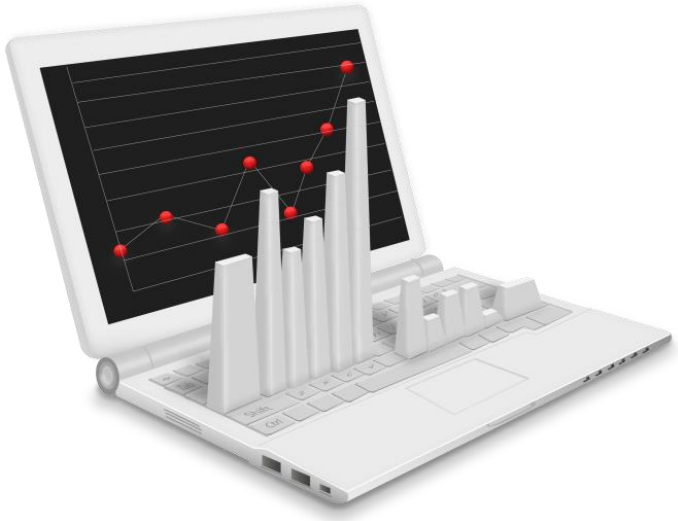
프로그래밍 언어론

객체지향 프로그래밍 언어의 예

컴퓨터공학과

조은선



충남대학교
CHUNGNAM NATIONAL UNIVERSITY

객체지향 프로그래밍 언어의 예

학습목표

- 다양한 객체지향 프로그래밍 언어의 예를 학습한다

학습내용

- SmallTalk
- C++
- Java

목 차

- 들어가기
- 학습하기
  - SmallTalk
  - C
  - Java
  - C#
  - Ada95
- 평가하기
- 정리하기

알고가기

추상데이터 타입과 객체지향언어의 클래스에 대한 설명으로 맞는 것은?

(1) 추상 데이터 타입과 객체지향언어의 클래스는 차이가 없다.

(2) 추상데이터 타입은 상/하위관계가 있지만 클래스는 아니다.

(3) 컴파일할때 어느 메소드를 호출할지가 결정되고 런타임에 그 결정된 메소드를 수행하면 동적 바인딩이다.

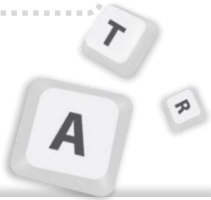
(4) 변수가 선언된 타입만으로는, 담고 있는 객체의 실제 타입을 정확히 알 수는 없다.

확인

Smalltalk

성격

- 객체지향 언어의 시초
- 순수 객체지향 언어
  - 모든 값은 객체이며 프로그램은 객체간의 메시지 교환으로 수행됨
  - 좀 느리다
- GUI 와 완벽한 라이브러리가 포함됨
- 타입이 없는 언어므로 동적 바인딩과 다형성은 극단적
  - 변수가 아닌 객체가 속한 클래스에 전적으로 의존하여 호출될 메소드가 결정됨



표현식 (Expression)

리터럴 (숫자, 스트링, 키워드..), 변수명 (모든 변수는 참조 타입),메시지, 블럭이 있음

메시지

- 단일인자연산 (*obj1 methodName1*)
- 이진연산 (`12 + 17`)
- 키워드 인자 연산 (`myArray at: 1 put: 5`)

메소드 정의

message_pattern [| temps |] statements

- message pattern 은 형식인자 선언과 동일, temps 는 메소드 이름
- 결과는 늘 객체

블럭

- 문장들의 리스트 (‘.’ 으로 문장 사이를 구분)

```
[index <- index + 1. sum <- sum + index]
```

- 단지 할 일을 '정의' 하고 있는 것

블록의 수행

- 단일인자 메시지 value에 인자로서 전달되어야함.

예
```
[…] value
```

- 변수에 지정된 후 수행도 가능

예
```
addIndex <- [sum <- sum + index]
        . . .
addIndex value
```

- 인자를 가질 수도 있음

예
```
[:x :y | statements]
```

Smalltalk의 문장(Statement)

지정문, 반복문, 선택문이 있음

지정문

```
index <- index + 1
```

반복문 (iteration)

whileTrue : 수행결과가 불린 값인 모든 블럭에서 정의된 메소드
결과가 참이면 인자로 주어진 블럭을 수행

```
[count <= 20]
    whileTrue: [sum <- sum + count.
                count <- count + 1]
```

timesRepeat : 모든 integer 객체에 정의된 메소드
그 integer 횟수 만큼 반복

```
xCube <- 1.
    3 timesRepeat: [xCube <- xCube * x]
```

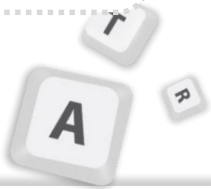
분기문

ifTrue: ifFalse : 모든 참거짓 객체에 정의된 메소드

```
[total = 0]
    ifTrue: […]
    ifFalse: […]
```

상속

- 단일 상속만 허용
- 상속시 상위 클래스의 구현을 모두 열람
- 상속시 재정의된 메소드는 기존 메소드와 별도로 취급됨
- 상속과 다형성은 별개
  - → 다형성은 상속뿐 아니라 이름 등이 동일한 모든 메소드 간에 형성



객체지향 언어의 대중화의 시초

- C의 모든 구조를 그대로 포함함
- 일반 타입들과 객체지향 개념이 융합된 타입시스템을 지원
- 속도 향상
  - 정적 바인딩을 기본으로 함

다중 상속 지원

동적 바인딩도 함께 지원

- 키워드 virtual로 메소드에 표시
  - 최상위 클래스 메소드에만 표시하면 됨

상속에도 접근제어를 도입

하위 클래스에서 수정될 가능성이 있는 메소드를 키워드 protected로 표시

- 이들 메소드는 하위클래스에서는 구현을 접근할 수 있으나 기타 클래스에서는 접근 불허

상속 자체에 public, private 키워드를 도입

```
class subclass_1 : public base_class { … };
class subclass_2 : private base_class { … };
```

- private 이면 ➔ 상위클래스의 어떤 데이터 구조도 접근 불가
- public 이면 ➔ public, private, protected 가 선언된 데이터구조나 메소드는 상속 후에도 그대로 유지됨 (default)

Java 의 특징

C++을 기반으로 함

보다 순수한 객체지향개념을 강화함

- 스택 기반 객체나 struct 등을 제거
- 동적 바인딩만을 지원, 다형성 강화
  - 기타 동적인 측면 강화로 다운로드 후 수행되는 인터넷에 적합하게 발전
- 기본 타입과 대응되는 클래스를 두고 변환 연산을 할 수 있도록 함

다수의 라이브러리 (API) 존재

- 언어에 내재된 것 + 다수의 사용자들이 제작한 것

인터페이스 (interface)

- 구현이 전혀 없이 메소드의 '선언'만으로 이루어진 타입
- 인터페이스 간의 상속등을 지원
- 관련 클래스와는 '구현(implements)' 관계를 유지

예 `Applet`은 상위 클래스, `Runnable`은 인터페이스일 때,

```
public class Clock extends Applet
                implements Runnable {..
```

  - 클래스 Clock 타입의 객체는 Applet이 정의한 모든 구현을 상속하고, Runnable이 선언한 모든 메소드를 구현하는 것을 약속함

- 만일 Clock에서 Runnable이 선언한 메소드를 구현하지 않은 부분이 있다면 오류
- 재사용성 극대화
  - 구현과 인터페이스의 완전한 분리
    - 구현을 전혀 몰라도 객체를 사용가능
    - 분산환경에서도 유리
- 인터페이스는 다중 상속, 클래스는 단일 상속
  - 구현이 간단하면서도, 동시에 모델링의 편리를 지원

평가하기

마지막으로 내가 얼마나 이해했는지를 한번 확인해 볼까요?

총 2문제가 있습니다.

START

평가하기 1

1. 다음 중 Smalltalk 과 관련된 설명으로 올바른 것을 고르시오.

① 클래스의 다중 상속을 허용한다.

② 클래스의 상속 시 상위 클래스의 구현을 열람할 수 없다.

③ 상속 관계의 클래스에 속하지 않더라도, 동일한 이름의 메소드들에 대해 다형성이 제공된다.

④ 메소드 수행의 결과는 숫자값 또는 객체이다.

확인

평가하기 2

2. 다음 중 Java의 인터페이스에 대한 설명으로 가장 거리가 먼 것은?

① 인터페이스에 정의된 메소드는 구현이 전혀 없이 선언만으로 이루어져있다.

② 각 인터페이스는 관련된 클래스가 존재할 수 있으며 implements라는 키워드로 연결된다.

③ 인터페이스 간에는 단일 상속만 허용되며 클래스 간에는 다중 상속이 허용된다.

④ 메소드의 body가 구체적으로 어떻게 구현되었는지 모르는 경우에도 인터페이스를 활용한다면 그 메소드를 호출하여 프로그래밍할 수 있다.

확인

정리하기

- **Smalltalk**

객체지향 프로그래밍 언어의 시초이며 모든 값이 객체이고 프로그램은 객체간의 메시지교환으로 수행되기 때문에 낯선 syntax를 가진다. Information hiding 개념이 없고, 동적 바인딩과 다향성이 제공된다.

- **C++**

C 언어가 주류를 이루었던 시점에서 객체지향 개념을 도입하기 위해 순수 객체지향 프로그래밍 기능들이 절충되어 도입되었다. 정적 바인딩과 동적바인딩이 함께 지원되며 접근제어, information hiding 기능이 추가되었다.

- **Java**

C++에 비해 동적인 면이 확대되고 인터페이스 등을 통해 재 사용성이 확대되었다. 다수의 라이브러리등과 함께 C++ 보다 심화된 객체지향성을 지원하고 있다.

"강의를 마치겠습니다. 수고하셨습니다."