

자 바 기 초

손병찬

sbc2000kr@naver.com



사전 부탁

- ✓ 핸드폰은 진동으로
- ✓ 수업중에 카톡하지 않기
- ✓ 수업 자료 외의 인터넷 금지
- ✓ 졸지 않기

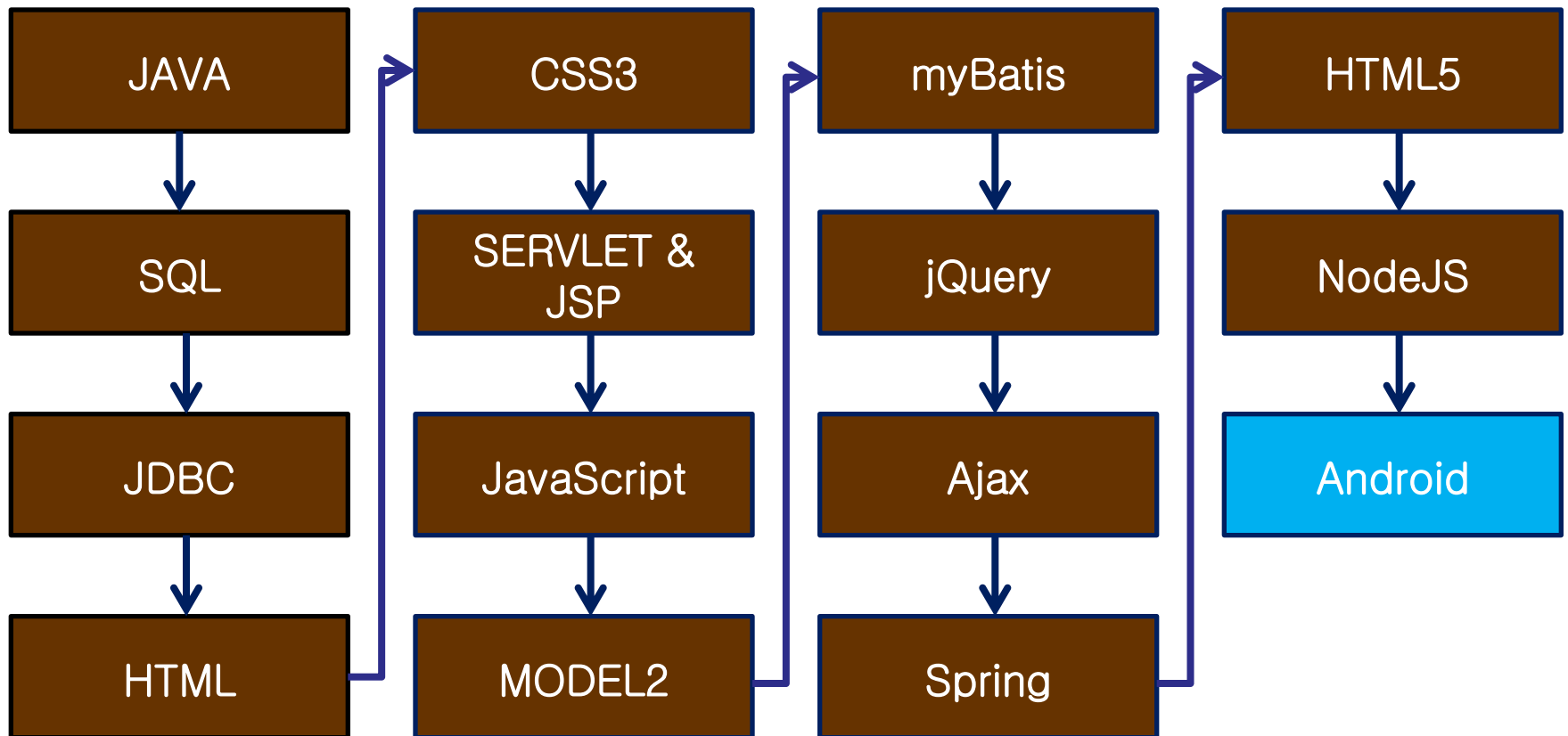


수업 진행

이론 + 실습 중심

응용은 ??

과정 로드맵





수업을 하기 위한 준비

0 절. 디렉토리 만들기

1 절. 필요파일 다운로드

- JDK : 1.8버전

- 편집기 : 이클립스 4.5 (Mars)

- 수업 디렉토리 만들기

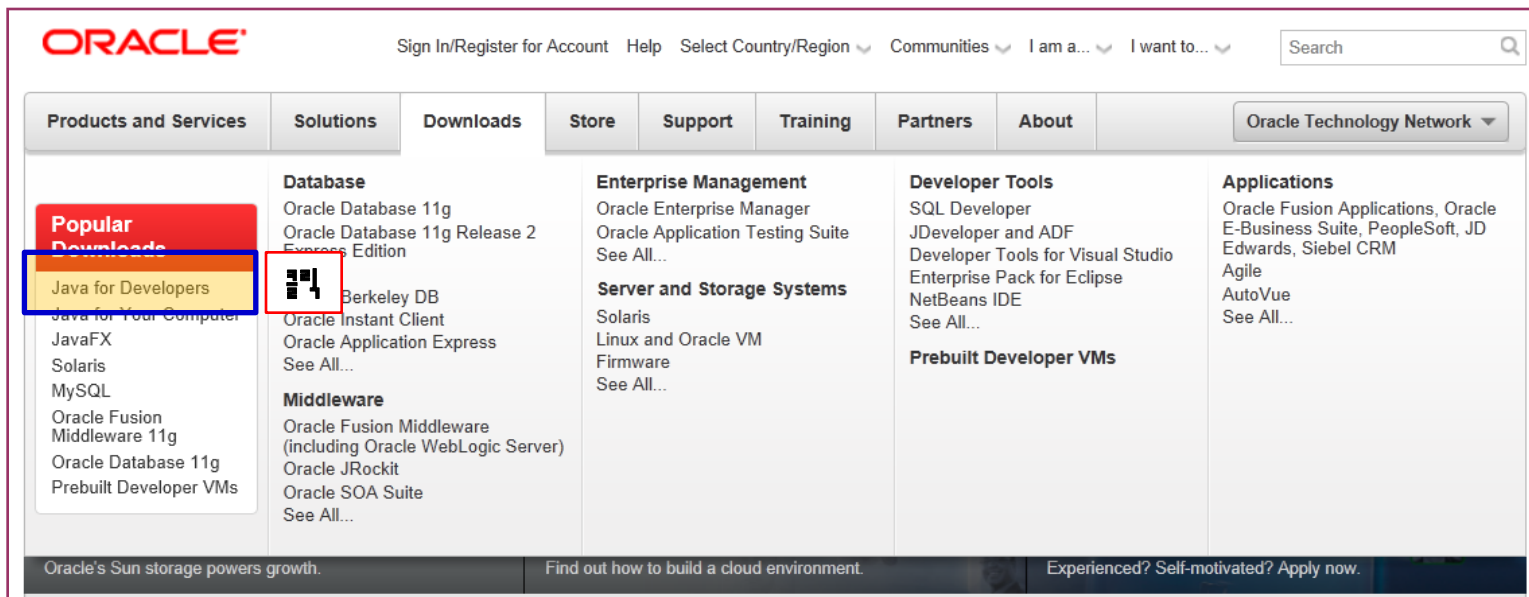
c:\java기수 : 수업 동안 사용될 루트 폴더

- **bin** : 설치 경로 (**Eclipse**)
- **workspace** : 작업 소스 폴더
- **setup** : 수업에 필요한 설치파일

- 다운로드 링크

<http://www.oracle.com>

- 자바 다운로드 선택



JDK 다운로드 링크 선택

Java SE Development Kit 7u5

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

2 ☐ Accept License Agreement **확인** Oracle Binary Code License Agreement

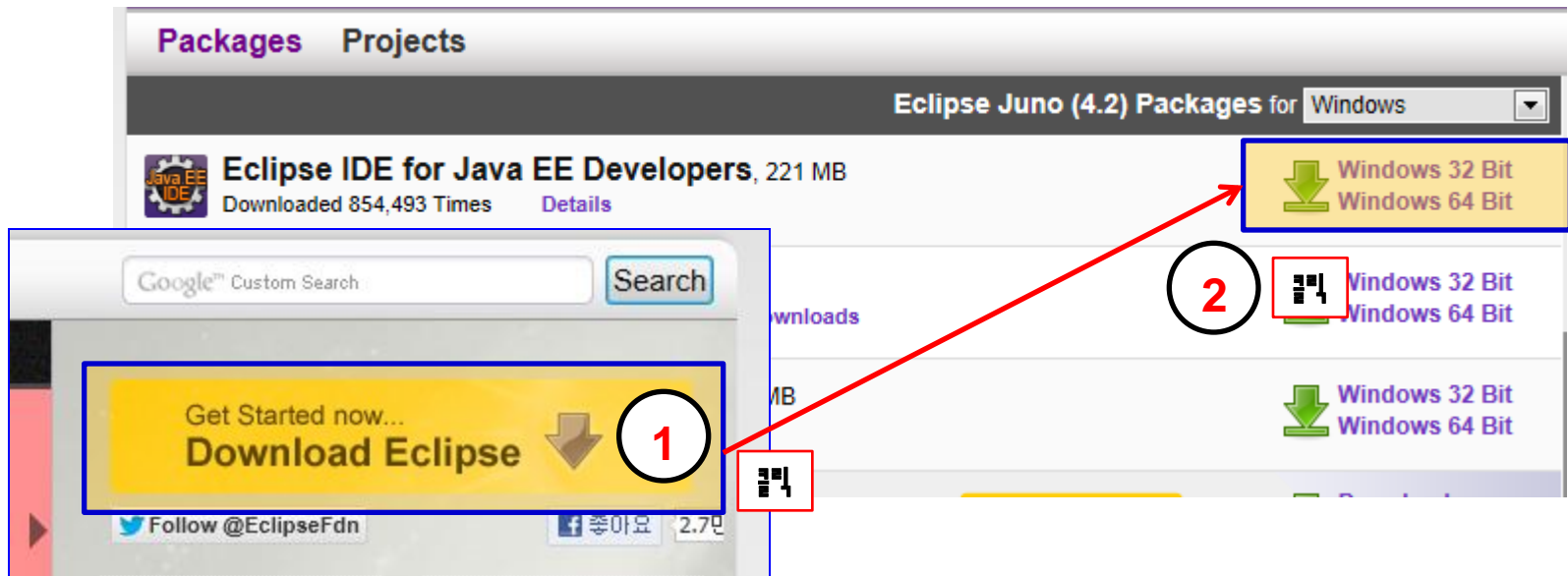
Product / File Description	File Size	Download
Here are the Java SE downloads in detail:		
Java Platform, Standard Edition		
Java SE 7u5 This release includes security enhancements and bug fixes. Learn more 1	JDK DOWNLOAD	JRE DOWNLOAD
"What Java Do I Need?" You must have a copy of	JDK 7 Docs	JRE 7 Docs
Solaris SPARC 64-bit	12.55 MB	3 다운로드 jdk-7u5-linux-i586.rpm
Solaris x64	14.39 MB	다운로드 jdk-7u5-linux-i586.tar.gz
Solaris x64	9.54 MB	다운로드 jdk-7u5-linux-x64.rpm
Windows x86	87.9 MB	다운로드 jdk-7u5-linux-x64.tar.gz
Windows x64	92.3 MB	다운로드 jdk-7u5-macosx-x64.dmg
		다운로드 jdk-7u5-solaris-i586.tar.Z
		다운로드 jdk-7u5-solaris-i586.tar.gz
		다운로드 jdk-7u5-solaris-sparc.tar.Z
		다운로드 jdk-7u5-solaris-sparc.tar.gz
		다운로드 jdk-7u5-solaris-sparcv9.tar.Z
		다운로드 jdk-7u5-solaris-sparcv9.tar.gz
		다운로드 jdk-7u5-solaris-x64.tar.Z
		다운로드 jdk-7u5-solaris-x64.tar.gz
		3 다운로드 jdk-7u5-windows-i586.exe
		다운로드 jdk-7u5-windows-x64.exe 확인

- Windows x86 → 32bit, Windows x64 → 64bit 일 경우 선택

- 다운로드 링크

<http://eclipse.org>

- Download Eclipse 버튼 클릭

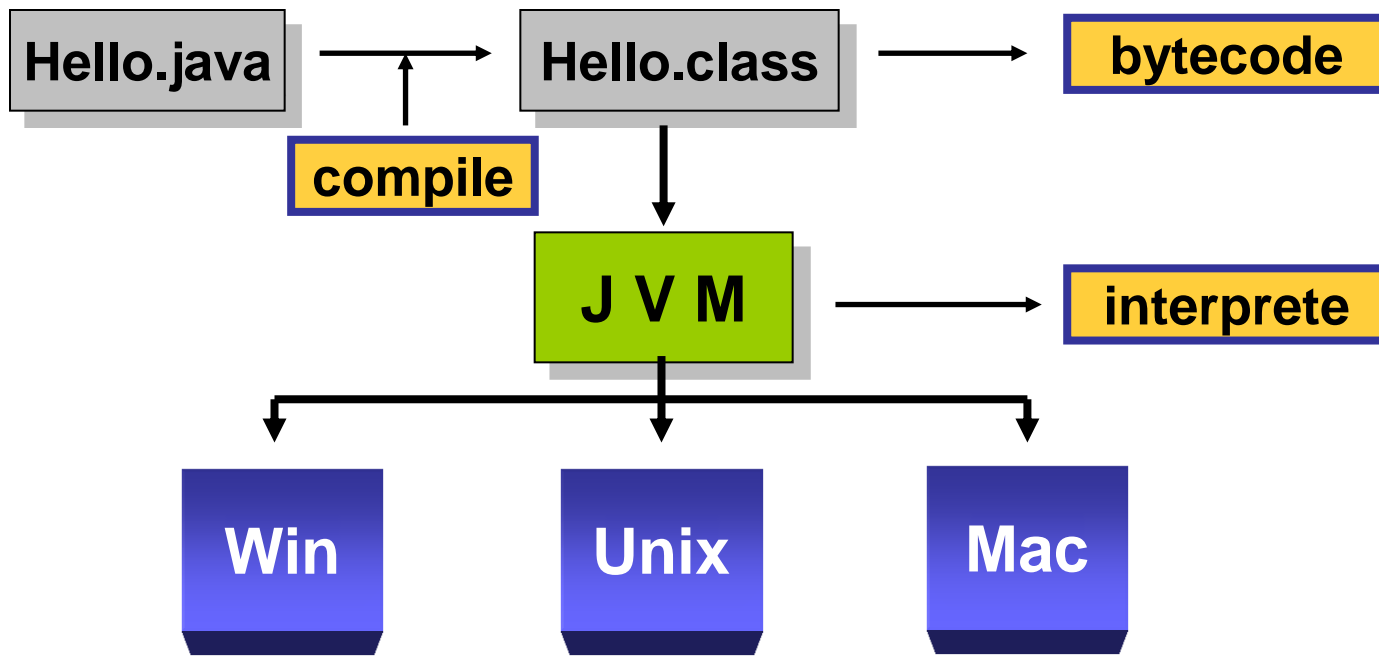




설치 및 실행

자바언어의 특징

- 단순(Simple) : 메모리 관리(Garbage Collection)
- 객체지향적(Object-Oriented) : 재사용
- 컴파일 + 인터프리터 : OS에 독립적, WORA





자바 환경 설정 및 설치

- Java 분야
 - J2SE
 - J2ME
 - J2EE
- Java 다운로드 : <http://www.oracle.com>
- JDK 설치 확인 : `java -version`
- 환경설정
 - JAVA_HOME 설정
 - PATH 설정



자바 환경 설정 및 설치

- 편집기 이클립스 설치
 - 다운로드 : <http://eclipse.org/>



HelloWorld

- **FileName : HelloWorld.java**
- **file save** : 파일 작성 후 파일을 저장한다.
 - 파일 저장 시 클래스의 이름과 파일명을 대소문자를 구별하여 저장한다.
 - **public class HelloWorld** 라면 파일명은 **HelloWorld.java** 로 저장한다.
- **compile** : 일반 텍스트를 **JVM**이 인식하는 **byte code** 로 만드는 과정
 - 파일이 저장되어 있는 디렉토리로 이동한다.
 - **javac** 파일명(확장자명 포함)
 - **javac HelloWorld.java**
 - **compile** 된 결과물 확인 : **HelloWorld.class**



HelloWorld

- **run : byte code**를 기계어 코드로 해석하고 실행하는 과정
 - 실행이 되기 위해서는 **main** 메서드가 반드시 필요하다
 - **java** 파일명 (확장자명 제외)
 - **java HelloWorld**



파일 작성시 참고사항

- 하나의 파일에 여러개의 클래스를 작성 할 때의 유의점
 - 하나의 **java** 파일에는 여러개의 클래스가 동시에 존재할 수 있다.
 - **public class** 형태의 선언은 파일내에서 하나만 가능하다.
 - **public class** 형태의 클래스가 파일명으로 적용되어야 한다.
public class Test -> Test.java
 - 만약, 여러개의 클래스가 정의되어 있고 **public class** 형태가 없다면 파일명으로 모든 클래스가 가능하다.



메인 메소드

- 실행 명령인 java 를 실행 시 가장 먼저 호출 되는 부분
- 만약, Application 에서 main() 메소드가 없다면 절대로 실행 될 수 없음
- Application의 시작 = 특정 클래스의 main() 실행
- 형태 (고정된 형태)

```
public static void main(String [ ] args) { }
```

명령형 매개변수

- 프로그램 실행 시 특정 값을 메인 메서드에 넘겨줄 수 있다.
- 실행 시 파일명 뒤에 값들을 추가.

```
java MainArgsTest my name is hong
```

- 명령형 매개변수는 모두 문자열로 취급된다. 0부터 접근한다.

```
public static void main(String [ ] args) { }
```

```
args [ 0 ] -> my
```

```
args [ 1 ] -> name
```

```
args [ 2 ] -> is
```

```
args [ 3 ] -> hong
```

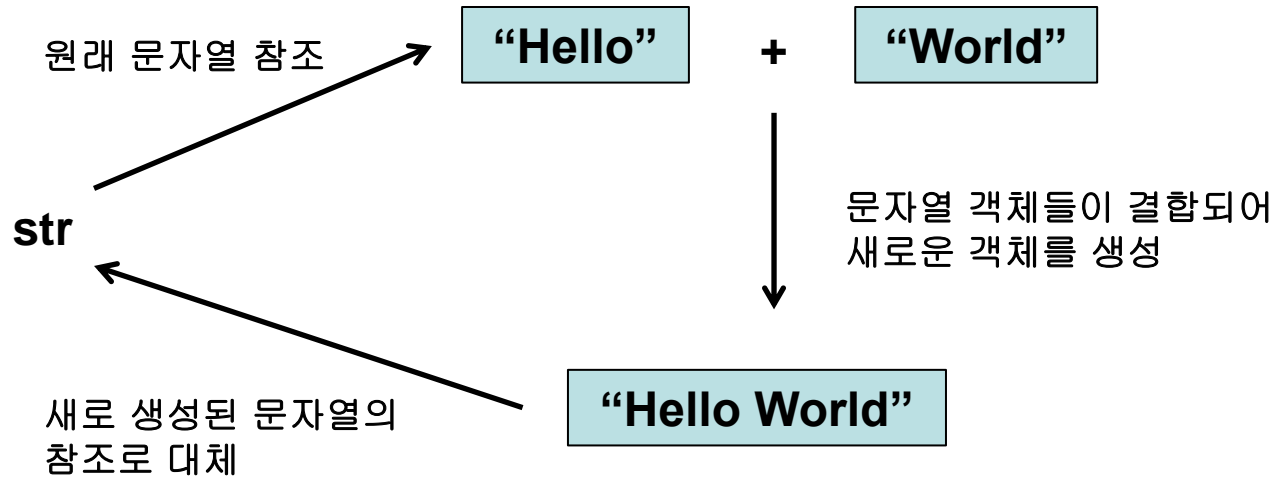


문자열의 + 연산

- 문자열과 + 연산을 수행하게 되면 결과는 문자열이 된다.
 - 문자열 + int = 문자열, 문자열 + char = 문자열, 문자열 + boolean = 문자열
- 문자열은 + 연산만 가능하다(-, *, / 등은 불가능)
- 문자열은 “ “ 으로 묶어서 표시
 - “java”, “100” ..

문자열의 결합

```
String str = "Hello";  
str += "World";
```





문자열을 숫자로 변경

- `int Integer.parseInt (String)`
- `int i = Integer.parseInt (“100“); // 100`
- 문자열이 숫자형태일 경우만 가능하다.

출력문

- print
- println
- printf
 - %d : 정수
 - %f : 실수
 - %c : 문자
 - %s : 문자열

```
System.out.println( "%s");
```

화면에 %s 가 출력

```
System.out.printf( "오늘은 날씨가 %s");
```

오류발생

```
System.out.printf( "오늘은 날씨가 %s", "화창하다");
```

오늘은 날씨가 화창하다

```
System.out.printf( "오늘 점심은 %d시에 %s" , 12, "강남식당");
```

오늘점심은 2시에 강남식당

```
System.out.printf( "오늘의 원달러 환율은 %f 원" , 1019.8);
```

오늘의 원달러 환율은 1019.800000 원

```
System.out.printf( "오늘의 원달러 환율은 %7.1f 원" , 1019.8);
```

오늘의 원달러 환율은 1019.8 원



식별자, 자료형, 연산자

- 클래스, 메소드, 변수의 이름
- 명명규칙
 1. 클래스 : 단어의 첫 글자를 대문자로 표기. 만약, 여러 개의 단어로 이루어져 있다면 각 단어의 첫글자를 대문자로 표기.
예> Hello, HelloWorld, BoardMng
 2. 멤버변수, 메소드 : 단어의 첫 글자를 소문자로 표기. 만약, 여러 개의 단어로 이루어져 있다면 각 단어의 첫글자를 대문자로 표기
예> name, cnt, main(), print(), juminNo, printName()
 3. 상수 : 모든 단어를 대문자로 표기. 만약, 여러 개의 단어로 이루어져 있다면 단어와 단어 사이를 '_'로 구분한다.
예> MAX, MIN, MAX_VALUE, SERVER_NAME



2. 변수

- 데이터를 저장할 메모리의 위치를 나타내는 이름
- 메모리 상에 데이터를 보관할 수 있는 공간을 확보
- 적절한 메모리 공간을 확보하기 위해서 변수의 타입 등장
- '=' 를 통해서 CPU에게 연산작업을 의뢰

2. 변수

메모리의 단위

- 0과 1을 표현하는 bit
- 8bit = 1byte
- 2bytes = word

40을 표현??

0	0	1	0	1	0	0	0
---	---	---	---	---	---	---	---

bit의 데이터 처리 - 2진수

- 전구의 불이 들어오고 나가는 처리

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	0	0	0	0	1	1	1
					4 +	2 +	1
							결과 7

2. 변수

- 변수

- 선언

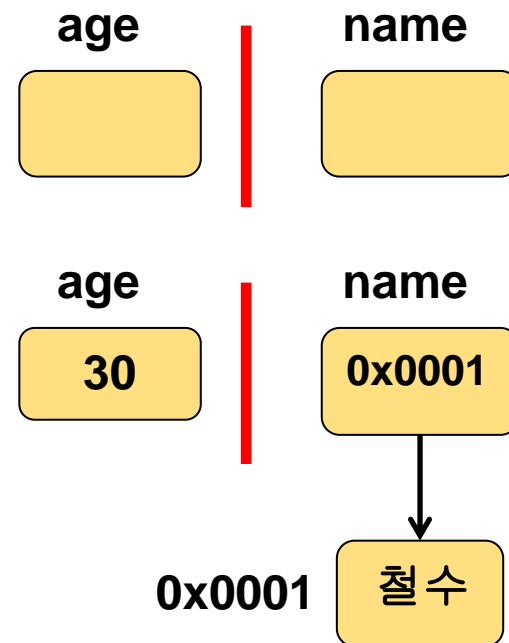
1. 자료형 변수명;
2. 예> `int age; String name; ...`

- 초기화

1. 변수명 = 저장할 값;
2. 예> `age = 30; name = "철수";`

- 선언과 초기화를 동시에

1. 자료형 변수명 = 저장할 값;
2. 예> `int age = 30;`



2. 변수

➤ 변수의 범위

- 변수가 참조 될 수 있는 영역
- 기준은 블록 단위

```
{  
    int a = 1;    // a를 선언하고 정의  
  
    // 선언 이후부터 a에 대한 참조가 시작, 여기부터 사용가능  
    // 선언전에는 b에 대한 참조를 사용할 수 없음(에러)  
    {  
        // a에 대한 참조 가능  
        // b에 대한 참조 불가능  
  
        int b = 2; // b를 선언하고 정의  
  
        // a와 b에 대한 참조 가능  
    }  
    // b에 대한 참조 불가능  
    // a에 대한 참조 가능  
}
```



2. 변수

퀴즈) 아래의 코드의 결과를 예측하고 그 이유를 생각하시오

```
int outer = 1;
{
    int inner = 2;
    System.out.println("inner = " + inner);
    System.out.println("outer = " + outer);
    int outer = 5;
}
```

```
System.out.println("inner = " + inner);
```

```
int inner = 3;
System.out.println("inner = " + inner);
System.out.println("outer = " + outer);
```

2. 자료형

• 자료형

- 기본 자료형과 참조 자료형(기본 자료형 8가지 외 모든 것)
- 기본 자료형 (맨 앞의 비트는 부호비트)

타입	세부타입	데이터형	크기	기본값	사용예
논리형		boolean	1bit	false	boolean b = true
문자형		char	2byte	null(\u0000)	char c = 'a', c1 = 65, c2 = '\uffff'
숫자형	정수형	byte	1byte	(byte)0	byte b = 100;
		short	2byte	(short)0	short s = 100;
		int	4byte	0	int i = 100;
		long	8byte	0L	long l = 100, l2 = 100L
	실수형	float	4byte	0.0f	float f = 3.1f, f2 = 3.1F;
		double	8byte	0.0d	double d = 3.1;

2. 형변환

예제작성

- 자료형의 크기 비교

byte < short < int < long < float < double

char < int < long < float < double

- 데이터 형변환

1) 묵시적(암묵적) : Implicit Casting

: 범위가 넓은 데이터 형에 좁은 데이터 형을 대입하는 것

: 예> byte b = 100; int i = b;

2) 명시적 : Explicit Casting

: 범위가 좁은 데이터 형에 넓은 데이터 형을 대입하는 것

: 형변환 연산자 사용 - (타입) 값;

: 예> int i = 100; byte b = i; (X), byte b = (byte) i; (O)

2. 자료형

예제작성

- 기본 자료형과 참조 자료형의 차이

	기본 자료형	참조 자료형
변수값	실제 사용할 값	객체 참조값
정의방식	Java 내부 이미 정의	클래스, 인터페이스, 배열, enum등..
생성방식	19, 3.14, true, 'a'	new 키워드 활용
초기화방식	default	생성자

```
int a = 100;
```

a

100

```
String s = new String("100");
```

s

0x0001

0x0001

"100"

2. 자료형

예제작성

기본 자료형 에서 꼭 기억해야 할 것들

모든 변수는 데이터를 담는 공간(상자)이다.
기본 자료형의 타입은 무조건 소문자로 시작한다.
상자의 타입은 담을 수 있는 데이터의 종류를 말한다.
각 타입마다 상자의 크기가 있다.

변수 선언시의 주의 점

적절한 변수의 타입을 써라.
변수의 이름은 결국 메모리상에서 찾아가는 이름 - 충돌 조심
상자를 최초로 만들 때에만 변수의 타입이 쓰인다.
이클립스는 미리 프로파일링을 통해서 불필요한 변수 선언을 처리

2. 변수, 상수, 자료형

예제작성

- 상수

- 변경될 수 없는 고정된 데이터
- 코드의 이해와 변경이 쉬움
- final 키워드를 이용해서 정의

예> int age; -> 변수, **final** double PI = 3.14 -> 상수

- 문자열 상수(이스케이프 문자)

문자상수	내용	문자상수	내용
\n	줄넘김	\"	“ 표시
\t	탭만큼 띄우기	\'	‘ 표시
\\	\\ 화면에 표시		

3. 연산자

예제작성

- 3항 연산자

형식

조건식 ? 수식-1 : 수식-2;

수식-1 : 조건식의 결과가 참(true) 일 때 수행되는 식

수식-2 : 조건식의 결과가 거짓(false) 일 때 수행되는 식

```
int a = 10;
```

```
int b = 5;
```

```
int max = ( a > b ) ? a : b ;
```

max 값은 ??

3. 연산자

예제작성

- 산술 연산자

연산자	사용법	설 명
+, -, *		
/	op1 / op2	op1을 op2로 나눈 몫을 구한다.
%	op1 % op2	op1을 op2로 나눈 나머지를 구한다.

byte, short 는 int 로 연산

피연산자 중 하나가 double 형일 경우, 연산이 수행되기 전 다른 피연산자도 double 형으로 변환

피연산자 중 하나가 float 형일 경우, 연산이 수행되기 전 다른 피연산자도 float 형으로 변환

피연산자 중 하나가 long 형일 경우, 연산이 수행되기 전 다른 피연산자도 long 형으로 변환

3. 연산자

예제작성

- 증감 연산자

연산자	사용법	설명	사용예
++	++op (선행처리)	1 증가	int a = 5; int b = a++; ??
	op++ (후행처리)		int a = 5; int b = ++a; ??
--	--op (선행처리)	1 감소	int a = 5; int b = a--; ??
	op-- (후행처리)		int a = 5; int b = --a; ??

```
int a = 5;
System.out.println(a++); 5
System.out.println(++a); 7
System.out.println(--a); 6
System.out.println(a  ); 6
System.out.println(a--); 6
System.out.println(a++); 5
```

3. 연산자

예제작성

- 비교 연산자 : 결과값으로 참(true), 거짓(false)이 반환

연산자	사용법	설명	사용예
>, >=, <, <=			
==	op1 == op2	서로 같은 경우	boolean b = ((10 % 2) == 0)
!=	op1 != op2	서로 같지 않은 경우	boolean b = ((10 % 2) != 0)
instanceof	op1 instanceof op2	객체의 타입을 비교	

가장 많이 사용되는 것
== , !=

3. 연산자

예제작성

- 논리 연산자 : 결과값으로 참(true), 거짓(false)이 반환

연산자	사용법	설명
&&	A && B	A와 B가 참일 경우만 참 반환 (A가 거짓일 경우 B는 실행하지 않는다)
	A B	A 또는 B 둘 중에 하나가 참일 경우 참 반환 (A가 참일 경우 B는 실행하지 않는다)
!	! A	A가 참이면 거짓, A가 거짓이면 참을 반환

3. 연산자

예제작성

- 배정연산자

연산자	사용법	설명
+=	op1 +=op2	op1 = op1 + (op2)
-=	op1 -=op2	op1 = op1 - (op2)
*=	op1 *=op2	op1 = op1 * (op2)
/=	op1 /=op2	op1 = op1 / (op2)

lhs op= rhs

rhs : right hand side (우변)

lhs : left hand side (좌변)



4. 조건문

예제작성

- if 문
- switch 문

4. 조건문

예제작성

- 단일 if

- if (조건식)
실행문장;
- if (조건식)
실행문장;
else
실행문장;
- if (조건식) {
.....
} else {
.....
}

주의사항

- 실행문장
때에는
- 조건식
반드시
구분해

```
int a = 3  
if(a = 3)  
if( 0 )
```

=> 에디팅

- else 절은 필요에 따라
기술했다.

```
if(true)  
System.out.println("1");  
System.out.println("1");
```

```
if(true) {  
System.out.println("1");  
System.out.println("1");  
}
```

4. 조건문

예제작성

- 다중 if

```
형식 - if (조건식) {  
    실행문장;  
} else if (조건식) {  
    실행문장;  
} else if (조건식) {  
    실행문장;  
} else {  
    실행문장;  
}
```

4. 조건문

예제작성

- 내포된 if


형식 - if (조건식) {
 if (조건식)

 if (조건식)

 else

}

블록 처리가
제대로 되지 않아
에러가 발생한다.



```
int a = 10, b = 5;  
if ( a > b ) {  
    if ( a == 10 )  
        System.out.println("a = 10");  
        System.out.println("a가 b보다 크다");  
    else  
        System.out.println("b가 a보다 크다");  
}
```

4. 조건문

예제작성

- switch

```
switch (수식) {  
    case 값1:  
        처리문장  
        break;  
    case 값2:  
        처리문장  
        break;  
    case 값n:  
        처리문장  
        break;  
    default:  
        묵시적  
}
```

주의사항

break문이 없을 경우
break를 찾을 때까지 선택된 case문 아래의 모든
문장을 실행

```
=====
switch( 1 ) {  
    case 1 :  
        System.out.println(1);  
    case 2 :  
        System.out.println(2);  
    default :  
        System.out.println(3);  
}
```

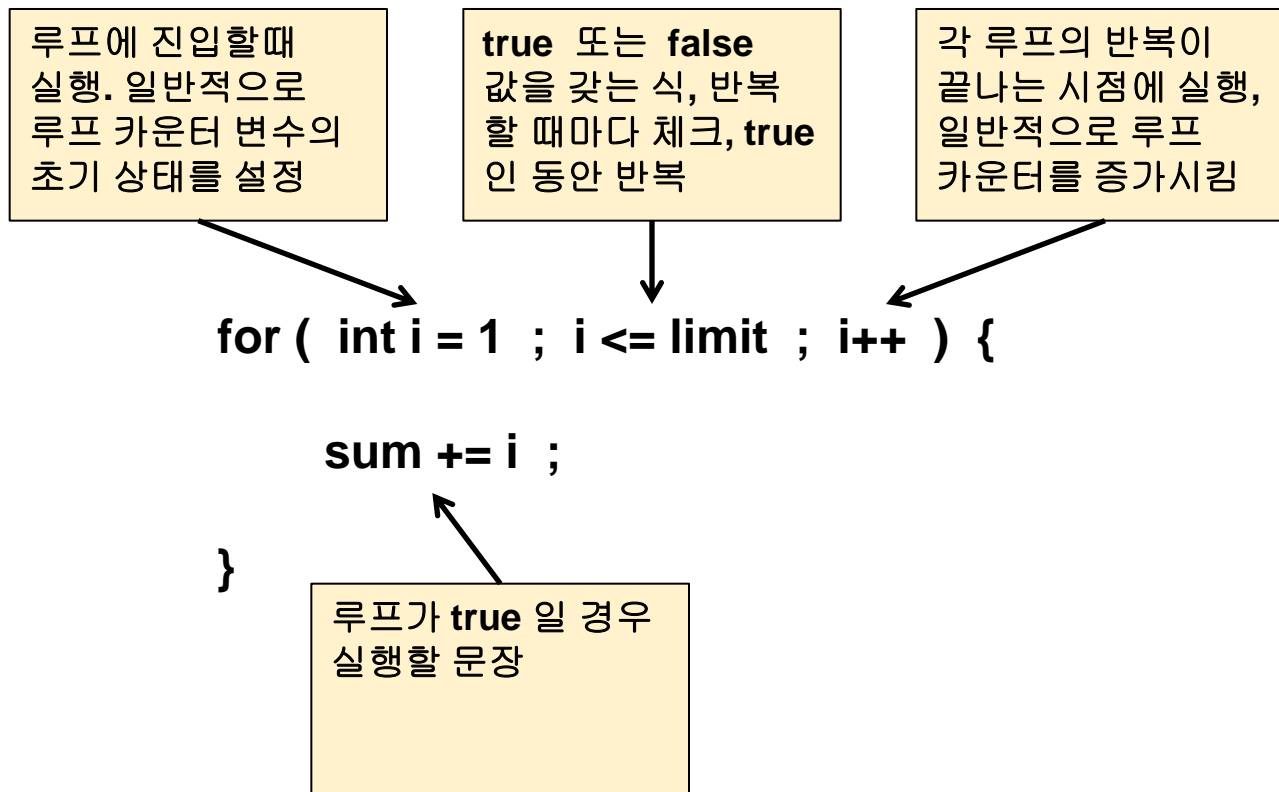
결과

1
2
3

4. 반복문

예제작성

➤ for



4. 반복문

예제작성

- for

- 형식 **for(1. 초기값 ; 2. 조건 ; 3. 증감) {**
 4. 반복문장들
 }
 5. 반복문 빠져나옴

- 실행순서

- 1 - 2(조건이 참일 경우) - 4 - 3
- 2(조건이 참일 경우) - 4 - 3
- 2(조건이 거짓일 경우) - 5

```
for(n=1; n < 6; n++)  
    System.out.print(n + " ");  
}
```

=====

예상결과??

1 2 3 4 5

4. 반복문

예제작성

- while

- ♦ 조건절로 지정된 조건이 참일 동안 **while** 블록을 실행

```
while(조건절) {  
    반복문장들  
}
```

예>

```
int a=10, b=20;
```

```
while(a > b)
```

```
System.out.println("이 문장은 영원히 나타나지 않는다");
```

4. 반복문

- do ~ while

do-while문은 조건을 나중에 평가한다
while 블록이 적어도 한번은 수행

```
do {  
    반복문장들  
} while(조건절);
```

ex> int a = 5, b = 10;

```
do {  
    System.out.println("무조건 실행됨");  
} while (a > b)  => 1번 실행됨.
```

while 와 **do-while**의 차이점
- **do-while** 은 무조건
한번은 실행. **While**은
실행이 안될 수 도 있다
다시 말해서
같은 조건일 경우 **do-while**
문만 실행 될 수 있다.

- break

break문의 3가지 역할

switch문에서 **switch**문을 벗어나는데 사용

반복문에서 반복루프를 벗어나는데 사용

중첩된 반복문을 한번에 빠져나갈때

- continue

반복문의 특정지점에서 제어를 반복문의 처음으로 보낸다

4. 제어

예제작성

- break

```
int i = 1;
while(i < 100) {
    if(i == 10) break;
    System.out.println(i + "자바의 세계로 오세요");
    i++;
}
```

결과는 ??

1 자바의 세계로 오세요!
2 자바의 세계로 오세요!
3 자바의 세계로 오세요!
4 자바의 세계로 오세요!
5 자바의 세계로 오세요!
6 자바의 세계로 오세요!
7 자바의 세계로 오세요!
8 자바의 세계로 오세요!
9 자바의 세계로 오세요!

4. 제어

예제 작성

- break

반복문이 중첩되었을 경우 가장 가까운 반복문을 빠져 나온다

```
int i, j;
for(i=1 ; i<=5 ; i++) {
    for(j=1 ; j<=i ; j++) {
        if (j > 3) break; // 내포된 반복문
        System.out.print(" * ");
    }
    System.out.println();
}
```

i = 1일 때
*
i = 2일 때
* *
i = 3일 때
* * *
i = 4일 때
* * *
i = 5일 때
* * *

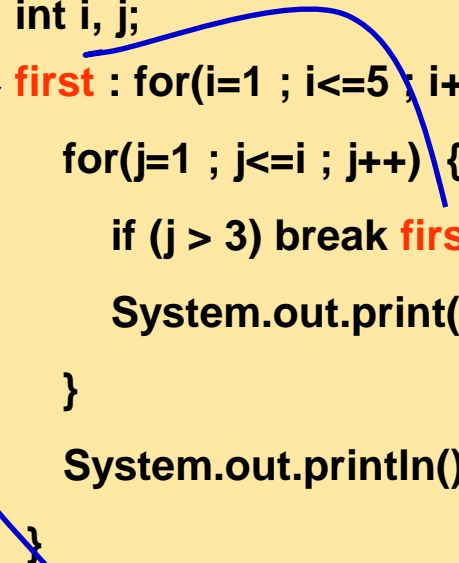
결과

```
*
* *
* * *
* * *
* * *
```

- break

중첩된 반복문을 한번에 빠져 나오기

```
int i, j;  
first : for(i=1 ; i<=5 ; i++) {  
    for(j=1 ; j<=i ; j++) {  
        if (j > 3) break first; // first라는 이름의 블록을 벗어난다.  
        System.out.print(" * ");  
    }  
    System.out.println();  
}
```



- continue

반복문의 특정지점에서 제어를 **반복문의 처음으로** 보낸다

```
예> class ContinueTest {  
    public static void main(String args[]) {  
        for(int i=0; i<10; i++) {  
            if (i%2 == 0) continue;  
            System.out.println(i + " 자바의 세계로 오세요! " );  
        }  
    }  
}
```

결과는??

1 자바의 세계로 오세요!
3 자바의 세계로 오세요!
5 자바의 세계로 오세요!
7 자바의 세계로 오세요!
9 자바의 세계로 오세요!



배 영

배열

- 배열이란?

나가수경연	임재범	김범수	박정현
1차 경연	40.1%	29.7%	30.2%
2차 경연	30.1%	35.7%	34.2%

만약, 가수가 늘어난다면??

String name = “임재범”;

String name2 = “김범수”;

String name3 = “박정현”;

double first = 40.1;

double first2 = 29.7;

double first3 = 30.2;

double second = 30.1;

double second2 = 35.7;

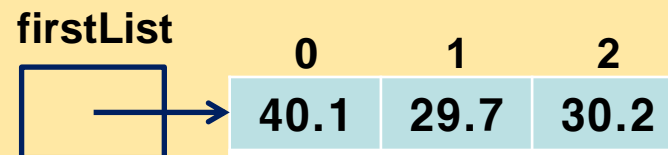
double second3 = 34.2;

- 배열이란?

- 같은 종류의 데이터를 저장하기 위한 자료구조
- 크기가 고정되어 있다(한번 생성된 배열은 크기를 바꿀 수 없다)
- 배열을 객체로 취급
- 배열의 요소를 참조하려면 배열이름과 색인(index)이라고 하는
- **int** 유형의 정수값을 조합하여 사용한다.

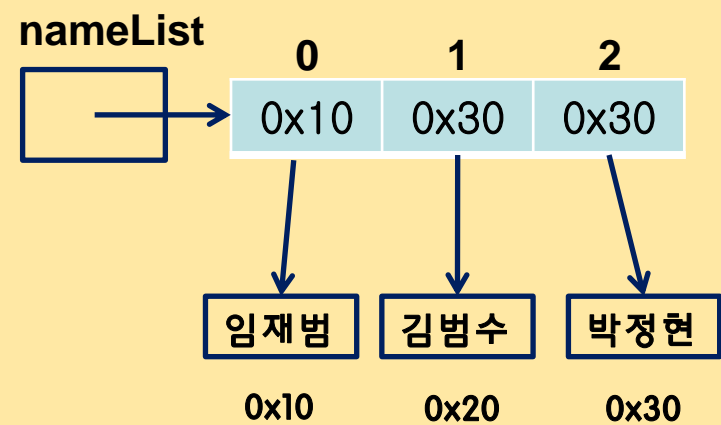
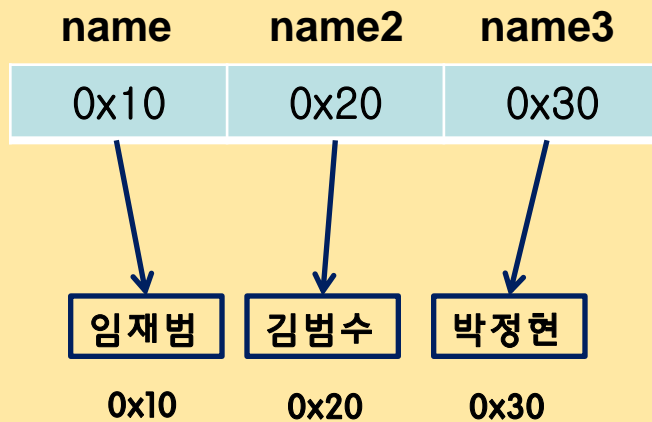
- 배열

first	first2	first3
40.1	29.7	30.2



배열

- 배열



- 배열의 선언

- [] 는 자바에서 배열을 나타낸다.
- [] 의 개수가 배열의 차원수를 나타낸다.
예를 들어, [] 일 경우 1차원, [][] 일 경우 2차원이 된다.

- 1차원 배열 선언

배열유형 배열이름 [] 또는 배열유형[] 배열이름

ex) int prime [], int [] prime

- 다차원 배열 선언

배열유형 배열이름 [][] 또는 배열유형[][] 배열이름

ex) int prime [][], int [][] prime

* 배열의 유형은 모든 것이 가능하다(기본형, 참조형)

배열

- 배열의 선언

타입	배열이름	선언
int	iArr	int [] iArr;
char	cArr	char [] cArr;
boolean	bArr	boolean [] bArr;
String	strArr	String [] strArr;
Date	dateArr	Date [] dateArr;

- 배열의 선언

`int [] dongList;`

dongList



하나의 값을 저장할 수
있는 메모리 생성

`int [][] aptInfoList;`

aptInfoList



하나의 값을 저장할 수
있는 메모리 생성

- 배열의 생성

- 1차원 배열

배열의 이름 = **new** 배열유형 [배열크기];

ex) **prime = new int [10]**

- 2차원 배열

배열의 이름 = **new** 배열유형[1차원배열개수][1차원배열의크기];

배열의 이름 = **new** 배열유형[1차원배열개수] **[]**;

ex) **prime = new int [3][2];**

prime = new int [3][];

- 자동 초기화

- 배열이 생성되면 자동적으로 배열요소는 기본값으로 초기화된다.

ex) int : 0

boolean : false

char : '\u0000'

참조형 : null....

- 멤버변수와 로컬변수 모두 배열이 생성이 되면 자동 초기화된다.

- 배열의 생성

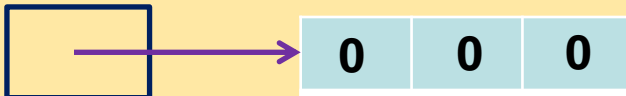
`int [] dongList;`

`dongList`



`dongList = new int[3];`

`dongList`



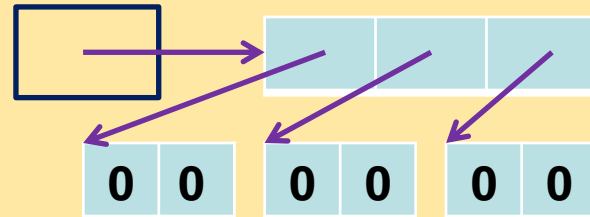
`int [][] aptInfoList;`

`aptInfoList`



`aptInfoList = new int[3][2];`

`aptInfoList`



- 배열의 생성

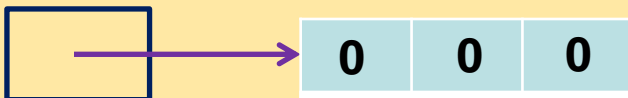
```
int [ ] dongList;
```

dongList



```
dongList = new int[3];
```

dongList



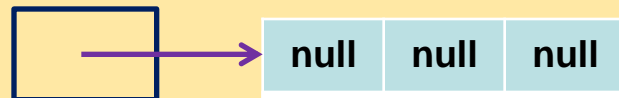
```
int [ ][ ] aptInfoList;
```

aptInfoList



```
aptInfoList = new int[3][ ];
```

aptInfoList



- 초기화

- 1차원 배열

배열이름[인덱스] = 값;

ex) `prime[0] = 100;`

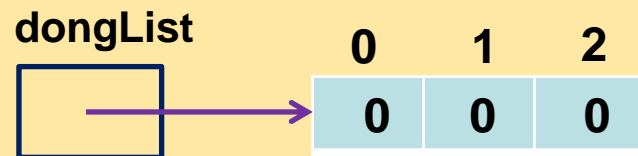
- 2차원 배열

배열이름[인덱스][인덱스] = 값;

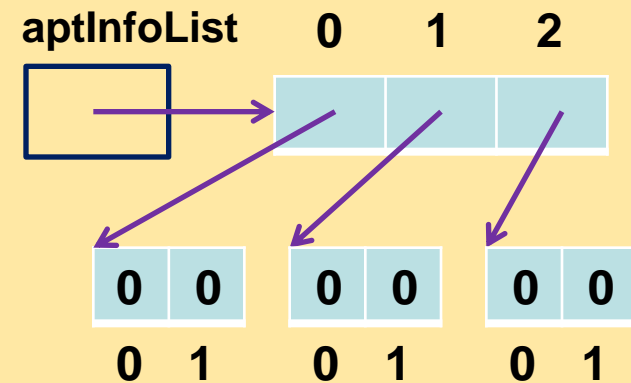
ex) `twoArr[0][1] = 100;`

- 배열의 초기화

```
int [ ] dongList = new int[3];
```



```
int [ ][ ] aptInfoList =  
    new int[3][2];
```



- 배열의 초기화

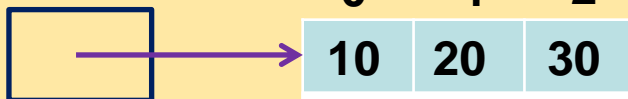
```
int [ ] dongList = new int[3];
```

```
dongList[0] = 10;
```

```
dongList[1] = 20;
```

```
dongList[2] = 30;
```

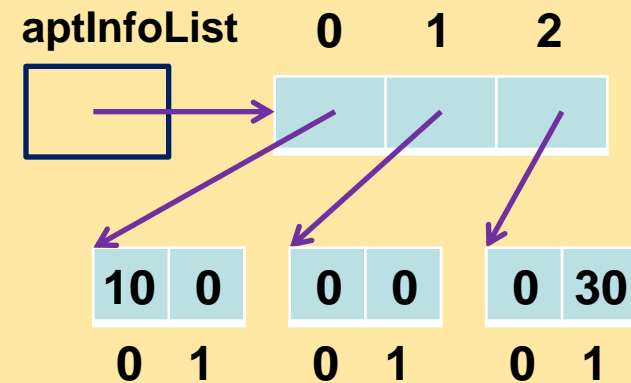
dongList



```
int [ ][ ] aptInfoList =  
    new int[3][2];
```

```
aptInfoList[0][0] = 10;
```

```
aptInfoList[2][1] = 30;
```



- 배열의 초기화

```
int [ ] dongList = new int[3];
```

- 배열의 인덱스는 0부터 시작
- 배열의 크기 : 배열이름.length
- 마지막 요소 인덱스 : 배열크기 - 1

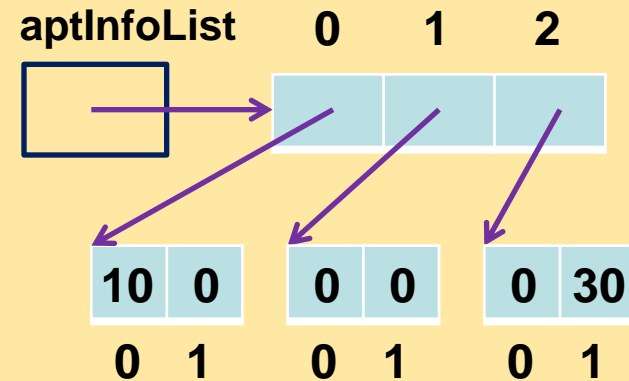
aptInfoList.length ?? 3

aptInfoList[0].length ?? 2

```
int [ ][ ] aptInfoList =  
    new int[3][2];
```

```
aptInfoList[0][0] = 10;
```

```
aptInfoList[2][1] = 30;
```



- 초기화

- { } 를 활용하는 방식 : 배열 선언 시에만 설정 가능

1차원 배열 : 배열유형 [] 배열명 = { 값, .. 값 };

ex) int [] prime = { 1, 2, 3 };

2차원 배열 : 배열유형 [][] 배열명 = { { 값1, 값2 }, { 값3, 값4 } };

ex) int [][] twoArr = { { 1, 2 }, { 3, 4 }, { 5, 6 } };

- new 배열타입[] { 값,}

ex) int [] prime = new int[] { 1, 2 };

- 배열관련 제공 API

- **`System.arraycopy(src, srcPos, dest, destPos, length)`**

src : 원본배열 **srcPos** : 원본배열의 복사 시작 위치(0부터 시작)

dest : 복사할 배열 **destPos** : 복사 받을 시작 위치

length : 복사할 크기

예) `String [] oriArr = {“봄”, “여름”, “가을”};`

`String [] destArr = new String[oriArr.length + 1];`

`System.arraycopy(oriArr, 0, destArr, 0, oriArr.length);`

`destArr[3] = “겨울”;`

`for(int i = 0; i < destArr.length; i++)`

`System.out.println(destArr[i]);`

- 배열관련 제공 API

- **Arrays.toString(배열객체)**

: 배열안의 요소를 [요소, 요소, ..] 의 형태로 출력

예)

```
for( int i = 0; i < destArr.length; i++)
```

```
    System.out.println(destArr[ i ]);
```

```
=====
```

단순 배열값 확인일 경우

```
System.out.println( Arrays.toString(destArr) );
```