

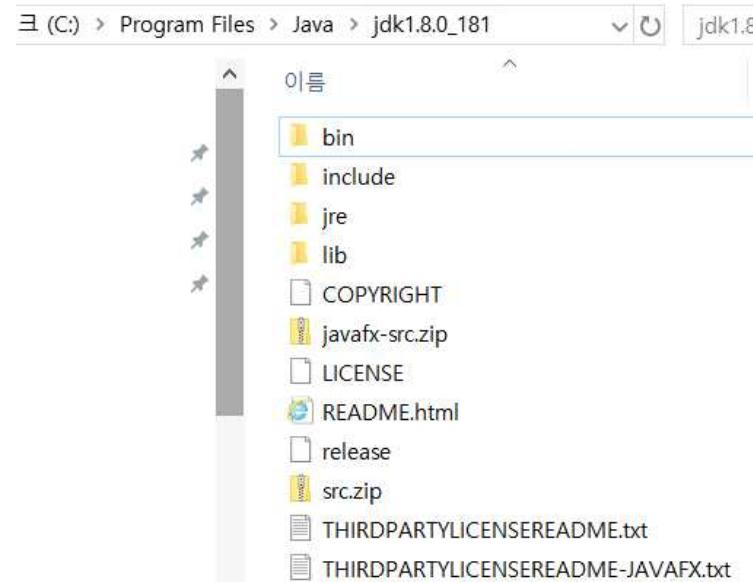
# Java Beginning 1

백성애 ([swanbaek@naver.com](mailto:swanbaek@naver.com))

# 1. 자바의 개발환경 구축

- 1) jdk 설치 -
- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- - j2se : Standard Edition [Core and Desktop]
- - j2ee : Enterprise Edition [Servlet/JSP/EJB...]
- - j2me : Micro Edition [Embedded]
  
- 2) C:\Program Files\Java\jdk1.8.0\_181 :  
JDK 소프트웨어가 설치되는 루트 디렉토리

# JDK(Java Development Kit) 의 구조



# 1. 자바 개발환경 구축

## 3) jdk환경변수 설정

- 제어판-시스템-고급-환경변수-새로만들기
- ㄱ)JAVA\_HOME 설정 : C:\Program Files\Java\jdk1.8.0\_181
- ㄴ) path설정 : %JAVA\_HOME%\bin;
- ㄷ) classpath설정 : .

## 2. 자바란?

- 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) 이어 jdk1.4/ 5.0 / 6.0/ 7.0/8.0/9.0/
- 8) 최근 10.0버전 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  
*Write once run anywhere*
- 2) 객체 지향언어: 프로그램에서 사용되는 객체들을 만들고 이것들을 조립/연결하여 전체 프로그램을 완성하는 프로그래밍 방식을 지향한다.  
=> 재사용성, 유연성, 생산성 향상
- 3) 멀티 스레드 지원 : Thread는 Process보다 작은 단위로 적은 메모리로 대용량 작업의 동시 다발적인 병렬 처리를 가능하게 한다.
- 4) 자동 메모리 관리 : -Garbage Collector(쓰레기 수집기)

### 3. 자바의 특징

#### 5) 동적 로딩과 동적 성능 확장 제공 :

애플리케이션이 실행될 때 모든 객체를 생성하지 않고 객체가 필요한 시점에 클래스를 동적으로 로딩해서 객체를 생성함.

#### 6) 함수적 스타일 코딩 지원:

최근 부각되고 있는 함수적 프로그래밍을 위해 람다식을 자바8부터 지원함.

#### 7) 풍부한 오픈소스 라이브러리 : 검증된 오픈소스 라이브러리들이 풍부하게 제공되어져 개발기간을 단축하고 안전성이 높은 애플리케이션 개발이 가능하다.

## 4. 자바 API 문서

- API란?
- Application Programming Interface.
- jdk에서 제공하는 표준 클래스 라이브러리들에 대한 설명서
- 웹브라우저로 아래 url로 들어가보자.
- <https://docs.oracle.com/javase/8/docs/api/>

## 5. 자바 컴파일 및 실행

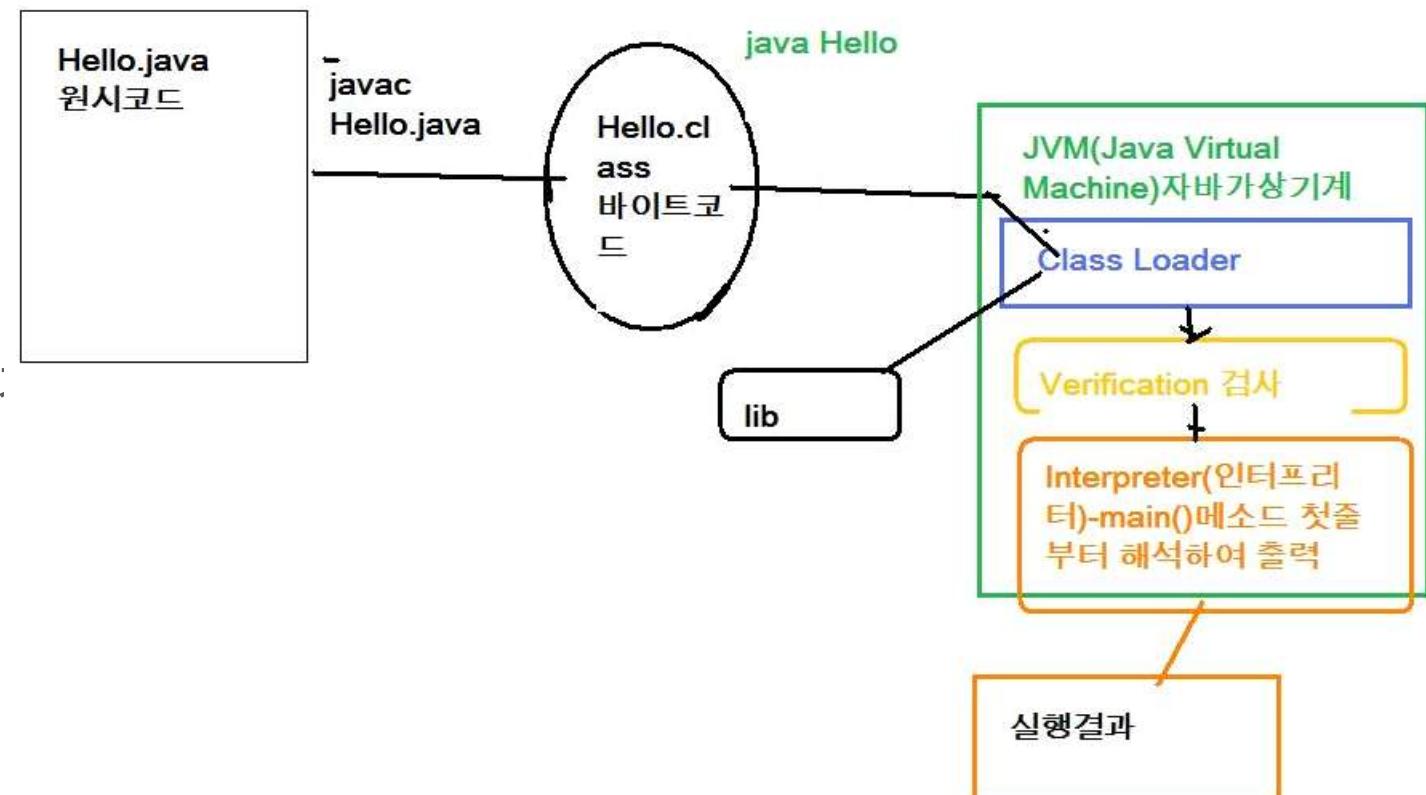
- 1) Hello.java 소스 파일 작성=> 원시 코드
- 2) 컴파일러(javac.exe)로 바이트코드 (.class) 파일 생성
- 3) 인터프리터(java.exe)

로 실행

도스에서 java파일이 있는  
곳까지 경로 이동후 아래와  
실행해보자.

컴파일: **javac Hello.java**

실행 : **java Hello**



## 6. 자바의 주석과 실행문

- 1) // : 단문 주석
- 2) /\* \*/ : 복문 주석
- 3) /\*\* \*/ 문서화 주석: javadoc를 이용해서 API문서를 작성하고자 할 때 사용
  
- 실행문은 변수 선언, 값 저장, 메소드 호출에 해당하는 코드를 말한다.
- 실행문의 마지막에는 반드시 세미콜론(;)을 붙여 실행문이 끝났음을 표시해주자.
- ex) int a=10;
- int b=20;
- int c = a + b;
- System.out.println(c);

# 7. 변수

## 1) 변수란?

- 데이터를 임시적으로 저장하는 메모리 공간.

즉 변수란 값을 저장하는 메모리 공간의 위치를 의미

## 2) 변수의 선언: 변수를 사용하기 위해서는 먼저 변수를 선언해야 한다.

타입      변수명

```
int        age; //정수값을 저장할 수 있는 age변수 선언  
double    grade; //실수값을 저장할 수 있는 grade변수 선언
```

## 3) 변수값 저장 : 변수에 값을 저장할 때는 대입연산자(=)를 사용한다.

```
int age = 22; //변수 선언과 동시에 값을 저장  
double grade; //변수 선언  
grade = 90.5; //값 저장
```

# 7. 변수

## 4) 변수의 명명 규칙

- 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 => (다음 페이지 참조)

# 7. 변수 - 자바의 예약어 (keyword)

---

abstract	do	if	package	synchronized
boolean	double	implements	private	this
break	else	import	protected	throw
byte	enum	instanceof	public	throws
case	extends	int	return	transient
catch	false	interface	short	true
char	final	long	static	try
class	finally	native	strictfp	void
continue	float	new	super	volatile
default	for	null	switch	while

---

## 변수 선언의 예

※ 잘못된 변수 선언의 예

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

※ 다음 변수는?

- int 변수=10;
  - int \$\$\$=20;
  - int \_myVar=30;
- > 모두 사용 가능.

# 8. 자바의 데이터 타입 - 기본자료형(Primitive Type)

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 -127 ~ 128  
| +---- short

+---- int

+---- long

- 실수형

+---- float

+---- double

+ ㄴ) 문자형 - char : '가' 'A' '\u0000'  
0 ~ 65535 [16비트]

+ ㄷ) 논리형 - boolean : true, false

종류 \ 크기	1	2	4	8
논리형	boolean			
문자형		char		
정수형	byte	short	int	long
실수형			float	double

# 변수의 기본값과 초기화

- 변수의 초기화: 변수에 초기값을 부여하는 것.
- 지역변수는 반드시 초기값을 부여하고 사용해야 함

자료형	기본값
boolean	false
char	'\u0000'
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d 또는 0.0
참조형 변수	null

## 8. 자바의 데이터 타입 - 참조형(Reference Type)

- 1) 클래스형
- 2) 배열
- 3) 인터페이스형
- 참조형은 new연산자를 이용해 초기화한다.
- Object obj=new Object();
- String str=new String("hello");
- String str2="hello";

# 9. 자바의 연산자 (Operator) 종류

- 연산자 (Operator): 기능을 수행하는 기호 ( +, -, \*, / 등)
- 피연산자 (Operand): 연산자의 작업 대상

1) 분리자 : . [] () ; ,

2) 단항 연산자: 항이 하나인 연산자

ㄱ) 중감연산자 : ++ --

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. 연산자 우선순위

종류	연산자	우선순위
괄호/대괄호	, [], ()	1
부정/증감	!, ++, --	2
산술	*, /, %, +, -	3
비트식	<<, >>, >>>	4
관계	<, >, <=, >=	5
	==, !=	6
비트식	&	7
	^	8
		9
논리	&&	10
		11
	? :	12
대입/할당	=, +=, -=, *=, >>=, ^=, ...	13

# 10. 연산자 우선순위

- 괄호의 우선순위가 제일 높다.
- 산술 > 비교 > 논리 > 대입
- 단항 > 이항 > 삼항
- 연산자의 연산 진행방향은 왼쪽에서 오른쪽(→)이다.
- 단, 단항, 대입 연산자만 오른쪽에서 왼쪽(←)이다.
- $3 * 4 * 5 \quad x = y = 5;$

## 10. 연산자 우선순위

$-x + 3$  단항 > 이항

$x + 3 * y$  곱셈, 나눗셈 > 덧셈, 뺄셈

$x + 3 > y - 2$  산술 > 비교

$x > 3 \&\& x < 5$  비교 > 논리

`int result = x + y * 3;` 항상 대입은 맨 끝에

- ※ 논리연산자에서 `&&`, `||` 연산자가 `|`, `||` 연산자 보다 우선순위가 높음.

# 11. 연산자 종류

## ○ 1) 증감 연산자

- ▶ 증가연산자(++) : 피연산자의 값을 1 증가시킨다.
- ▶ 감소연산자(--) : 피연산자의 값을 1 감소시킨다.

```
int i = 5;
```

```
int j = 0;
```

전위형	$j = ++i;$	$++i;$ $j = i;$	값이 참조되기 전에 증가시킨다.
후위형	$j = i++;$	$j = i;$ $i++;$	값이 참조된 후에 증가시킨다.

# 11. 연산자 종류

- 2) 부호 연산자 : +, -    3) 논리부정연산자: !

- ▶ 부호연산자(+,-) : ‘+’는 피연산자에 1을 곱하고  
‘-’는 피연산자에 -1을 곱한다.
- ▶ 논리부정연산자(!) : true는 false로, false는 true로  
피연산자가 boolean일 때만 사용 가능

```
int i = -10;
```

```
boolean power = false;
```

```
i = +i;
```

```
power = !power;
```

```
i = -i;
```

```
power = !power;
```

# 11. 연산자 종류

- 4) 비트별 NOT 연산자 : ~

- 정수를 2진수로 표현했을 때, 1을 0으로 0은 1로 바꾼다.

- 정수형에만 사용 가능.

2진수	10진수
0 0 0 0 1 0 1 0	10
1 1 1 1 0 1 0 1	-11
1 1 1 1 0 1 0 1	-11
0 0 0 0 0 0 0 1	+ ) 1
1 1 1 1 0 1 1 0	-10

# 11. 연산자 종류

## ○ 5) 이항 연산자 : 사칙연산(+,-,\*,/), 나머지 연산(%) 등

이항연산자는 연산을 수행하기 전에 피연산자의 타입을 일치시킨다.

- int보다 크기가 작은 타입은 int로 변환한다.

( byte, char, short → int )

- 피연산자 중 표현범위가 큰 타입으로 형변환 한다.

byte + short → int + int → int

char + int → int + int → int

float + int → float + float → float

long + float → float + float → float

float + double → double + double → double

## 11. 연산자 종류

- 6) 삼항 연산자 : 변수선언문  $=(\text{조건식})? \text{값1}: \text{값2};$
- 조건식이 true이면 값1을 변수에 대입하고, false이면 값2를 변수에 대입한다.

## 12. 자바의 제어문

주 제어문	보조 제어문
1) 조건문 : if, if~else, if~else if~else	1) break 문 2) continue 문
2) switch~case 문	단독으로 쓰이지는 못 하고 주제어문과 함께 사용된다.
3) 반복문 - for 문 - while 문 - do~while 문	

# 12. 제어문

- 1) 조건문

- if문

- if ~ else 문

- if ~ else if ~ else

- 문: 다중 if문

```
if(조건식) {  
    // 조건식의 결과가 true일 때 수행될 문장을  
}
```

```
if(조건식) {  
    // 조건식의 결과가 true일 때 수행될 문장을  
} else {  
    // 조건식의 결과가 false일 때 수행될 문장을  
}
```

```
if(조건식1) {  
    // 조건식1의 결과가 true일 때 수행될 문장을  
} else if(조건식2) {  
    // 조건식2의 결과가 true일 때 수행될 문장을  
    // (조건식1의 결과는 false)  
} else if(조건식3) {  
    // 조건식3의 결과가 true일 때 수행될 문장을  
    // (조건식1과 조건식2의 결과는 false)  
} else {  
    // 모든 조건식의 결과가 false일 때 수행될 문장을  
}
```

# 12. 제어문

- 2) switch ~ case 문

- 조건식에는 변수, 수식 들이 올수 있으며

- int형 이하의 정수형과

- String만 사용 가능

- (String은 jdk7.0부터 사용가능)

- 조건식의 계산결과와 일치하는 case문으로 이동 후 break문을 만날 때까지 문장들을 수행한다.(break문이 없으면 switch문의 끝까지 진행한다.)
- 일치하는 case문의 값이 없는 경우 default문으로 이동한다.

```
switch (조건식) {  
    case 값1 :  
        // 조건식의 결과가 값1과 같을 경우 수행될 문장들  
        //...  
        break;  
    case 값2 :  
        // 조건식의 결과가 값2와 같을 경우 수행될 문장들  
        //...  
        break;  
    //...  
    default :  
        // 조건식의 결과와 일치하는 case문이 없을 때 수행될 문장들  
        //...  
}
```

# 13. 제어문 - 반복문

## ○ 1) for 루프문

- 초기화, 조건식, 증감식 그리고 수행할 블럭{} 또는 문장으로 구성

```
for (초기화; 조건식; 증감식) {  
    // 조건식이 true일 때 수행될 문장들을 적는다.  
}
```

[참고] 반복하려는 문장이 단 하나일 때는 중괄호{}를 생략할 수 있다.

1. 초기화 → 2. 조건식 → 3. 수행될 문장 → 4. 증감식



## 13. 제어문 - 반복문

### ○ 2) while루프문

- 조건식과 수행할 블럭{} 또는 문장으로 구성

```
while (조건식) {  
    // 조건식의 연산결과가 true일 때 수행될 문장들을 적는다.  
}
```

## 13. 제어문 - 반복문

### o 3) do ~ while 루프문

- while문의 변형. 블럭{}을 먼저 수행한 다음에 조건식을 계산한다.
- 블럭{}이 최소한 1번 이상 수행될 것을 보장한다.

```
do {  
    // 조건식의 연산결과가 true일 때 수행될 문장들을 적는다.  
} while (조건식);
```

# 14. 제어문 - 보조제어문(break, continue)

- 1) break문
  - 단독으로는 사용되지 못하고 주 제어문과 함께 사용됨.
  - 가장 가까운 반복문 또는 switch문을 벗어난다.
  
- 2) continue문
  - 자신이 포함된 반복문을 계속 수행한다.
  - continue이후 문장은 수행되지 않는다.

# 15. 배열

- [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

# 15. 배열

- 1차원 배열

데이터형 배열명[] = new 데이터형[배열의 크기];

- 2차원 배열

데이터형 배열명[][] = new 데이터형[배열의 크기][배열의 크기];

예 1)

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

## 15. 배열

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 index 초과 오류
```

발생

예3) 선언, 메모리 할당, 초기화를 한꺼번에 하는 방법

법

```
int [] c = {1, 2, 3, 4, 5};
```

## 15. 배열

\*\*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]);

# 15. 배열

## [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;
```

## 15. 배열

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];
```

# 15. 배열

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] |

# 16. 객체지향프로그래밍 -OOP

OOP란? Object Oriented Programming

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.

# 16. 객체지향프로그래밍 -OOP

## 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...

# 16. 객체지향프로그래밍 -OOP

- 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

# 17. 클래스와 객체

## 1) 클래스란?

- 객체를 정의해 놓은 것. 객체를 생성할 때 사용하는 틀. 설계도.
- ex) 봉어빵 틀=> 클래스, 봉어빵 => 객체
- 설계도 => 클래스, 집 => 객체

## 2) 객체란?

실제로 존재하는 사물 또는 개념. 속성과 행동양식을 갖는다.

## 17\_2. 클래스의 구조

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

## 17\_3. 클래스의 멤버 - 변수

### ▶ 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;

\*\***객체명**으로 접근해야 한다.

#### ㄴ) 클래스변수(static 변수)

ex) static int b=10;

\*\***클래스명**으로 접근해야 한다.

## 17\_4. 클래스의 멤버 - 생성자

- ▶ 2) 생성자(멤버 변수의 초기화):

객체를 생성할 때 호출된다.

생성자 이름과 클래스 이름은 같아야 한다.

반환타입이 없다.

- ▶ ex)

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

## 17\_5. 클래스의 멤버 - 메소드

- ▶ **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;
```

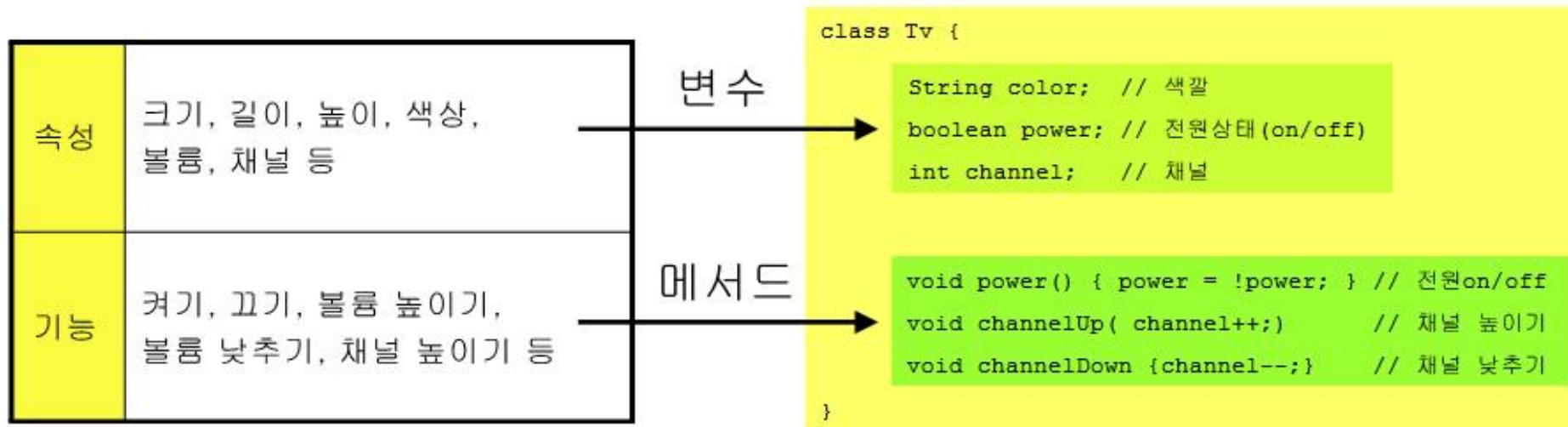
```
}
```

```
public static void mySub(){
```

```
}
```

# 18. 객체의 속성과 기능(행동양식)

- ▶ 객체는 속성과 기능으로 이루어져 있다.
  - 객체는 속성과 기능의 집합이며, 속성과 기능을 객체의 멤버(member, 구성 요소)라고 한다.
- ▶ 속성은 변수로, 기능은 메서드로 정의한다.
  - 클래스를 정의할 때 객체의 속성은 변수로, 기능은 메서드로 정의한다.



# 19. 객체 생성과 사용

## ▶ 인스턴스의 생성방법

클래스명 참조변수명; // 객체를 다루기 위한 참조변수 선언

참조변수명 = new 클래스명(); // 객체생성 후, 생성된 객체의  
주소를 참조변수에 저장

```
Car c = new Car();
c.color ="red";
c.velocity = 100;
c.run();
```

# 20. OOP특징 - 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.

# 20. OOP특징 - 추상화

```
|58     위의 집이란 객체를 자바 프로그램에 추상화해보면
|59     class House
|60     {
|61         int room;
|62         String ownerName;
|63         String addr;
|64
|65         public void existAt(String addr){
|66             System.out.println(addr+"에 위치하다");
|67         }
|68     }
|69     class HouseTest
|70     {
|71         public static void main(String args[]){
|72             House h=new House();
|73             h.existAt("300번지");
|74
|75             House h2=new House();
|76             h2.existAt("100번지");
|77
|78
|79         }
|80     }|||||||||||||||||||
```

# 21. 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;  
    }  
}
```

## 22. OOP - 다형성

### 3. 다형성(Polymorphism)

: 여러 가지 형태를 가질 수 있는 성질을 의미.

한 타입의 참조변수로 여러 타입의 객체를 참조할 수 있도록 함으로써 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

## 22. OOP - 다형성

### ▶ 오버로딩 조건

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

## 22. OOP - 다형성

### 2) 오버라이딩(Overriding)

...상속 개념과 맞물려 사용

부모로부터 상속 받은 메소드를

재정의해서 사용하는 것

: 재정의

## 22. OOP - 다형성

### 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.

## 23. 오버로딩(다중정의)

- Overloading
- **1)생성자 오버로딩**
- -생성자란?
- : 객체가 생성될 때 최초로 실행되는메소드를 의미.
- 생성자 구성시 유의할 점
  - a) 생성자 이름은 클래스명과 동일하게
  - b) 반환타입을 가져선 안된다.

## 23. 오버로딩(다중정의)

- 생성자의 주요 역할
- : 멤버 변수를 초기화 하는 일
- -사용자가 생성자를 구현하지 않았을 경우
- ->컴파일러는 default생성자를 제공해줌
  
- ※ 그러나 사용자가 생성자를 하나라도 구현했다면, 그 때는 컴파일러가 제공해 주는 기본생성자는 사라진다.
- -자바에서는 생성자를 다양하게 오버로딩 함으로써 다양한 초기값을 부여하고 있다.

## 24. 생성자 안에서 this()

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.

※ 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.

## 25. 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.

### \*\*\*\*\*

# 26. 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]

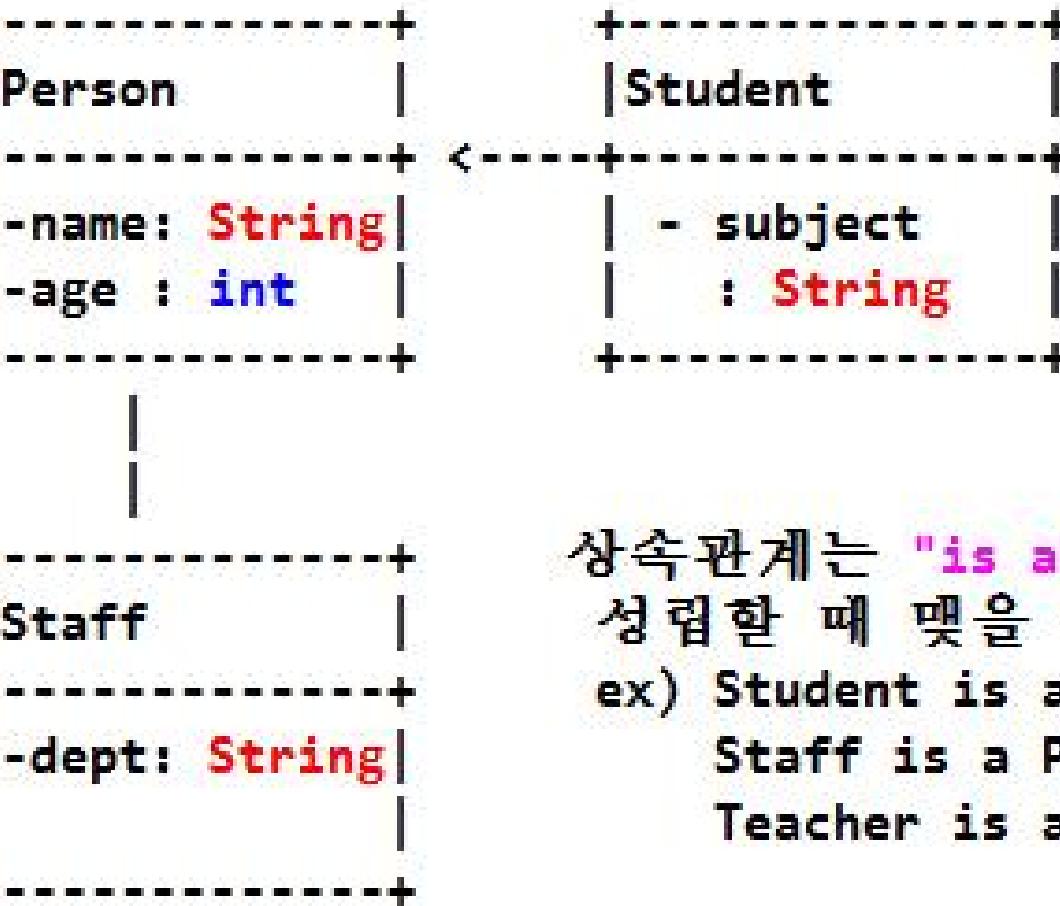
좌표를 나타내는 Point클래스를 설계해보자.

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 메소드를 구성해보자.
8. main()메소드를 갖는 PointTest클래스를 만들어 Point객체 생성해 자기가 구성한 메소드를 호출해보자.

## 27. 상속성

- 기존 클래스에 작은 기능이나 특성을 추가하여 새로운 클래스로 만드는 것을 의미.
- 즉 부모클래스를 만들고, 그 부모클래스에 있는 속성과 기능을 자식 클래스에서 상속받아, 새로운 기능과 속성을 추가하는 것.
- 상속 개념을 적용함으로써 개발시간 단축, 재사용성 등에 놀라운 장점이 있다

## 27. 상속성



상속관계는 "is a" 관계가  
성립할 때 뜻을 수 있다.

ex) Student is a Person  
Staff is a Person  
Teacher is a Person

\*\*자바에서 상속을 받을 때는 **extends** 란 키워드  
를 사용한다.

자바는 단일 상속 개념이므로 **extends**로 상속  
받을 수 있는 클래스는 단 하나뿐이다.