Java – 객체지향 – 2

컴소과 동아리 REFERENCE

제작자: 김명준

목 차

상속

상 속

상속: 기존의 클래스를 재사용하여 새로운 클래스를 작성하는 것
두 클래스를 조상과 자손의 **관계**를 맺어주는 것

```
class Book {
    String name;
    int price;
}

class Java extends Book {

}
```

extends 키워드를 사용해 상속 관계를 맺어줌

조상: Book, 자손: Java

자손은 조상의 **모든 멤버를 상속받음**

(**생성자, 초기화블록 제외**)

즉, 자손의 멤버개수는 조상보다 적을 수 없음

상 속

상속을 하는 이유?

- 코드의 중복을 제거할 수 있음
- 유지보수하기 쉬워짐

조상의 변경은 자손에 영향을 미치지만,

자손의 변경은 조상에 아무런 영향을 미치지 않음

즉, 조상은 변경이 적게 설계해야함

상 속

포함: 클래스의 멤버로 **참조 변수**를 선언한 것

자바에선 **단일 상속**만 지원하기 때문에 나머지는 포함 관계로 맺음

```
class BookStore {
    Book book = new Book();
}
```

상속 관계: is-a(~은 ~이다)

포함 관계: has-a(~은 ~을 가지고 있다)

오버라이딩

오 버 라 이 딩

오버라이딩: 조상 클래스로부터 상속받은 메소드의 내용을

상속받는 클래스에 맞게 변경하는 것

오버라이딩의 조건

- 메소드의 선언부가 일치해야 함
- 접근 제어자는 조상 클래스의 메소드보다 좁은 범위로 변경 X
- 조상 클래스의 메소드보다 더 많은 예외를 선언 X

static 메서드는 오버라이딩 X

오버라이딩

super: 조상을 가리키는 참조변수

조상의 멤버와 자신의 멤버를 구별하는 데 사용

super(): 조상의 생성자를 호출할 때 사용

조상의 멤버들도 초기화되어야 함

Object 클래스를 제외한 모든 클래스의 생성자 첫 줄에는

this()나 super()를 호출해야 함

없다면 컴파일러가 자동적으로 super();를 첫 줄에 삽입

패키지

패키지

패키지: 서로 관련된 클래스와 인터페이스의 묶음

소스 파일에 첫 번째 문장에 단 한 번 선언 `package ex;`

모든 클래스는 반드시 하나의 패키지에 속해야 함

import문: 클래스를 사용할 때 패키지명을 생략할 수 있게 해줌

`import java.util.*;` java.lang 패키지는 import 하지 않아도 사용 가능

static import문: static 멤버를 사용할 때 클래스 이름을 생략 가능하게 해줌

제어자

제 어 자

제어자: 부가적인 의미를 부여

제어자의 종류

- 접근 제어자(단 하나만 사용 가능)
- 그 외 제어자(여러 개 사용 가능)

제 어 자

접근 제어자

제어자	같은 클래스	같은 패키지	자손클래스	전체
public	O	O	O	O
protected	O	O	O	
default	O	O		
private	O			

접근 제어자를 사용하는 이유

- 외부로부터 데이터를 보호하기 위해서
- 외부와 내부를 분리하기 위해서

제 어 자

제어자	대상	의미
final	클래스	변경될 수 없는 클래스, 확장될 수 없는 클래스가 된다. final로 지정된 클래스는 다른 클래스의 조상이 될 수 없다.
	메서드	변경될 수 없는 메서드. final로 지정된 메서드는 오버라이딩을 통해 재정의 될 수 없다.
	멤버변수 지역변수	변수 앞에 final이 붙으면 값을 변경할 수 없는 상수가 된다.

상수의 종류

- final static: 변하지 않고 공통적으로 쓰는 경우
- final: 변하면 안되지만 각각 달라야 하는 경우

다형성

다형성

다형성: 조상 타입의 참조 변수로 자손 타입의 인스턴스를 다룰 수 있는 것

참조 변수의 현재 타입에 따라 쓸 수 있는 멤버가 정해짐

참조변수는 서로 상속 관계에 있는 타입간의 형 변환만 가능함

형 변환할 때 주의점: 실제 인스턴스가 무엇인지

즉, 형 변환할 때 반드시 instanceof 연산자로 확인한 후에 형 변환

```
참조변수 instanceof 타입
```

다형성

멤버 변수는 참조 변수 타입에 영향을 받고

메소드는 실제 인스턴스 타입에 영향을 받음

Example.java를 실행

참조형 매개변수는 메소드 호출 시,

자신과 같은 타입 또는 자손 타입의 인스턴스를 넘겨줄 수 있음

- 다형적 매개변수
- 하나의 배열로 여러 타입의 객체를 다루기

추상 클래스

추상 클래스

추상 클래스: 추상 메소드를 포함하고 있는 클래스
생성자와 멤버변수, 메소드를 가질 수 있음 **But, 객체 생성은 X**

추상 메소드: 선언부만 있고 구현부가 없는 메소드

```
abstract 반환타입 메소드이름([매개변수]);
```

자손마다 다르게 구현될 것으로 예상되는 경우에 사용

추상 메소드는 **자손 클래스에서 오버라이딩으로 구현**

인터페이스

인터페이스

인터페이스: **추상 메소드의 집합**
추상 메소드와 상수만 멤버로 가질 수 있음

```
public interface 인터페이스_이름 {


}
```

클래스와는 달리 다중 상속을 지원

```
public interface 인터페이스_이름 extends 인터페이스_이름1, 인터페이스_이름2 {


}
```

인 터 페 이 스

인터페이스 구현: **implements** 키워드를 사용

```
public class 클래스_이름 implements 인터페이스_이름 {

}
```

구현할 때 주의점: **public을 필수로 지정해야 함**
인터페이스의 메소드들은 모두 **public abstract**

인터페이스도 다형성을 적용할 수 있음

- 매개변수: 해당 인터페이스를 구현한 클래스의 인스턴스를 받음
- 반환 타입: 해당 인터페이스를 구현한 클래스의 인스턴스를 반환

인터페이스

인터페이스의 장점

- 개발 시간을 단축 가능
- 표준화 가능
- 관계를 맺어줄 수 있음(다형성)
- 독립적인 프로그래밍이 가능

감사합니다

컴소과 동아리 REFERENCE