JAVA Beginning

백 성 애

1. 자바 개발 환경 구축

- 1)jdk설치-
- http://www.oracle.com/technetwork/java/javase/downloads/index.html
- - j2se : Standard Edition [Core and Desktop]
- - j2ee : Enterprise Edition [Servlet/JSP/EJB…]
- - j2me : Mobile

- 2) c:\Java\jdk1.7.0 : JDK 소프트웨어가 설치되는 루트 디렉토리

- jdk1. 7.0 -----bin : 컴파일러, 인터프리터 등
  기타 개발 및 실행도구들
- +-- demo: 데모 코드(샘플파일)들
- +-- include
- +-- include-old : Native Code를 위한
  C헤더 파일들
- +-- jre : 개발도구를 제외하고 실행에
  필요한 최소한의 환경만을 제공
- +-- lib : 실행파일들에 의해 사용되는
  파일들

1. 자바 개발 환경 구축

- 3)jdk환경변수 설정
- 제어판-시스템-고급-환경변수-새로만들기
- ㄱ)JAVA_HOME 설정 : c:\Java\jdk1.7.0
- ㄴ)path설정 : %JAVA_HOME%\bin;
- ㄷ)classpath설정 : .

2. 자바의 역사

- 1) 1991년 sun사 Green Project 출범
  –James Gosling을 주축으로 Oak라는 언어 개발
- [가전기기에서 사용할 목적] 하드웨어 독립적인 언어로 구상됨
- 2) 1995년 : sun사와 netscape사 협약
- 3) 1996년 : 자바 지원 netscape 2.0 발표
- 4) 1997년 : jdk1.1 발표
- 5) 1998년 : jdk1.2 발표
- 6) 2000년 : jdk1.3 발표
- 7) 최근 : jdk1.4/ 5.0 / 6.0/ 7.0/8.0버전 발표

3. 자바의 특징

- 1) 플랫폼 독립성 : JVM(Java Virtual Machine)이 해당 플랫폼마다 제공되어져, 이를 설치하면 어떤 운영체제에서 작성된 자바 파일이든지 동일한 실행을 제공한다.
- 2) 객체 지향언어 : 재사용성, 유연성, 프로그램 생산성 향상
- 3) 멀티 스레드 지원 : Thread는 Process보다 작은 단위로 동시 다발적으로 작업 수행이 가능.
- 4) 자동 메모리 관리
  -Garbage Collector(쓰레기 수집기)
- 5) 동적인 성능 확장 제공: Applet

4. 자바의 주석 처리 방법

- 1) // : 단문 주석
- 2) /* */ : 복문 주석
- 3) /** */ 문서화 주석
  -javadoc를 이용해서 API문서를 작성하고자 할 때 사용

5. 클래스 구조

- 1) 패키지 선언: 최상단에 위치
  import문 보다도 먼저 와야 한다.
- 2) import 문 : 사용하고자 하는 패키지 경로를 기재
- 3) class 선언 : class키워드로 선언하고 클래스 이름을 기재.
  이 때 주의. 클래스명==파일명

6. 클래스의 멤버

- 1) 변수

: 데이터를 임시적으로 저장하는 메모리 공간.
즉 변수란 값을 저장하는 메모리 공간의 위치를 의미.
값을 담는

2) 변수의 종류

ㄱ)멤버변수(instance 변수)

ex) int a=10;

**객체명으로 접근해야 한다.

ㄴ)클래스변수(static 변수)

ex) static int b=10;

**클래스명으로 접근해야 한다.

6. 클래스의 멤버

- 2) 생성자(멤버 변수의 초기화):
  객체를 생성할 때 호출된다.
  생성자 이름과 클래스 이름은 같아야 한다.
  반환타입이 없다.
- ex)

```
class Hello{
    public Hello(){
                a=20;
        }
 }
```

6. 클래스의 멤버

- 메소드 (method)
- 1)public static void main(String args[]){ }
  : 실행시 제일 먼저 JVM에서 호출해주는 메소드.
  프로그램 시작이자 끝이 된다.

- 2)사용자 정의 메소드

```
public int myFunction(int a){
            메소드가 하는 일
    this.a=a;
    return a;
}
public static void mySub(){
}
```

7. 변수의 명명규칙

- 영문자와 숫자를 섞어 쓸 수 있으나,
  숫자로 시작되어선 안된다.
- 한글/한자도 변수명으로 사용가능
- 특수문자는 변수로 사용할 수 없다.
  단, 언더바(_), $는 식별자로 사용 가능
- 변수명은 명사형으로 지으며, 소문자로 시작.
- keyword는 사용 불가
- **keyword->교재 참조

※ 잘못된 변수 선언의 예

- int 9nine : 숫자로 시작 불가
- int hey&bar: &라는 특수문자 사용 불가
- int char : 예약어는 사용 불가

- 다음변수는?
  - int 변수=10;
  - int $$$=20;
  - int _myVar=30;

  ---> 모두 사용 가능.

8. 자바의 자료형

```
1) Primitive Type : 기본 자료형
2) Reference Type : 참조형 ex) String s="Hi";
          |                    String s=new String("Hi");
          |
          +--ㄱ) 클래스형
          +--ㄴ) 인터페이스형
          +--ㄷ) 배열
```

```
1) Primitive Type
    |
    + ㄱ) 수치형--정수형----byte
    |              |   +------short
    |              +------int
    |              |   +------long
    |              |
    |              |-실수형
    |                  +-----float
    |                  +-----double
    |
    +ㄴ) 문자형 - char    : '가' 'A' '\u0000'
    |                       0~ 65535[16비트]
    +ㄷ) 논리형 - boolean : true, false
```

9. 자바 연산자 종류

1) 분리자 : . [] () ; ,

2) 단항 연산자: 항이 하나인 연산자
   ㄱ)증감연산자 : ++ --
   ㄴ)부호연산자 : + -
   ㄷ)비트별 NOT연산자 : ~
   ㄹ)논리 부정 연산자 : !

3) 산술 연산자 : * / % + -
4) 쉬프트연산자 : << >> >>>
5) 비교 연산자 : < <= > >= instanceof
6) 비트 연산자 : & ^ |
7) 논리 연산자 : && ||
8) 조건 연산자 : ?:
9) 할당 연산자 : = += *= /= -=
   <<= >>= >>>= &= ^= |=

10. 자바의 제어문

주 제어문	보조 제어문
1)조건문 : if, if~else, if~else if~else 2) switch~case문 3) 반복문 - for문 - while문 - do~while문	1)break 문 2)continue 문 단독으로 쓰이지는 못하고 주제어문과 함께 사용된다.

11. Wrapper 클래스

Primitive Type(기본자료형)	Reference type(참조형)
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

기본자료형을 마치 랩으로 포장해놓은 것 같다하여 래퍼 클래스라고 함.
기본자료형은 단순한 연산에 사용되지만 래퍼 클래스는 참조형이므로 변수와 다양한 메소드등을 가져 많은 기능을 수행한다.

12. 배열(Array)

- [1] 배열이란?
  ..동종의 데이터들을 묶어 저장해놓은 자료구조
  - 비슷한 구조의 것을 하나의 데이터 구조에 번호를 매겨 저장하는 방식을 의미.
  - 이런 방법은 데이터의 저장, 정렬, 검색을 매우 유용하게 할 수 있어 편리하다.

[2] 배열 사용 방법

1) 선언
2) 메모리 할당
3) 초기화

12. 배열(Array)

- -1차원 배열
- 데이터형 배열명[]=new 데이터형[배열의 크기];
- -2차원 배열
- 데이터형 배열명[][]=new 데이터형[배열의 크기][배열의 크기];

- 예1)
- int a[]; //1)배열 선언
- a=new int[2]; //2)메모리 할당
- a[0]=10; //3) 초기화
- a[1]=20;

12. 배열(Array)

- 예2) 선언과 메모리 할당을 동시에 하는 방법
- int b[]=new int[3]; //1)+2)
- b[0]=100; //3) 초기화
- b[1]=200;
- b[2]=300;
- b[3]=400; [x] //배열 index초과 오류 발생
- 예3) 선언,메모리할당, 초기화를 한꺼번에 하는 방법
- int [] c={1,2,3,4,5};

12. 배열(Array)

- **배열에 저장된 값을 꺼내오고자 한다면...
- 그때는 index를 이용해 꺼내온다.
- index는 0부터 시작.
- 이때 주의. 인덱스가 배열 크기를 벗어나지
- 않도록 주의.
- 예) System.out.println(a[0]);

12. 배열(Array)

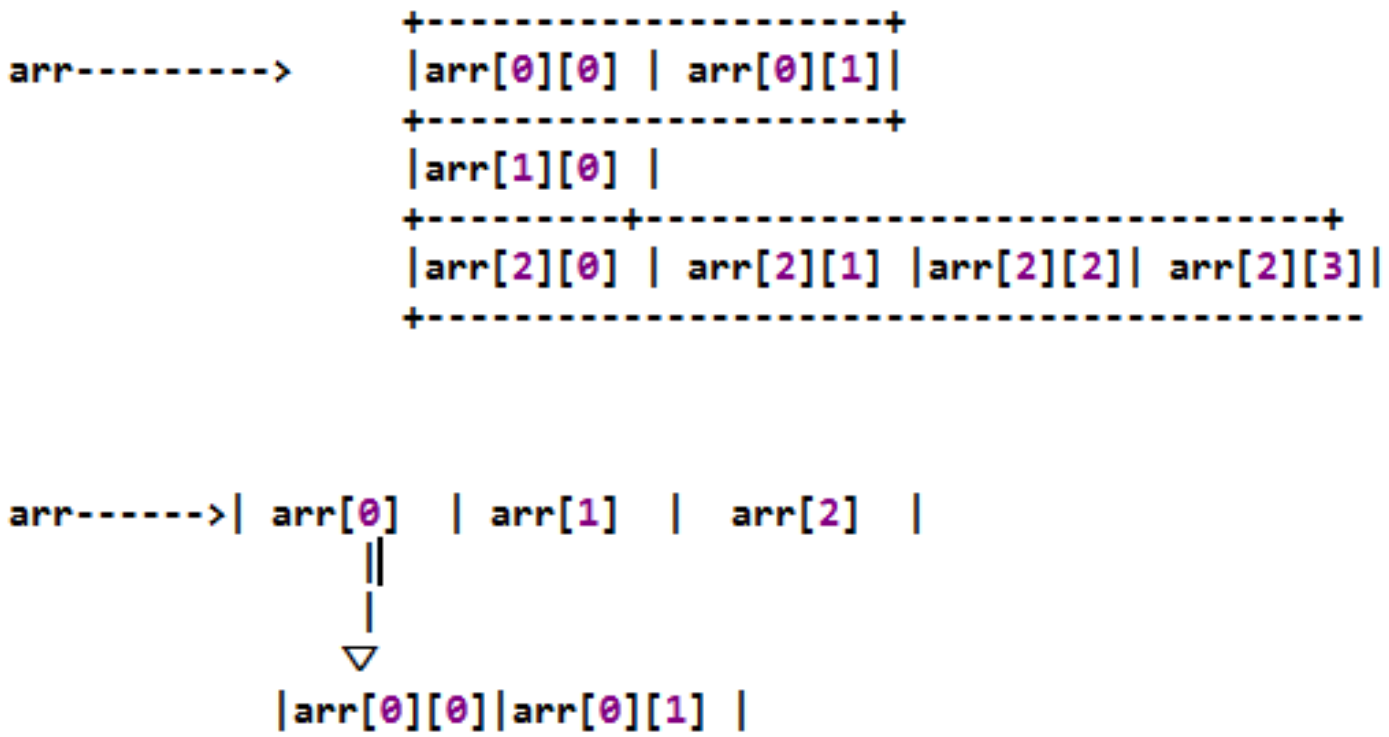
- [3] 다차원[-2차원] 배열 사용 방법
- 1) 선언
- int arr[][];
- int [][] arr; int []arr[];
- 2) 메모리 할당-배열 생성
- arr=new int[3][2]//3행 2열
- 3) 초기화
- arr[0][0]=1;
- arr[0][1]=2;
- arr[1][0]=3;
- arr[1][1]=4;
- arr[2][0]=5;
- arr[2][1]=6;

12. 배열(Array)

- 예1) 선언과 동시에 생성하고 초기화
- int arr[ ][ ]={ {1,2,3},{10,20,30} };

- 예2) 이차원 배열의 경우, 배열을 생성할 때
- 행의 크기를 고정시키고, 각 행에 대한
- 열의 크기를 가변적으로 줄 수 있다.
- int []arr[]=new int[3][ ];
- //행의 크기를 3으로 고정. 열의 크기는
- // 나중에 할당.
- **열의 크기 할당 방법**
- arr[0]=new int[2];
- arr[1]=new int[1];
- arr[2]=new int[4];

12. 배열(Array)


arr--------->
arr[0][0] | arr[0][1]
arr[1][0]
arr[2][0] | arr[2][1] arr[2][2] | arr[2][3]
arr------>| arr[0] | arr[1] | arr[2] |
arr[0][0]|arr[0][1]

13. Object Oriented Programming

OOP란?

1. 구조적 프로그래밍, 절차지향 프로그래밍

   ex] C 프로그래밍

   : 일을 처리하는 순서와 과정을 프로그래밍으로 구현한 것.

   - Procedure, Process를 중시함
   - 순서, 과정이 달라지면 새로운 작업 모델이 필요함
   - 재사용성 불가

   반면 OOP는

   프로세스 중심이 아닌 객체 중심으로 프로그래밍 하는 것.

   -모듈화, 재사용이 좋다.

13. Object Oriented Programming

2. Object Oriented Programming

ex]C++, Java 프로그래밍

: 인간의 현실세계를 프로그램에 반영한 것. 즉, 현실세계에 존재하는 object(객체,물체) 개념을 Program에 반영한 것

1) object : 물건, 물체를 의미.

유무형의 물체.

ex) 집, 사람, 컴퓨터

정치, 경제, 사회, 공기...

13. Object Oriented Programming

2) object의 특징: 객체는 속성과 행위(행동양식)를 갖는다.

따라서 프로그램에 객체를 반영할 때는
먼저 속성과 행동양식을 뽑아내는 과정이
필요한데...이를 객체 모델링이라 한다.

-OOP 순서

1) 프로그램에 필요한 객체를 뽑아냄
2) 객체 모델링
3) 클래스 구성
5) 객체 생성 및 사용

13. Object Oriented Programming

- 3. OOP의 주요 특징

- 1. 추상화
- 2. 은닉성(캡슐화)
- 3. 다형성
- 4. 상속성

- – object를 프로그램에 반영하는 작업을 추상화라고 함

13. OOP - 추상화

- 1. 추상화(Abstraction)란?
- ...어떤 물체(object)에서 주된 특징을 부각시켜
- 표현하고, 나머지 부분은 과감하게 생략하는 것

- OOP에서 사용되는 추상화도 이와 비슷하다.
- 한 물체를 대표하는 속성(명사)과 기능(동사)
- 를 추출해내는 것을 프로그래밍에서는 추상화
- 라고 한다.
- ex) 집을 프로그래밍으로 추상화해보면...
- House
- |-속성(attribute): 방수, 주인이름, 지붕색...
- +-행위(behavior) : 세를 놓다. 수리하다.
- 청소하다. 사다. 팔다....
- 위의 속성은 멤버변수로...
- 행위 또는 기능은 메소드로 표현한다.

13. OOP – 추상화

```
358     위의 집이란 객체를 자바 프로그램에 추상화해보면
359     class House
360     {
361         int room;
362         String ownerName;
363         String addr;
364
365         public void existAt(String addr){
366             System.out.println(addr+"에 위치하다");
367         }
368     }
369     class HouseTest
370     {
371         public static void main(String args[]){
372             House h=new House();
373             h.existAt("300번지");
374
375             House h2=new House();
376             h2.existAt("100번지");
377
378
379         }
380     }//////////////////////////
```

13. OOP – 캡슐화

2. 은닉화(Encapsulation)
   - data를 캡슐화하고
   - data에 접근할 때는 메소드로...
     [setXXX()/getXXX()]

```
class House{
    private int room;
     public void setRoom(int r){
          room=r;
     }
     public int getRoom(){
            return room;
     }
}
```

13. OOP-다형성

3. 다형성(Polymorphism)
: 여러 가지 형태를 가질 수 있는 성질을 의미.
한 타입의 참조변수로 여러 타입의 객체를 참조할 수 있도록 함으로 써 다형성을 프로그램적으로 구현해 놓은 특징이 바로 다형성이다.

1) 오버로딩(Overloading)
[생성자 오버로딩, 메소드 오버로딩]
...메소드 이름을 동일하게 주되
매개변수의 데이터 타입과, 갯수, 순서를 다르게 주어서 구성하는 것
: 중복정의/ 다중정의

13 OOP-다형성 (Overloading)

- 오버로딩 조건

- 오버로딩하려는 메소드 이름이 같아야
- 메소드의 매개변수의 데이터형이 다르거나, 갯수가 다르거나, 순서가 달라야 한다.
- 메소드의 반환타입은 신경 안써도 됨
  (같아도 되고 달라도 됨)

13 OOP-다형성 (Overriding)

2) 오버라이딩(Overriding)

…상속 개념과 맞물려 사용

부모로부터 상속 받은 메소드를

재정의해서 사용하는 것

: 재정의

13 OOP-다형성 (Overriding)

오버라이딩 조건

- 오버라이드 하려는 메소드가 부모 클래스에 존재해야 한다.
- 메소드 이름이 동일해야 한다.
- 메소드의 매개변수 갯수, 데이터타입이 같아야 한다.
- 메소드의 반환타입도 같아야 한다.
- 메소드의 접근 지정자는 부모클래스와 동일하거나, 접근 범위가 넓어야 한다.
- Exception의 경우 부모 클래스의 메소드와 동일하거나 더 구체적인 Exception을 발생시켜야 한다.

13. OOP-생성자 Overloading

Overloading

1)생성자 오버로딩

-생성자란?

: 객체가 생성될 때 최초로 실행되는메소드를 의미.

-생성자 구성시 유의할 점

a) 생성자 이름은 클래스명과 동일하게

b) 반환타입을 가져선 안된다.

13. OOP-생성자 Overloading

-생성자의 주요 역할
  : 멤버 변수를 초기화 하는 일
  -사용자가 생성자를 구현하지 않았을 경우
  ->컴파일러는 default생성자를 제공해줌

  ※ 그러나 사용자가 생성자를 하나라도 구현했다면, 그 때는 컴파일러가 제공해주는 기본생성자는 사라진다.

-자바에서는 생성자를 다양하게 오버로딩 함으로써 다양한 초기값을 부여하고 있다.

13. OOP-생성자 Overloading

2) 생성자 안에서 this()의 사용

-this()는 자기 자신의 생성자를 호출하는 메소드

-한 클래스 안에 여러 개의 생성자가 오버로딩된 형태로 존재하고, 그 기능이 유사할 때, this라는 키워드를 이용해서 자기 자신의 다른 생성자를 호출할 수 있다.

13. OOP-생성자 Overloading

- 이 때 주의할 점

ㄱ)this()는 생성자 안에서만 호출해야 한다.

ㄴ)this()를 호출할 때는 반드시 생성자의 첫 번째 문장이어야 한다.

ㄷ)또한 생성자 안에서 this()와 super()를 함께 쓸 수 없다.

13. OOP – this의 사용

※ this 의 사용**********

1) this.변수 : 자기클래스의 멤버변수(인스턴스 변수)를 접근할 때 사용

2) this.메소드: 자기 클래스의 멤버 메소드를 접근할 때 사용

3) this() : 자기 자신의 생성자 호출 시 사용

*** this 라는 키워드는 static 메소드안에서는 사용할 수 없다.

****

13. OOP – super의 사용

※ super 의 사용**********

1) super.변수 : 부모클래스로부터 물려받은 변수
2) super.메소드: " " 메소드
3) super() : 부모클래스의 생성자
..super()역시 생성자 안에서만 호출 가능하며, 생성자의 맨 첫줄에 위치 해야 한다.
super 라는 키워드도 static 메소드 안에서 사용 불가.
super() 는 부모클래스에 생성자가 오버로딩된 형태로 여러 개 존재할 때 그 중에서 어떤 생성자를 호출할 지 결정할 수 있다.

**그러나 super()생성자를 사용자가 명시적으로 호출하지 않는다면, 컴파일러는 자식클래스 생성자에서 super()의 디폴트 생성자를 자동으로 호출한다.

ex)

```
class Parent
{
        String name;
        public Parent(String n){
                name=n;
        }//인자 생성자----
}//////////////////
  class Son extends Parent
  {
        public Son(){
                super("아무개");
                //만일 위 문장이 없다면 에러발생
                //컴파일러가 super()를 자동호출
                //하므로...
        }//---------
  }//////////////////////
```

[실습문제]
좌표를 나타내는 Point클래스를 설계해보자.
1. 클래스이름: Point
2. 멤버변수: x, y좌표를 기억시킬 변수
3. 2의 멤버변수를 캡슐화한다.
4. 캡슐화한 변수에 접근할 set/get계열 메소드를 구성한다.
5. Point클래스의 생성자를 구성한다.
...3가지 형태로 오버로딩해보자.
6. this()를 이용해서 멤버변수 값을 초기화하자.
7. 메소드 구성:
1) Point클래스의 x, y좌표값을 증가 감소시켜주는 메소드
2) x와 y값이 같은지를 비교하는 메소드
를 구성해보자.
8. main()메소드를 갖는 PointTest클래스를 만들어
Point객체 생성해 자기가 구성한 메소드를 호출해보자.

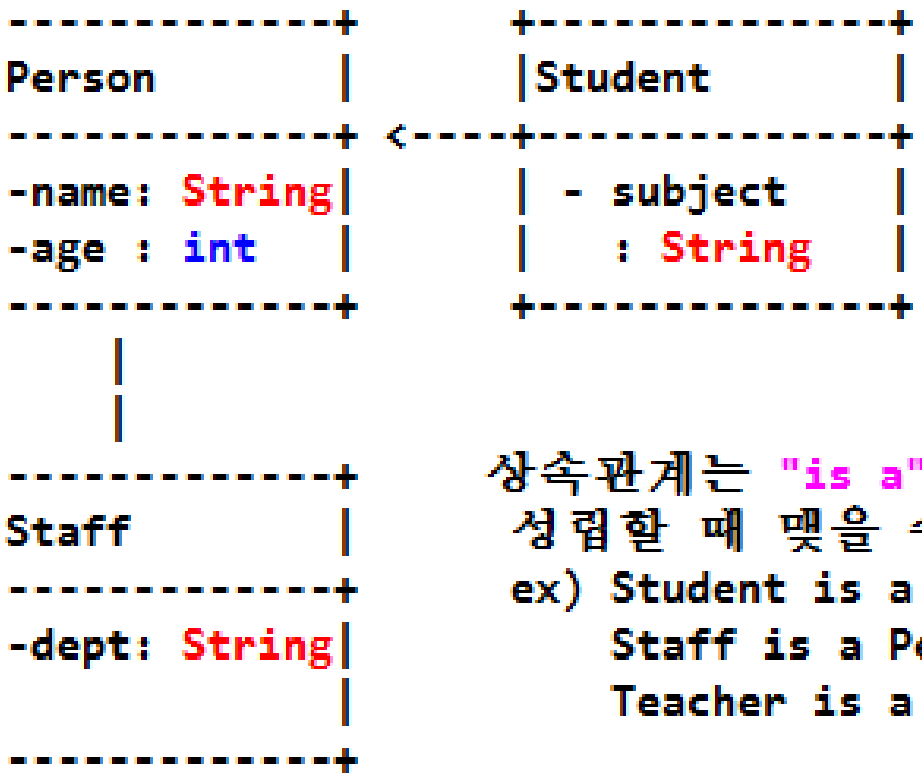
13. OOP - 상속성(Inheritance)

기존 클래스에 작은 기능이나 특성을 추가하여
새로운 클래스로 만드는 것을 의미.

즉 부모클래스를 만들고, 그 부모클래스에
있는 속성과 기능을 자식클래스에서 상속받아,
새로운 기능과 속성을 추가하는 것.

-상속 개념을 적용함으로써 개발시간 단축,
재사용성 등에 놀라운 장점이 있다.

13. OOP – 상속성(Inheritance)



**자바에서 상속을 받을 때는 extends 란 키워드를 사용한다.
자바는 단일 상속 개념이므로 extends 로 상속받을 수 있는 클래스는 단 하나뿐이다.

14. 패키지 (package)

패키지(package)란?

1) 클래스들 + 인터페이스들의 집합

2) 패키지를 사용하는 이유

i) 관리의 용이성

ii) 배포의 목적

3) 패키지는 유사한 기능을 가진 클래스와 인터페이스를 묶어 관리하도록 하며,
개발이 끝난 후 jar파일로 묶어 배포한다

14. 패키지 (package)

4) 패키지 구성

```
package 패키지이름;
 import java.util.*;
 class Test
 {
 }
```

패키지 선언은 최상단에 위치해야 한다.

-->import 문 보다도 먼저 와야 함.

14. 패키지 (package)

```
5)
-예제 1 의 구조--> AAA.java/CCC.java 파일 참조
myjava/day11[ROOT]
    |
    +--myPack : 패키지
        +----------AAA.java[AAA.class,BBB.class]
        +----------CCC.java[CCC.class, DDD.class]

 - 예제2의 구조 [import문을 익히기 위한 예제]
>myjava/day11[루트]
         |
         +--pack : 패키지
              + demo: 하위 패키지---Demo.java
              +----------------------My1.java
              +----------------------My2.java
         |
         +----TestPack.java [루트에 존재]
```

14. 패키지 (package)

-예제3의 구조[접근지정자 관련..]

```
myjava/day11[ROOT]
      |
      +--myPack : 패키지
      |     +----------AAA.java[AAA.class,BBB.class]
      |     +----------CCC.java[CCC.class, DDD.class]
      |
      +--yourPack: 패키지
            +----------FFF.java
```

:위 예제는 yourPack이란 패키지에 있는 FFF클래스에서 myPack의 AAA클래스와 BBB 클래스의 객체를 생성해 접근하려 할 때의 예제이다.

AAA클래스의 접근지정자=> public

BBB클래스의 접근지정자=> 생략형
:같은 패키지일 때만 접근 가능

14. 패키지 (package)

Access Modifier [접근지정자]의 접근 범위******

Modifiers	Same Class	Same Package	Sub Class	Universe
public	yes	yes	yes	yes
protected	yes	yes	yes	
default	yes	yes		
private	yes			

14. 패키지 (package)

-예제3의 구조[접근지정자 관련...]
..-d옵션을 주어서 컴파일 해본다.

```
myjava/day11
        |
        +--linux
        |     +---java:패키지
        |           |
        |           +--Parent.java
        |           +--Son1.java
        |
        +---windows
              +----java:패키지
                    +--Son2.java
```

15. 추상클래스 (Abstract Class)

- 한개 이상의 추상 메소드를 가지는 클래스

- 추상메소드(abstract method)란?
  ...메소드 몸체(body)없이 선언만 하는 것
  이 때 메소드 앞에 abstract 란 modifier를 붙여 준다.

ex) abstract public void sub();

- 추상메소드를 한개라도 가진 클래스는 역시 class 앞에 abstract를 붙여주어 추상클래스로 만들어야 한다.

15. 추상클래스 (Abstract Class)

- 추상 클래스를 상속받은 클래스에서는 추상 메소드를 강제적으로 오버라이딩해야 한다. 그렇지 않을 경우 그 자식 클래스도 추상 클래스가 되어야 함.

- 추상 클래스는 타입선언은 할 수 있으나 new 해서 객체 생성은 할 수 없다.
...반드시 상속을 통해서만 완성됨.
상속받은 concrete 클래스로는 객체 생성이 가능하다.

15. 추상클래스 (Abstract Class)

일반 클래스의 멤버 (concrete class)	추상 클래스의 (abstract class)	인터페이스 (interface
1. 멤버변수 2. 클래스변수 3. 생성자 4. 일반메소드 5. 클래스메소드 6. final변수 등	1 +2 +3 +4 +5 +6 가 다 멤버로 들어가고 이에 추가로+ 7.추상메소드가 추가됨	1.final변수와 2. 추상메소드로만 구성됨

16. final 지정자 (modifier)

final 지정자(modifier)

-final 지정자는 abstract 와 반대 개념으로 이해하자.

	abstract	final
클래스	상속받게 할 목적	상속받지 못하도록
메소드	강제로 오버라이딩 시키려는 목적	오버라이딩을 못하게
변수	x	값할당 못하게 상수로 만듬 final public static을 함께 쓴다.

17. 인터페이스 (Interface)

모든 메소드가 추상메소드이고,
모든 속성이 상수(final 변수)로 구성된 틀
: 멤버가 추상메소드+ 상수로만 구성됨.

- 인터페이스를 상속받을 때는 implements 란 키워드를 사용한다.

- 인터페이스는 다중 상속이 가능

17. 인터페이스 (Interface)

- 인터페이스에 있는 추상메소드는 abstract 란 키워드를 생략한다...너무나도 당연하기에

- 인터페이스의 변수 또한 public static final 을 생략해도, 컴파일러가 자동으로 붙여준다.

- 인터페이스 구현 방법----------

```
interface MyInter{
    void sub();
    //추상메소드---public과 abstract를
    //                생략해도 자동으로 붙는다.
}
```

---

17. 인터페이스 (Interface)

```
class MyClass implements MyInter
  {
        //인터페이스를 상속받으면 반드시
        //추상메소드를 오버라이딩 해야 한다.
        public void sub(){
              System.out.println("sub()");
        }
  }
```

17. 인터페이스 (Interface)

- 인터페이스가 인터페이스를 상속받을 때는 extends라는 키워드를 사용하며,
  이 때 extends 로 여러 개의 인터페이스를 상속받는 것이 가능하다.
  ex)

```
interface MyInter3
                    extends MyInter1, MyInter2{
            void func();
 }
```

17. 인터페이스 (Interface)

- 인터페이스도 타입 선언은 할 수 있으나,
  new 해서 객체 생성은 할 수 없다.
  반드시 그 인터페이스를 상속받은 자식 클래스
  객체로 생성해야 한다.

인터페이스를 사용하는 이유*******
인터페이스를 상속받은 클래스들에 대해
인터페이스 레퍼런스를 통해 동일하게 접근할
수 있기 때문
****