

2020-1

객체프로그래밍 실습

송인서

공과대학 IT미디어공학과 4학년(17)

songinseo0910@duksung.ac.kr

010-9610-9779

QnA 및 공지용  slack

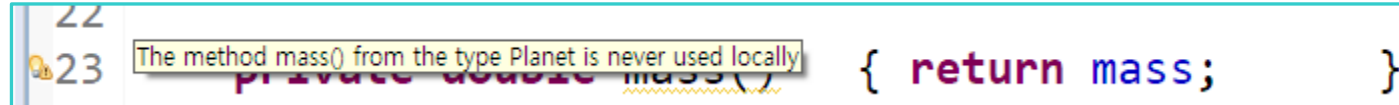
2020-1-dswu-it-java.slack.com



과제

- 폼에 다음 문제의 답을 입력해 제출

1. Enum 내에서 선언된 상수는 내부적으로 어떤 키워드를 사용해 정의되나?(3가지)
2. Java에서 클래스 다형성(Polymorphism)을 구현하기 위해 있는 두 가지 기능은?
3. Java의 클래스 계층도에서 최상위에 있는 클래스는?
4. 메소드 오버로딩을 위한 조건은?
5. 실습에서 다음 경고를 무시하기 위해 사용한 어노테이션은?



```
22  
23 The method mass() from the type Planet is never used locally { return mass; }
```

6. String[] args를 이터레이션하기 위해 사용한 구문은?
7. 메소드 오버라이딩 시 주의해야 할 점을 두가지 쓰세요.

Enum(열거형)

교재 134p 6.4 Enum 타입

- 레퍼런스 타입 객체
- 상수들의 집합을 관리할 때 사용
 - 요일, 영어 달 이름, 계절, 메뉴, 태양계 행성 등...
- 상수 키워드에 어떤 값이나 논리를 부여-> 상수를 클래스처럼 사용할 수 있게 해줌
 - Java의 enum은 기능이 매우 풍부한 편
 - 적절하게 잘 쓰면 코드의 가독성도 올라가고 유지보수도 쉬워짐

예제 - Planet

- 교재 137~138p 참조하여 파일을 작성해봅시다.(파일명 Planet.java로 생성)

	//mass	//radius	
<i>MERCURY</i>	(3.303e+23,	2.4397e6),
<i>VENUS</i>	(4.869e+24,	6.0518e6),
<i>EARTH</i>	(5.976e+24,	6.37814e6),
<i>MARS</i>	(6.421e+23,	3.397e6),
<i>JUPITER</i>	(1.9e+27,	7.1492e7),
<i>SATURN</i>	(5.688e+26,	6.0268e7),
<i>URANUS</i>	(8.686e+25,	2.5559e7),
<i>NEPTUNE</i>	(1.024e+26,	2.4746e7);

public static final double *G* = 6.67300e-11;

```

double surfaceWeight(double otherMass) {
    return otherMass * surfaceGravity();
}

public static void main(String[] args) {
    if (args.length != 1) {
        System.err.println("Usage: java Planet <earth_weight>");
        System.exit(-1);
    }
    double earthWeight = Double.parseDouble(args[1]);
    double mass = earthWeight/EARTH.surfaceGravity();
    for (Planet p : Planet.values())
        System.out.printf("Your weight on %s is %f\n", p,
            p.surfaceWeight(mass));
    }
}

```

args[1]을 args[0]으로 수정

만약에 175kg을 인수로 가지는 명령문으로 Planet.class를 실행하며 다음과 같은 결과를 얻을 것이다.

IDE 없이 컴파일해보기

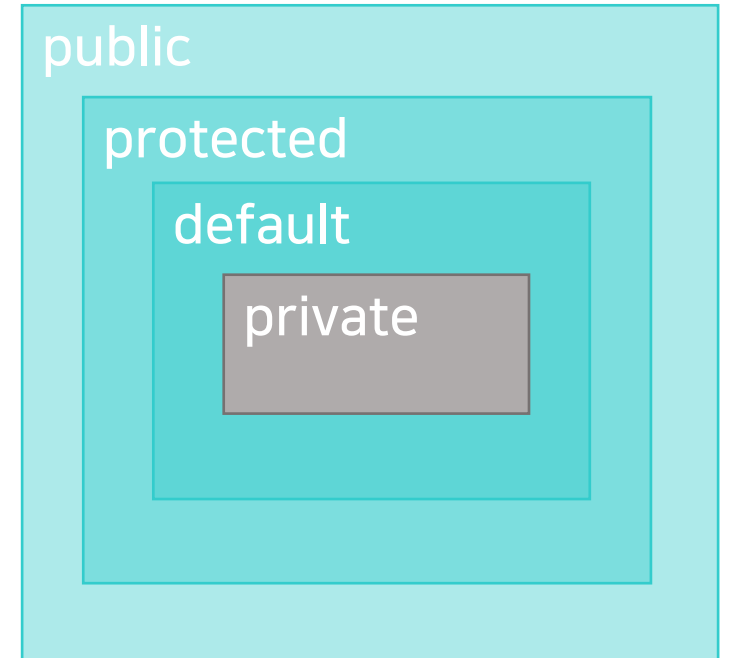
- cmd를 열어 javac 입력 후 enter를 눌러 명령어가 등록되어 있는지 확인
 - javac는 .java 파일을 컴파일해 .class 파일을 생성해주는 명령어(JDK에 포함)
 - 만약 javac 명령어를 인식하지 못하면 다음 블로그를 참고하여 환경변수를 수정할 것
<https://1duri1.tistory.com/250>
 - 환경변수 수정한 경우 프롬프트 재시작 한 후 확인

접근지정자

교재 167p

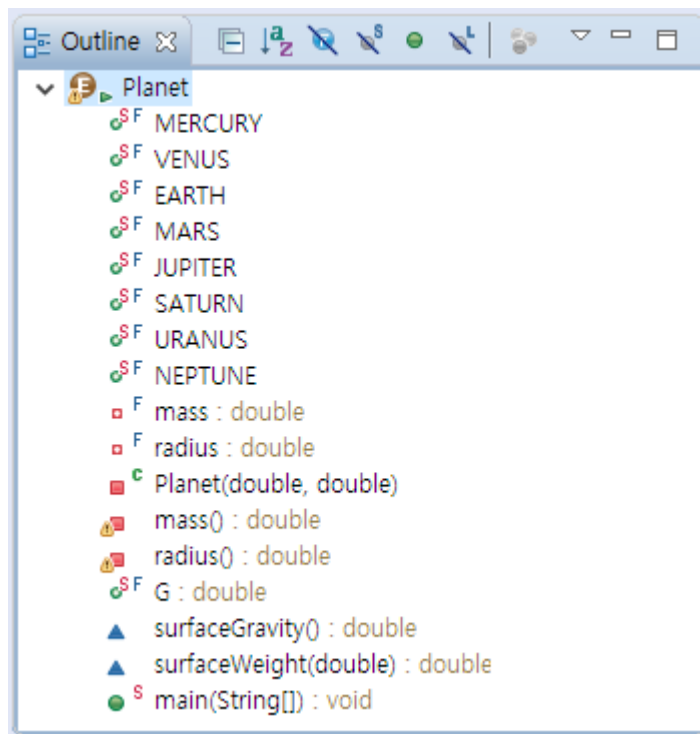
[표 7.1] 멤버 접근 지정자에 대한 액세스 제어

	해당클래스	해당패키지	해당클래스의 서브클래스	기타 (다른 패키지)
public	O	O	O	O
protected	O	O	O	X
*default	O	O	X	X
private	O	X	X	X



참조 – outline view

<https://stackoverflow.com/questions/1561336/what-do-the-icons-in-eclipse-mean>




























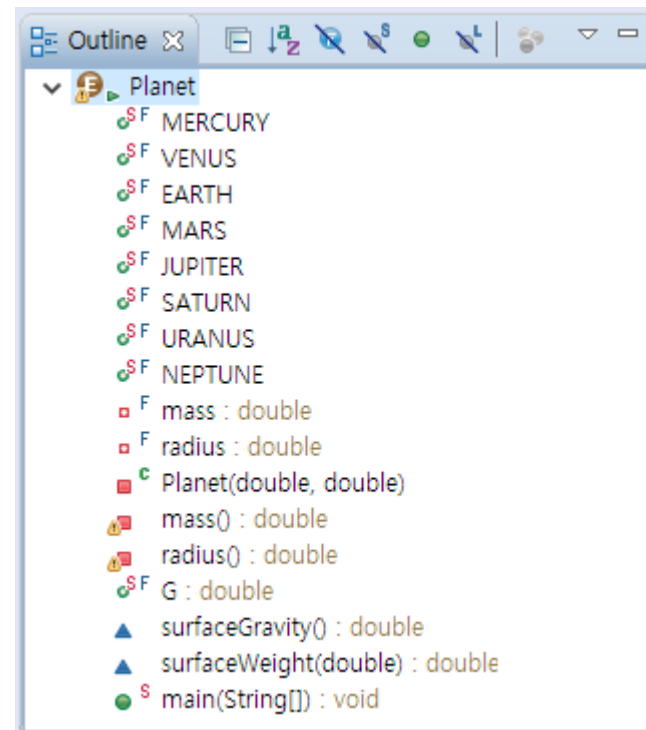
	class (public)
	interface (public)
	<u>enum type (public)</u>
	annotation type (public)
	package visible class
	private class
	protected class
	default field (package visible)
	private field
	protected field
	public field
	default method (package visible)
	private method
	protected method
	public method

참조 – outline view

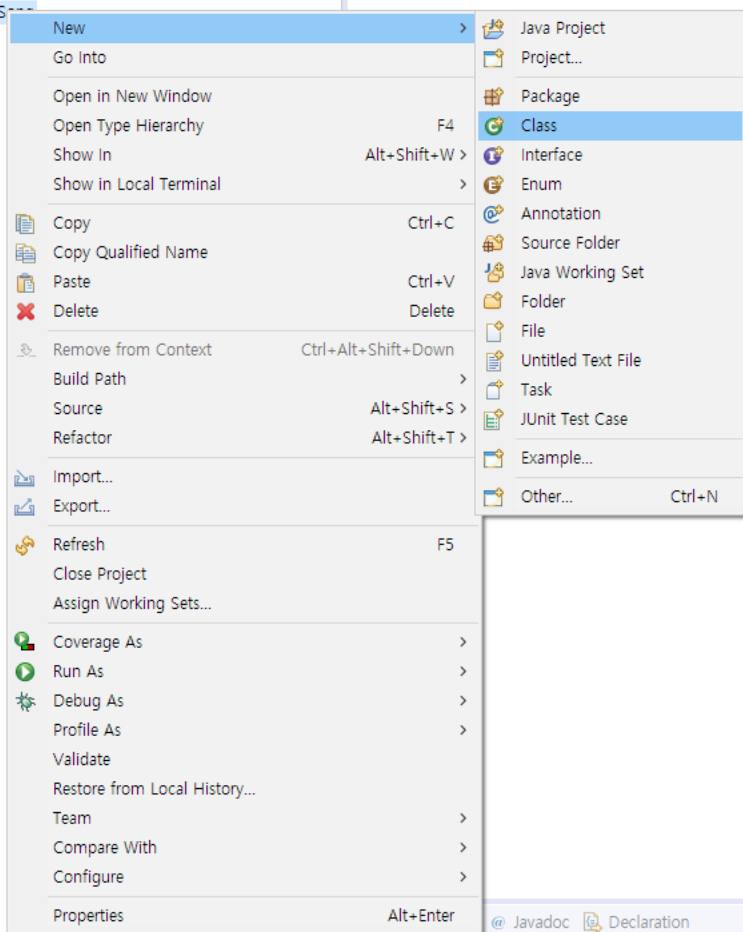
<https://stackoverflow.com/questions/1561336/what-do-the-icons-in-eclipse-mean>

Object adornments

	marks project as Java project		native method
	decorates files and folders if they are on the build path of their enclosing Java project		transient field
			volatile field
	decorates Java projects and working sets that contain build path errors		type with public static void main(String[] args)
	this Java element causes an error		implements method
	this Java element causes a warning		overrides method
	this Java element is deprecated		type with focus in Type Hierarchy or Quick Outline/Hierarchy
	constructor		maximal expansion level in Call Hierarchy
	abstract member		recursive call in Call Hierarchy
	final member		compilation unit containing an abstract class as primary type
	static member		compilation unit containing an interface as primary type
	default method		compilation unit containing an <u>enum</u> as primary type
	synchronized member		compilation unit containing an annotation as primary type



프로젝트 디렉토리에서 우클릭



New Java Class

Java Class

The use of the default package is discouraged.

Source folder: InseoSong/src Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name: Week1

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☒ public static void main(String[] args) 체크

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Finish Cancel

Week5

체크

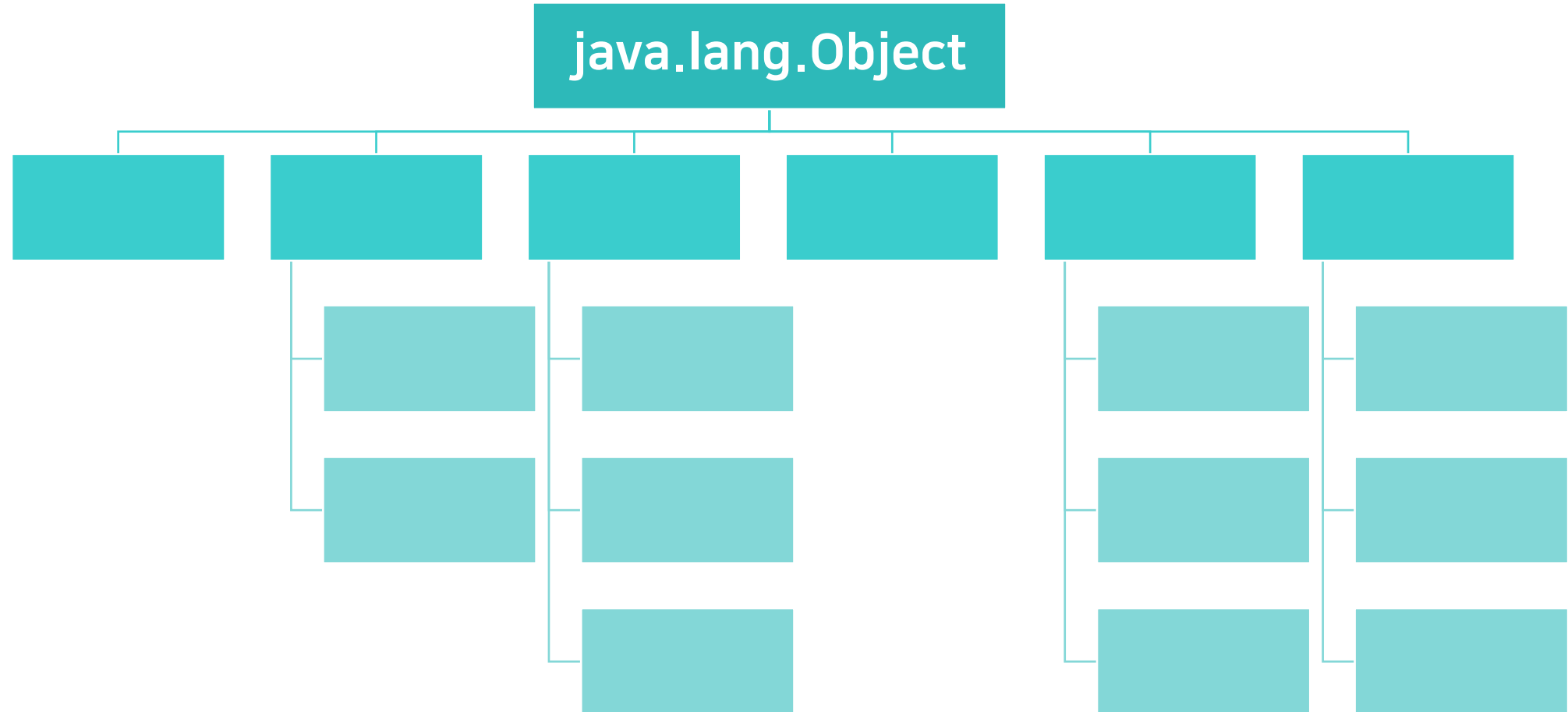
Updates Available

Updates are available for your software.
Click to review and install updates.
Set up [Reminder options](#)

상속(Inheritance)

- 다형성(Polymorphism)을 구현하기 위한 기능
- 객체 간의 공통된 필드, 메소드 등을 중복 선언하지 않고 재사용 할 수 있도록 해줌
- 키워드 extends 를 사용해 상속 관계 표현
 - 서브클래스 extends 수퍼클래스
- 자바의 모든 클래스는 java.lang.Object를 상속받음
 - java.lang.Object 는 모든 클래스의 수퍼클래스
 - 모든 클래스는 java.lang.Object의 서브클래스

상속(Inheritance)



메소드 오버라이딩(Overriding)

toString

```
public String toString()
```

Returns a string representation of the object. In general, the `toString` method returns a string that "textually represents" this object. The result should be a concise but informative representation that is easy for a person to read. It is recommended that all subclasses override this method.

The `toString` method for class `Object` returns a string consisting of the name of the class of which the object is an instance, the at-sign character '@', and the unsigned hexadecimal representation of the hash code of the object. In other words, this method returns a string equal to the value of:

```
getClass().getName() + '@' + Integer.toHexString(hashCode())
```

Returns:

a string representation of the object.

예제 - 메소드 오버라이딩(toString())

```
@Override  
public String toString() {  
    return width + "*" + height + "*" + lenght;  
}
```

예제 - 메소드 오버라이딩(equals())

```
@Override
public boolean equals(Object obj) {
    if(obj instanceof Box) {

        Box other = (Box)obj;
        boolean w = (this.width == other.width);
        boolean h = (this.height == other.height);
        boolean l = (this.lenght == other.lenght);

        return w && h && l;

    }else {
        return false;
    }
}
```