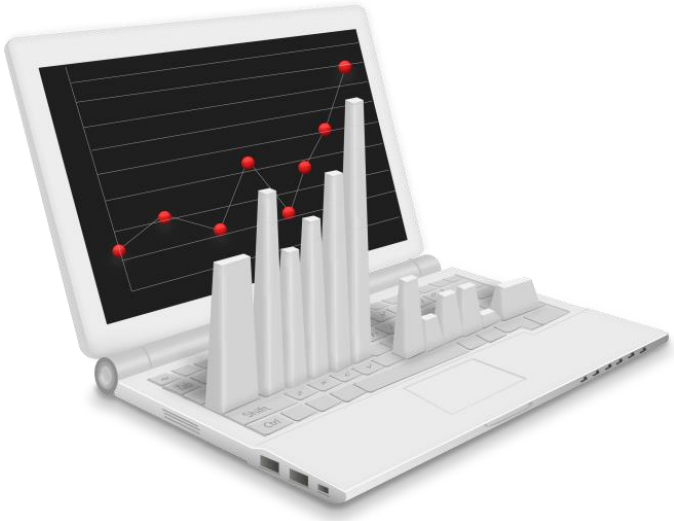
프로그래밍 언어론

객체지향 프로그래밍 개념

컴퓨터공학과

조은선



충남대학교
CHUNGNAM NATIONAL UNIVERSITY

객체지향프로그래밍 개념

학습목표

- 객체지향 프로그래밍의 기본 개념에 대해 학습한다.

학습내용

- 객체 지향 프로그래밍 개념
- 상속

목 차

- 들어가기
- 학습하기
  - 객체 지향 프로그래밍 개념
  - 상속
- 평가하기
- 정리하기

알고가기

추상 데이터 타입과 관련이 적은 것은?

(a) 정보 은닉의 단위가 된다.

(b) 함수와 데이터를 묶어 캡슐화하는 단위가 된다.

(c) Ada의 package나 C++, Java의 class가 여기 속한다.

(d) 외부로 구현과 관련된 부분은 노출하고, 인터페이스 부분은 감추는 기능이 있다.

확인

| 추상데이터타입의 한계

추상 데이터 타입

장점

- **Encapsulation Information hiding, 분리 컴파일..**
  → 구조화된 프로그래밍이 가능, 효과적인 소프트웨어 생산, 유지 보수에 유리

그러나

- **재사용성이 여전히 좋지만은 않고,**
- **실세계에서 자주 등장하는 상/하위 관계 표현이 미흡**
  (두 추상데이터타입은 평면적으로 대등한 관계를 가지고 독립적으로 존재)

"재사용성이 보다 좋고, 실제 세계를 더 잘 표현할 수 있는 개념이 필요"

객체 지향 개념(Object oriented concept)

추상 데이터 타입과 **비슷함**.(캡슐화)

- **추상 데이터 타입 자체를 클래스 (class) 라 하고**
- **클래스에 속하는 각 개체를 객체 (object) 라 하고,**
  해당 클래스에 대한 사례 (instance) 라 하고,
- **객체에 대한 연산 (부프로그램) 들을 메소드 (method) 라 함.**
  - 메소드는, 인터페이스를 나타내는 헤더 선언과 구현으로 나뉨

추상 데이터 타입과 **다름**

- **상속 (inheritance)**
- **다형성 (polymorphism)**
- **기타**

객체 지향 개념과 타입

클래스와 타입

- 클래스도 실세계의 대상을 표현하고, 공간의 효율적 사용과 타입검사를 지원할 수 있음
- 객체를 담는 변수 (이하, '객체 변수') 는, 그를 instance로 가지는 클래스의 타입을 갖게 됨

예

```
Person p = <Person: 이름=홍길동,
                    주소=대전시 궁동,
                    연산= {printData()..}>
```

- 추상데이터 타입과 마찬가지

상속 (inheritance)

상속 (inheritance)

- 두 개의 클래스가 가지는 부모 자식 관계

  예 사람-학생, 운송수단-기차, 재생도구-오디오재생도구

- 부모 클래스는 **상위 클래스 (super class)** , 자식 클래스는 **하위 클래스 (subclass)** 라 함.
- 하위클래스에 속하는 객체는 상위클래스에도 속하는 것으로 간주
  - 집합의 포함관계와도 비슷
  - 실세계에 많이 내재됨

상속 (inheritance)과 재사용성

- **하위클래스 정의 = 상위클래스 정의 + alpha**
  - 상위 클래스의의 모든 데이터선언과 메소드를 이어 받아야하며, 추가적으로 데이터 선언과 메소드들을 더 가질 수 있음

예 학생 = 사람 + 학번, 학점 ...

- **새로운 클래스를 만들때 유리**
  - 기존의 적당한 클래스로부터 상속받아 개발 시간 단축
  - 상위클래스의 내용을 완전히 모르더라도 상속 받을 수 있음
  - 개발을 레고 조립처럼 컴포넌트화 하기에 유리

상속과 상/하위 타입

상/하위타입

- 하위타입 객체는 상위 타입의 객체가 수행하는 모든 연산을 지원하여, 상위타입 객체를 기대하는 모든 위치에서 사용 가능

예 `Student` 는 `Person` 의 하위타입으로 설정 가능

클래스 상속의 재사용성은 상/하위 타입과 유사

- 하위클래스의 데이터구조와 메소드가 상위클래스의 데이터구조와 메소드를 모두 가지고 있음 → 상/하위 타입!
- 상위 클래스 타입의 변수가 하위 클래스 객체를 가질 수 있다.

예

```
Person p = <Student: 이름=홍길동, 주소=대전시 궁동,
                     학번=20810, 학점=4.3,
연산={printData()..}>
```

상속과 상/하위 타입 관계의 논점

- 데이터 구조와 메소드의 재사용이 된다고 반드시 타입의 상/하위 구조를 부여할 필요가 있을까?

  예 클래스 `Person` 에서 클래스 `Pet` 상속 받음

- 상/하위 타입으로 간주되는 경우는 반드시 동일한 데이터 구조와 메소드(구현)을 가져야만 할까?

  예 클래스 `Student`를 만들 때 클래스 `Person` 에서 상속받지 않았다면?

오버라이드 (override)

- 상위클래스가 제공하는 기존의 메소드의 구현을 그대로 상속하지 않고 자신에 맞게 변경해서 상속받는 것

예

클래스 Person의 printData()는 주민번호와 이름과 주소만을 출력, 클래스 Student는 이를 오버라이드하여 printData()에서 학번, 학점까지 더 출력하도록 할 수 있음

평가하기

마지막으로 내가 얼마나 이해했는지를 한번 확인해 볼까요?

총 3문제가 있습니다.

START

평가하기 1

1. 다음중 객체지향 프로그래밍을 위한 개념과 추상데이터 타입의 개념간의 공통이 아닌 것은?

① 객체지향 프로그래밍을 위한 클래스는 추상 데이터 타입과 같이, 많은 경우 데이터 타입으로 존재한다.

② 객체지향 프로그래밍을 위한 클래스와 추상 데이터 타입은 모두 대상 데이터에 대한 연산 (부프로그램)들을 포함하는 개념이다.

③ 객체지향 프로그래밍을 위한 클래스와 추상 데이터 타입은 모두 정보 은닉을 지원하는 개념이다.

④ 객체지향 프로그래밍을 위한 클래스와 추상 데이터 타입은 모두 상속을 지원하는 개념이다.

확인

평가하기 2

2. “( ) 클래스는 자신의 ( ) 클래스의의 모든 데이터선언과 메소드를 이어 받아야하며, 추가적으로 데이터 선언과 메소드들을 더 가질 수 있다” 에서 두 괄호에 들어갈 단어를 다음 보기중 고르시오.

(1) 상위-하위 (2) 하위 – 상위 (3) 상위-상위 (4) 하위--하위

확인

평가하기 3

3. Student 가 Person의 하위 클래스이고, s와 p가 각각 Student와 Person의 인스턴스라고 할 때 다음 중 타입 오류가 발생하는 것을 모두 고르시오.

(1) Person x = p; (2) Person x = s (3) Student y = p (3) Student y = s

확인

정리하기

- **객체지향 개념**

추상데이터타입과 유사하나 재사용성과 실제 세계를 잘 표현하는 개념으로 상속(inheritance)를 도입한 개념이다.

- **상속**

클래스간의 상/하위 관계를 정의하는 것으로 재사용성이 좋아지고, 새로운 클래스를 만들때 개발시간이 단축된다. 상/하위 타입 관계와 유사하나 반드시 같을 필요는 없다.

"강의를 마치겠습니다. 수고하셨습니다."