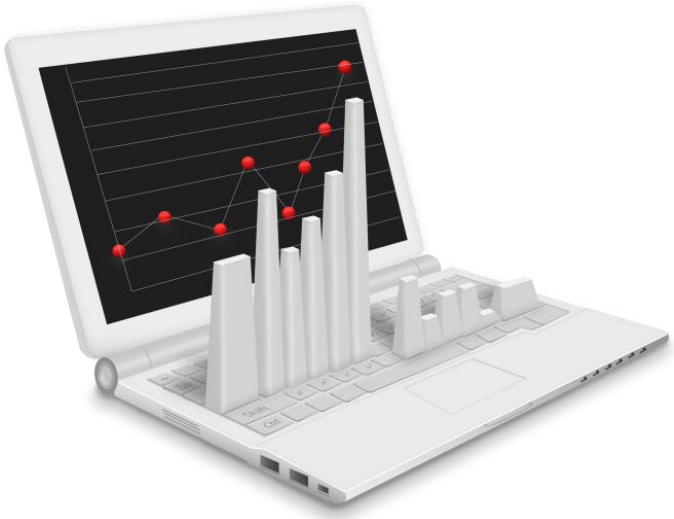
프로그래밍 언어론

다형성 (Polymorphism)

컴퓨터공학과

조은선



충남대학교
CHUNGNAM NATIONAL UNIVERSITY

학습목표

- 객체지향 프로그래밍 개념의 다형성에 대해 학습하고 기타 성질에 대해 논의한다.

학습내용

- 다형성 (polymorphism)
- 기타 객체지향 프로그래밍의 논점

목 차

- 들어가기
- 학습하기
  - 다형성
  - 객체지향 프로그래밍의 논점
- 평가하기
- 정리하기

알고가기

다음은 상위-하위 클래스관계에 대한 설명으로 거리가 가장 먼 것을 고르시오.

(1) 객체를 담는 변수 (이하, ‘객체 변수’) 는, 그를 instance(사례)로 가지는 클래스의 타입을 갖게 됨

(2) 상위클래스 정의는 하위 클래스가 정의하는 내용에 추가의 정의를 가진다.

(3) 새로운 클래스를 만들 때 기존의 적당한 클래스로부터 상속받으면 개발 시간을 단축한다.

(4) 집합의 포함관계와도 비슷하다.

확인

상속과 상/하위 타입

- 변수가 선언된 타입만으로는, 담고 있는 객체의 실제 타입을 정확히 알 수는 없다.

예 변수 p가 가리키는 것이 Person 타입 객체일 수도 있지만, Student 타입의 객체일 수도 있다.

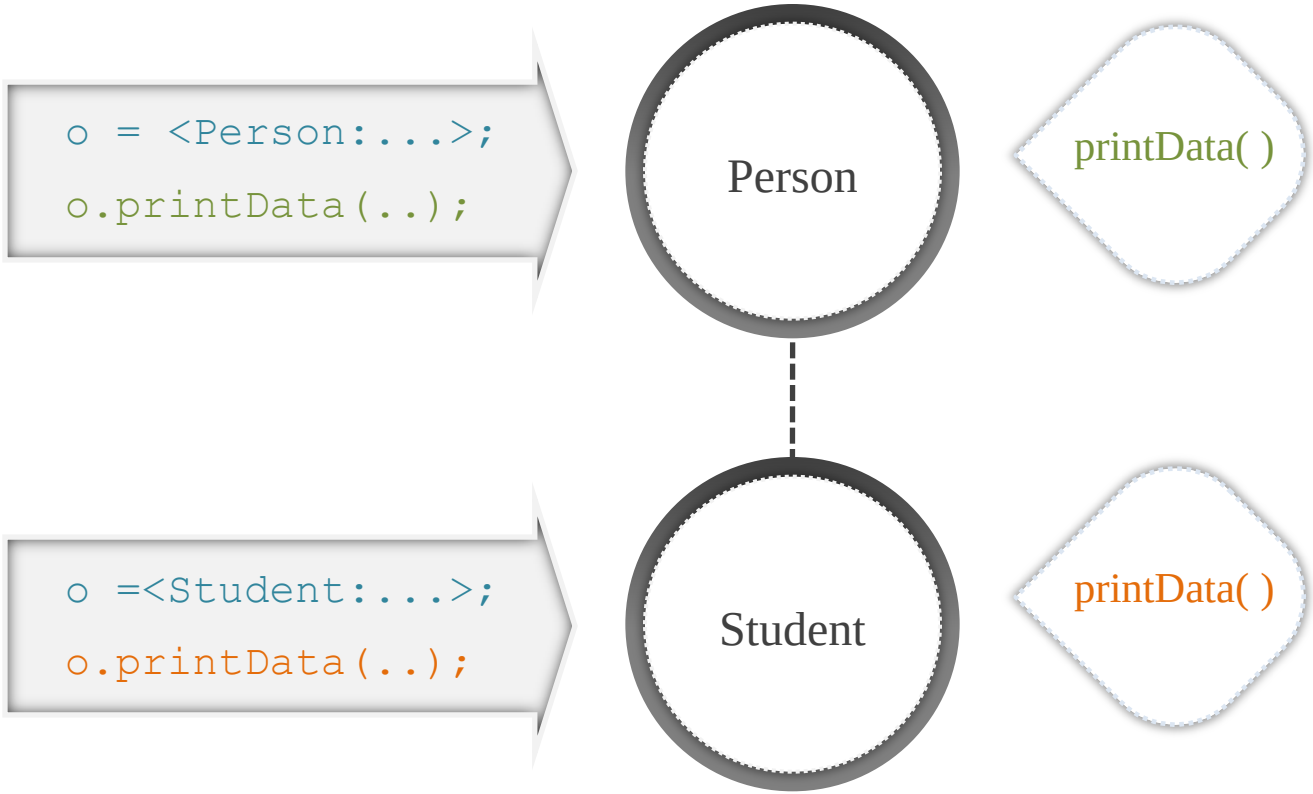
- 이러한 변수를 이용하여 오버라이드된 메소드를 호출하면?

예

```
Person p = <Student: 이름=홍길동, 주소=대전시 궁동, 학번=20810,
                학점=4.3, 연산={printData()..}>
// 객체변수 p는 Person 타입으로 선언,
// 지정되는 객체는 Student 클래스 (타입)의 사례
p.printData();          // 과연, 학번, 학점이 출력될까?
```

다형성이란?

- 메소드 호출의 모양은 같아도, 다른 메소드가 선택될 수 있는 성질



다형성 (Polymorphism)

메소드 호출과 바인딩

정적 바인딩 (Static binding) - C++

- 컴파일시: 어느 메소드를 호출할 지 결정, 코드 생성
  (변수의 타입 정보를 이용)
- 런타임 : 결정된 메소드 수행

동적 바인딩 (Dynamic binding) - Java, Smalltalk

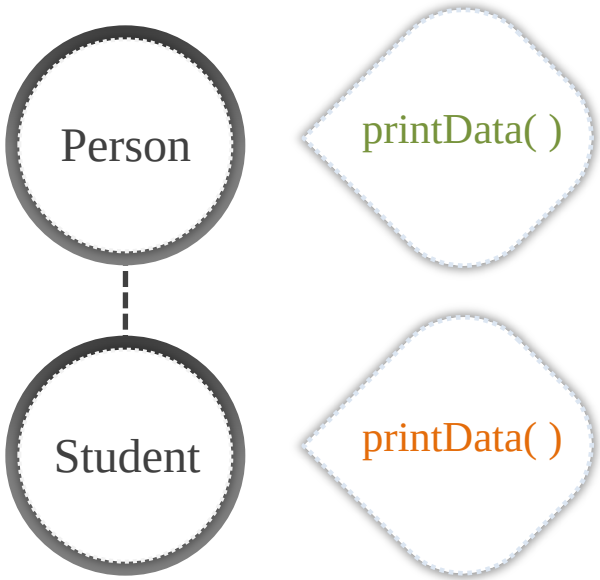
- 컴파일시 : 어느 메소드를 호출할 지를 계산하는 코드를 생성
  (객체의 타입 정보를 이용)
- 런타임 : 메소드를 결정하는 코드 수행 + 결정된 메소드 수행

→다형성!

다형성 (Polymorphism)

메소드 호출과 바인딩

```
Person p;
if (user-input)
        p = <Person:...>;
else
        p =<Student:...>;
p.printData();
```



정적 바인딩 (Static binding) - C++

- if 문이 참/거짓일 때 모두 : `p.printData()`는 `Person`의 것을 사용

동적 바인딩 (Dynamic binding) - Java, Smalltalk

- if 문이 참일 때 : `p.printData()`는 `Person`의 것을 사용
- if 문이 거짓일 때 : `p.printData()`는 `Student`의 것을 사용

객체 지향 프로그래밍 논점

모든 데이터 값을 객체로?

- 정수, 실수, 문자 등도 모두 객체화
  - → 성능 저하
- 기본 타입과 데이터 타입이 공존
  - → 두가지 타입시스템으로 복잡

모든 객체를 힢에 위치시킬 것 인지

- 변수가 객체를 참조하는 것을 구현하기에 일관성이 있어 편하고,
- 제거할 때 암묵적인 쓰레기 수집이 가능
- 수행 효율은 떨어짐

다중 상속을 허용할 것인지

- 다중상속은 편리하나 구현이 복잡

객체 지향 프로그래밍 논점 (계속)

상/하위 클래스가 상/하위 타입인가?

동적 바인딩 vs. 정적 바인딩

➔ 동적 바인딩은 편리하나 비용이 들고 정확한 타입 검사가 실행순간까지 미루어짐

상속 시 하위 클래스가 상위 클래스의 구현을 열람하는 것을 허용할지

허용 : 상위클래스가 구현을 수정을 하면 하위 클래스도 수정되어야할 수 있으므로 재사용성에 문제

불허 : 상위 클래스의 구현 내용을 감안하지 못하므로 효율성을 극대화 시킬 수 없음

평가하기

마지막으로 내가 얼마나 이해했는지를 한번 확인해 볼까요?

총 2문제가 있습니다.

START

평가하기 1

1. 다음 중 다형성과 관련된 설명으로 거리가 먼 것은?

① 변수의 타입만으로 그 변수가 가지고 있는 값이 어떤 클래스에 속하는 객체인지 판단 가능하다.

② 다형성을 제공하는 프로그래밍 언어에서는 메소드 호출의 모양은 같아도, 다른 메소드가 선택되어 수행될 수 있다.

③ 정적 바인딩은 컴파일시에 어느 메소드를 호출할 지 결정하여 코드를 생성한다.

④ 동적 바인딩은 런타임에 메소드를 결정하는 코드와 결정된 메소드를 모두 수행한다.

확인

평가하기 2

2. 객체지향 프로그래밍 언어 설계 시 결정 해야 할 사항으로 거리가 먼 것은?

① 상속시 하위 클래스가 상위 클래스의 구현을 열람하는 것을 허용할지 결정

② 모든 객체를 힢에 위치시킬 것인지 결정

③ 단일 상속을 허용할 것인지 결정

④ 모든 데이터 값을 객체로 할 것인지 결정

확인

정리하기

- **다형성**

메소드의 호출 모양은 같아도 다른 메소드가 선택될 수 있는 성질이다. 객체지향 언어에서는 상/하위 클래스 간에 같은 이름이나 오버라이드된 다른 정의를 가지는 메소드가 정의될 수 있으므로 다형성이 제공된다.

- **객체지향 언어 설계의 논점**

모든 데이터값을 객체로 할지, 모든 객체를 힢에 위치시킬것인지, 다중상속을 허용할 것인지, 상/하위 클래스를 상/하위 타입관계로 할 것인지, 동적바인딩을 할 것인지, 상위클래스의 구현을 열람하는 것을 허용할 것인지 등의 논점들이 있다.

"강의를 마치겠습니다. 수고하셨습니다."