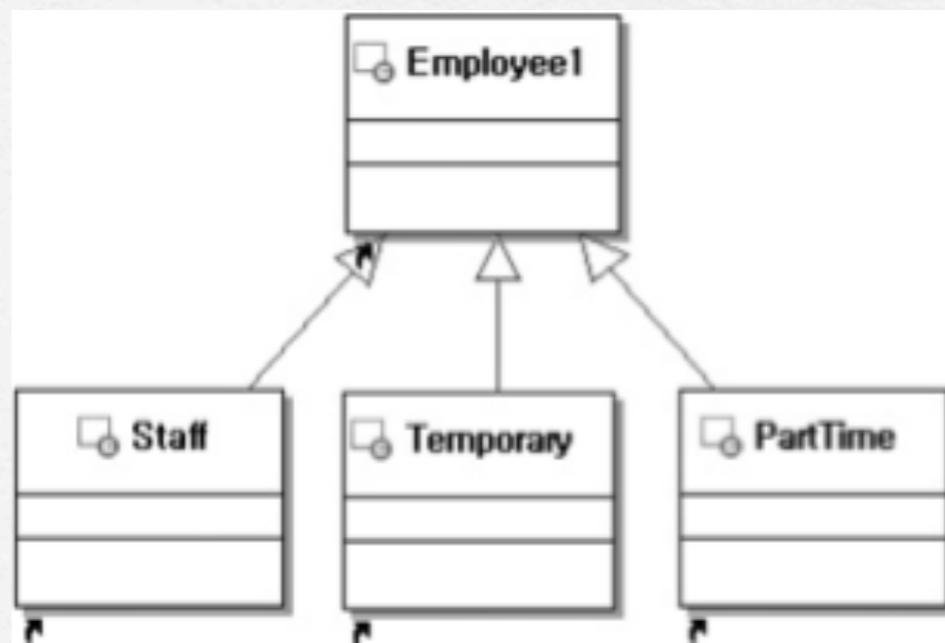
- 일반화관계는 _____를 의미
- 일반화관계는 다른 의미로 _____라고도 한다.



[그림 4-18] 차와 버스, 트럭, 택시(일반화관계)

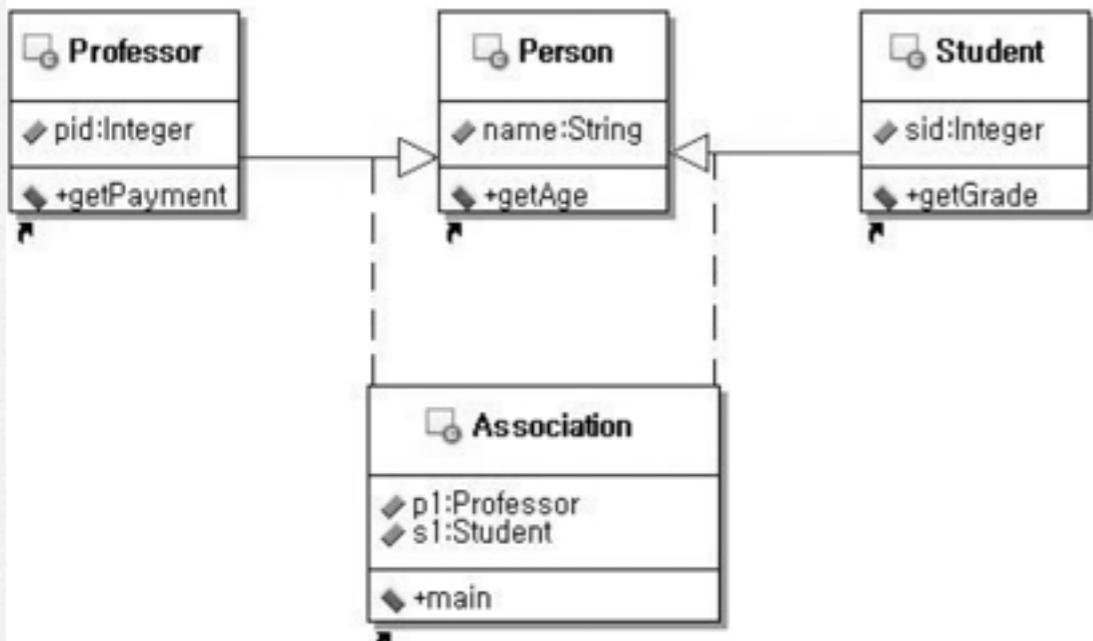


[그림 4-19] 학생과 초등학생, 중학생, 고등학생, 대학생
(일반화관계)



[그림 4-20] 사원과 정사원, 계약사원, 아르바이트생
(일반화관계)

□ [예제 4-1] 사람과 교수, 학생 간의 일반화와 연관관계



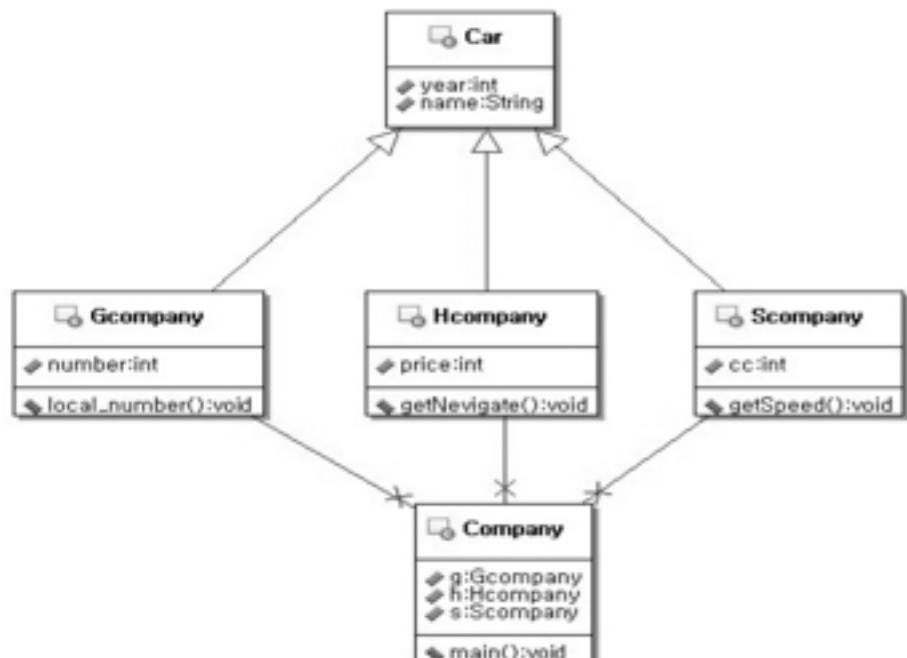
[그림 4-21] 사람과 교수, 학생 간의 일반화관계

[코드 4-3] 일반화관계 자바 코드

```
01 class Person
02 {
03     String name;
04     int getAge()
05     {
06         return 49;
07     }
08 }
09
10 class Student extends Person
11 {
12     int sid;
13     int getGrade(){
14         return sid-200400;
15     }
```

```
16 }
17
18 // Person으로 상속받기 때문에 name, getAge(), pid, getPayment()로 구성
19 class Professor extends Person
20 {
21     int pid;
22     int getPayment()
23     {
24         return pid+10000;
25     }
26 }
27
28 class Inheritance
29 {
30     public static void main(String[] args) {
31         Student s1 = new Student();
32         Professor pl = new Professor();
33         s1.name = "홍길동";
34         s1.sid = 200401;
35         System.out.println("Student name : "+s1.name+ "Student
ID : "+s1.sid);
36         System.out.println("Student Age : "+s1.getAge()+
"Student Grade : "+s1.getGrade());
37         pl.name = "홍교수";
38         pl.pid = 1016;
39         System.out.println("Professor name : "+pl.name+
"Professor ID : "+pl.pid);
40         System.out.println("Professor Age : "+pl.getAge()+
"Professor Payment : "+pl.getPayment());
41     }
42 }
```

□ [예제 4-2] 자동차와 자동차 제품 간 일반화 및 연관관계



[코드 4-4] 일반화(상속)관계 자바 코드

```

01  class Car
02  {
03      String name;
04      int year;
05      int getYear()
06      {
07          return year;
08      }
09  }
10  class Gcompany extends Car
11  {
12      int number;
13      int getLocal_number(){
14          return number;
15      }
16  }
17  class Hcompany extends Car
18  {
19      int price;
  
```

[그림 4-22] 자동차와
자동차 제품 간 일반화관계

```

20      int getNavigate() {
21          return price+10000000;
22      }
23  }
24
25
26  class Scompany extends Car
27  {
28      int cc;
29      int getSpeed(){
30          return cc+1300;
31      }
32  }
33
34  public class Company
35  {
36      public static void main(String args[])
37      {
38          Gcompany g = new Gcompany();
39          Hcompany h = new Hcompany();
40          Scompany s = new Scompany();
41          g.name = "해간자";
42          g.number = 2001;
43          System.out.println("Car name: "+g.name+"Car number: "+g.number);
44          System.out.println("Car year: "+g.getYear()+"Car number: "+g.getLocal_number());
45          h.name = "소나파";
46          h.price = 10000000;
47          System.out.println("Car name: "+h.name+"Car price: "+h.price);
48          System.out.println("Car year: "+h.getYear()+"Car number: "+h.getNavigate());
49          s.name = "SMS";
50          s.cc = 2500;
51          System.out.println("Car name: "+s.name+"Car price: "
  
```

의존관계

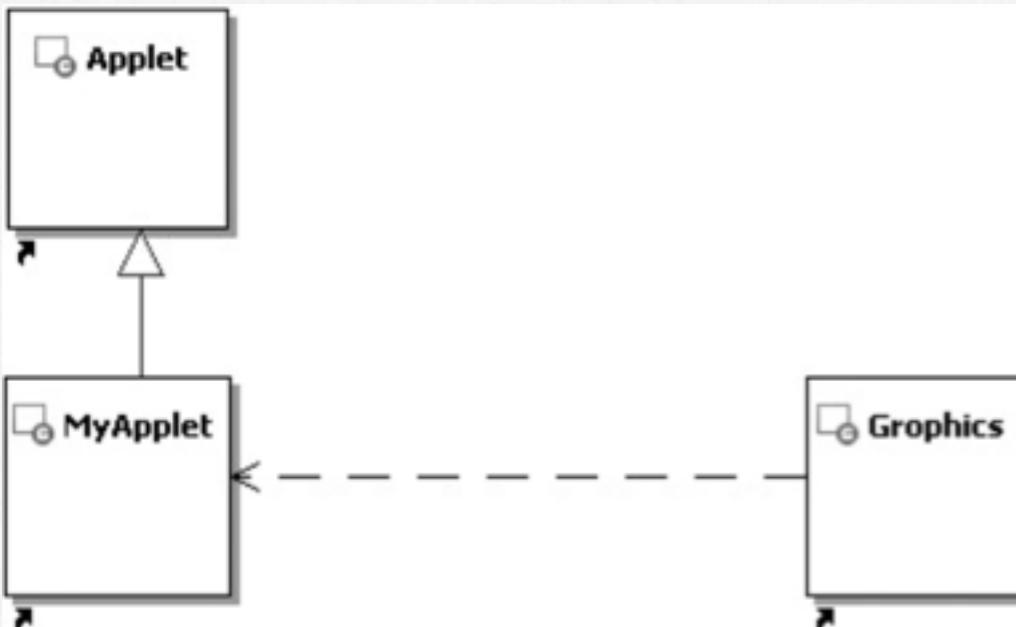
- 의존관계 : _____ 관계
- 다른 클래스의 멤버 함수를 사용하는 경우
- 하나의 클래스에 있는 멤버 함수의 인자가 _____의 관계
를 의미



[그림 4-23] TV와 리모컨의 의존관계

- 다른 의존관계
 - 수업 → 교수
 - 전화기 → 버튼
 - 세탁기 → 손잡이
 - 자동차 → 기어

- 의존관계에 있어서 클래스 A가 클래스 B의 객체를 생성하는 경우
- 예 : 애플릿 프로그램인 MyApplet의 paint() 메서드에서 text를 출력하기 위한 g.drawString() 메서드 호출할 경우

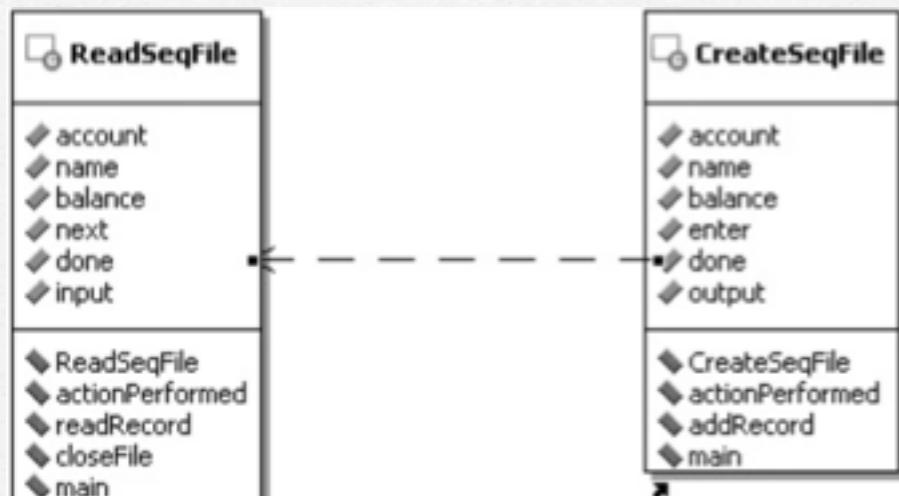


[그림 4-24] 메소드 호출의 경우(의존관계)

[코드 4-5] 애플릿 프로그램

```
01 public class MyApplet extends Applet
02 {
03     init();
04     public void start();
05     public void paint(Graphics g)
06     {
07         g.drawString("Hello Applet World",10,20);
08     }
09 }
```

- 예제 4-3 계좌를 저장하고 읽는 프로그램
- 이 프로그램은 seq를 생성하여 계좌를 만든다.
- 계좌에 값을 넣고서, 저장을 하고난후 seq를 읽는 프로그램을 돌려서 계좌에 들어간 값과 이름을 출력하는 프로그램



[그림 4-25] 계좌를 저장하고 읽는 프로그램

```

69     {
70         System.out.println(io.toString());
71         System.exit(1);
72     }
73 }
74
75 private void closeFile()
76 {
77     try
78     {
79         input.close();
80         System.exit(0);
81     }
82     catch (IOException ie)
83     {
84         System.out.println(io.toString());
85         System.exit(1);
86     }
87 }
88
89 public static void main(String args[])
90 {
91     new ReadSeqFile();
92 }
93 }

```

[25.44] 기본간접이 자바 제작 ①

①
②

```

34     add(balance);
35
36     next = new Button("출금"); // 파일로부터 데이터를 읽는 버튼
37     next.addActionListener(this);
38     add(next);
39
40     done = new Button("완료"); // 프로그램을 종료하는 버튼
41     done.addActionListener(this);
42     add(done);
43     setVisible(true);
44
45     public void actionPerformed(ActionEvent e) {
46         if (e.getSource() == next)
47             readRecord(); // 데이터를 한 레코드씩 읽는 메서드
48         else
49             closeFile();
50     }
51
52     public void readRecord() {
53         int accountNo;
54         double d;
55         String namedata;
56         try
57         {
58             accountNo = input.readInt(); // 정수형인 계좌번호를 읽는다.
59             namedata = input.readUTF(); // 문자열인 이름을 읽는다.
60             d = input.readDouble(); // 실수인 잔고를 읽는다.
61
62             /*입어들인 데이터를 관련된 텍스트 필드에 출력한다.*/
63             account.setText(String.valueOf(accountNo));
64             name.setText(namedata);
65             balance.setText(String.valueOf(d));
66         }
67         catch(EOFException eof){closeFile();}
68         catch (IOException io)
69     }

```

- 위의 코드에 의해서 다음 코드가 영향을 받는다.

[코드 4-7] 의존관계의 자바 코드 ②

```

01 import java.io.*;
02 import java.awt.*;
03 import java.awt.event.*;
04 public class CreateSeqFile extends Frame implements
ActionListener

73     try
74     {
75         output.close(); //파일을 닫는다.
76     }
77     catch (IOException io)
78     {
79         System.err.println(io.toString());
80     }
81     System.exit(0); //프로그램을 종료한다.
82 }
83
84
85 public static void main(String args[])
86 {
87     new CreateSeqFile();
88 }

```

①

④

```

05 {
06     private TextField account, name, balance;
07     private Button enter, done;
08     private DataOutputStream output; //필터 스트림 객체
09     public CreateSeqFile(){
10         super("고객파일을 생성");
11         try
12         {
13             output = new DataOutputStream(new
FileOutputStream("client.doc"));
14         }
15         catch (IOException e)
16         {
17             System.err.println(e.toString());
18             System.exit(1);
19         }
20         setSize(250, 130);
21         setLayout(new GridLayout(4,2));
22         add(new Label("계좌번호"));
23         account = new TextField(); //계좌번호 입력 필드
24         add(account);
25         add(new Label("이름"));
26         name = new TextField(20); //이름 입력 필드
27         add(name);
28         add(new Label("잔고"));
29         balance = new TextField(20); //잔고 입력 필드
30         add(balance);
31         enter = new Button("입력"); //입력된 데이터를 저장하는 버튼
32         enter.addActionListener(this); //이벤트와 연결
33         add(enter);
34         done = new Button("종료"); //입력을 종료하는 버튼
35         done.addActionListener(this); //이벤트와 연결
36         add(done);
37         setVisible(true);
38     }

```

②

```

39     public void addRecord()
40     {
41         int accountNo = 0;
42         Double d;
43         if(!account.getText().equals("")){ //계좌번호의 입력 체크
44             try
45             {
46                 accountNo = Integer.parseInt(account.getText());
47                 if(accountNo > 0){
48                     output.writeInt(accountNo); //필터 스트림을 통하여
49                     //정수를 저장한다.
50                     output.writeUTF(name.getText()); //문자열을 저장한다.
51                     d=new Double(balance.getText()); //실수 객체 생성
52                     output.writeDouble(d.doubleValue()); //실수 저장
53                 }
54                 account.setText(""); //텍스트 필드를 삭제
55                 name.setText("");
56                 balance.setText("");
57             }
58             catch (NumberFormatException nfe)
59             {
60                 System.err.println("정수를 입력해야 합니다.");
61             }
62         }
63         catch(IOException io){
64             System.err.println(io.toString());
65             System.exit(1);
66         }
67     }
68
69     public void actionPerformed(ActionEvent e)
70     {
71         addRecord(); //입력된 데이터를 파일에 저장한다.
72         if(e.getSource() == done) {

```

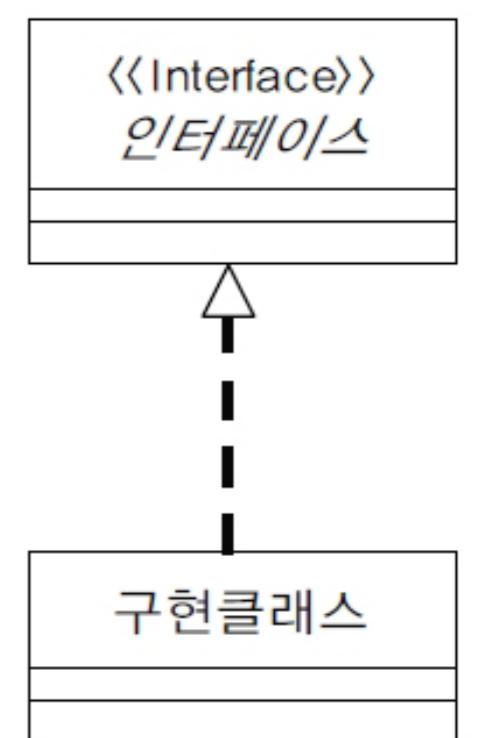
③

추상클래스와 인터페이스

- 추상클래스
 - 이탤릭체로 클래스명을 표시하며 스테레오타입을 이용하여 _____로 표기
- 인터페이스
 - 스테레오타입을 이용하여 _____로 표기하고, 이탤릭체로 인터페이스명을 표시
- 실체화 관계
 - 추상클래스나 인터페이스를 상속받아 자식클래스가 추상메서드를 구현할 때 사용



[그림 4-26] 추상클래스와 인터페이스



[그림 4-27] 실체화관계

오늘의 강의 요약

- 클래스 다이어그램
- 클래스와 객체 개념
- 클래스 추출
- 클래스간 관계

Q & A

More Information?

오유수, Ph.D.

yoosoo.oh@daegu.ac.kr

공대5호관 5506D호

