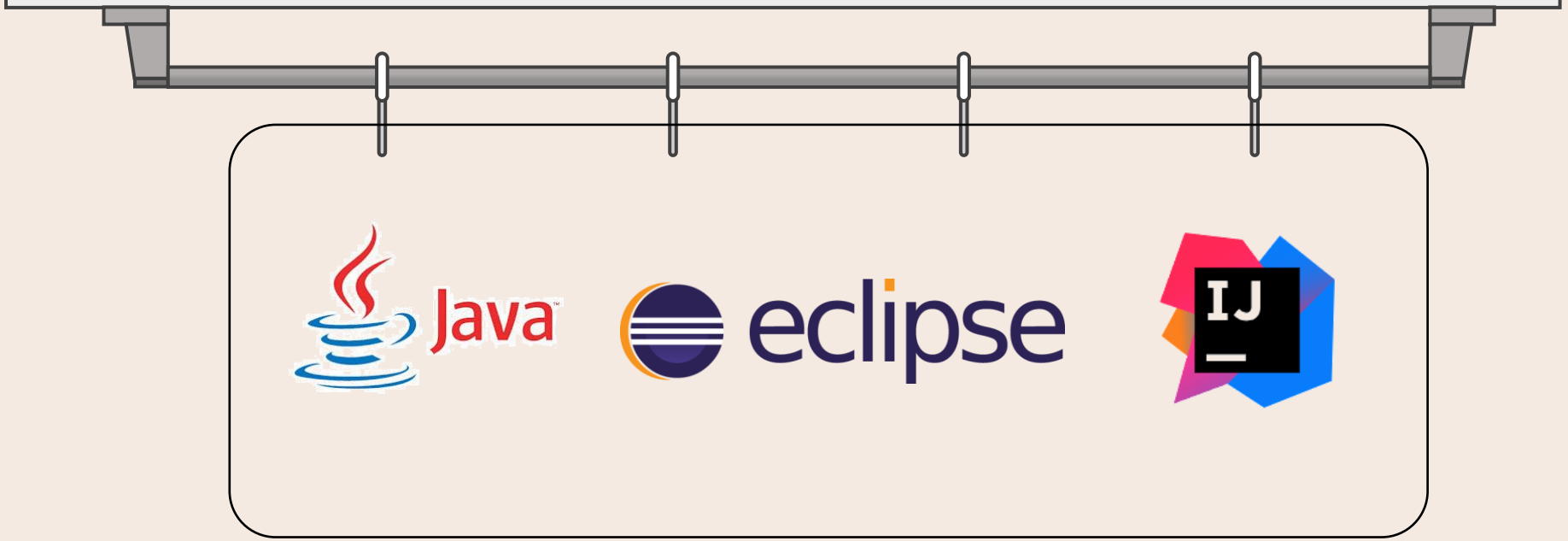
자바 Programming

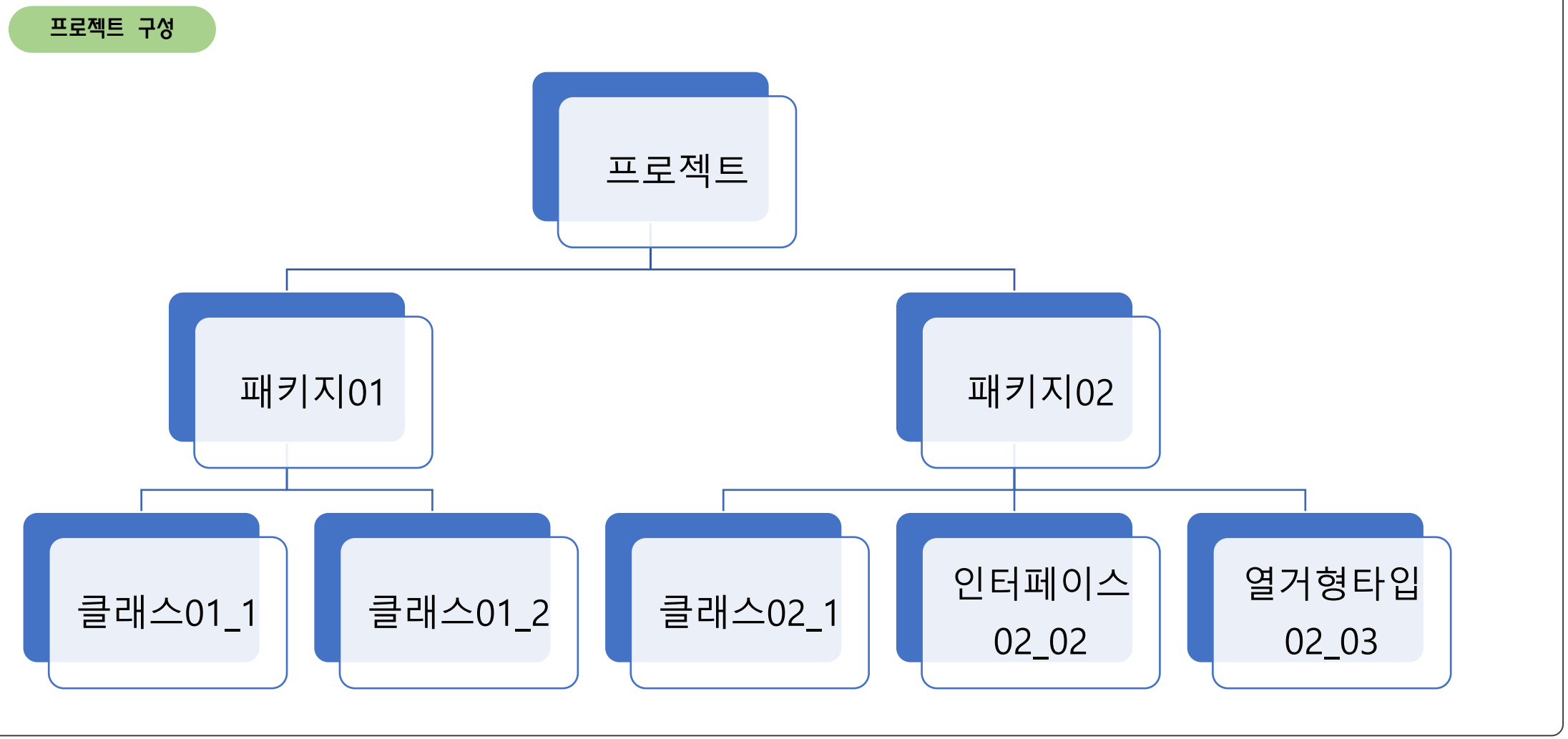
자바 개요



자바 개요

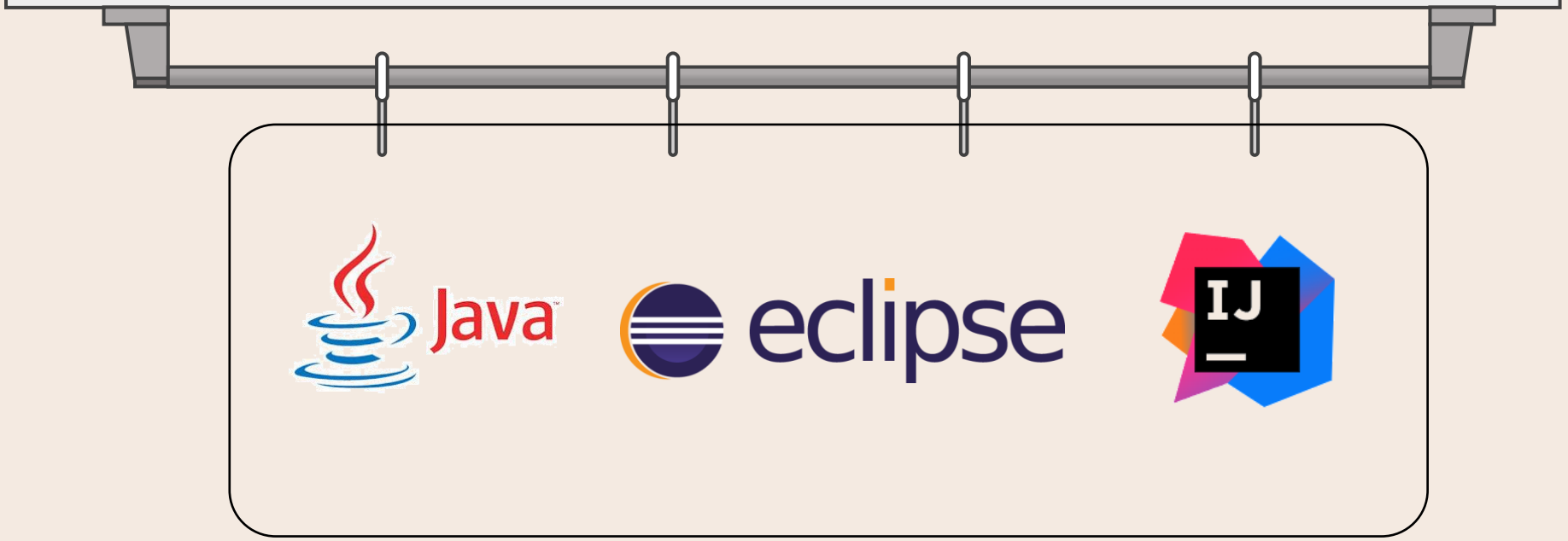
프로젝트 구성

자바에서 프로젝트는 다음과 같은 요소들로 구성이 되어 있습니다.



자바 Programming

변수와 연산자



변수와 연산자

자바 프로그래밍

자바 프로그래밍은 크게 다음과 같이 3단계로 이루어집니다.

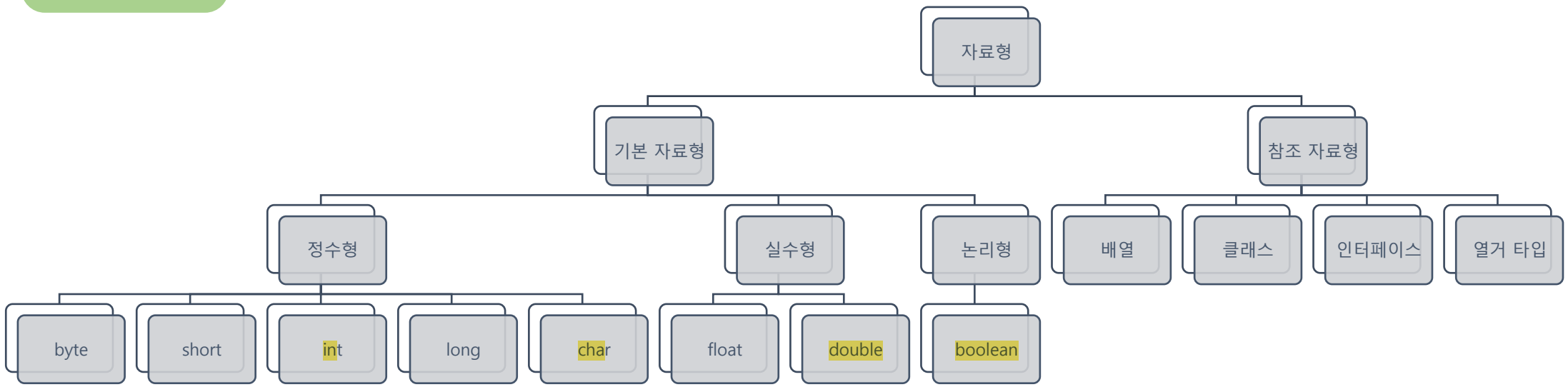
자바 기초	객체 지향	자바 심화
• 변수 • 연산자 • 제어문 • 배열 • 메소드	• 클래스/객체 • 생성자 • 접근 지정자 • this/static • 상속 • 패키지 • 추상 클래스 • 인터페이스	• 예외 처리 • 쓰레드 • 컬렉션 • 네트워크 • GUI

변수와 연산자

자료형

자바의 자료형은 크게 기본 자료형과 참조 자료형으로 구분됩니다.

자바의 자료형



정수형	문자형	실수형	논리형
byte(1바이트)	char(1바이트)	float(4바이트)	boolean(1바이트)
short(2바이트)		double(8바이트)	
int(4바이트)			
long(8바이트)			

자료형의 바이트

예를 들어서, 정수 데이터가 1개가 필요하다고 가정하면 컴퓨터 메모리에 4바이트를 저장 공간이 필요하다는 의미입니다.

변수와 연산자

출력 관련 메소드

자바에서 출력과 관련 메소드들을 살펴 봅니다.

println(), print(), printf() 메소드는 모두 콘솔에 출력을 수행하는 데 사용되지만, 각 메소드의 동작 방식과 사용 목적이 조금씩 다릅니다.

메소드	설명	주요 특징
println	문자열 또는 값을 출력한 후 줄 바꿈(newline)을 추가합니다. 출력 후 자동으로 다음 줄로 넘어갑니다.	"ln"은 "line"의 약자로, 줄 바꿈이 포함됨을 의미합니다.
print	문자열 또는 값을 출력하지만, 줄 바꿈을 하지 않습니다.	줄 바꿈 없이 같은 줄에 계속 출력을 이어서 진행할 때 사용됩니다.
printf	형식을 지정하여 출력할 수 있는 메서드로, 포맷 문자열을 사용합니다. 서식, 형식이 포맷이다.	C 언어의 printf와 유사하며, 줄 바꿈 기능이 없습니다. 다양한 형식 지정자(%d, %s, %f 등)를 사용하여 값을 포맷팅합니다.

%d는 정수
%s는 문자열
%f는 실수

사용 예시

```
System.out.println("홍");
System.out.println();
System.out.println("길");
System.out.println("동");

System.out.print("대한 ");
System.out.print("민국 ");
System.out.print("화이팅 ");
```

실행 결과

```
홍

길
동
대한 민국 화이팅
```

변수와 연산자

특수 문자

문자열 내에서 특정한 의미를 가지는 특수한 문자들을 의미하며, 이스케이프(escape) 시퀀스라고 합니다.

문자열 내에서 특별한 의미를 가지는 몇 가지 특수 문자들이 있습니다.

이러한 특수 문자(eacape code)는 프로그래밍할 때 사용할 수 있도록, 미리 정의해 놓은 문자열 조합을 의미합니다.

이러한 특수 문자들은 백슬래시(\)와 함께 사용되며, 주요한 특수 문자들과 그 의미는 다음과 같습니다.

항목	설명
\n	줄 바꿈 문자. 문자열 내에서 이를 사용하면 다음 줄로 이동합니다.
\t	탭 문자. 문자열 내에서 이를 사용하면 탭 공백을 삽입합니다.
\\	백슬래시 자체를 나타내는 이스케이프 시퀀스입니다.
\'	작은 따옴표를 나타내는 이스케이프 시퀀스입니다.
\"	큰 따옴표를 나타내는 이스케이프 시퀀스입니다.
\r	캐리지 리턴(줄 바꿈 문자, 현재 커서를 가장 앞으로 이동)
\f	폼피드(줄 바꿈 문자, 현재 커서를 다음 줄로 이동)
\a	벨 소리(출력시 PC 스피커에서 소리가 난다.)
\b	백 스페이스 문자
\000	널 문자

변수와 연산자

서식(format) 문자열

문자열에 형식을 지정하여 값을 출력하거나 조합할 때 사용됩니다.

서식(format)이란 숫자, 문자, 날짜의 포맷을 지정하는 동작이고, 자바에서는 printf() 메소드 또는 String.format()가 서식을 지정해주는 역할을 합니다.
그리고, 다양한 데이터 유형을 출력하기 위해 형식 지정자(Format Specifiers)를 활용합니다.
예를 들어서 서식 지정자를 사용하면 엔터 키를 누르는 기능이 없으므로, '\n' 등의 특수 문자를 엔터키 효과를 줄 수 있습니다.

메소드	설명	주요 특징
%d	10진수 정수 출력	System.out.printf("%d", 10);
%f	소수점이 포함된 실수 출력	System.out.printf("%.2f", 3.14159); 소수점 둘 째 자리까지 표시하라.
%s	문자열 출력	System.out.printf("%s", "Hello");
%c	단일 문자 출력	System.out.printf("%c", 'A');
%b	불리언 값 출력	System.out.printf("%b", true);
%n	줄 바꿈	System.out.printf("Line 1%nLine 2");
%x	16진수 출력	System.out.printf("%x", 255);
%o	8진수 출력	System.out.printf("%o", 8);

데시멀 decimal

```
System.out.printf("이름 : %s님\n", name);
System.out.printf("나이 : %d세\n", age);
```

변수와 연산자

주석(comment)

주석(Comment)은 프로그램 코드 내에 포함된 설명이나 메모를 의미합니다.

주석은 프로그램을 실행할 때 무시가 되며, 주석은 코드를 이해하기 쉽게 만들고, 다른 사람과 협업할 때 코드의 의도를 명확히 전달할 수 있습니다.

또한 나중에 코드를 수정할 때 이전에 작성한 주석을 참고하여 작업할 수 있습니다.

Python에서는 크게 한 줄 주석과 여러 줄 주석을 사용할 수 있습니다.

항목	설명
한 줄 주석	코드 한 줄에 대한 주석을 작성하고자 할 때 사용합니다. 슬래시 기호 2개 `//`를 사용하며 기호 뒤에 오는 모든 내용은 주석으로 처리가 됩니다.
여러 줄 주석	여러 줄에 걸쳐 주석을 작성할 때 사용합니다. 여러 줄 주석은 `/*`로 시작하고, `*/`로 종료합니다.

사용 예시

```
int x = 5 // 이것은 한 줄 주석입니다.


/*
이것은
여러 줄 주석입니다.
여러 줄에 걸쳐서 설명할 때 사용합니다.
*/
```

변수와 연산자

식별자 명명 규칙

식별자(identifier)란 변수/배열/메소드/클래스/인터페이스/예외 객체 등에 붙여 지는 고유한 이름을 의미합니다.

식별자 명명 규칙

다음과 같은 규칙을 따라야 합니다.

(1)첫 글자는 '_', '$', 영문 대문자, 소문자 (한글 가능)

(2)글자수에 제한 없습니다.

(3)공백 문자 및 특수 문자는 사용이 불가능합니다.

(4)숫자는 첫 글자가 아닐 때 사용 가능합니다.

(5)예약어(reversed word)/키워드는 사용 불가능합니다.

(6)기타 단순 약속 (대, 소문자의 규칙)

문제 풀기

다음 중 올바른 변수 명을 모두 고르시오.

1) sum 2) 300sum 3) System 4) MyHome

다음 변수들은 문법적인 오류가 없나요?

String age ;

int name ;

사용 예시

```
int a=10, A=20; (O)
int _a=100; (O)
int 100a=15; (X)
int kor25=100; (O)
int k g h = 50; (X)
int System=100; (X)
```

표기법

낙타봉(Camel Case) 표기법

String welcomeMyHome

헝가리언 표기법

int iage ;

String strName ;

변수와 연산자

식별자 명명 규칙

다음은 식별자(identifier)로 사용할 수 없는 자바의 예약어 목록입니다.

자바 예약어 목록

abstract / assert / boolean / break / byte / case / catch / char / class

const / continue / default / do / double / else / enum / extends / final

finally / float / for / goto / if / implements / import / instanceof / int

interface / long / native / new / package / private / protected / public

return / short / static / super / switch / synchronized / this / throw

throws / try / void / while

변수와 연산자

변수와 상수

변수는 수정 가능, 상수는 수정이 불가능한 식별자입니다.

변수(Variable)는 프로그램 실행 중에 값의 변경이 가능한 식별자를 의미하고, 상수(Constants)는 프로그램 실행 중에 절대 변경할 수 없는 식별자를 의미합니다.

특징	변수(Variable)	상수(Constant)
값 변경 여부	값을 변경할 수 있음	값을 변경할 수 없음
선언 방법	데이터 타입과 변수명만 선언	final 키워드와 함께 선언
초기화 요구 사항	초기화하지 않아도 되며, 나중에 할당 가능	선언 시 반드시 초기화 필요
용도	변하는 값을 저장할 때 사용	변하지 않는 고정된 값을 저장할 때 사용
명명 규칙	보통 소문자 카멜 케이스(myValue) 사용	보통 대문자와 언더스코어(MAX_VALUE) 사용

변수와 상수의 공통점

둘 다 특정 데이터 타입을 가지며, 메모리 공간에 값을 저장합니다.

변수와 상수 모두 클래스, 메서드, 블록 등 다양한 스코프에서 선언 가능하며, 접근 제한자를 사용할 수 있습니다.

변수와 연산자

변수와 상수

변수는 수정 가능, 상수는 수정이 불가능한 식별자입니다.

변수(Variable)는 프로그램 실행 중에 값의 변경이 가능한 식별자를 의미하고, 상수(Constants)는 프로그램 실행 중에 절대 변경할 수 없는 식별자를 의미합니다.

자료형	설명	키워드	예시
상수	프로그램 실행 중에 값 변경이 불가한 데이터를 의미합니다.		
정수형	정수형은 정수를 나타내는 자료형입니다. 양수, 음수, 0을 모두 포함합니다.	int	int 변수 = 123 ;
실수형	부동 소수점형은 실수를 나타내는 자료형입니다. 소수점을 가지고 있거나 지수 표기법을 사용할 수 있습니다. float 형은 숫자 뒤에 f를 붙여 줍니다.	float 또는 double	float 변수 = 45.678f ; double 변수 = 12.3456 ;
문자형	1개의 영문자, 숫자, 특수 문자 등을 표현. 문자는 반드시 외따옴표로 둘러 싸야 합니다.	char	char 변수 = ´A´ ;
문자열형	일련의 문자열의 집합으로써, 반드시 쌍따옴표로 묶어서 표현해야 합니다.	String	String 변수 = "Good" ;
논리형	참과 거짓을 표현하기 위하여 사용하는 타입입니다.	boolean	boolean flag = true ;

변수와 연산자

변수의 선언과 값의 할당

변수를 선언하는 방법과 값을 할당하는 방법에 대하여 살펴 봅니다.

변수 선언

```
// 타입을 먼저 작성하고, 변수의 이름을 명시합니다.
// 선언 구문 이후에 세미 콜론을 붙여 줍니다.
int age ; // 정수형 데이터를 위하여 변수 age를 준비할께요.

// 사람의 이름(문자열) 정보를 위하여 변수 name을 준비할께요.
String name ;

// 동일한 타입을 2개 이상 선언하고자 하는 경우, 타입은 1번만 명시하고 콤마를 사용하면 됩니다.
int korean, englinsh, math ;

// 다음과 같이 동일한 타입이더라도, 별개의 라인으로 작성할 수 있습니다.
String hello ;
String world ;
String everyone ;
```

```
// 선언이 된 이후, 재선언이 불가능합니다.
int age ; // 이미 선언이 되었으므로, 오류 발생

// 선언이 된 이후, 다른 타입으로 재선언이 불가능합니다.
String age ; // 이전 선언시 정수형이었습니다.
```

변수와 연산자

변수의 선언과 값의 할당

변수를 선언하는 방법과 값을 할당하는 방법에 대하여 살펴 봅니다.

값의 할당

```
// 사람의 이름은 문자열로 처리할 수 있습니다.
String name ;

// 사람의 나이는 정수형 처리할 수 있습니다.
int age ;

// = 기호의 우측 값 "홍길동" 을 왼쪽 변수 name에 대입(할당)합니다.
name = "홍길동" ;

// = 기호의 우측 값 20 을 왼쪽 변수 age에 대입(할당)합니다.
age = 20 ;

// 변수는 선언과 동시에 값을 대입할 수 있습니다.
double height = 175.8 ;
```

```
// 변수 name의 이전 값은 "홍길동"이었는 데 이번에 "김철수"로 변경하겠습니다.
name = "김철수" ;

// 변수 age에는 정수 데이터를 할당할 수 있는 문자열 "박영희"로 변경하려고 하므로 오류입니다.
age = "박영희" ;
```

변수와 연산자

변수의 선언과 값의 할당

변수를 선언하는 방법과 값을 할당하는 방법에 대하여 살펴 봅니다.

값의 할당


```
int x ;

x = 3 ;

x = 4 ;
```


```
int y = 5 ;
```

변수와 연산자

자바 변수(Variable)의 연산

자바 프로그램에서 수식은 연산자와 피연산자로 구성이 되어 있습니다.

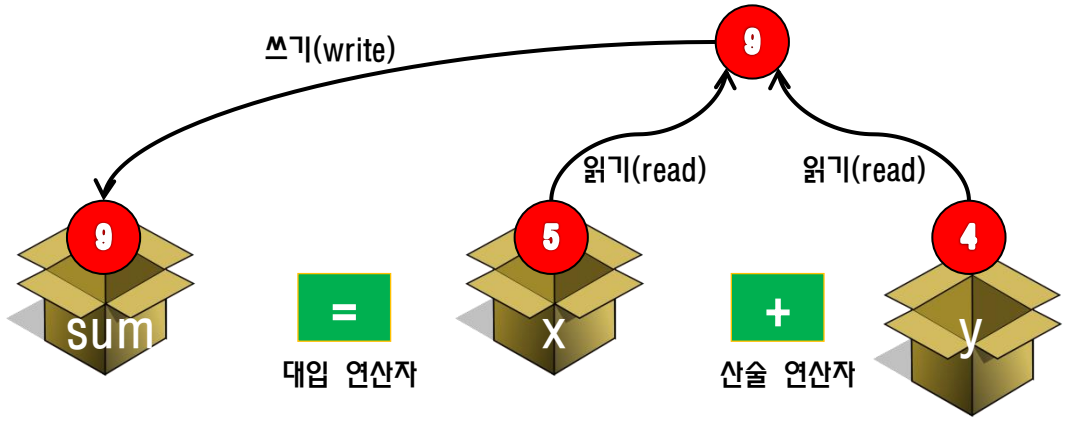
항목	설명
연산자(Operator)	연산 작업을 수행하기 위해 사용하는 기호(symbol)나 키워드입니다.
피연산자(Operand)	연산자의 입력 값으로 사용되는 데이터입니다. 즉, 연산자가 처리하는 대상으로 리터럴, 변수, 상수, 결과 값 등이 될 수 있습니다.

사용 예시

변수 2개의 값을 덧셈하여 다른 변수에 대입하기

```
int x = 5
int y = 4
int sum = x + y
```

선언, 정의 X
할당, 대입 O



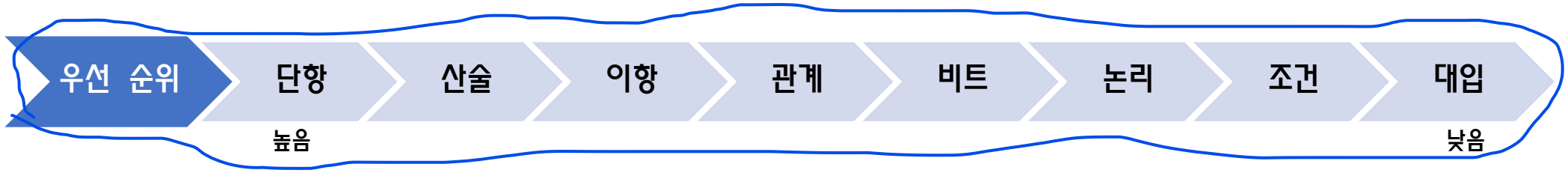
연산자의 우선 순위 : 산술 > 대입

변수와 연산자

자바의 연산자

자바 프로그램에서는 다음과 같은 연산자들이 지원됩니다.

연산자 우선 순위



연산자의 특징

연산은 코드의 좌측에서 우측 방향으로 진행하되, 곱셈/나눗셈이 덧셈/뺄셈보다 우선 순위가 높습니다.

예시) 3 + 5 * 2

소괄호를 사용하면 연산자의 우선 순위를 변경할 수 있습니다.

예시) (3 + 5) * 2

연산자는 실행 우선 순위가 존재합니다. 단산이관비논조대

연산자는 단항, 산술, 이항, 관계, 비트, 논리, 조건, 대입 연산자 순으로 우선 순위가 높습니다.

변수와 연산자

자바의 연산자

자바 프로그램에서는 다음과 같은 연산자들이 지원됩니다.

연산자의 종류

이름	연산자	처리 기능
부호 연산자	+ -	-는 부호 반전 역할
산술 연산자	+ - * / %	산술 계산
증감 연산자	++ --	1씩 증가/감소
대입 연산자	= += -= *= /= &= |=	연산 결과 대입
관계 연산자	< <= > >= == !=	크기 비교
논리 연산자	&& || !	논리 계산
비트 연산자	& | ~ ^ >> <<	비트 단위 처리
조건 연산자	조건식 ? 참 : 거짓	조건 연산의 간결한 표현
캐스트 연산자	(type)	형변환

연산자 사용 예시

```
int x = 10;// ´=´ 기호를 대입 연산자라고 합니다.(값 10을 x에 대입)
int y = 20 ;
int sum = x + y ; // ´+´ 기호를 산술 연산자라고 부릅니다..
```

변수와 연산자

자바의 연산자

피연산자의 개수에 따른 연산자의 분류는 다음과 같습니다.

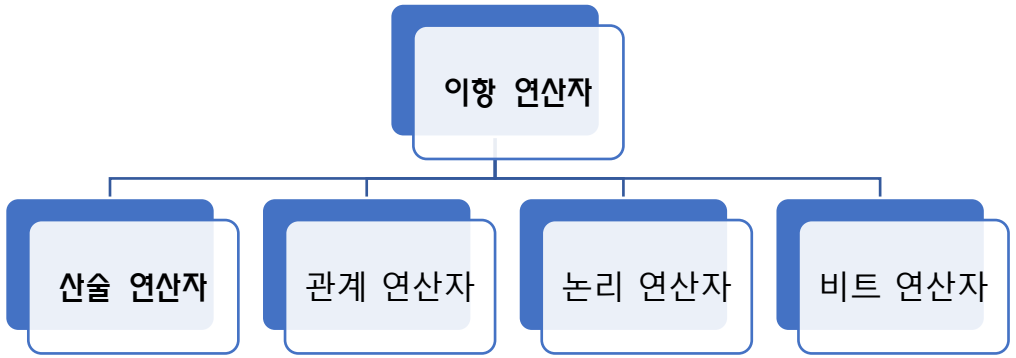
피연산 수에 따른 분류

이름	연산자
단항 연산자	+(부호) -(부호) ++ -- ! ~ (type)
이항 연산자	+ - * / %
대입 연산자	= += -= *= /= &= |= ^= >>= <<=
관계 연산자	< <= > >= == !=
논리 연산자	&& || !
삼항 연산자	? :

사용 형태에 따른 항의 개수

연산자	사용 예시
단항 연산자	-5, ++a, b--
이항 연산자	5 + 4, x * y
삼항 연산자	int a = 3, b = 5 ; int result = 0 ; result = a >= b ? a - b : b - a ;

이항 연산자는 두 개의 피연산자를 취하는 연산자입니다.



변수와 연산자

산술 연산자

피연산자에 대하여 덧셈, 뺄셈, 곱셈, 나눗셈 및 나머지 연산을 수행하는 수치 연산자입니다.

크게 단항 연산자와 이항 산술 연산자로 나뉘어 집니다.

연산은 좌측 방향에서 우측으로 연산이 수행이 되되, 곱셈/나눗셈/나머지(*, /, %) 연산이 +, - 연산자보다 먼저 실행됩니다.

소괄호를 사용하면 연산의 우선 순위를 변경할 수 있습니다.

데이터 타입이 다른 경우 작은 데이터 타입이 큰 타입으로 변경된 이후에 연산이 됩니다.

구분	연산자	의미	수학적 표현	자바 표현	자바 결과	비고
단항 산술 연산자	+	양수	+3	+3	3	
	-	음수	-2(-2)	-2(-2)	4	동일 부호이면 +가 됩니다.
이항 산술 연산자	+	덧셈	3+2	3+2	5	
	-	뺄셈	10-2	10-2	8	
	*	곱셈	xy 또는 x*y	2*4	8	
	/	나눗셈	x/y	8/2	4	
	%	나머지	x mod y	9%2	1	모두 정수형이어야 함.

변수와 연산자

증감 연산자

변수의 값을 1씩 증가시키거나 감소시키는 데 사용하는 연산자입니다.

++ 기호와 -- 기호 2가지 종류가 있습니다.

연산자의 위치에 따라서 전위/후위라는 용어가 붙습니다.

변수가 아닌 수식에는 사용이 불가능합니다.(예시 : (x+y)++)

실수형에도 적용이 가능합니다.

´전위´는 모든 연산자 보다 먼저 실행됩니다.

´후위´는 모든 연산자 보다 나중에 실행됩니다.

-후위 증감 연산자는 해당 줄의 연산이 끝나면 증감된다.

-연산은 해당 변수의 값이 사용되는 순간을 의미하며, 변수가 계산에 쓰이거나, 다른 변수에 할당되거나, 출력되는 등 값이 실제로 참조되는 시점을 뜻한다.

형태	연산자 호칭	연산의 의미
++x	전위 증가 연산자	연산하기 전에 x값을 1 증가시킵니다. [연산하기 전에 우선 +1 수행하세요]라는 의미입니다. x의 값을 우선 +1증가 후 연산 수행한다는 개념입니다.
x++	후위 증가 연산자	연산한 다음에 x값을 1 증가시킵니다.
--x	전위 감소 연산자	연산하기 전에 x값을 1 감소시킵니다.
x--	후위 감소 연산자	연산한 다음에 x값을 1 감소시킵니다.

전위 연산자는 모든 연산자 중 우선순위 1등

후위 연산자는 모든 연산자 중 우선순위 꼴등

변수와 연산자

관계 연산자

관계 연산자 = 비교 연산자

두 개의 값 또는 표현식을 비교하여 관계를 평가하는 데 사용되어, 비교 연산자라고 부르기도 합니다.

이러한 관계 연산자들은 두 개의 값 또는 표현식을 비교하여 관계를 평가하고, 이를 통해 참(true) 또는 거짓(false)을 반환합니다.

조건문 특히 제어문(for, while, do…while)의 조건 검사식으로 주로 사용되며, 프로그램에서 값들의 상대적인 크기나 동등 여부를 비교할 때 유용하게 사용됩니다.

예를 들면 `5 > 3`의 실행 결과는 `true`가 됩니다. 제어문 : 조건문(if, if-else, switch), 반복문(for, while, do-while), 분기문(break, continue, return)

관계 연산자

구분	항목	설명	세부 설명	사용 예시	
비교 연산자	<	~보다 작음	첫 번째 숫자가 두 번째 숫자 보다 작은지 검사합니다.	3 < 8	true
	>	~보다 큼	첫 번째 숫자가 두 번째 숫자 보다 큰지 검사합니다.	3 > 8	false
	<=	작거나 같음	첫 번째 숫자가 두 번째 숫자 보다 작거나 같은지 검사합니다.	3 <= 8	true
	>=	크거나 같음	첫 번째 숫자가 두 번째 숫자 보다 크거나 같은지 검사합니다.	3 >= 8	false
항등 연산자	==	같음	두 대상이 같은지 검사합니다.	3 == 3 1 == 7	true false
	!= <>	같지 않음	두 대상이 같지 않은지 검사합니다.	3 != 3 1 != 7	false true

변수와 연산자

논리 연산자

논리적인 조건을 판단하거나 여러 조건을 결합하여 참(true)과 거짓(false)을 평가할 때 사용됩니다.

논리 연산자들은 조건을 결합하거나 평가할 때 사용이 되며, and, or, not 등의 3가지 주요한 논리 연산자가 있습니다.
예를 들어, if 문에서 여러 조건을 결합하여 논리적인 표현식을 만들거나, while 문에서 조건이 충족될 때까지 반복하는 등의 상황에서 사용됩니다.
논리 연산자는 조건문뿐만 아니라 필터링, 조건부 할당 등의 다양한 상황에서 유용하게 사용됩니다.

항목	설명	간단 설명
and 연산자	모든 조건이 참(true)일 때만 전체 표현식이 참이 됩니다.	이고, 그리고
or 연산자	조건 중 하나라도 참일 때 전체 표현식이 참이 됩니다.	이거나, 또는
not 연산자	피연산자의 불리언 값을 반대 상태로 변환합니다.	부정

논리 연산자 우선 순위

우선 순위 : not(!) > and(&&) > or(||)

높음 ~ 낮음

논리곱&논리합

식1	식2	논리곱 (&&)	논리합(||)
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

사용 예시

수식	식1	식2	연산자	결과
4 < 5 && 3 != 6	4 < 5	3 != 6	&&	true
4 > 7 || 3 == 6	4 > 7	3 == 6	||	false

boolean bool = !(4 > 7 || 3 == 6) ; 결과는 참

예문)

```
boolean a = true;
boolean b = false;
boolean c = true;

boolean result = !a || b && c;
System.out.println(result);
```

다음 페이지에 계속…

변수와 연산자

논리 연산자

논리적인 조건을 판단하거나 여러 조건을 결합하여 참(true)과 거짓(false)을 평가할 때 사용됩니다.

논리곱 부울 대수

논리곱(AND)을 "곱"이라고 부르는 이유는, 불 대수(Boolean Algebra)에서의 동작이 숫자 곱셈의 규칙과 유사하기 때문입니다. 불 대수에서 true는 1, false는 0으로 표현됩니다.
논리곱 연산은 다음과 같은 진리표와 결과를 가집니다:

식1	식2	논리곱 (&&)	곱셈 결과
0	0	0	0*0=0
0	1	0	0*1=0
1	0	0	1*0=0
1	1	1	1*1=1

논리곱은 부울대수의 연산 중 하나이다.

부정 연산자

값	!(논리부정)
false	true
true	false

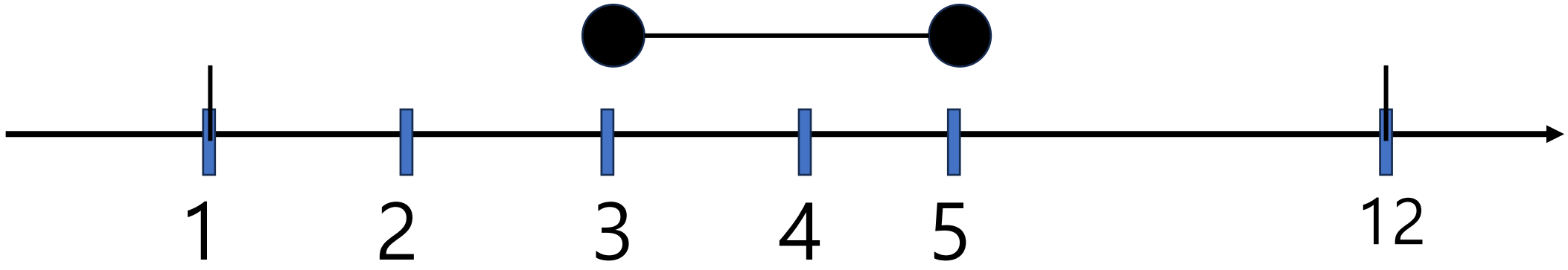
변수와 연산자

풀어 봅시다.

다음 물음에 대한 답을 작성해 보세요.

논리 연산자 문제

변수 month의 값이 3이상이고, 5이하이면 계절 "봄"입니다.
이것을 연산자로 표현해 보세요.



month >= 3 && month <= 5

ex)

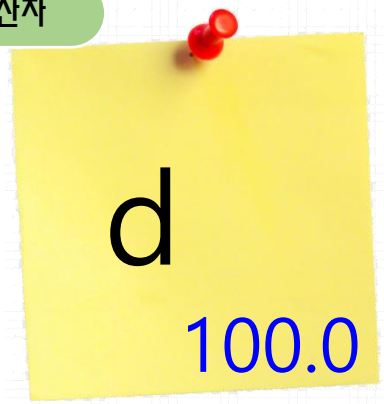
```
int month = 4;
if(month>=3 && month<=5){
    System.out.println("3보다 크고 5보다 작다.");
}
```

변수와 연산자

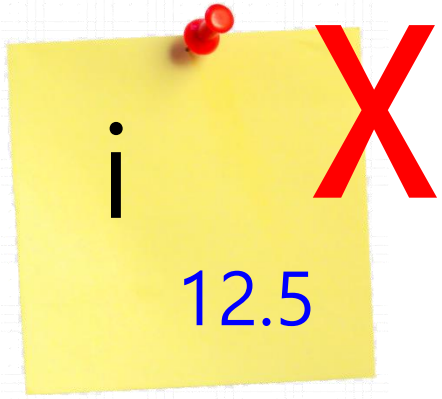
Casting 연산자

Casting(캐스팅)이란 데이터의 타입을 다른 타입으로 일시적으로 변경 하는 것을 말합니다.

형변환 연산자




```
// 암시적 형변환
doube d = 100 ;
```




```
// 명시적 형변환
int i = 12.5; ;
```

형변환의 유형

종류	설명	변경 주체
암시적 형변환 (Implicit Casting)	자바 시스템이 자동으로 데이터의 형을 변경해 줍니다. 예시 double d = 100 ; 정수 5가 double 형에 저장이 되어야 하므로 내부적으로 실수화 되어야 합니다.	자바 가상 머신(자바 시스템)
명시적 형변환 (Explicit Casting)	개발자가 직접 캐스팅하는 것을 말합니다. 예시 int i = (int)12.5 ; 실수 값 12.5를 명시적 캐스팅하여 정수 값 12으로 바꿔 줍니다. 참고 : 정수형 변수 i는 정수 부분의 값만 취하므로 0.5는 버리고, 12가 됩니다. 이때 (int)를 캐스팅(캐스트) 연산자라고 부릅니다.	개발자

변수와 연산자

Casting 연산자

문자간의 비교는 자바 Virtual Machine이 정수형으로 형변환을 수행한 다음 비교합니다.

아스키코드가 유니코드의 하위집합에 포함된다.

아스키 코드란 알파벳를 10진수로 만들어서 컴퓨터의 내부에서 처리하기 위한 코드를 말합니다.

즉, 내부적으로 (암시적으로) 캐스팅이 됩니다.

예를 들면, 알파벳 ´A´는 숫자 65으로 변경이 됩니다.

항목	설명
암기 사항	A(65), a(97), 숫자 0(48)
아스키 코드의 특징	65+26-1 1) 형변환이 되므로 문자간의 뺄셈이 가능합니다. int result = ´a´-´A´= 97 – 65 = 32 2)비교, 판단 가능 문자간의 비교를 할 수 있습니다. 3)어떤 문자ch가 소문자인가요? char ch = ´b´ ; ch가 ´a´ 보다 크거나 같고, ´z´ 보다 작거나 같으면 ch >= ´a´ && ch <= ´z´

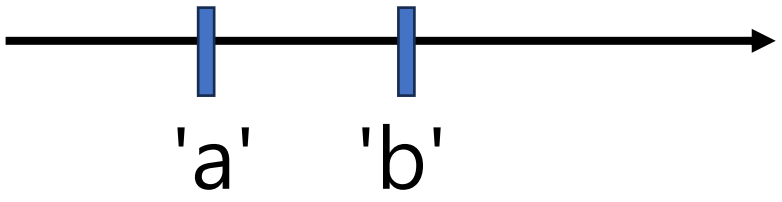
a는 아스키코드 97
z는 아스키코드 122

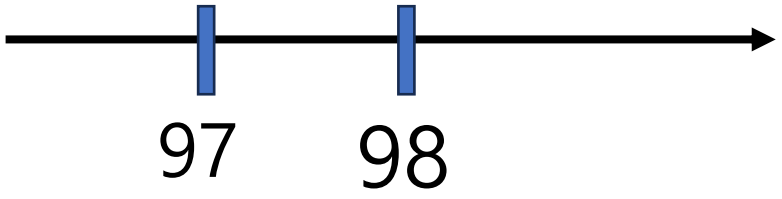
OX 문제

자바에서 char은 정수형 데이터 타입입니다.

참/거짓 판단

'a' > 'b'





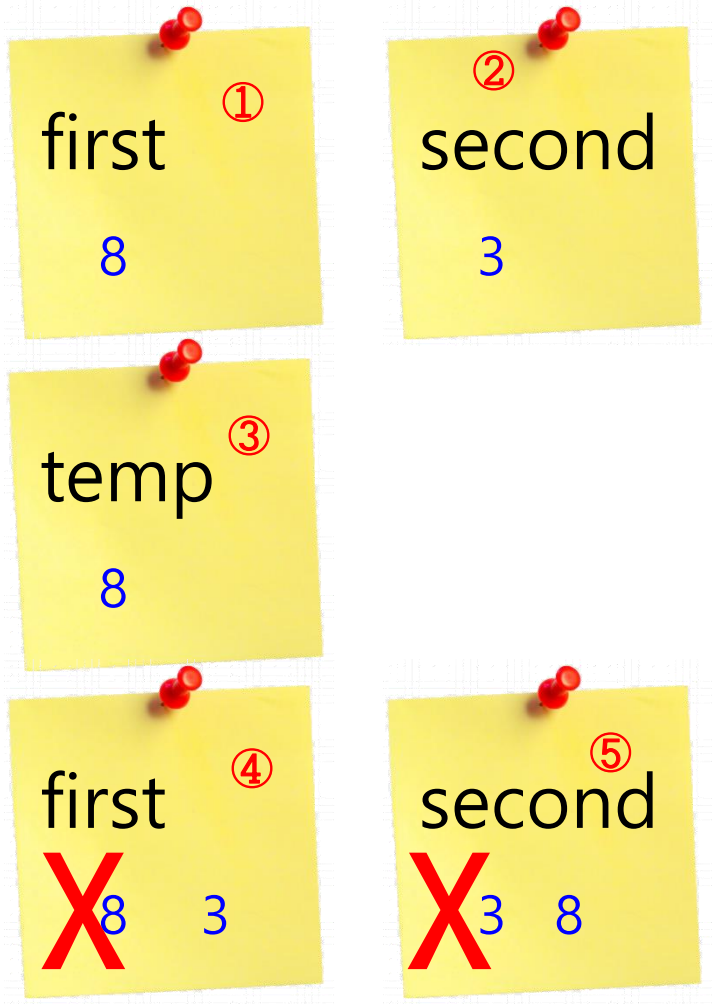
변수와 연산자

swap 기법

swap이란 2개 이상의 변수의 값을 서로 맞바꾸기 기능을 수행하는 알고리즘입니다.

swap 기법

```
int first = 8 ; ①
int second = 3 ②

int temp ;
temp = first ; ③

first = second ; ④
second = temp; ⑤
```

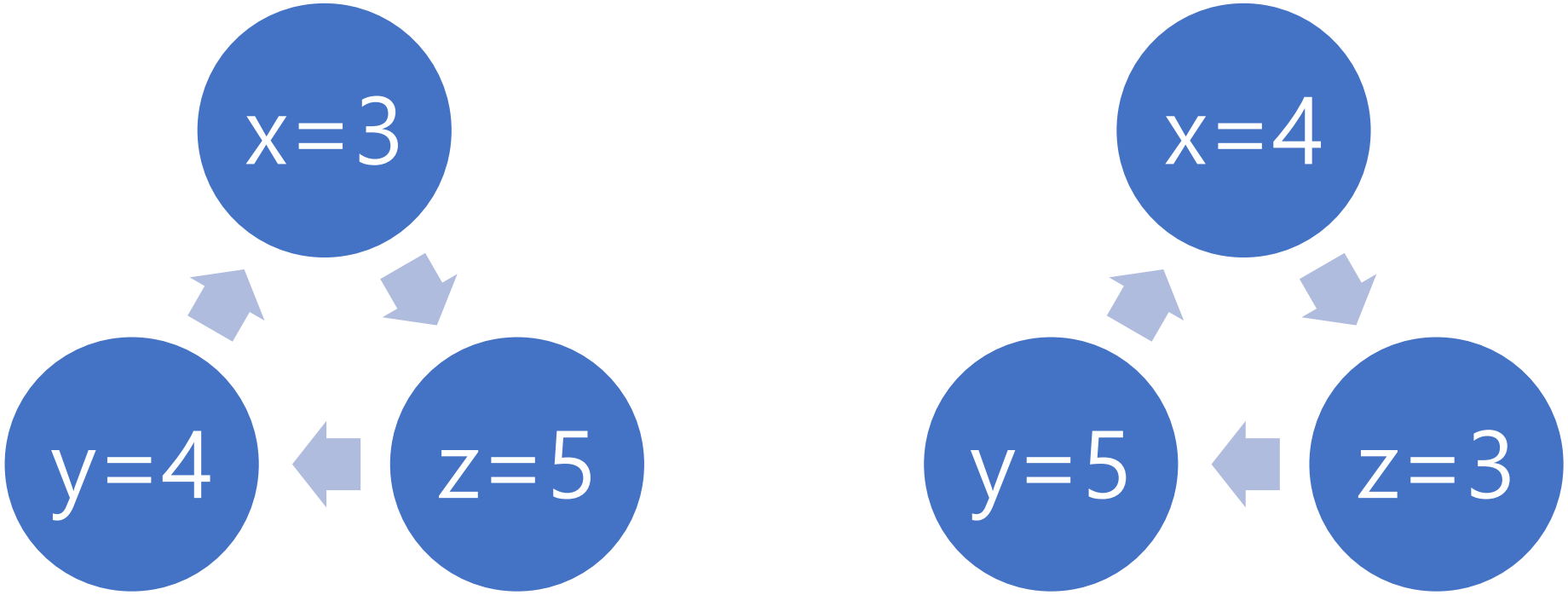


변수	값
first	8(X) 3
second	3(X) 8
temp	8

변수와 연산자

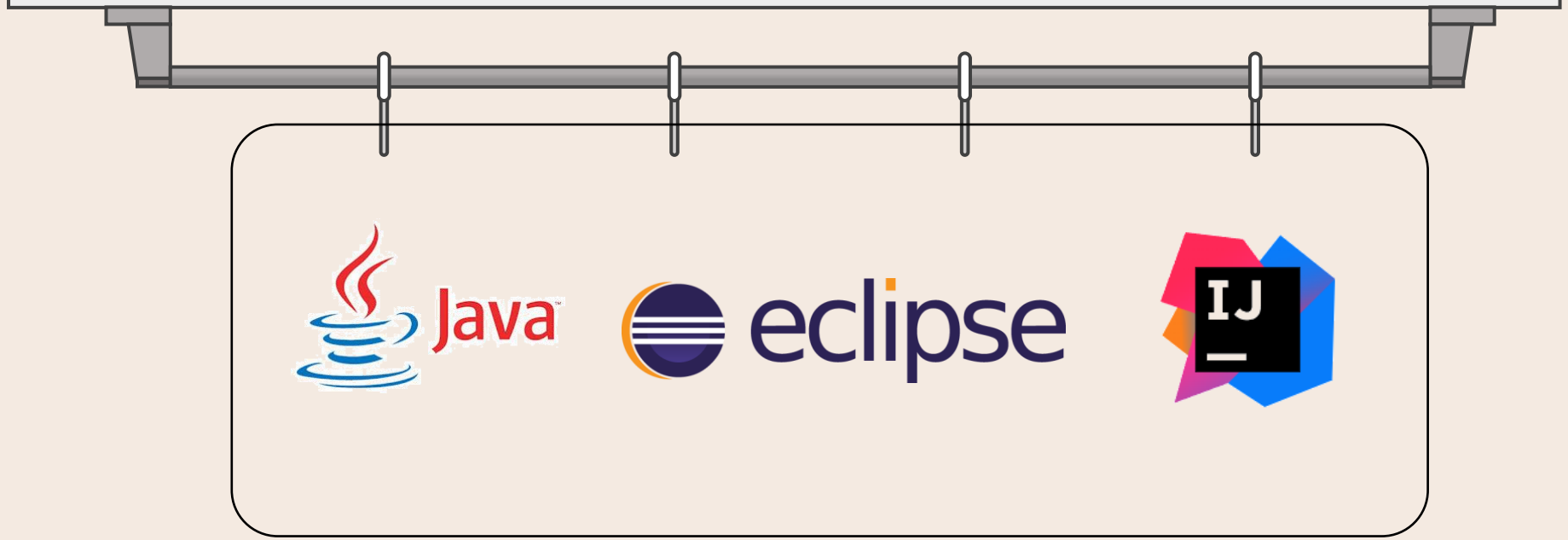
swap 기법

swap 기법을 이용하여 변수들의 값을 변경합니다.



자바 Programming

제어문



제어문

제어문 개요

프로그램의 수행 순서를 제어하거나, 문장들의 반복 수행 횟수를 조절해주는 문장입니다.

제어문은 조건문(선택문), 반복문, 보조 제어문 등으로 구성이 되어 있습니다.

분기(선택)문

반복문

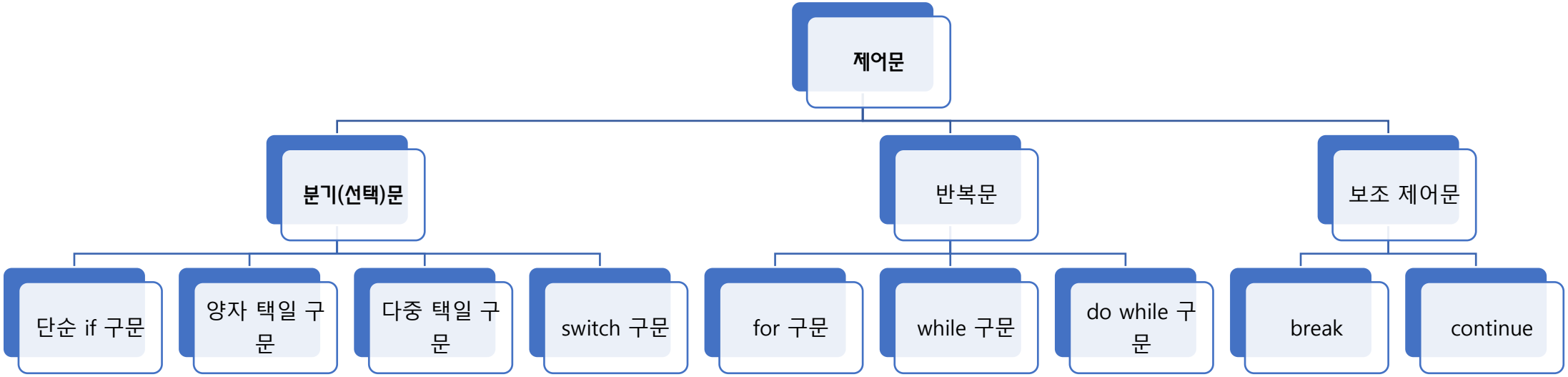
```
안녕하세요. 1
안녕하세요. 2
안녕하세요. 3
…
…
…
안녕하세요. 998
안녕하세요. 999
안녕하세요. 1000
```

제어문

제어문 개요

프로그램의 수행 순서를 제어하거나, 문장들의 반복 수행 횟수를 조절해주는 문장입니다.

제어문은 조건문(선택문), 반복문, 보조 제어문 등으로 구성이 되어 있습니다.



경우의 수 세기.

초기식,조건식,증감식 잘 세우기.

for문은 반복 횟수가 명확할 때 사용.
while문은 반복 횟수가 명확하지 않을 때 사용.

제어문

제어문 개요

프로그램의 수행 순서를 제어하거나, 문장들의 반복 수행 횟수를 조절해주는 문장입니다.

제어문은 조건문(선택문), 반복문, 보조 제어문 등으로 구성이 되어 있습니다.

구분	제어문	설명
선택문 (조건문)	if 문(단순 if 문) if … else 문 : 가장 많이 사용(양자 택일 구문) if … else … if 문 (다중 택일 구문) switch case 문	프로그램의 흐름을 조절/실행 순서를 변경하는 구문입니다.
반복문	for 문 while 문 do while 문	특정 구문을 반복적으로 실행시키는 구문입니다.
보조 제어문	break 문 continue 문	제어문에 대하여 보조적인 역할을 수행하는 구문입니다.

제어문

경우의 수

어떤 조건이나 상황에 따라 여러 가지 가능한 결과를 표현하거나 처리하는 개념입니다.

경우의 수

어떤 사건(일)이 일어날 수 있는 경우의 가짓수(outcomes)를 수로 표현한 것입니다.

1회의 시행에서 일어날 수 있는 사건의 가짓수를 n이라고 할 때 이 때의 경우의 수를 n이라고 합니다.

경우의 수는 확률과 매우 밀접한 관계를 가지는데 이는 각 사건이 일어날 확률들의 관계를 알 수 있다면 경우의 수를 통해 각 확률을 구할 수 있기 때문입니다.

항목	경우의 수
정수 x가 홀수인지 짝수인지 판별하기	2
동전 던지기	2
정수 x가 짝수인지 판별하기	1
주사위의 눈 판별하기	6
주사위 던질 때 홀수인지 짝수인지 판별하기	2
정수 x를 3으로 나누었을 때의 나머지	3
이번 달의 계절 판별하기	4
학점 판별하기	5
수우미양가 판별하기	5

제어문 개요

왼쪽 질문에 대한 조건식(오른)을 만들어 보세요.

조건식 만들기

코멘트	조건식
변수 x는 짝수인가요 ?	x%2 == 0
변수 x는 3의 배수인가요 ?	x%3 == 0
변수 x는 y의 약수인가요 ?	y%x == 0
변수 x가 3의 배수이거나 5의 배수라면 ?	x%3 == 0 || x%5 == 0
변수 ch1는 소문자가 맞습니까?	ch1 >= 'a' && ch1 <= 'z'
봄은 3월(month)부터 5월까지입니다.	month >= 3 && month <= 5

제어문

제어문 단순 if 구문

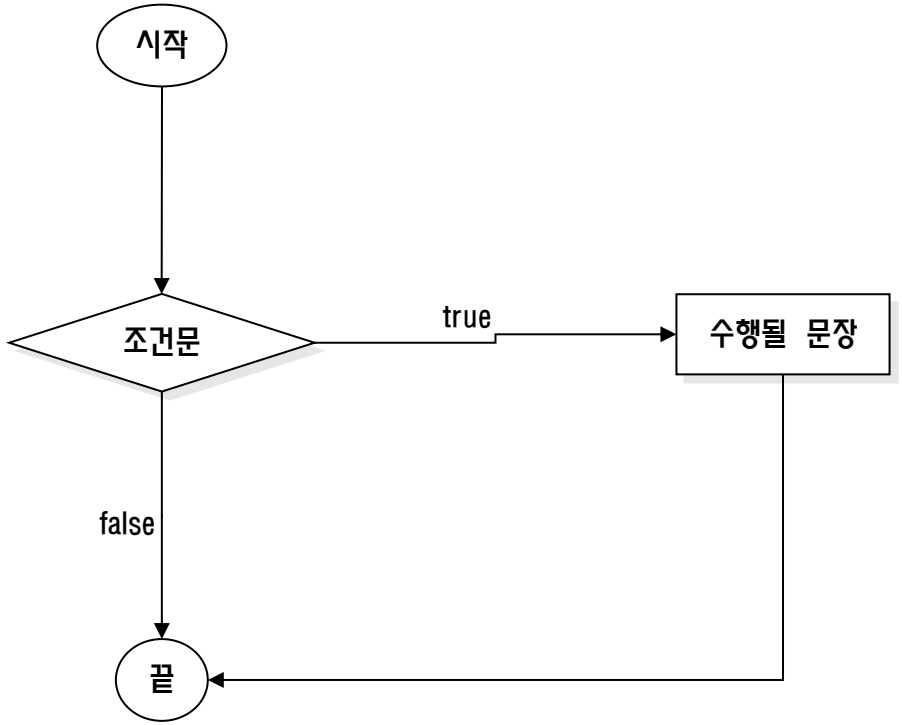
조건식이 참(true)일 경우에만 특정 블록의 코드를 실행하는 조건문입니다.

단순하게 if 구문만 존재하여 일반적으로 [단순 if 구문]이라고 부릅니다.
"조건문"에는 boolean 타입의 변수 또는 결과 값이 true/false인 연산 수식이 들어 갑니다.
일반적으로 관계 연산자 또는 논리 연산자의 결과 값이 들어 갑니다.
"조건문"이 true인 경우에만 해당 블록{…} 내의 코드들이 실행됩니다.

사용 형식

```
if(condition){
    // to_do_list...
}
```

condition(조건식)이 참일 경우에만 body 안쪽의 내용이 실행 됩니다.
여기서 condition은 참 또는 거짓으로 평가될 수 있는 표현식이며, 조건이 참일 때 실행할 코드가 실행됩니다.
조건이 거짓일 경우에는 해당 코드 블록은 건너 뛰고
다음 코드가 실행됩니다.
body는 {와 }로 둘러 싸인 영역을 의미합니다.



제어문

제어문 양자 택일 if 구문

특정 조건에 따라 두 가지 실행 경로 중 하나를 선택하도록 하는 제어문입니다.

if 구문의 참/거짓(2가지 경우의 수)에 따라서 하나의 블록을 실행하므로 [양자 택일 구문]이라고 부릅니다.

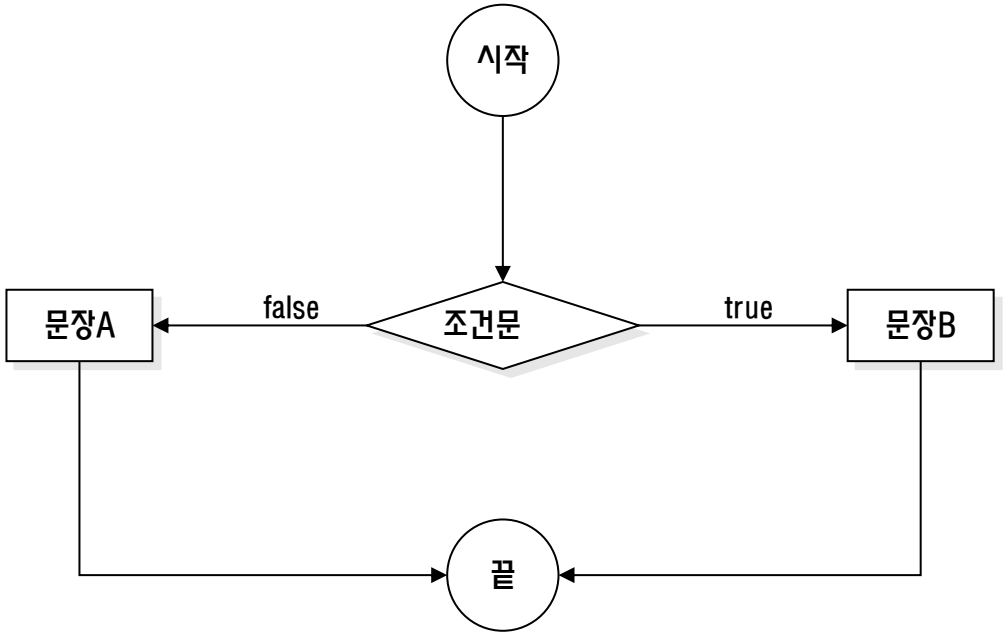
양자 택일 문의 축소된 형태가 [조건 연산자]입니다.

여러 개의 문장이 수행되면 반드시 중괄호를 묶어 주도록 합니다.

조건식의 참/거짓 여부에 따라 단, 하나의 블록만 실행이 됩니다.

사용 형식

```
if(condition){

    // condition이 참(true)이면 수행될 영역입니다.

}else{

  // condition이 거짓(false)이면 수행될 영역입니다.

}
```



제어문

제어문 다중 택일 if 구문

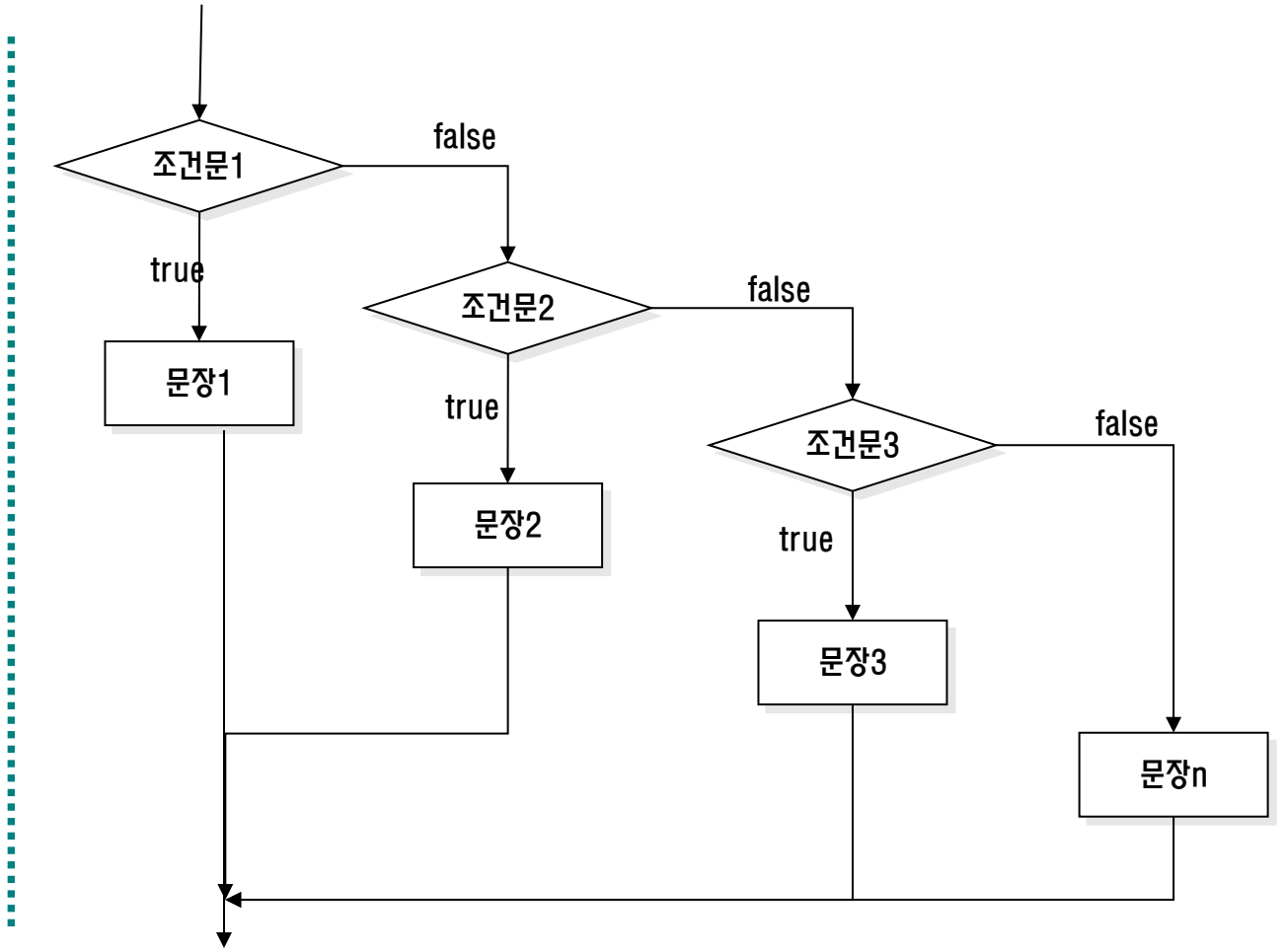
여러 개의 조건을 순차적으로 검사하여, 해당 조건이 참일 때 특정 코드를 실행하도록 하는 제어문입니다.

셋 이상의 조건에서 선택하고자 하는 경우로써 일반적으로 [다중 택일 구문]이라고 부릅니다.

골격을 만들고 입력한다.

사용 형식

```
if(condition1){
        문장1;

}else  if(condition2){

        문장2;

}else  if(condition3){
        문장3;

} [  else{
        문장n;
} ]
```



제어문

break 구문

반복문(예: for 또는 while 루프) 내에서 사용되며, 반복문을 즉시 종료하는 데 사용됩니다.

break 문이 실행되면 반복문을 더 이상 진행하지 않고 즉시 종료하고, 반복문 다음에 오는 코드로 실행이 이동합니다.
주로 반복문 내에서 특정 조건이 충족되면 반복을 종료해야 할 때 사용됩니다.
예를 들어, 어떤 조건이 충족되면 루프를 종료하고 루프 바깥의 코드로 제어를 이동해야 하는 경우에 break 문을 사용할 수 있습니다.

사용 예시

아래는 while 루프에서 break 구문을 사용하는 예시입니다:
다음 코드는 0부터 9까지의 숫자를 출력하면서 i가 5가 되는 시점에서 break 문이 실행되어 반복문을 종료합니다.
결과적으로 출력되는 숫자는 0부터 4까지이며, i가 5일 때 반복문이 종료되었습니다.

```
int i = 0 ;
while (i < 10){
      System.out.print(i);
      i += 1 ;
      if (i == 5){
            break;
      }
}
```

제어문

continue 구문

반복문(예: for 루프, while 루프) 내에서 사용되며, 현재 반복을 중지하고 다음 반복으로 넘어가도록 합니다.

continue 문을 만나면 현재 반복 블록의 나머지 코드는 실행되지 않고, 다음 반복으로 넘어가게 됩니다.
continue 문은 주로 조건에 따라 특정 상황에서 반복을 건너뛰고자 할 때 사용됩니다.
예를 들어, 반복문에서 특정 조건을 만족하는 경우에는 해당 반복을 건너뛰고 다음 반복으로 넘어가는 것이 유용할 수 있습니다.

사용 예시

1부터 10까지의 숫자 중에서 홀수만 출력하는 예시입니다.
for 루프를 사용하여 1부터 10까지의 숫자를 반복하고, 각 숫자가 짝수인 경우에는 continue 문을 만나서 반복을 건너 뜁니다.
그러므로 짝수인 경우에는 print(i) 문이 실행되지 않고, 홀수인 경우에만 해당 숫자가 출력됩니다.

```
for (int i=1 ; i < = 11 ; i++) {
      if (i % 2 == 0){
            continue ;
            System.out.print(i) ;
      }
}
```

제어문 switch case 구문

주어진 변수 또는 표현식의 값을 평가하여 여러 조건 중 하나와 일치하는 경우 해당 블록을 실행하는 제어문입니다.

다중 선택문으로 case 다음의 표현식에 따른 분기를 수행합니다.

case 문장 이후의 루틴을 수행하지 않을 경우 반드시 break 문을 입력하여 강제 종료하여야 합니다.

연속적인 값의 비교를 사용하는 관계 연산자에는 사용할 수 없습니다.

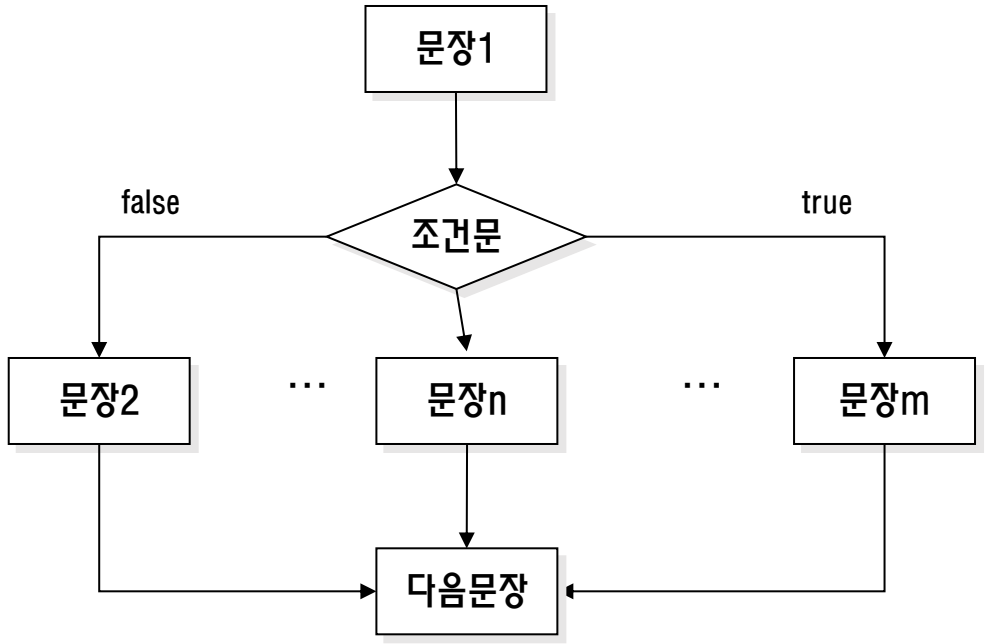
표현식은 byte, short, int, char, String 자료형을 사용할 수 있습니다.

수식에서 []는 옵션 사항입니다.

switch는 갯수가 유한할 때, IF는 갯수가 무한할 때 사용하면 좋다.

사용 형식

```
switch(표현식){
case 값01 :
      문장1;
      [break;]
case 값02 :
      문장2;
      [break;]
…
case 값0n :
      문장n;
      [break;]
[default:]
      문장n+1;
}
```



switch … case 구문 확인 사항

switch … case 구문에 대하여 다음 물음에 답해 보세요.

확인해 봅시다.

switch ... case 구문은 정수식 또는 문자열인 경우에만 사용이 가능합니다. (O X) 정답 : O

case 구문의 끝에는 반드시 세미콜론을 붙여야 합니다. (O X) 정답 : X 콜론

switch ... case 구문에서 다중 if 구문의 else 절과 유사한 구문은 ( ) 구문입니다. 정답 : default

break 구문은 언제 사용하면 좋을까요. Switch구문을 빠져나가고 싶을 때

반복문 for 개요

1부터 10까지의 종합에 대하여 반복문을 생각해해 봅니다.

for 구문

특정 부분의 문장을 반복적으로 수행하고자 하는 경우에 사용합니다.

제어 변수란 반복을 수행하기 위하여 사용되는 변수를 의미합니다.

일반적으로 i라는 제어 변수를 많이 사용합니다.

관련 용어

용어	설명	사용 예시
초기식	최초 딱 1번 실행되는 수식	int i = 1
조건식	비교/판단을 위한 근거가 되는 수식	i <= 10 또는 i < 11
증감식	제어 변수 증가/감소를 위한 수식	i = i + 1 또는 i += 1 또는 i++
제어 변수	반복문 중간 중간에 수시로 바뀌는 변수	i를 제어 변수라고 부릅니다.

<=가 아닌 <를 사용하는 방식이 배열할 때 유리함.

사용 형식

```
for(초기식 ; 조건식 ; 증감식){
      //반복 수행될 문장
}
```

```
int total = 0 ;

for( int i = 1 ; i < 11 ; i++ ){
      total += i ;
}

System.out.println("종합 : " +  total );
```

반복문 중첩 for 개요

다중(중첩) for 구문은 for 구문 안에 for 구문이 중첩이 되어 들어 있는 형태를 말합니다.

중첩 for 구문은 ´구구단 표´ 또는 다차원 배열(2차원 배열, 3차원 배열 등)에 많이 활용이 되고 있습니다.

이때 외부 for 구문에서 사용하는 제어 변수와 내부 for 구문의 제어 변수는 다른 변수이어야 합니다.

예를 들어서, I, j, k 등의 형식으로 사용하면 됩니다.

구구단 표

```
2 * 1 = 2    2 * 2 =  4    2 * 3 =  6   … 2 * 8 = 16    2 * 9 = 18
3 * 1 = 3    3 * 2 =  6    3 * 3 =  9   … 3 * 8 = 24    3 * 9 = 27
…
9 * 1 = 9    9 * 2 = 18    9 * 3 = 27   … 9 * 8 = 72    9 * 9 = 81
```

사용 형식

```
for(초기식; 조건식; 증감식){
    문장1;

    for(초기식; 조건식; 증감식){
       문장2;
    }

    문장3;
}
```

제어문

while 구문

조건이 참(true)인 동안 반복적으로 코드 블록을 실행하는 루프(반복문)입니다.

조건식이 참일 경우에만 반복문이 계속 실행됩니다.

기본 사용법

while 문의 일반적인 구조는 다음과 같습니다.

조건은 불리언 값(true 또는 false)을 반환하는 표현식입니다.

조건이 참일 경우에는 반복적으로 실행할 코드가 반복적으로 실행되고, 조건이 거짓일 경우에는 반복문이 종료됩니다.

```
while (조건){
    반복적으로 실행할 코드
}
```

사용 예시

1부터 5까지의 숫자를 출력하는 코드를 while 문을 사용하여 작성할 수 있습니다.

코드에서는 i가 5 이하일 때까지(즉, i가 5보다 작거나 같을 때까지) i를 출력하고 i를 1씩 증가시킵니다.

결과적으로 이 코드는 1부터 5까지의 숫자를 출력하고, i가 6이 되면 while 루프가 종료됩니다.

```
int i = 1 ;
while (i <= 5){
    print(i)
    i += 1
}
```

무한 while 구문

조건이 항상 참(true)으로 설정되어 있어서 종료 조건이 없는 상태에서 루프가 반복적으로 실행되는 구문입니다.

이 구문은 일반적으로 특정 조건이 충족될 때까지 반복 작업을 계속해야 할 때 유용합니다.

무한 while 루프는 주의해서 사용해야 합니다.

종료 조건을 명시적으로 설정하지 않으면 루프가 영원히 실행되어 프로그램이 멈추지 않을 수 있습니다.

따라서 무한 while 루프를 사용할 때는 반드시 적절한 종료 조건을 설정해야 합니다.

기본 사용법

무한 while 루프의 구조는 다음과 같습니다.

여기서 true는 불리언 값으로 항상 참을 나타내는 상수입니다.

따라서 이 while 루프는 항상 참인 조건으로 설정되어 있어서 무한히 반복됩니다.

```
while (true){
    반복적으로 실행할 코드
}
```

사용 예시

다음 코드에서는 사용자가 "종료"를 입력할 때까지 반복하여 사용자로부터 입력을 받고, 사용자가 "종료"를 입력하면 루프를 종료합니다.

그렇지 않은 경우에는 사용자의 입력을 처리하는 코드가 실행됩니다.

```
while (true){
    String user_input = "종료";
    if (user_input == "종료"){
        break ;

    }else{
        print("입력값:", user_input);
    }
}
```

제어문

어떠한 제어문이 좋을까요?

다음 각각의 질문에 사용하기 적절한 구문들을 언급해 보세요.

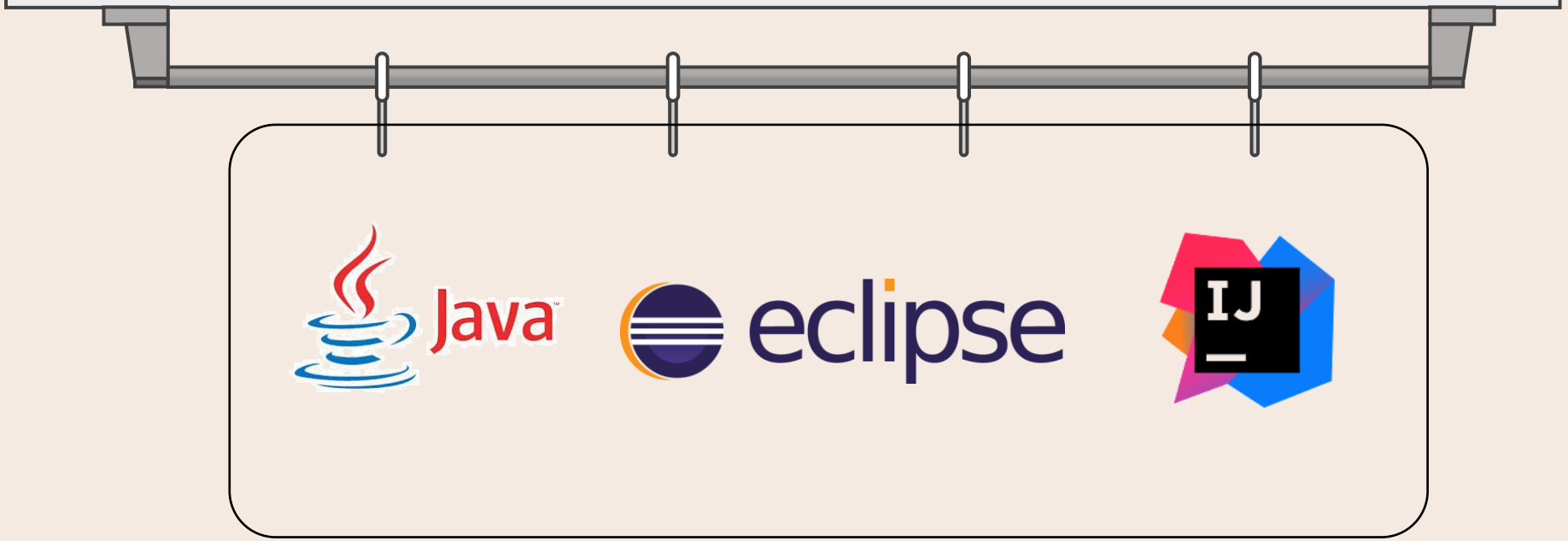
요구 사항

주어진 문제에 대하여 어떠한 제어 구문을 사용하면 좋을 지 판단해 보세요.

문제	사용할 구문 및 배경 지식
1부터 100까지의 총합 구하기	for 구문 또는 while 구문
입력된 정수만큼 한 줄에 5개씩 별을 출력해보기	for 구문, 단순 if 구문
1부터 10까지의 정수 중에서 홀수와 짝수의 합계 각각 구하기	for 구문, 양자 택일(if...else), switch(i%2) 구문
1부터 10까지 정수 중에서 3의 배수 또는 5의 배수의 합 구하기	for 구문, 단순 if(i%3 ==0 || i%5==0) 구문
1부터 50까지 정수 중에서 3의 배수가 아닌 수의 총합과 3의 배수의 총합의 차이 값을 구하기	for 구문, 양자 택일(if...else), switch(i%3) 구문
학생 10명의 국어 시험에 대한 학점 평가하기	for 구문, 다중 택일 구문
1 + 1/2 + 1/3 + 1/4 + ... + 1/99 + 1/100의 총합 구하기	for 구문, 형변환
1 - 1/2 + 1/3 - 1/4 + ... + 1/99 - 1/100의 총합 구하기	for 구문, 형변환, 양자 택일 구문
숫자 업다운 게임	무한 while loop, 다중 택일 구문

자바 Programming

배열(array)



배열(array)

배열 개요

다음은 배열과 관련된 항목들입니다.

항목	설명
정의	자료형이 동일한 여러 개의 값을 메모리에 연속적 배치해 놓은 자료 구조를 의미합니다.
3요소	배열 이름, 데이터 타입, 요소의 개수
특징	배열의 각각의 원소들을 요소(Element)라고 부릅니다. 배열 요소의 접근은 배열이름[index]의 형식으로 접근할 수 있습니다. 요소의 개수를 구하는 방법은 배열이름.length라는 메소드를 사용하면 됩니다. 배열의 요소 개수를 n : 배열명[0] ~ 배열명[n-1]으로 접근할 수 있습니다. 자바에서 대괄호 기호 []는 무조건 배열입니다. 배열의 요소 1개는 일반 변수와 동등하게 취급할 수 있습니다. 데이터 타입마다 요소들의 기본 값이 존재합니다. 예를 들어서, 정수형 배열은 모든 요소의 기본 값이 0입니다.
장점	식별자 이름 1개로 배열 표현이 가능합니다. 반복문을 사용할 수 있습니다.
단점	배열 요소의 개수가 정해지고 나면 더 이상 수정이 불가능합니다.
선언 방법	① new 연산자를 이용하는 방법 ② 해당 배열의 내용을 직접 초기화하는 방법

배열(array)

배열 개요

다음은 배열과 관련된 항목들입니다.

배열 요소의 기본 값

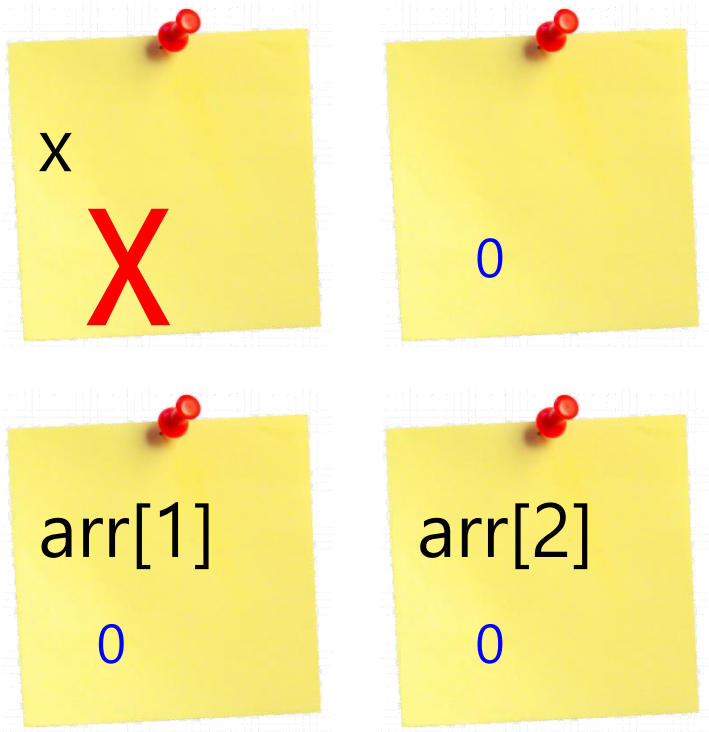
일반 변수와는 다르게 배열에 대한 요소들은 기본 값을 가지고 있습니다.

배열이 초기화 되지 않은 경우 각각의 타입에 맞는 암시적인 기본 값을 가지고 있습니다.

데이터 타입	기본 값
byte, short, int, long	0
float, double	0.0
char	공백 문자
boolean	false
참조 자료형	null

```
int x ;
int[] arr = new int[3] ;
```

색인=첨자=요소번호=인덱스



배열(array)

배열 개요

다음은 배열과 관련된 항목들입니다.

문제 풀어 보기

OX 문제 :

배열 요소 1개는 일반 변수와 동등하게 취급할 수 있습니다.

배열의 요소 번호는 1부터 시작합니다.

배열 타입마다 요소들의 기본 값이 존재합니다.

배열을 생성하는 방법은 크게 ( ) 가지가 있습니다.

하나는 ( )을 이용하는 방법이고, 다른 하나는 ( )입니다.

요소 개수를 구하는 방법은 ( )라는 메소드를 사용하면 됩니다.

OX 문제 :

O X O

2

new 연산자

초기화 기법

배열이름.length

배열(array)

배열 생성(초기화)

다음은 배열 초기화 관련된 문법입니다.

초기화 기법

```
int [] 배열_이름 = { 85, 90, 75 } ;
```

생성된 결과 배열 이름이 a 이고 배열 원소의 개수가 3 라고 가정 한다면

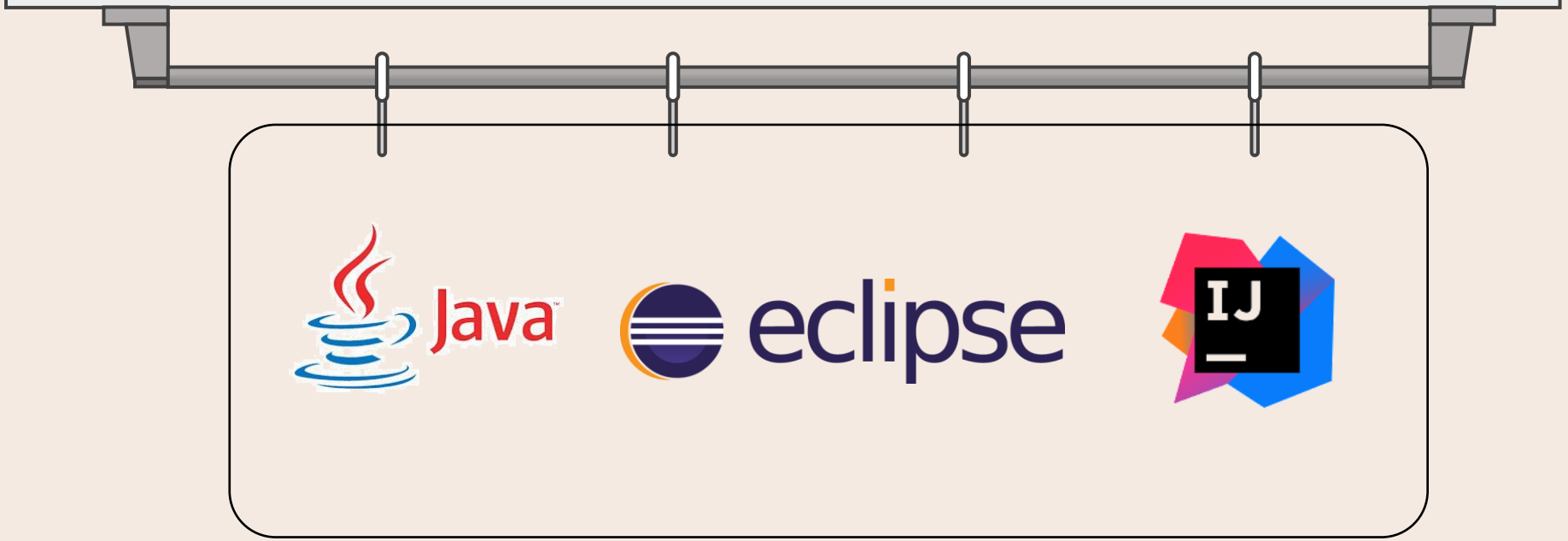
A. 배열 a[0] ~ a[2]가 생성됩니다.
B. 타입이 정수형이므로 12(4*3) 바이트의 메모리 공간 할당이 이루어 집니다.

선언 예시

```
int[] arr01 = {30, 15, 70, 10} ;
int[] arr02 = new int[]{10, 20, 30};
String[] arr03 = {"a", "b", "c"} ;
String[] arr04 = new String{"a", "b", "c"} ;

일반적으로 초기화 기법은 요소들이 이미 정해진 경우에 유용하게 사용될 수 있습니다.

String[] soshi = { "제시카", "티파니", ... } ;
```

자바 Programming

클래스(class)



클래스(class)

사람에 대한 정보 표현

같은 국적을 가진 2사람에 대한 정보를 출력하는 프로그램을 고려해 봅시다.

요구 사항

동일한 국적을 가진 김유신(yusin)과 유관순(soon) 2사람에 대한 정보를 관리하려고 합니다.
다음과 같은 변수 정보를 이용하여 출력하는 프로그램을 작성해 보세요.

출력 결과

김유신에 대한 정보
국적(nationality) : 대한 민국
이름(name) : 김유신
키(height) : 172.5
몸무게(weight) : 72.5
취미(hobby) : 당구
혈액형(blood) : AB

풀어 보세요.

변수는 총 몇 개가 필요한가요?

한 사람에 대한 정보는 국적/이름/키/몸무게/취미/혈액형 등이 존재합니다.
2사람이고, 변수는 각각 6개가 필요하므로 전체는 6*2=12, 총 12개의 변수가 필요합니다.

클래스(class)

소프트웨어 모델링

데이터를 표현하기 위하여, 다음 항목들을 어떤 방식으로 전개해 나갈 것인가를 설계하는 기법을 말합니다.

소프트웨어 모델링

점검할 사항
변수가 몇 개 필요한가요?
변수들의 데이터 타입은 무엇인가요?
데이터를 어떤 방식으로 보여줄 것인가요.

이전 문제에서 각 사람에게는 변수가 각각 6개 필요합니다.
변수 6개를 하나의 묶음으로 처리하려면 어떻게 해야 할까요?
이때 필요한 개념이 객체 지향 프로그래밍(Object Oriented Programming)입니다.

현실 세계

	현실 세계		프로그래밍
속성	국적 이름 키 몸무게 취미 혈액형	변수	nationality name height weight hobby blood
동작(행위)	정보를 출력합니다.	메소드	void display(){…}

변수는 특정 시점의 상태를 저장해둔 정적인 개념이고, 메소드는 어떠한 역할이나 기능을 수행해주는 동적인 개념입니다.

클래스(class)

메소드(method) 개요

다음은 메소드와 관련된 항목들입니다.

항목	설명
정의	입력된 데이터를 가공/데이터 연산을 수행한 후 결과물을 사용자에게 되돌려 주는 기능/역할을 수행하는 소스 코드. 다른 프로그램에서는 이것을 함수라고 부릅니다.
메소드 3요소	1) 메소드 이름 2) 매개 변수 3) 반환 타입
메소드 이름	식별자 작성 규칙과 동일하게 적용됩니다.
매개 변수	메소드 외부에서 사용자가 입력(변수, 배열, 객체 형태)해 주어야 하는 어떤 데이터 다른 용어로 파라미터, 인자, 인수라고도 부릅니다. 매개 변수는 소괄호 안에 작성되어야 하며, 매개 변수는 존재하지 않아도, 소괄호는 반드시 작성해 주어야 합니다.
반환 타입	반환 값이 없으면 void형으로 선언합니다. return을 사용할 수 없음. return 다음의 자료형과 동일합니다.
return 구문	메소드의 최종 결과 값을 호출한 곳으로 되돌려 주고자 하는 경우에 사용합니다. return 구문 이후에 일체 어떠한 코드도 허용하지 않습니다.
void 키워드	메소드의 반환 타입이 존재하지 않을 때 사용하는 키워드입니다. return이라는 키워드를 사용할 필요가 없습니다.
메소드 사용 이유	재사용성(반복적이고 일상적인 코드를 중복하여 코딩하기 싫어서)이 용이 합니다. 반복적인 코드를 여러 번 작성하지 않아도 됩니다.
특징	메소드는 다른 메소드와 동등한 위치에서 코딩해야 합니다. 메소드 안에 메소드 작성을 할 수 없습니다. 식별자에 소괄호가 붙어 있으면 무조건 메소드로 인식하면 됩니다.

클래스(class)

메소드(method) 개요

다음은 메소드와 관련된 항목들입니다.

항목	설명
메소드 호출	작성된 메소드를 실행시키는 것을 의미합니다.
매개 변수 작성 규칙	매개 변수가 2개 이상이면 콤마로 구분합니다. 매개 변수 이름은 통상적으로 x, y, m, n 등으로 작성합니다. 형식) 타입01 이름01, 타입02 이름02, ... 예시) int x, int y
이미 알고 있는 메소드	println() : 문자열을 출력 후 엔터키 누름 printf() : 서식 지정자를 이용한 문자열 출력 abs() : 절대 값을 구해줍니다

클래스(class)

메소드 예시

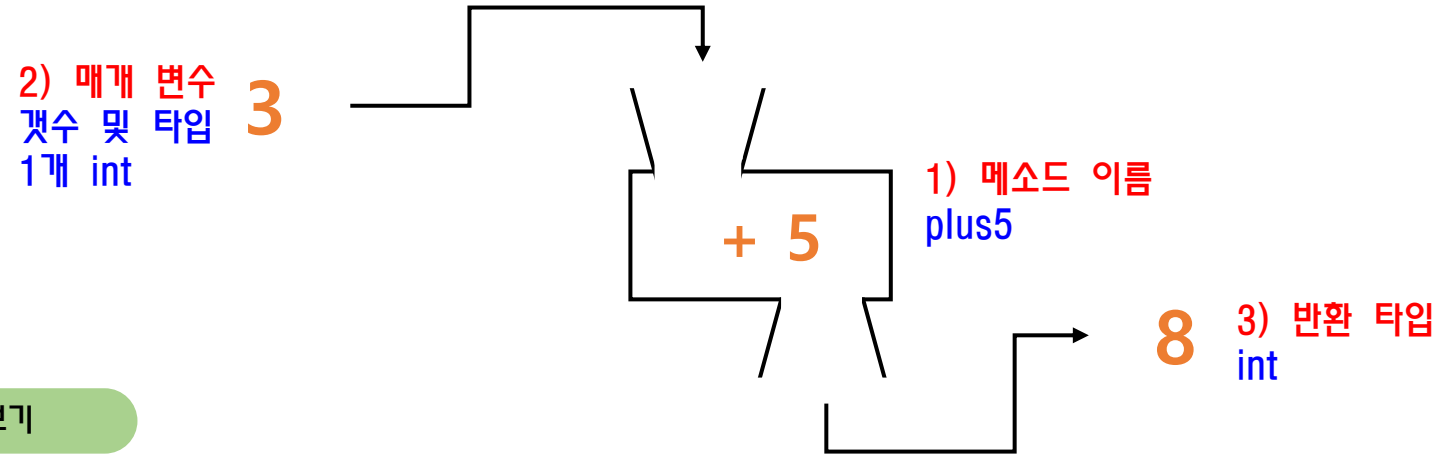
1개의 정수를 매개 변수로 입력 받아서 +5를 수행하는 메소드를 생각해 봅니다.

사용 형식

```
[static] 반환타입 메소드이름(매개변수리스트){
    //적절한 로직 처리
    retrun 반환해야할_값 ;
}
```

노트(note)

메소드 정의시 static이라는 키워드가 붙어 있습니다.
이 항목은 정적 메소드란 뜻으로 객체를 생성하지 않고, 사용자가 사용이 가능합니다.
좀 더 상세한 내용은 Class의 static 단원에서 배우도록 하겠습니다.



만들어 보기

위의 메소드를 코드로 구현하면 다음과 같습니다.

```
int plus5(int x){
    int result = x + 5 ;
    retrun result ;
}
```

메소드(method) 개요

다음 빈칸 및 내용에 대한 정답을 제시해 보세요.

풀어 봅시다

1) 변수와 메소드를 구분하는 방법은 ( )의 유무입니다.
2) void 키워드는 언제 사용하나요?
3) 메소드의 3가지 구성 요소는 ( ), ( ), ( )입니다.
4) 반환 타입이 존재하는 메소드는 반드시 ( ) 키워드를 사용해야 합니다.

1) 소괄호
2) 값을 반환하지 않는 경우
3) 이름, 매개 변수의 개수와 타입, 반환 타입
4) return

클래스(class)

클래스와 객체

다음은 클래스와 관련된 항목들입니다.

항목	설명
정의	사물의 특성을 소프트웨어적으로 추상화해서 모델링 해놓은 것입니다. 객체(구체적인 물건)를 생성하기 위한 틀(Template) 또는 청사진(Blueprint)
선언 방법	클래스를 선언하려면 class 키워드를 사용합니다. 필요에 따라서는 접근 지정자(수정자)가 필요합니다. 클래스 이름은 통상적으로 첫 글자는 대문자로 작성하는 경향이 있습니다.
특징	사물의 특성을 변수(속성)와 메소드(행위)로 만든 개념입니다. 기존 자료형을 이용하여 새롭게 만드는 사용자 정의 타입이라고 봐도 좋다. 무엇인가를 만들어 내기 위한 소프트웨어적인 설계도/프로토타입입니다.
구성 요소	[변수] + [메소드] + [생성자]
생성자	객체 생성시 오직 1번만 호출이 되는 특수 메소드를 의미합니다. 디폴트 생성자는 매개 변수가 없는 눈에 보이지 않는 생성자를 의미합니다.

클래스(class)

클래스와 객체

다음은 클래스와 관련된 항목들입니다.

클래스 작성 방법

```
[접근지정자] class 클래스이름  [extends 슈퍼클래스이름]{ //클래스 헤더
            //코딩할 영역을 클래스 바디(body)라고 부릅니다.
            // 클래스 멤버(변수, 생성자, 메소드)
}
```

```
class Account{
            String name ;
            int no ;
            int balance ;
}
```

멤버 변수

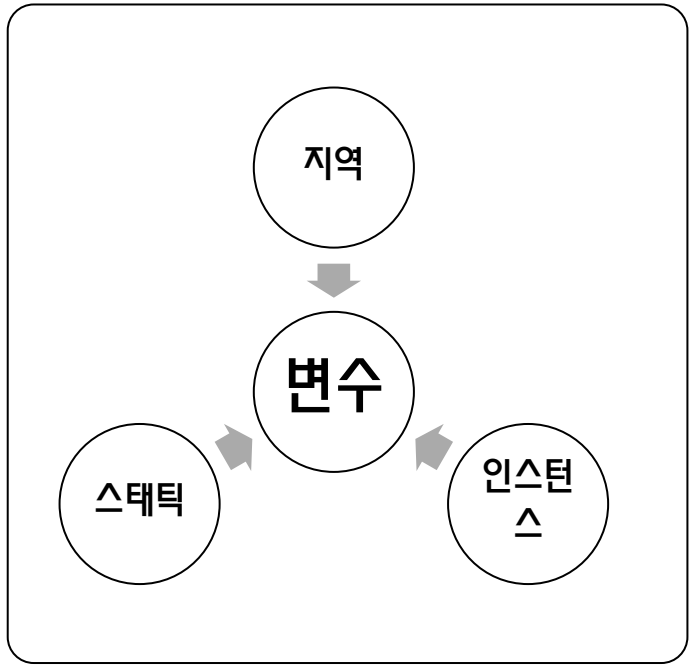
클래스 내부에 들어 있는 변수
객체가 참조를 하기 위한 변수
인스턴스 변수라고 부릅니다.
예시에서는 name, no, balance가 여기에 해당됩니다.

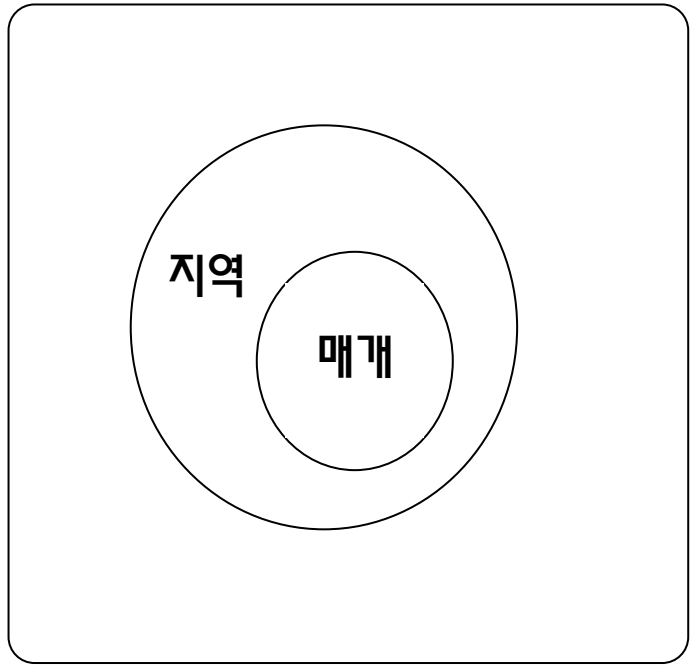
클래스(class)

변수의 종류

자바에서 사용 가능한 변수의 종류는 다음과 같습니다.

변수의 종류





클래스(class)

변수의 종류

자바에서 사용 가능한 변수의 종류는 다음과 같습니다. 인스턴스변수는 필드의 한 종류이다.

항목	지역(local) 변수	인스턴스(instance) 변수	스태틱(static) 변수
생성 위치	특정 메소드 내부에 생성	클래스 내부/메소드 외부에 생성	클래스 내부/메소드 외부에 생성
생성 시점	메소드 호출시 생성과 소멸을 반복하므로, automatic 변수라고 합니다.	객체 생성시 생성됩니다.	main 메소드 보다 먼저 메모리에 로딩됩니다.
기본 값	존재하지 않습니다.	타입별로 존재합니다. String(null), int(0), double(0.0) 등등	타입별로 존재합니다. 인스턴스 변수와 동일합니다.
초기화 필요성	반드시 초기화가 필요합니다.	경우에 따라서 초기화	경우에 따라서 초기화
초기 값 지정	변수 선언시 지정하면 됩니다.	변수 선언시 지정하면 됩니다.	변수 선언시 지정하면 됩니다.
static 키워드	사용하지 않습니다.	사용하지 않습니다.	사용합니다.
this 키워드	사용이 불가능합니다.	사용 가능합니다.	사용이 불가능합니다.
멤버 변수	X	O	X
목적	특정 메소드 내에서 국한적으로 참조하고자 하는 경우에 사용합니다.	객체가 가지는 어떠한 정보를 저장하기 위함입니다.	모든 객체들이 공유하기 위하여 1개만 생성됩니다.
기타	매개 변수도 지역 변수입니다.	non-static 변수	정적 변수 = 클래스 변수

어떠한 변수가 적당한가요?

홍길동 님의 통장 비밀 번호 지역변수, 이유 : 특정 메소드 내에서만 사용되도록 하여 메모리 누수와 보안 문제를 방지한다.

동일한 반의 학생들의 담임 선생님 이름 스태틱 변수

2개의 메소드에서 공통으로 참조하기 위한 평균(average)이라는 변수 인스턴스 변수, 이유 : 평균값이 특정 객체에 종속될 경우 사용. 평균은 객체별로 달라질 가능성이 많으므로 적합하다.

데이터를 임시로 저장하기 위한 변수 지역변수

클래스(class)

변수의 종류

자바에서 사용 가능한 변수의 종류는 다음과 같습니다.

항목	설명
지역 변수	특정 메소드 내에서 선언이 되어 사용되는 변수를 의미합니다. 이 메소드 밖에서 접근이 불가능합니다. 블록이 열리면 자동으로 생성되고, 블록을 빠져 나가면 자동 삭제되므로, automatic 변수라고 합니다. 매개 변수 역시 지역 변수의 한 종류입니다. 기본 값이 없으므로 반드시 초기화가 되어야 합니다. 오류 메시지) The local variable total may not have been initialized

final 키워드	상수(constant)를 선언하고자 하는 경우에 사용.(상수의 의미 파악이 쉬움)
	최초 생성시 오직 1번만 셋팅이 가능하고, 이후에는 읽기 전용입니다.
	일반적으로 불변의 진리(수학 공식에 사용되는 상수 값)에 사용됩니다.
	관례상 대문자로 많이 사용합니다.(C 언어의 영향)
	단, 한번만 초기화가 됨.(변경 불가능)
	사용 예시
	final int MAX = 1000;
	public final double PI = 3.14 ; //강제성 부여
	
	PI = 5.14 ; (이건 오류 : 얘는 읽기 전용)
	The final field(변수) Account4.PI cannot be assigned

클래스(class)

자바의 메모리 구조

자바의 메모리 구조는 크게 Static(공유 영역), Stack(스택), Heap(힙)이라는 3개의 큰 영역으로 나누어 집니다.

항목	설명
static(공유 영역)	메소드에 대한 바이트 코드(byte code)가 저장되는 영역입니다. static(클래스) 변수가 할당되는 곳입니다. main 메소드보다 먼저 메모리에 로딩됩니다.
스택(stack)	사람이 직접 객체 변수에 접근하여 지시(read/write)할 수 있습니다. 즉, 객체 이름을 사람이 직접 접근할 수 있습니다. 메소드와 관련된 매개 변수들의 저장소입니다. 메소드 내의 지역 변수 및 임시 변수의 저장소입니다. 스택 메모리는 자동 생성/자동 소멸됩니다.
힙(heap)	객체 사용시 중요한 메모리 영역입니다. 힙 메모리를 이용하기 위해 new 연산자를 사용합니다. 클래스 객체, 문자열 객체, 배열 객체 등등. 사용자가 직접 참조 불가, 레퍼런스 변수로 참조가 가능하다 인스턴스 변수가 존재하는 영역으로 사람이 접근하지 못합니다. 반드시 객체가 있어야만 의미가 있습니다. ↔ static 변수

클래스(class)

자바의 메모리 구조

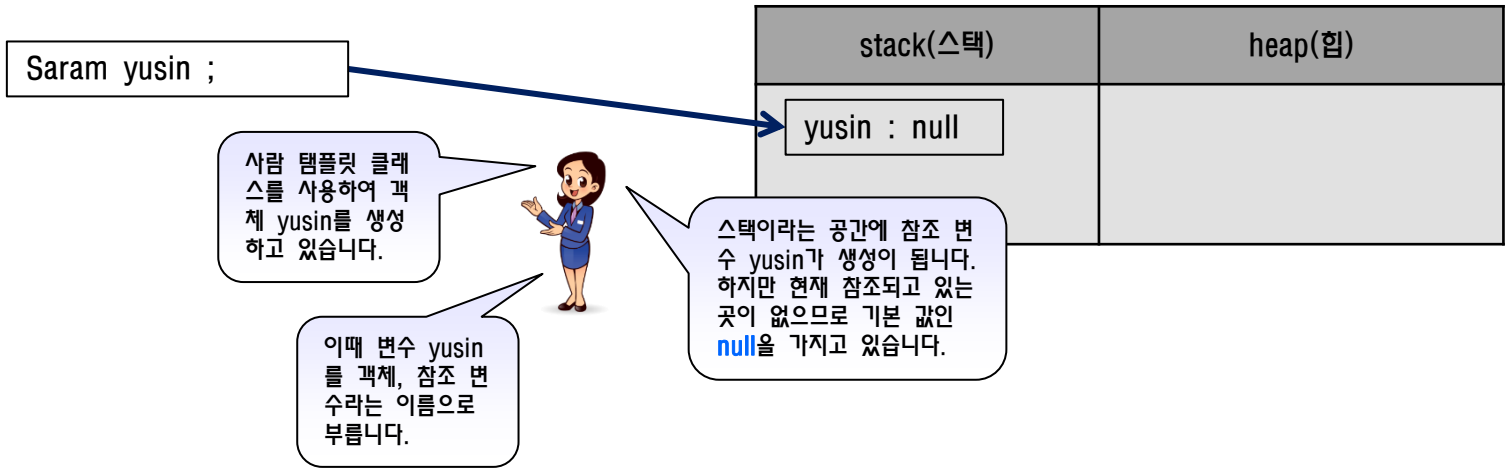
사람 정보를 저장하기 위한 탬플릿 클래스를 다음과 같이 정의합니다.

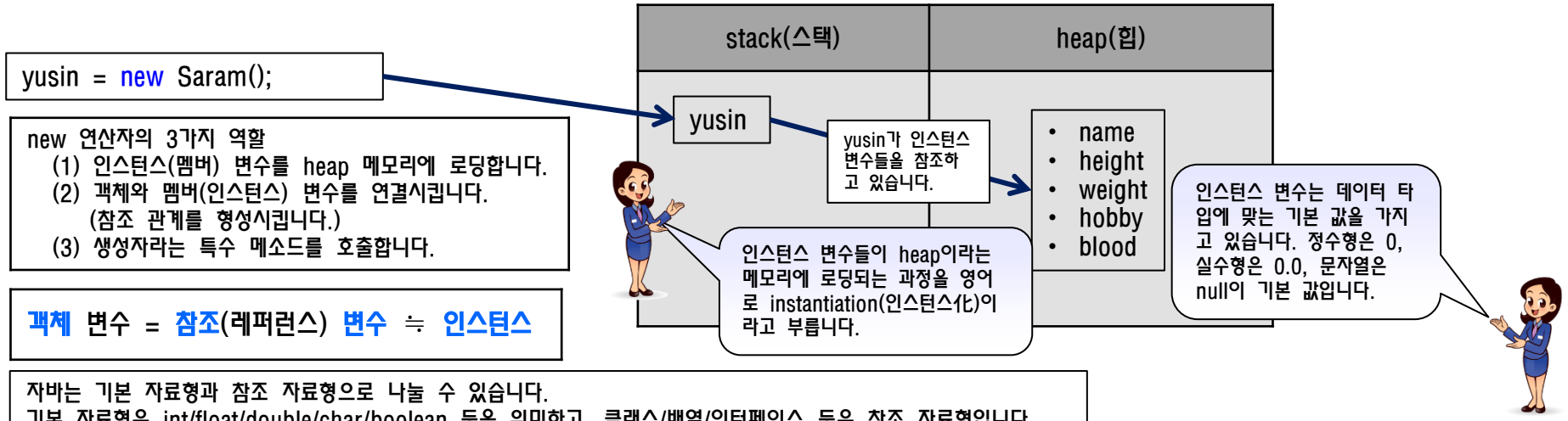
```
public class Saram {
	static String nationality ;
	String name ;
	double height ;
	double weight ;
	String hobby ;
	String blood ;

	void display(){
		System.out.println("국적 : " + nationality);
		System.out.println("이름 : " + name);
		System.out.println("키 : " + height);
		System.out.println("몸무게 : " + weight);
		System.out.println("취미 : " + hobby);
		System.out.println("혈액형 : " + blood);
	}
}
```

클래스(class)

자바의 메모리 구조

null이란 '참조를 하고 있지 않는'의 의미로 사용합니다. 혹자는 이 자체를 'null 참조'라고 명명하는 사람도 있습니다.





new 연산자의 3가지 역할
(1) 인스턴스(멤버) 변수를 heap 메모리에 로딩합니다.
(2) 객체와 멤버(인스턴스) 변수를 연결시킵니다.
(참조 관계를 형성시킵니다.)
(3) 생성자라는 특수 메소드를 호출합니다.

객체 변수 = **참조**(레퍼런스) **변수** ≒ **인스턴스**

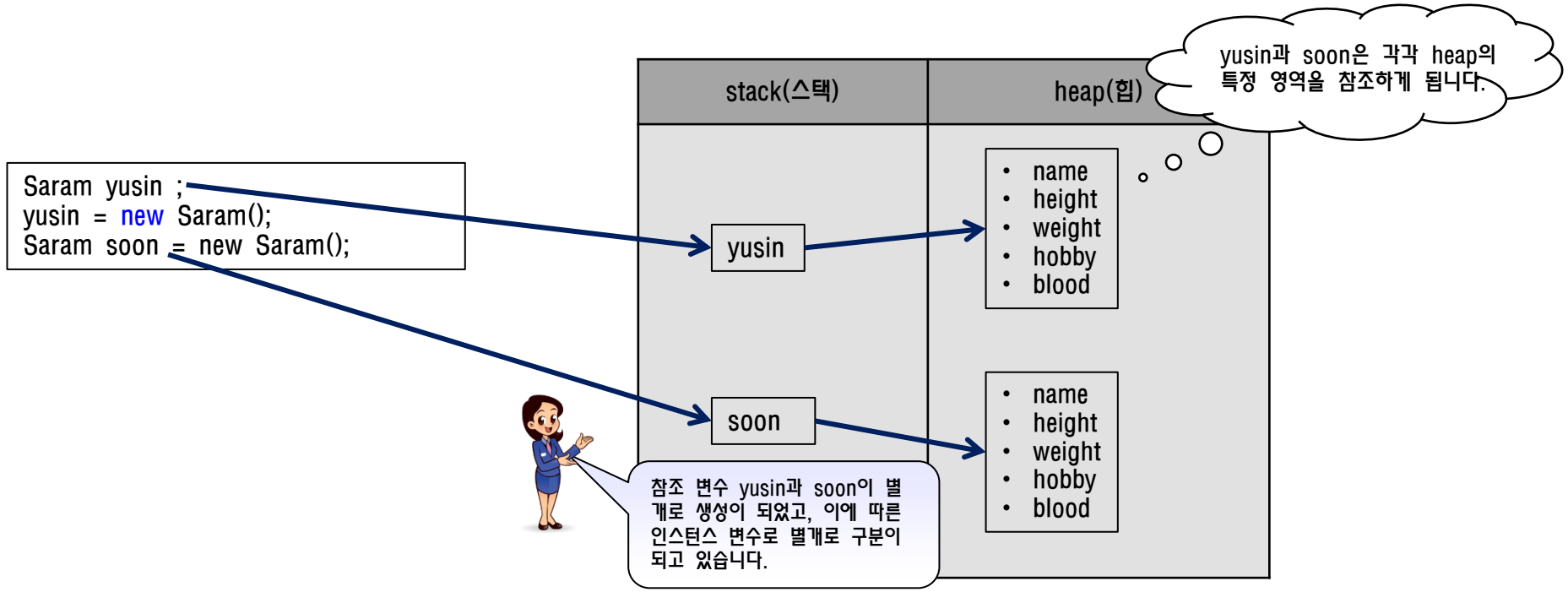
자바는 기본 자료형과 참조 자료형으로 나눌 수 있습니다.
기본 자료형은 int/float/double/char/boolean 등을 의미하고, 클래스/배열/인터페이스 등은 참조 자료형입니다.

클래스(class)

자바의 메모리 구조

객체 변수와 heap 메모리

객체의 수가 2개 이상이면 heap 영역에 각각의 데이터 공간이 만들어 지고, 이를 참조하기 위한 객체 변수(yusin, soon)가 만들어 집니다.

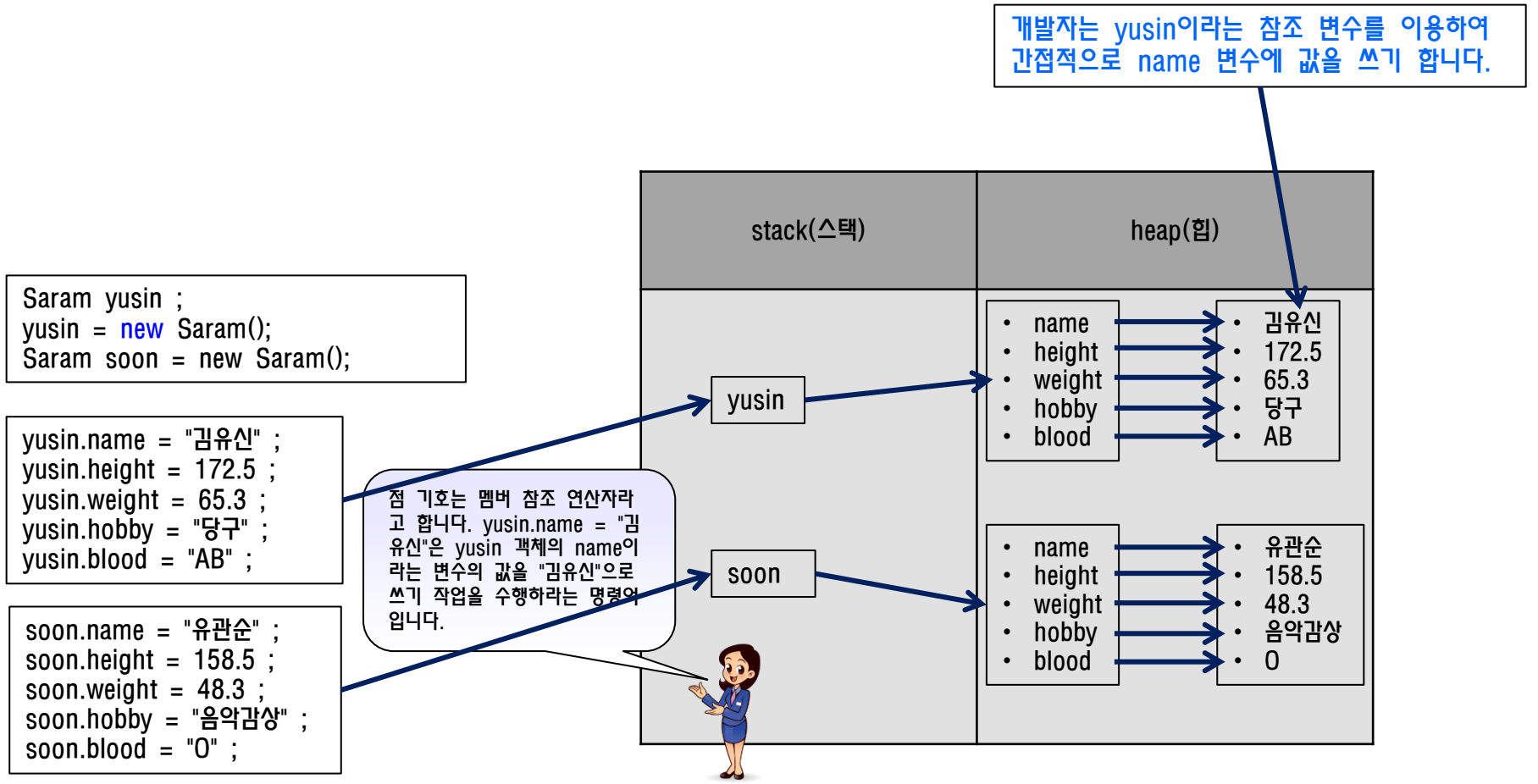


클래스(class)

자바의 메모리 구조

멤버 참조 연산자를 사용하여 데이터에 쓰기 작업을 수행할 수 있습니다.

객체의 멤버 변수를 값을 읽거나 쓰기 작업을 수행하고자 한다면, 멤버 참조 연산자인 dot(.)을 사용하면 됩니다.
다음 예시는 각 객체 변수가 가지고 있는 인스턴스 변수에 값을 지정(write)하는 예시입니다.

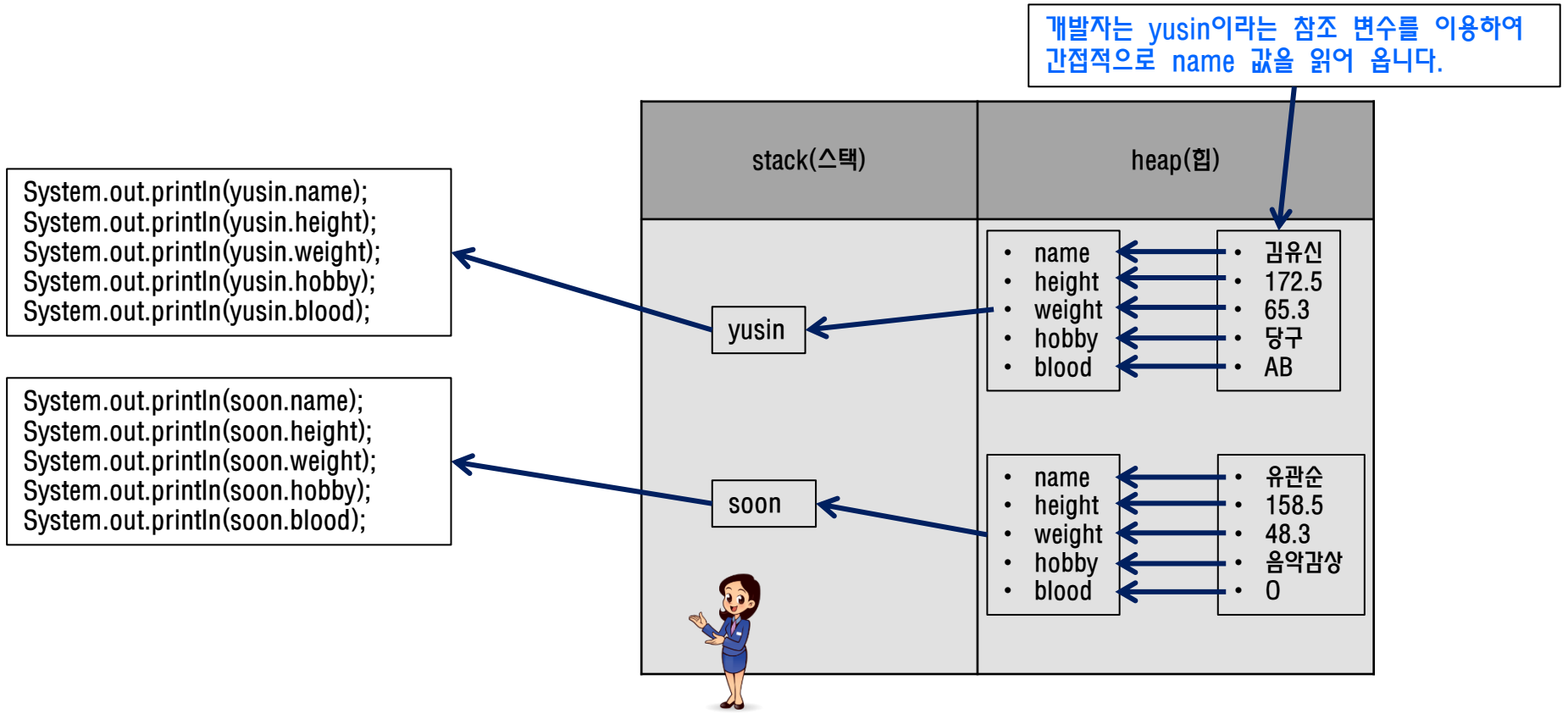


클래스(class)

자바의 메모리 구조

멤버 참조 연산자를 사용하여 데이터 읽기를 수행할 수 있습니다.

객체의 멤버 변수를 값을 읽거나 쓰기 작업을 수행하고자 한다면, 멤버 참조 연산자인 dot(.)을 사용하면 됩니다.
다음 예시는 각 객체 변수가 가지고 있는 인스턴스 변수의 값을 읽어와(read) 출력하는 예시입니다.



클래스(class)

자바의 메모리 구조

자바의 메모리 구조에서 객체(Object) 참조와 비교는 다음과 같이 이루어집니다.

항목	설명
대입(=) 연산자	대입(=) 연산자를 사용하게 되면 두 객체는 동일한 주소를 참조하게 됩니다. yusin = soon ; 라고 코딩하게 되면 soon가 참조하고 있던 위치를 yusin 역시 동일하게 참조하게 됩니다. yusin가 참조하고 있던 위치는 가상 가상 머신(JVM)에 의하여 release됩니다.
등위(== 또는 !=) 연산자	==와 !=는 각각 2개의 객체의 참조가 같은 객체인가 또는 다른 객체인가를 결정하고 true/false을 반환합니다. if(yusin == hee)는 참조의 위치가 동일한가를 물어 보는 문장입니다.
equals 메소드	최상위 클래스인 Object 클래스의 equals 메소드를 가지고 두 객체가 같은 것인지/아닌 것인지 판별 가능 if(pt1.equals(pt2))

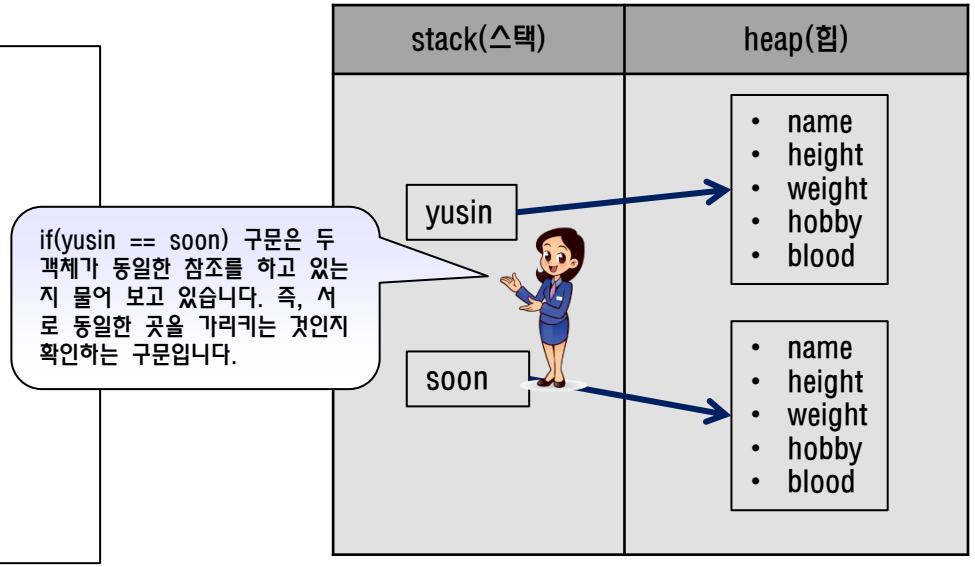
```
int x = 3, y = 4 ;
if( x == y ){} // x와 y는 동일한 값인가요?

Saram yusin = new Saram();
Saram soon = new Saram();
if(yusin == soon){} // 동일한 곳을 바라보고 있습니까?

x = y ; // x의 값을 4로 치환하시오.
yusin = soon ; // soon의 참조를 yusin이 복사합니다.
```

클래스(class)

자바의 메모리 구조

자바의 메모리 구조에서 객체(Object) 참조와 비교는 다음과 같이 이루어집니다.

```
Saram yusin ;
yusin = new Saram();

Saram soon = new Saram();

if(yusin == soon){
      System.out.println("참");

}else{
      System.out.println("거짓");

}

// "거짓"이 출력됩니다.
```



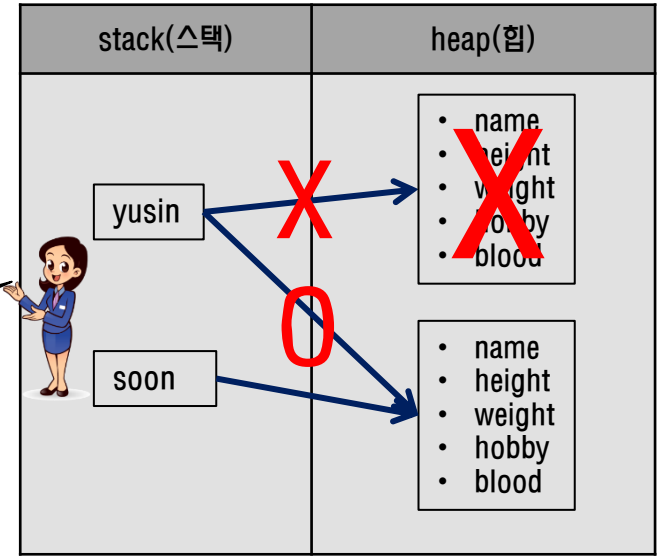
클래스(class)

자바의 메모리 구조

자바의 메모리 구조에서 객체(Object) 참조와 비교는 다음과 같이 이루어집니다.

```
Saram yusin ;
yusin = new Saram();
Saram soon = new Saram();

if(yusin == soon){
      System.out.println("참");
}else{
      System.out.println("거짓");
}

yusin = soon ;
```

yusin = soon는 soon이 참고하는 있는 곳을 yusin도 참조하도록 하겠다는 의미입니다. 이전에 yusin이 참조하던 연결 고리는 JVM에 의하여 release됩니다.

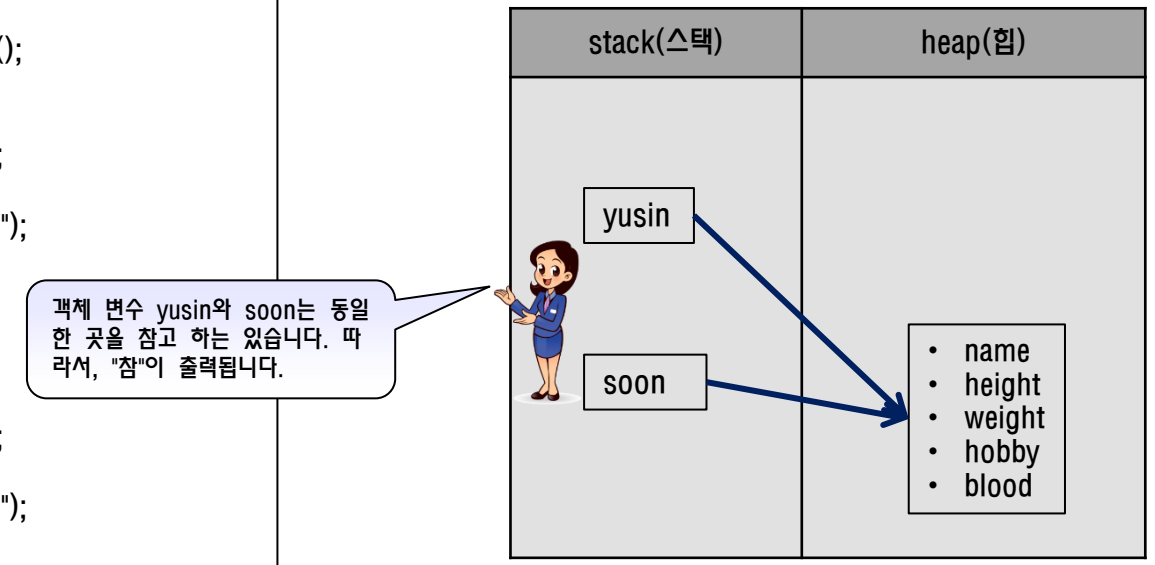


클래스(class)

자바의 메모리 구조

자바의 메모리 구조에서 객체(Object) 참조와 비교는 다음과 같이 이루어집니다.

```
Saram yusin ;
yusin = new Saram();
Saram soon = new Saram();

if(yusin == soon){
      System.out.println("참");
}else{
      System.out.println("거짓");
}

yusin = soon ;

if(yusin == soon){
      System.out.println("참");
}else{
      System.out.println("거짓");
}
// 참이 출력됩니다.
```

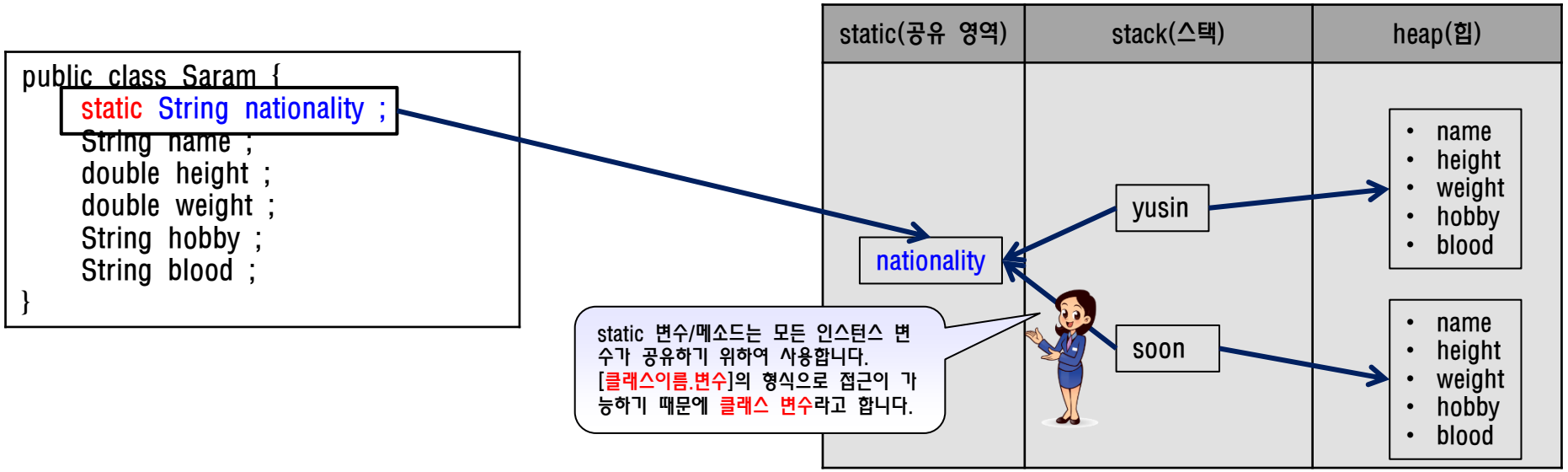


클래스(class)

자바의 메모리 구조

객체 생성과 자바의 메모리 구조 변동 사항에 대하여 살펴 보도록 합니다.

항목	설명
static(공유 영역)	static 변수 main 메소드보다 먼저 메모리에 로딩됩니다.
stack(스택)	사람이 직접 객체 변수에 접근하여 지시(read/write) 예제 : yusin, soon
heap(힙)	인스턴스 변수가 존재하는 영역으로 사람이 접근하지 못합니다. 반드시 객체가 있어야만 의미가 있습니다. ↔ static 변수



클래스(class)

접근 지정자(제어자)

데이터의 접근 허용 범위

접근 지정자(Accsss Modifier)란 클래스 내에 들어 있는 변수 및 메소드의 접근 허용 범위를 [어디까지 허용할 것인가]를 나타내는 척도로써,
[접근 제어자] 또는 [접근 제한자]라고도 합니다.
공개(public) 메소드란 접근이 불가능한 private 변수를 클래스 외부에서 간접적으로 참조하기 위하여 제공하는 메소드를 말합니다.
예를 들어서, 통장의 잔액은 직접 확인이 불가능합니다.
잔액 조회라는 간접적인 기능을 이용하여 조회가 가능합니다.

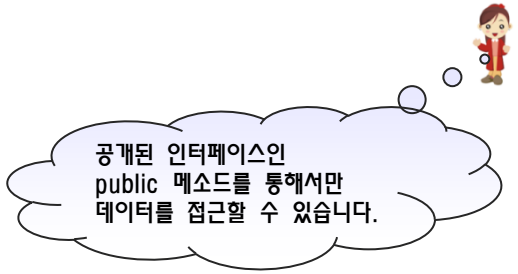
통장 클래스

- 예금주(개방)
- 계좌 번호(개방)
- 비밀 번호(숨김)
- 잔액(숨김)

항목	설명
잔액 조회	숨겨져 있는 잔액을 간접적으로 조회합니다.
입금/인출	숨겨져 있는 잔액을 간접적으로 변경합니다.

```
public Integer get잔액(){
    return 잔액 ;
}
```

```
public void set잔액(Integer 잔액){
    this.잔액 = 잔액;
}
```



클래스(class)

접근 지정자(제어자)

데이터의 접근 허용 범위

다음은 Saram 클래스에서 이름과 키에 대하여 getter과 setter를 적용한 예시입니다.

Saram 클래스

숨겨진 데이터

```
private String name;
private double height
```

getter 사용

```
public String getName(){
    return name ;
}

public double getHeight(){
    return height ;
}
… 중략
```

setter 사용

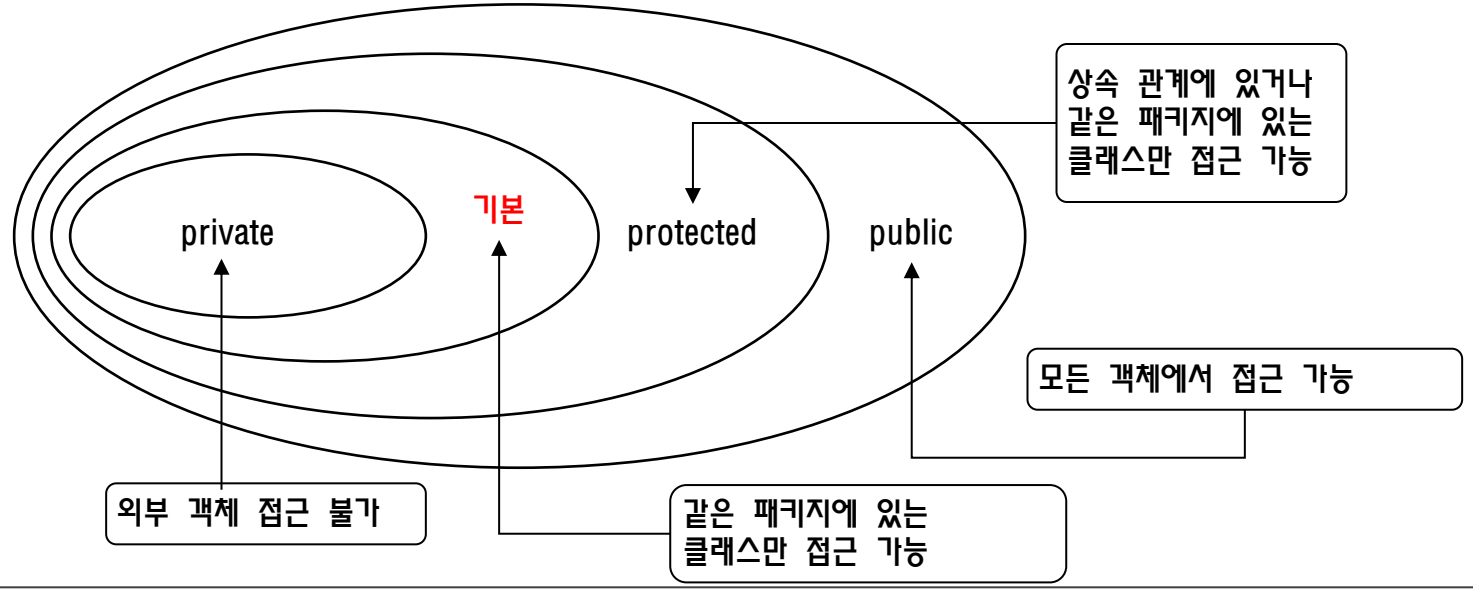
```
public void setName(String name){
    this.name = name;
}

public void setHeight(double height){
    this.height = height ;
}
… 중략
```

클래스(class)

접근 지정자(제어자)

접근 지정자와 패키지와의 관계

종류	public	protected	기본	private
동일 클래스	가능	가능	가능	가능
동일 패키지의 모든 클래스	가능	가능	가능	불가능
동일 패키지의 서브 클래스	가능	가능	가능	불가능
다른 패키지의 서브 클래스	가능	가능	불가능	불가능
다른 패키지의 일반 클래스	가능	불가능	불가능	불가능
비고	전체 공개	일존 공개	일존 공개	비공개



클래스(class)

접근 지정자(제어자)

게터 메소드와 세터 메소드에 대하여 살펴 봅니다.

private 변수는 일반적으로 공개(public) 메소드를 이용하여 접근할 수 있는 여지를 남깁니다.
메소드에 사용할 수 있는 접근 제어자는 public, protected, private, default(접근 제어자가 없는 형태)가 올 수 있는 데,
getXXX() 메소드와 setXXX() 메소드에는 주로 public을 사용합니다.
마찬 가지로 불특정 다수의 접근을 허용하겠다는 의미로 public이 사용됩니다.

-	getter	setter
정의	비공개(private) 변수의 값을 읽어 오기 위한 공개(public) 메소드	비공개(private) 변수의 값을 설정하기(쓰기) 위한 공개(public) 메소드
이름	get + 변수의 첫 단어를 대문자로 변환합니다.	set + 변수의 첫 단어를 대문자로 변환합니다.
매개변수	없음	1개인데 변수의 타입과 동일합니다.
반환 타입	해당 변수의 타입과 동일합니다.	void

사용 형식	public 반환타입 get변수의첫글자만대문자(){
	return 해당변수 ;
	}
		public void set변수의첫글자만대문자(변수의타입 변수이름){
		해당변수 = 변수이름 ;
		}

사용 예시	public int getBalance(){
	return balance ;
	}
		public void setBalance(int _balance){
		balance = _balance ;
		}

캡슐화

캡슐(내용을 알 수 없는) 약처럼, 주요 데이터는 접근 지정자로 데이터를 숨기는 개념입니다.
이것은 private 접근 지정자를 두고 하는 말입니다.

클래스(class)

식별자 표기법

식별자 표기법에는 CamelCase 표기법과 헝가리언 표기법 등이 있습니다.

CamelCase 표기법

Camel case는 복합 단어를 사용할 때 주로 사용되는 표기법 중 하나입니다.
이 표기법은 여러 단어를 하나의 단어로 이어서 표기하는 방식으로, 각 단어의 첫 글자를 대문자로 표기하며 나머지 글자는 소문자로 표기합니다.

예를 들어, "helloWorld", "myFavoriteColor", "openAIChatbot"과 같은 식으로 표기합니다.
각 단어의 첫 글자가 대문자로 시작하므로 단어들이 낙타의 등 모양과 비슷해져서 "Camel case"라는 이름이 지어졌습니다.

헝가리언 표기법

프로그래밍 언어에서 변수 및 함수의 인자 이름 앞에 데이터 타입을 명시하는 코딩 규칙이다.
찰스 시모니(Charles Simonyi)가 마이크로소프트의 개발 책임자로 있을 때 제안했으며,
80년대 당시에는 IDE라는게 다들 부실했기 때문에 물론 볼랜드의 Turbo C 같은 예외도 있었지만
이 규칙이 엄청난 센세이션을 불러 일으켰다.

하지만 지금은 MS도 공식 가이드라인에서 사용하지 말 것을 권고[1]하고 있다.
헝가리안 표기법이라는 명칭은 제안자인 찰스 시모니가 헝가리인이라서 붙은 것이다.

클래스(class)

[이론] this 키워드

특정 객체가 변수 또는 메소드를 호출할 때 복사가 되는 특수 형태의 참조 변수

this 예약어는 객체 자기 자신을 가리키는 레퍼런스 변수입니다.

항목	설명
특징	자바 컴파일러가 자동으로 제공하는 레퍼런스 변수입니다. 객체 생성시 인스턴스 내부에서 자동 생성됩니다. 메소드 내 자신을 참조하기 위한 레퍼런스 변수로써 선언하지 않고 지역 변수처럼 사용이 가능합니다. 필요에 따라서는 명시적으로 작성할 수 있습니다.

용도	클래스 내의 인스턴스 변수, 메소드, 생성자에서 사용이 가능합니다.
	객체의 멤버 변수(필드)와 메소드의 매개 변수와 동일한 경우 이것을 구분하기 위하여 사용합니다.
	`public void 어떤메소드(String name){`
	`this.name = name ;`
	`}`
	
	동일 클래스내에서 다른 생성자를 호출하고자 하는 경우에 사용합니다.
	this 생성자의 사용(다른 메소드보다 최우선 실행되어야 합니다.)
	`public void 생성자(){`
	`this(매개변수) ;`
	`}`

클래스(class)

메소드의 종류

자바의 메소드는 크게 static 메소드와 인스턴스 메소드로 구분이 됩니다.

항목	설명
static 메소드 (클래스 메소드)	객체와 무관하게 실행이 될 수 있는 메소드로써, 클래스 이름으로 직접 접근이 가능합니다. 객체가 생성이 되지 않아도 가능합니다.(경제성) static이라는 키워드를 사용하고, this 연산자 사용이 불가능합니다. 내부에서는 반드시 클래스 변수를 가지고 처리. 인스턴스 변수 사용 불가. 예시 : 수학의 Math 클래스의 PI 상수 및 절대값(abs)
인스턴스 메소드	반드시 객체가 생성이 되어야 참조가 가능한 메소드입니다. static 이란 단어를 쓰지 않고(non static), this 참조 변수를 사용할 수 있습니다.

static 메소드로 참조시의 특징

참조하는	참조 되는	참조 방법
static 메소드 (메인)	static 변수	static 변수이므로 클래스 이름으로 참조가 가능합니다.
	인스턴스(non-static) 변수	객체를 생성하여 dot(.) 연산자로 참조해야 합니다.
	static 메소드	static 메소드이므로 클래스 이름으로 참조가 가능합니다.
	인스턴스(non-static) 메소드	객체를 생성하여 dot(.) 연산자로 참조해야 합니다.

클래스(class)

메소드 오버로딩

메소드 이름은 동일하게 작성하되 매개 변수 자료형의 타입이나 개수를 다르게 정의하는 것을 의미합니다.

메소드 시그너쳐(signature) = 메소드의 이름 + 자료형 + 매개 변수의 개수

크게 생성자 오버로딩(클래스 단원에서 배움)과 메소드 오버로딩이 있습니다.

요구 사항

다음 오버로딩된 메소드의 개수는 몇 개인가?

이름	반환 타입
Add(3, 5)	void Add(정수, 정수)
Add(1.1, 2.0)	void Add(실수, 실수)
int result = Add(3, 5, 7)	int Add(정수, 정수, 정수)
Add(100, 200)	void Add(정수, 정수)
Add(2.0, 3.0)	void Add(실수, 실수)
Add(5, 4.0)	void Add(정수, 실수)

정답 : 4 개

이론적으로 오버로딩은 최대 (무한) 개 까지 가능합니다.

오버로딩을 하는 이유

동일한 기능을 수행하는 메소드를 만들 때 마다 매번 이름을 정하는 것은 굉장히 성가신 일입니다.

동일한 이름을 사용하면서 동일한 기능을 메소드로 구현하면 데이터 유형에 구애 받지 않고 자유롭게 사용할 수 있도록 하고자 하는 취지입니다.

클래스(class)

생성자(constructor)

생성자의 정의 및 특징, 그리고 작성 방법에 대하여 살펴 보도록 합니다.

항목	설명
정의	객체가 heap 메모리에 생성된 이후 자바 시스템에 의하여 자동으로 한번 호출되는 특수 메소드를 말합니다.
기본 생성자	생성을 하지 않더라도 눈에 보이지 않고, 매개 변수가 없는 생성자가 하나 존재합니다. 이것을 기본(디폴트) 생성자라고 부릅니다.
특징	생성자가 한 개이면 그 생성자가 기본 생성자가 됩니다. 생성자는 상속이 되지 않습니다. overloading은 가능하지만 overriding은 허용하지 않습니다. 생성자를 명시적으로 만들게 되면 눈에 보이지 않던 생성자는 사라집니다.
작성 방법	생성자 이름은 반드시 클래스 이름과 동일해야 합니다.(일반적으로 public 키워드를 사용합니다.) 반환 타입은 명시할 필요가 없습니다.
용도	인스턴스 변수를 초기화하는 데 사용합니다. 초기화가 필요한 작업들이 여기에 명시되어야 합니다.
주의	매개 변수가 있는 생성자가 존재하는 경우 매개 변수가 없는 생성자를 사용하려면 반드시 매개 변수가 없는 생성자를 구성해야 합니다.

클래스(class)

생성자(constructor)

생성자에 대하여 다음 물음에 답해 보세요.

확인해 봅시다.

생성자는 자바 시스템에 의하여 여러 번 호출이 가능합니다. (O X) 객체가 heap 메모리에 생성된 이후 자바 시스템에 의하여 자동으로 한번 호출되는 특수 메소드를 말합니다.

개발자가 생성자를 정의해도 눈에 보이지 않는 생성자는 계속 존재합니다.(O X)

생성자 구현시 반환 타입은 반드시 명시해야 합니다.(O X)

생성자는 주로 static | 인스턴스 | 지역 변수를 초기화하는 용도로 사용됩니다.(3가지 중 택일)

2개 이상의 생성자가 정의되는 것을 생성자 (오버로딩)이라고 합니다.

생성자의 이름은 반드시 (클래스)의 이름과 동일하게 작성해야 합니다.

생성자와 메소드의 공통점과 차이점은 무엇인가요.

공통점) 동작, 행위

차이점)

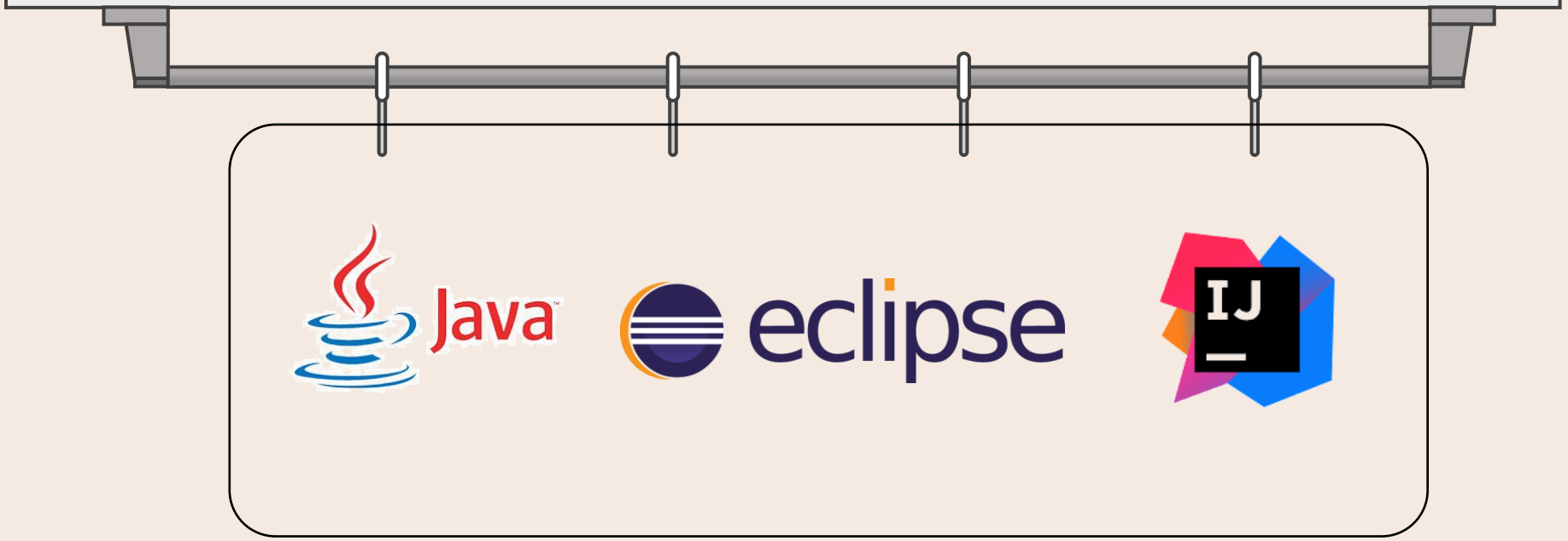
반환 타입 사용 유무

호출이 가능하나?

개발자가 직접 호출 가능한가의 여부

자바 Programming

패키지, 상속, 다형성



패키지, 상속, 다형성

[이론] 패키지(package) 개요

패키지에 대하여 알아 봅시다. https://docs.oracle.com/en/java/javase/12/docs/api/index.html

유사하거나 동등 레벨에서 취급하고자 하는 여러 개의 참조 자료형(클래스, 인터페이스, 예외 등)를 묶어 놓은 폴더의 개념(꾸러미)입니다.
서로 관련 있는 클래스/인터페이스를 하나의 단위로 묶는 것을 의미합니다.

패키지, 상속, 다형성

[이론] 패키지(package) 개요

패키지에 대하여 알아 봅시다. https://docs.oracle.com/en/java/javase/12/docs/api/index.html

항목	설명
정의	유사한 여러 개의 참조 자료형(클래스, 인터페이스, 예외 등)를 묶어 놓은 폴더의 개념(꾸러미). 서로 관련 있는 클래스/인터페이스를 하나의 단위로 묶는 것을 의미합니다.
특징	패키지(package)선언은 주석문을 제외하고 반드시 소스 파일의 첫 줄에 와야 합니다. 단, 기본 패키지는 명시하지 않습니다.
구성 요소	참조 자료형인 클래스, 인터페이스, 예외, 열거형 자료(Enum) 등등이 포함됩니다.
패키지 작성법	package 패키지경로명;

import 키워드	특정 패키지 내의 클래스를 사용하고자 할 때 사용하는 구문입니다.
	import 패키지이름.클래스이름 ;
	동일한 패키지의 모든 클래스를 나타낼 때는 *를 이용합니다.
	이클립스에서는 Ctrl + Shift + o (알파벳 오)를 사용하고, IntelliJ에서는 Alt + Enter 키를 사용하면 자동 import를 실행할 수 있습니다.
	
	import java.util.Calendar ; //Calendar 클래스를 좀 사용하겠습니다.
	import java.lang.*; //java.lang 하위의 모든 클래스를 좀 사용하겠습니다.

dot(.)	폴더 구분자를 의미합니다.
java.lang.*	이 패키지는 암시적으로 import가 되어 있습니다. 그래서, System 클래스는 임포트를 하지 않아도 됩니다.
rt.jar	jdk가 제공해주는 자바 패키지 리스트를 담고 있는 파일(압축 파일) 이 파일을 압축 해제해보면 모든 클래스가 보입니다.

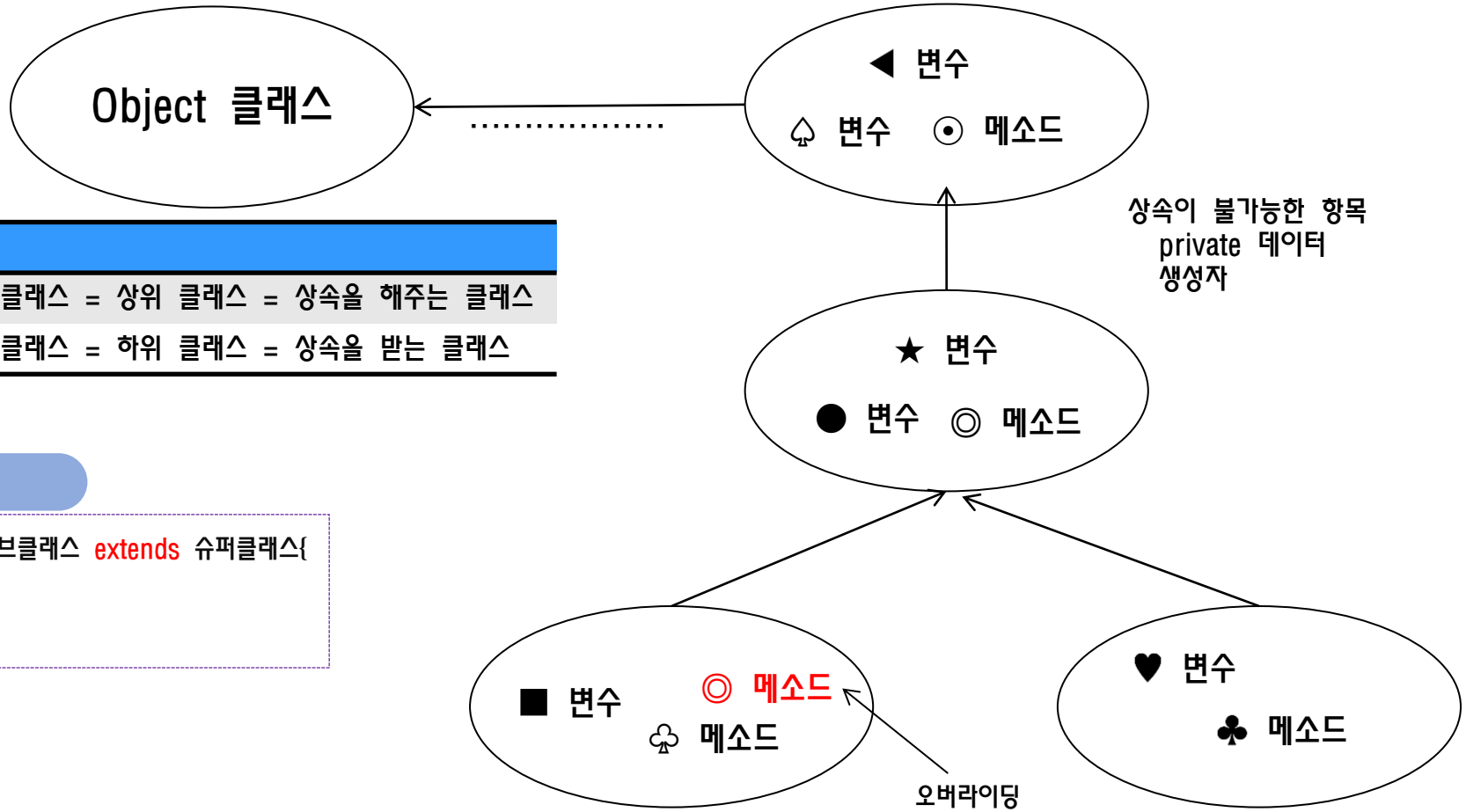
패키지, 상속, 다형성

[이론] 상속(Inheritance) 개요

상속의 개념에 대하여 살펴 보도록 합니다.

부모 클래스가 가지고 있는 변수와 메소드를 그대로 내려 받아서 자식 클래스가 사용할 수 있도록 허가권을 취득 하는 것을 말합니다.

단일 상속만 지원됩니다.(자바는 기본적으로 다중 상속이 불가능합니다.)



항목	설명
수퍼 클래스	부모 클래스 = 상위 클래스 = 상속을 해주는 클래스
서브 클래스	자식 클래스 = 하위 클래스 = 상속을 받는 클래스

사용 형식

```
접근지정자 class 서브클래스 extends 슈퍼클래스{


}
```

패키지, 상속, 다형성

[이론] 상속(Inheritance) 개요

상속(Inheritance)의 문법/특징은 다음과 같습니다.

부모 클래스가 가지고 있는 변수와 메소드를 그대로 내려 받아서 자식 클래스가 사용할 수 있도록 허가권을 취득 하는 것을 말합니다.
단일 상속만 지원됩니다.(자바는 기본적으로 다중 상속이 불가능합니다.)

항목	설명
특징	슈퍼 클래스의 변수와 메소드를 서브 클래스에서 재사용이 가능합니다. 코드의 재사용성(재활용)에 유용합니다. Java는 단일 상속만 허용합니다. 단일 상속의 보완은 인터페이스를 이용하여 보완합니다.
상속이 불가능한 항목	private인 변수(필드)나 메소드는 상속이 불가능합니다. 그리고 생성자는 상속이 불가능합니다.
상속과 생성자	상속 관계에서 생성자는 수퍼 클래스의 생성자가 먼저, 서브 클래스의 생성자가 나중에 호출이 됩니다. 서브 클래스의 생성자에는 다음 문장이 암시적으로 숨겨져 있습니다. super() ; 의 의미는 수퍼 클래스의 생성자중 매개 변수가 없는 생성자를 호출하라.

상속의 문법	// extends는 연장, 확장의 뜻을 내포하고 있습니다.
	[접근지정자] class 서브클래스 extends 수퍼클래스{
	//어떤 코드
	}

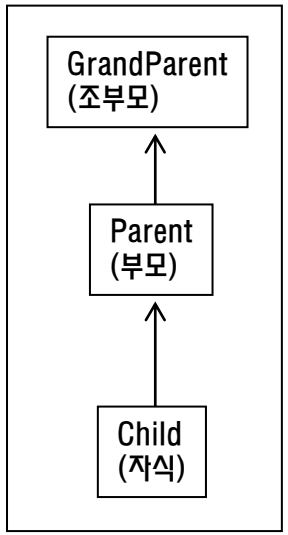
패키지, 상속, 다형성

[이론] super 키워드

super 예약어는 서브 클래스가 슈퍼 클래스를 지칭하고자 할때 사용하는 키워드입니다.

슈퍼 클래스라고 함은 상속 관계에 있는 모든 클래스를 말합니다.

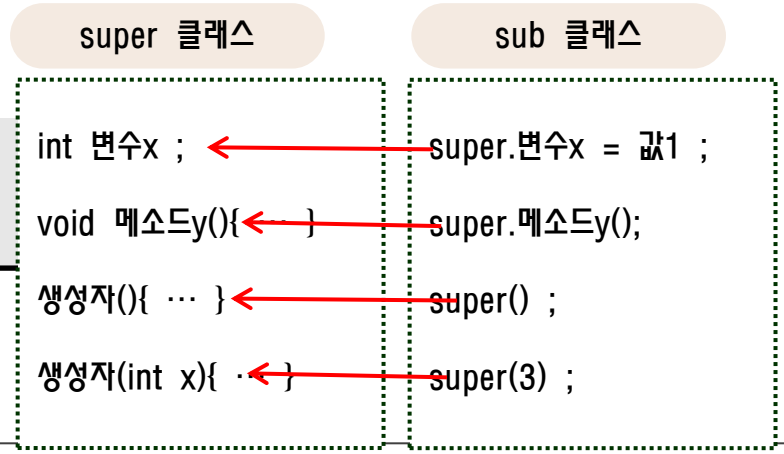
그림에서 Parent와 GrandParent는 둘 다 super 키워드로 접근이 가능합니다.



항목	설명
용도 및 사용 형식	변수 참조 : super.객체변수, 메소드 참조 : super.메소드(매개변수) super(…) : 부모의 생성자를 명시적으로 호출합니다. 단, 접근 지정자가 private이면, super 키워드로 접근이 안 됩니다.

주의 사항	슈퍼 클래스의 생성자를 호출시 매개 변수를 넘기는 경우에는 반드시 서브 클래스의 생성자의 첫 번째 문장에 super(매개변수)를 기록해야 합니다.
	예를 들어서 super("김철수", 20) ; 을 호출한다고 하면 다음과 같이 코딩해야 합니다.
	public Child생성자(){
	super("김철수", 20) ;
	//이하 어떤 코드들…
	}

사용 예시	super() : 매개 변수가 없는 생성자를 호출합니다. super("제시카", 20) : 매개 변수(문자열과 정수)가 있는 생성자를 호출합니다. 이럴 경우 암시적인 super()는 더 이상 호출되지 않습니다.



패키지, 상속, 다형성

Object 클래스

자바의 최상위 클래스로써, 모든 클래스는 Object를 암시적으로 상속 받습니다.

중요 메소드	설명
protected Object clone()	객체를 복사하는 데 사용합니다.
public boolean equals(Object obj)	두 객체의 내용이 동일한지 알아볼 때 사용합니다.
public int hashCode()	개체 식별자 정수 해시 코드 값을 반환합니다. 해시 코드는 객체를 식별하기 위한 고유한 정수 값을 의미합니다.
protected void finalize()	객체를 더 이상 사용하지 않을 때 쓰레기 수집 기능 수행합니다. Deprecated : 이 메소드는 더 이상 사용하지 않도록 권장합니다.
public Class getClass()	객체의 클래스 이름을 Class형으로 반환합니다. 객체가 어떤 클래스로 만들어졌는지를 알려 줍니다.
public String toString()	해당 객체 정보를 문자열 형태로 출력합니다. getClass() + "@" + hashCode()의 16진수 값 println() 메소드로 객체 이름을 출력하는 경우 암시적으로 toString() 메소드를 호출해 줍니다.
public void notify()	대기 중인 스레드를 하나 다시 시작(Wakes up) 시킵니다.
public void notifyall()	대기 중인 스레드를 모두 다시 시작 시킵니다.
public void wait()	스레드의 작동을 중지하고 대기 상태로 만들어 줍니다. 다른 쓰레드가 notify() 메소드 또는 notifyAll() 메소드를 이용하여 호출할 때 까지 대기합니다.

패키지, 상속, 다형성

overriding(재 정의)

오버라이딩(overriding)에 대하여 살펴 봅니다.

슈퍼 클래스에서 정의되어 있는 메소드를 서브 클래스의 취지에 맞도록 다시 정의하는 것을 말합니다.

재(再) 정의라고 합니다. ↔ 중복 정의(overloading)

단, 생성자와 private 변수와 메소드 오버라이딩의 대상이 될 수 없습니다.

항목	설명
특징	슈퍼 클래스의 메소드는 super 키워드를 이용하여 접근이 가능합니다. 단, private 변수나 메소드는 getter을 이용하여 접근이 가능합니다. 오버라이딩시에 슈퍼 클래스의 메소드는 은닉(concealment)됩니다.
제한 조건	메소드의 이름 및 매개 변수가 반드시 일치해야 합니다. 슈퍼/서브의 시그니처가 반드시 동일해야 합니다.
메소드 은닉 (concealment)	오버라이딩을 하게 되면 부모의 메소드가 본의 아니게 자식의 메소드에 의하여 숨겨지는데 이를 메소드 은닉화 현상이라고 합니다.

패키지, 상속, 다형성

[이론] 레퍼런스 형변환

레퍼런스 자료형은 형변환은 다음과 같은 경우에 필요합니다.

이러한 형변환은 프로그램의 유연성을 높여주고 다형성을 활용할 수 있도록 해줍니다. 그러나 형변환은 항상 주의해서 사용해야 합니다.
잘못된 형변환은 런타임 에러를 유발할 수 있습니다.
따라서 형변환을 사용할 때는 반드시 안전하게 캐스팅될 수 있는지를 확인하는 등 주의가 필요합니다.

항목	설명	사용 예시
상위 클래스 또는 인터페이스로의 형변환	여러 개의 클래스가 하나의 상위 클래스 또는 인터페이스를 구현하고 있을 때, 상위 타입으로 레퍼런스를 선언하면 여러 타입의 객체를 동일한 타입으로 취급할 수 있습니다.	class Animal { /* ... */ } class Dog extends Animal { /* ... */ } class Cat extends Animal { /* ... */ } Animal myPet = new Dog(); // 업캐스팅
인터페이스 간 형변환	여러 인터페이스를 구현한 클래스가 있을 때, 해당 인터페이스로 형변환하여 다양한 인터페이스를 구현한 객체로 작업할 수 있습니다.	interface Engine { /* ... */ } interface GPS { /* ... */ } class Car implements Engine, GPS { /* ... */ } Engine myEngine = (Engine) myCar; // 다운캐스팅

하위 클래스로의 형변환	상위 클래스로 선언된 레퍼런스를 하위 클래스 타입으로 형변환할 때가 있습니다.
	이는 업캐스팅된 객체를 다시 다운캐스팅하여 원래의 타입으로 돌리는 경우 등에 사용됩니다.
		class Animal { /* ... */ }
		class Dog extends Animal { /* ... */ }
		
		Animal myAnimal = new Dog(); // 업캐스팅
		Dog myDog = (Dog) myAnimal; // 다운캐스팅

제네릭에서의 형변환	제네릭을 사용하는 경우, 특정한 타입으로 캐스팅이 필요할 수 있습니다.	List<?> myList = new ArrayList<String>(); List<String> myStringList = (List<String>) myList; // 형변환

패키지, 상속, 다형성

[이론] 레퍼런스 형변환

레퍼런스 자료형도 형변환이 가능합니다. (단, 상속 관계의 클래스에만 의미가 있습니다.)

항목	설명
승급(promotion)	서브 클래스가 일시적으로 부모 클래스의 형태로 변환이 되는 것을 의미합니다.(업 캐스팅) 서브 클래스 → 슈퍼 클래스 슈퍼클래스이름 레퍼런스변수 = new 서브클래스이름();
승급의 특징	서브 클래스의 변수는 일시적으로 사용이 불가능합니다. 이 현상을 참조 영역의 축소라고 합니다. 컴파일러에 의한 암시적 형변환, 명시적인 형변환도 가능합니다. Parent01 obj = new Child01();
강등(degradation)	다운 캐스팅 (반드시 명시적 형변환이 필요합니다.) 중요 : 강등이 되면 원래의 변수 값을 다시 사용할 수 있습니다. 슈퍼 클래스 → 서브 클래스 서브/슈퍼 클래스의 메소드 및 변수는 모두 사용 가능
강등의 예시	Object obj = ... ; String munja = (String)obj ; int x = (int)3.14 ;
형 변환 사용처	컬렉션 처리나 파일 입출력 등에 사용됩니다. 웹 프로그래밍(회원 가입, 정보 수정 등등)에 사용됩니다. 스마트 폰 개발(안드로이드에서는 버튼, 체크 박스 등을 object 처리후 형 변환)

패키지, 상속, 다형성

[이론] 레퍼런스 형변환

레퍼런스 자료형도 형변환이 가능합니다. (단, 상속 관계의 클래스에만 의미가 있습니다.)

승급시 변수, 메소드의 접근 범위

승급이 이루어진 경우 변수 및 메소드의 접근 범위는 다음과 같습니다.

항목	수퍼	서브	설명
변수(필드)	접근 가능	접근 불가능	수퍼 클래스 내의 변수에만 접근이 가능합니다. 서브 클래스 내의 변수는 일시적으로 접근이 불가능합니다. 강등이 이루어 지면 접근이 가능합니다 .
메소드	있음	없음	어떤 메소드가 수퍼에만 존재하는 경우, 상속을 받고 있으므로, 수퍼 클래스의 메소드가 실행이 됩니다.
메소드	없음	있음	강등 기법을 통해서만 서브 클래스의 메소드 접근이 가능합니다.
오버라이딩	있음	있음	서브 클래스의 메소드가 우선적으로 실행이 됩니다. 수퍼 클래스의 메소드는 본의 아니게 숨겨 지는데, 이것을 메소드 은닉화 현상이라고 합니다. 수퍼의 메소드는 super.메소드이름()으로 명시적 호출할 수 있습니다.

패키지, 상속, 다형성

instanceof 연산자

특정 객체가 어떠한 클래스로 생성된 것인지를 확인하기 위한 연산자입니다.

반환 값은 true 또는 false 값을 리턴합니다.
if, for 등의 제어문에 사용이 가능합니다.
객체 instanceof 클래스
    [객체]는 [클래스]로 만들어진 인스턴스인가요?
obj instanceof Book
    객체 obj는 Book 클래스로 만들어진 인스턴스입니까?

사용 예시

```
private static void display(Object obj) {
   // 입력되는 매개 변수는 무조건 Object로 승급됩니다.

   if(obj instanceof Integer){
      Integer su = (Integer)obj;
      System.out.println("정수 : " + su);

   }else if(obj instanceof String){
      String mystr = (String)obj ;
      System.out.println("문자열 : " + mystr);

   }else if(obj instanceof Book){
      Book mybook = (Book)obj ; // 강등
      System.out.println("나의 책 정보 : \n" + mybook.toString());
   }
}
```

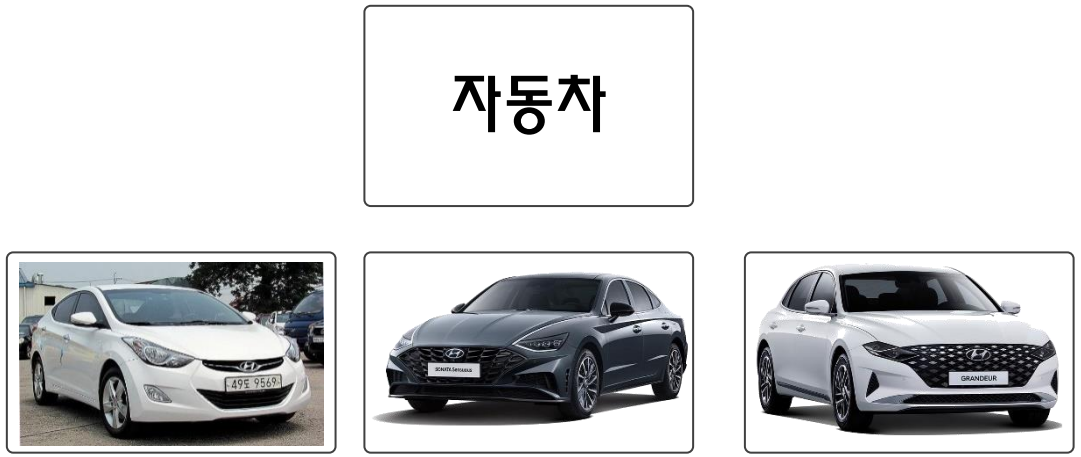
패키지, 상속, 다형성

Polymorphism(다형성)

다형성이란 부모(상위) 객체를 통해 자식(하위) 객체를 표현하는 방법을 말합니다.

수퍼 클래스 타입으로 여러 가지 서브 타입을 취할 수 있다는 개념입니다.

항목	설명
다형성 예시	레퍼런스 형변환, 오버로딩(다중 정의), 오버라이딩(재 정의)
사용 이유	관리의 편리성 : 부모의 이름으로 서브 클래스들을 관리할 수 있습니다. 예를 들어서 ´아반떼´, ´소나타´, ´그랜저´라고 하지 않고, 자동차1, 자동차2, 자동차3 등으로 표현할 수 있습니다. 중복된 코드는 슈퍼 클래스에서만 작성하면 되므로 코드의 양을 줄일 수 있습니다. 수퍼 클래스의 관점에서 서브 클래스들을 배열 형식으로 처리하면 되므로 반복문을 사용할 수 있습니다.
전제 조건	두 클래스는 반드시 상속 관계이어야 합니다.



형변환과 다형성

승급이 되는 경우, 서브가 가지고 있는 변수는 일시적으로 사용이 불가능합니다.

승급이 되는 경우, 서브가 가지고 있는 변수는 일시적으로 사용이 불가능합니다.
단, 원래 타입으로 강등이 되면 서브가 가지고 있는 변수의 값은 다시 사용이 가능합니다.
예를 들어서, Avante 클래스의 comment는 승급시 접근이 불가능 했었습니다.
다시 이것을 강등하여 접근할 수 있었습니다.

어떤 메소드가 수퍼 클래스에만 존재한다면 상속이라는 개념을 이용하여 서브 클래스는 이 메소드를 사용할 수 있습니다.
만약 이 메소드를 오버라이딩하게 되면 프로그램에서 호출시 서브 클래스의 메소드가 실행됩니다.
이때 수퍼 클래스의 메소드는 은닉화 현상이 발생합니다.

패키지, 상속, 다형성

형변환과 다형성

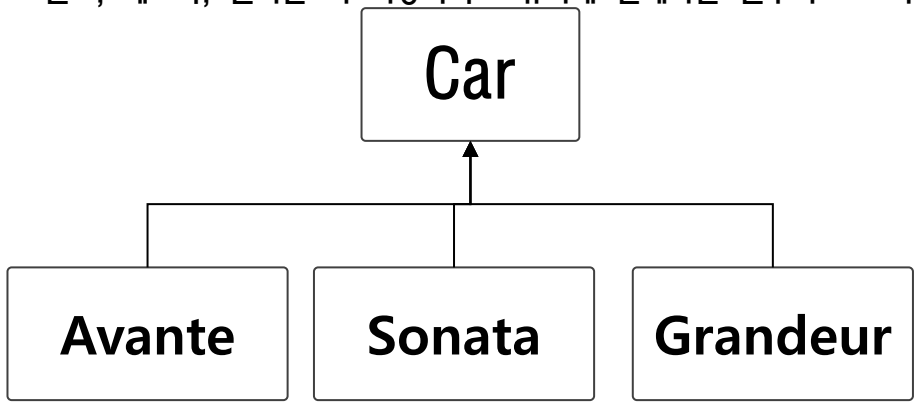
패키지 polymorphism를 구현하기 전에 일반화의 개념부터 살펴 보도록 하겠습니다.

일반화는 보편적인 변수나 메소드는 수퍼 클래스에 작성하고, 특화된 변수나 메소드는 서브 클래스에 작성하는 기법을 의미합니다.

Car 클래스는 Avante, Sonata, Grandeur의 슈퍼 클래스로 사용합니다.

세 차종에서 공통적으로 사용하였던 이름과 단가 변수는 Car 클래스에 두도록 합니다.

코멘트, 제조사, 연비는 각 차종마다 고유하게 존재하는 변수이므로 각 서브 클래스에 작성하도록 합니다.



클래스	항목	설명
Person	멤버 변수	private String name ; private Car car = null ; double tax = 0.0 ; // 세금
	메소드	public void ride(Car car){ ... } private void calcTax(int price){…}
Car	슈퍼 클래스	private String name ; private int price ; 생성자 및 메소드를 필요하면 구현하도록 합니다.
Avante	서브 클래스	private String comment ; //코멘트
Sonata	서브 클래스	private String maker ; //제조사
Grandeur	서브 클래스	private double fuel ; //연비
	생성자, getter()	필요한 경우에 생성하도록 합니다.
CarMain	-	main 메소드가 존재하는 메인 클래스입니다. 배열을 사용하는 법도 다루어 봅니다.

패키지, 상속, 다형성

클래스 이름과 파일 이름

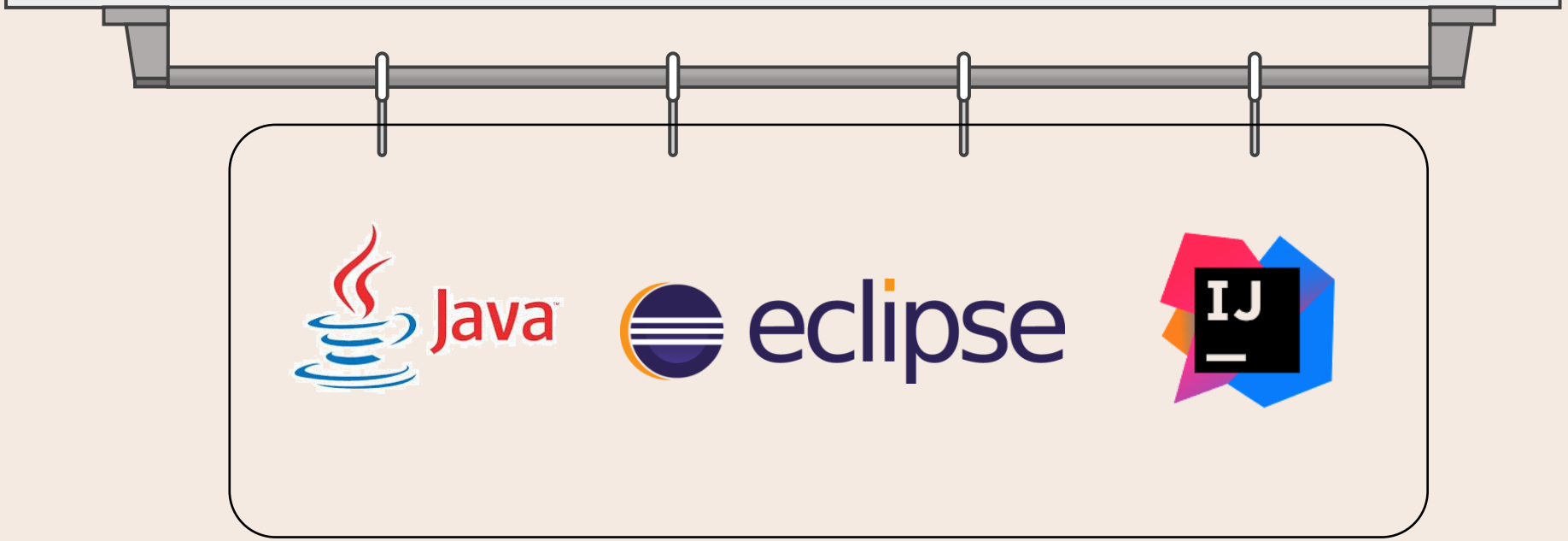
다음과 같이 final 키워드를 다뤄 보는 프로그램을 구현해 보세요.

경우에 따라서는 하나의 파일 내에 2개 이상의 클래스를 작성하는 경우가 있습니다.
이럴 경우 다음과 같이 사용법에 주의를 해야 합니다.

항목	설명
class A{ ... } class B{ ... } class C{ ... }	1개의 파일 내에 좌측과 같이 3개의 클래스가 있다고 가정합니다. 자바는 클래스 이름과 파일 이름이 동일해야 한다는 규칙이 있는데, 좌측의 예시에서는 3개의 클래스 중에 파일 전체를 대표할 수 있는 이름으로 파일명을 정하면 됩니다.
public class A{ ... } class B{ ... } class C{ ... }	클래스 A가 public이므로 반드시 파일 이름이 A.java가 되어야 합니다.
public class A{ ... } public class B{ ... } class C{ ... }	하나의 파일에는 2개 이상의 public가 존재하면 안됩니다. 좌측의 예는 잘못된 Case입니다. 클래스 A, B 중에서 메인(main) 클래스를 하나 지정하고, 나머지 클래스는 public 키워드를 제거하도록 합니다.

자바 Programming

추상 클래스, 인터페이스



추상 클래스, 인터페이스

[이론] 추상 클래스

다음 대화 내용을 살펴 보겠습니다.

도형의 면적을 구하는 방법은 무엇인가요?

어떤 도형을 말하는지 구체적으로 알려주시겠어요?

사각형이요.

사각형의 면적은 밑변과 높이 값을 곱하면 됩니다.

추상 클래스, 인터페이스

[이론] 추상 클래스

추상 클래스에 대하여 살펴 봅니다.

항목	설명
추상 메소드	메소드 이름 앞에 abstract 키워드를 붙여 주는 메소드를 말합니다. 내용물이 존재하지 않는 형태의 모델 개념으로, 불완전한 메소드입니다. 어떠한 구체적인 항목을 만들어 내기 위한 추상적으로 미리 디자인된 개념으로 이해하시면 됩니다.
추상 클래스	클래스 이름 앞에 abstract 키워드를 사용하는 클래스
목적/이유	업무 규칙을 위한 강제적인 표준 규범을 잡기 위하여 사용합니다.
특징	실제 구현할 내용이 없으므로 메소드 바디{...} 대신 세미콜론(;)를 붙여 줘야 합니다. 일반적인 클래스들이 포함할 수 있는 변수 및 메소드 역시 구현이 가능합니다. 자생력이 없어 인스턴스 생성이 불가능한 클래스.(cannot instantiate ...) new 연산자 사용이 불가능합니다. 다만, 승급시 수퍼 클래스로 사용 가능합니다. 자바에서 지원되는 클래스 중 Calendar 클래스가 여기에 해당합니다. 추상 클래스를 상속 받는 추상 클래스는 추상 메소드를 오버라이딩 하지 않아도 됩니다.
제한 조건	추상 메소드가 있는 클래스는 반드시 추상 클래스 이어야 합니다. 추상 클래스를 상속 받는 서브 클래스는 반드시 추상 메소드를 오버라이딩으로 구체화(implementation)해야 합니다.

OX 문제

추상 클래스는 일반 변수를 포함할 수 없습니다.

추상 클래스는 일반 메소드를 포함할 수 없습니다.

추상 클래스 A를 상속 받는 추상 클래스 B는 추상 클래스 A의 모든 추상 메소드를 오버라이딩해야 합니다.

추상 클래스, 인터페이스

[이론] 인터페이스

인터페이스의 기본 개념에 대하여 살펴 봅니다.

항목	설명
목적/이유	자바에서 지원하지 않는 다중 상속을 충족시키기 위하여 도입한 개념입니다.
특징	추상 메소드와 final 키워드를 사용한 상수(constant)로만 구성되어 있습니다. 상속을 받고자 하는 경우 extends 키워드대신 implements 키워드를 사용합니다. 인터페이스간의 상속은 extends 키워드를 사용합니다.
변수	모든 변수에는 public static final 키워드가 암시적으로 선언되어 있습니다. final 변수에 대해서는 반드시 초기 값을 부여 해야 합니다.
메소드	모든 메소드에는 public abstract 키워드가 암시적으로 선언되어 있습니다.
제한 조건	사용되는 메소드는 모두 추상 메소드이어야 합니다. 상속 받는 서브 클래스는 모두 overriding해야 합니다.
인터페이스 사용처	컬렉션(자료 구조), 자바의 GUI(스윙, java fx 등등)에 사용됩니다. 스마트 폰 개발(안드로이드)에 사용됩니다. 웹 프로그래밍 과정, 자바 프레임워크 과정 등에 사용됩니다.

사용 형식

```
[접근_지정자] interface 인터페이스이름 [extends 인터페이스명1, 인터페이스명2,…,인터페이스명n] {
        상수들의 모음 ;
        추상_메서드들의_모음;
}
```

추상 클래스, 인터페이스

[이론] 추상 클래스와 인터페이스

인터페이스에 대하여 다음 문제들을 풀어 봅니다.

OX 문제

인터페이스에는 일반 메소드가 들어갈 수 있습니다.

인터페이스의 모든 변수는 [인터페이스이름.변수이름]으로 접근할 수 있습니다.

인터페이스의 모든 변수는 수정이 가능합니다.

인터페이스 간의 상속에서 하위 인터페이스는 상위 인터페이스의 모든 메소드를 반드시 오버라이딩해야 합니다.

빈 칸 채우기

인터페이스는 자바의 (단일 상속)을 보완하기 위한 방편으로 설계된 개념입니다.

추상 클래스, 인터페이스

[이론] 추상 클래스/인터페이스

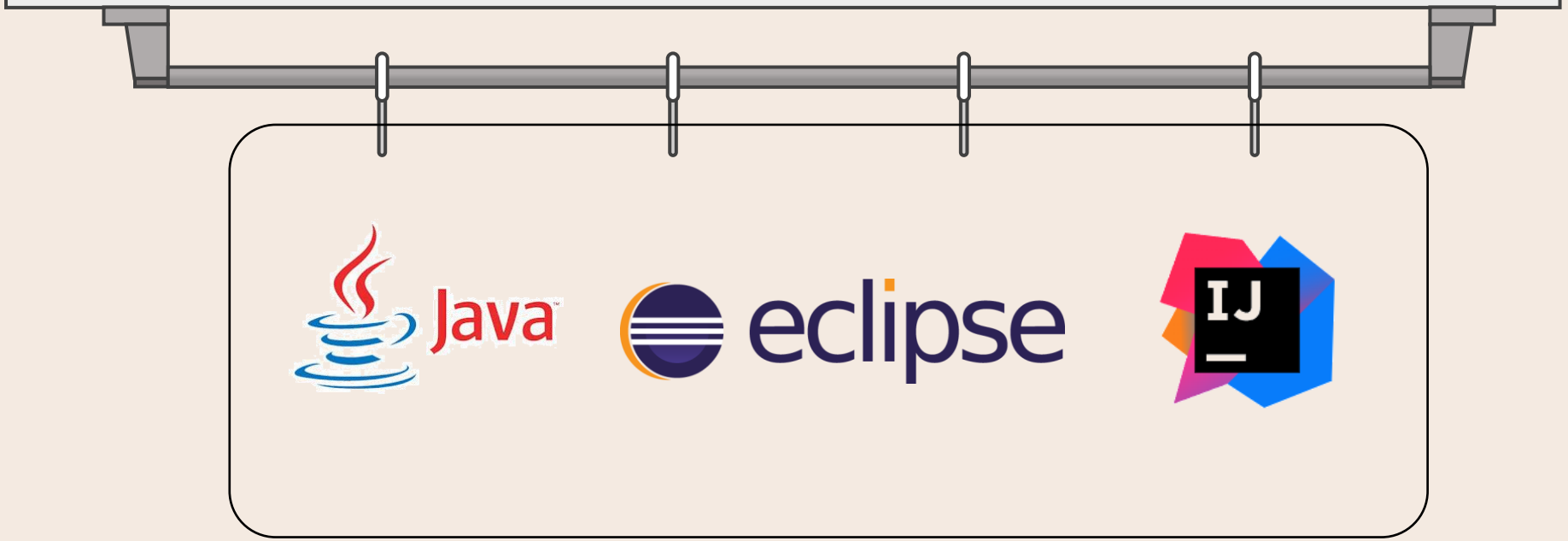
다음 요구 사항대로 프로그램을 구현해 보세요.

종류	추상 클래스	인터페이스
일반 메소드	정의할 수 있습니다.	정의를 한다면, 해당 메소드는 static 메소드 또는 default 메소드이어야 합니다.
추상 메소드	0개 이상 존재합니다.	존재하는 모든 메소드는 추상 메소드입니다. (static 또는 default 메소드는 제외)
일반 변수	변수 및 상수(constant) 모두 정의 가능합니다.	모두 final이 붙으므로, 상수(constant)가 됩니다.
상속 및 구현	한 개의 클래스만 상속 가능합니다.(단일 상속)	여러 개의 인터페이스를 상속할 수 있습니다.(다중 상속)
메소드 구현	일반 메소드는 필요에 의하여 오버라이딩하면 됩니다. 추상 메소드는 반드시 오버라이딩되어야 합니다.	모든 메소드는 반드시 오버라이딩되어야 합니다. (static 또는 default 메소드는 제외)
키워드	abstract	interface
상속	extends 키워드를 사용합니다.	implements 키워드를 사용합니다.
용도	-	이벤트 프로그래밍(자바 GUI, 안드로이드 등등)

super	sub	상속 키워드
class	class	extends
interface	class	implements
interface	interface	extends
class	interface	X

자바 Programming

주요 유틸리티 클래스



주요 유틸리티 클래스

주요 클래스 개요

다음 클래스들은 자바에서 많이 사용되는 유틸리티 클래스입니다.

패키지	클래스	설명
java.lang	Object	모든 클래스의 슈퍼 클래스
java.lang	Math	수학과 관련된 클래스
java.util	Date	날짜와 시각에 대한 클래스
java.util	Calendar	날짜와 시각에 대한 조작을 수행하는 추상 클래스입니다.
java.lang	Wrapper	기본 자료형에 대한 클래스(int, char, double 등) Integer와 같이 기본 자료형을 감싸서 제공하는 랩퍼 클래스들
java.lang	System	표준 입출력 클래스
java.lang	String	문자열 관련 클래스
java.lang	Thread	쓰레드 기능을 제공하는 메소드
java.lang	Class	객체를 생성한 클래스에 대한 정보를 얻어 내기 위한 클래스
java.lang	StringBuilder	String 클래스의 단점을 보완한 클래스
java.lang	StringBuffer	String 클래스의 단점을 보완한 클래스
java.util	StringTokenizer	문자열로부터 토큰을 추출하는 클래스

주요 유틸리티 클래스

주요 클래스 개요

다음 클래스들은 자바에서 많이 사용되는 유틸리티 클래스입니다.

패키지	클래스	설명
java.util	Arrays	
java.util	Random	
java.text	SimpleDateFormat	
java.text	DecimalFormat	

주요 유틸리티 클래스

Calendar 클래스

밀레니엄 버그와 관련하여 Date 클래스를 위한 보조 클래스로 설계되었습니다.

개념 살펴 보기

날짜와 시각에 대한 조작을 수행하는 추상 클래스로써. 객체 생성은 Calendar now = Calendar.getInstance(); 으로 사용됩니다.

메소드	설명
abstract void add(int field, int amount)	지정한 필드에 시간을 더하기 또는 빼기 연산을 합니다.
boolean after(Object when)	현재 Calendar 객체가 when 객체보다 미래의 날짜이면 true
boolean before(Object when)	현재 Calendar 객체가 when 객체보다 과거의 날짜이면 true
void clear(int field)	지정된 필드를 정의되지 않은 상태로 변경합니다.
int get(int field)	인자로 전달된 필드의 값(년, 월, 일, 시, 분, 초)을 반환합니다.
static Calendar getInstance()	default Timezone과 Locale을 사용하여 Calendar 객체를 반환합니다.
void set(int year, int month, int date)	년, 월, 일(DAY_OF_MONTH)을 이용하여 시각을 설정합니다.
void setTimeInMillis(long millis)	Calendar의 현재 시각을 인자로 전달된 long 형의 값으로 설정합니다.
long getTimeInMillis()	캘린더 객체의 값을 밀리 세컨드 값으로 리턴해줍니다.
int getActualMaximum(int)	명시된 field의 최대 값을 리턴해줍니다. 예를 들어 12월의 마지막 날은 31입니다.
int cimpareTo(Calendar another)	두 개의 Calendar 객체를 비교합니다.
boolean equals(Object obj)	두 개의 Calendar 객체가 같으면 true를 반환합니다.
Date getTime()	Calendar 객체를 Date 객체로 반환합니다.
void setTime(Date date)	Date 객체의 값으로 Calendar 객체를 설정합니다.

주요 유틸리티 클래스

Calendar 클래스

밀레니엄 버그와 관련하여 Date 클래스를 위한 보조 클래스로 설계되었습니다.

개념 살펴 보기

날짜 관련 많이 사용되는 항목들을 static final 형식의 읽기 전용 상수로 미리 정의해 두고 있습니다.

상수(constant)	설명
AM_PM	HOUR이 오전(AM)이면 0, 오후(PM)이면 1을 반환합니다.
DAY_OF_MONTH	한 달 중의 날짜 수를 의미하는 상수(1 ~ 31) 값입니다.
DAY_OF_WEEK	Sunday ~ Saturday까지의 요일을 나타내는 상수(1 ~ 7)
DAY_OF_YEAR	한 해 중에서 몇 일째인지를 나타내는 상수 값입니다.
DAY_OF_WEEK_IN_MONTH	한달 중 해당 요일이 몇 번째인지 나타내는 상수 값입니다.
HOUR	12시간제(오전 오후의 구분 없이 몇 시인지를 나타내는 상수)
HOUR_OF_DAY	24시간제를 의미합니다. 하루 중 시각을 나타내는 상수(0 ~ 23) 값입니다.
MONTH	달을 의미하는 상수 값으로 배열 형식이 됩니다.(1월이 0입니다.(0 ~ 11)
WEEK_OF_MONTH	이번 달에서 몇 번째 주인지를 나타내는 상수 값입니다.
WEEK_OF_YEAT	올해의 몇 번째 주인지를 나타내는 상수 값입니다.
DATE	

주요 유틸리티 클래스

Object 클래스

모든 클래스들에 대하여 최상위에 존재하는 클래스입니다.

메소드	설명
protected Object clone()	객체를 복사하는 데 사용됩니다.
public boolean equals(Object obj)	두 객체의 내용이 동일한지 알아볼 때 사용됩니다.
public int hashCode()	개체 식별자 정수 해시 값을 반환합니다. 해시코드는 검색을 용이하게 만들기 위한 고유 정수 값.
protected void finalize()	객체를 더 이상 사용하지 않을 때 쓰레기 수집 기능 수행.
public Class getClass()	객체의 클래스 이름을 Class 형으로 반환합니다.
public String toString()	객체의 문자열을 반환합니다.
public void notify()	대기 중인 스레드를 하나 다시 시작합니다.
public void notifyall()	대기 중인 스레드를 모두 다시 시작합니다.
public void wait()	스레드의 작동을 중지하고 대기 상태로 만들어 줍니다.

주요 유틸리티 클래스

Object 클래스

toString() 메소드 오버라이딩

주요 클래스들은 Object 클래스의 toString()을 미리 오버라이딩하여 자신의 취향 대로 미리 만들어 두고 있습니다.
예를 들면, Integer 클래스는 정수 값을 취급해주는 클래스인데, 해당 정수의 값을 문자열 형태로 출력을 하기 위하여 미리 Override 오버라이딩했습니다.

클래스	설명
Object	객체를 다음 정의대로 문자열 형태로 리턴해줍니다. 정의 : getClass().getName() + '@' + Integer.toHexString(hashCode()) 레퍼런스 변수 호출 : 레퍼런스변수.toString()가 호출(암시적).
String	문자열 형식의 해당 문자열을 그대로 출력해줍니다.
Integer	숫자 형식의 해당 문자열을 그대로 출력해줍니다.

주요 유틸리티 클래스

Object 클래스

equals() 메소드 오버라이딩

항목	자료형	설명
== 키워드	기본 자료형	값의 내용이 동일한지 확인합니다.
	레퍼런스 변수	동일한 인스턴스를 참고 하고 있는 지 확인할 때 사용합니다.
equals() 메소드	레퍼런스 변수	레퍼런스 변수가 동일한 내용을 저장하고 있는지를 따질 때 사용합니다.

```
int x = 3, y = 4 ;
if(x == y){} // 값 비교

if(객체a == 객체b){} // 참조 비교
```

equals() 메소드의 기본 행동은 참조 비교 입니다.
이것을 값 비교로 바꿔서 내가 추구하고자 하는 대로 변경할 수 있습니다.

주요 유틸리티 클래스

Math 클래스

수학과 관련된 상수(static, final)와 메소드(static 메소드)로 구성된 클래스입니다.

개념 살펴 보기

모든 상수에는 static과 final 키워드가 붙어 있어, 클래스 이름으로 접근 가능하고, 읽기 전용입니다.
모든 메소드는 static 키워드가 붙어 있어 클래스 이름으로 접근이 가능합니다.

메소드	설명
static double abs(double a)	절대 값
static double ceil(double a)	올림 값
static double floor(double a)	버림 값
static int round(double a)	반올림
static double max(double a, double b)	큰 수 구하기
static double min(double a, double b)	작은 수 구하기
static double random()	0 <= 결과 < 1.0의 임의의 double 값.
static double signum (double d)	양수이면 1, 음수이면 -1을 0이면 0을 반환합니다.
static double pow (double a, double b)	숫자 a의 b승을 구합니다.
static double sqrt (double a)	a의 루트(평방근)을 구합니다.
static double hypot(double x, double y)	직각 삼각형의 빗변의 길이를 구해줍니다.

주요 유틸리티 클래스

Wrapper 클래스

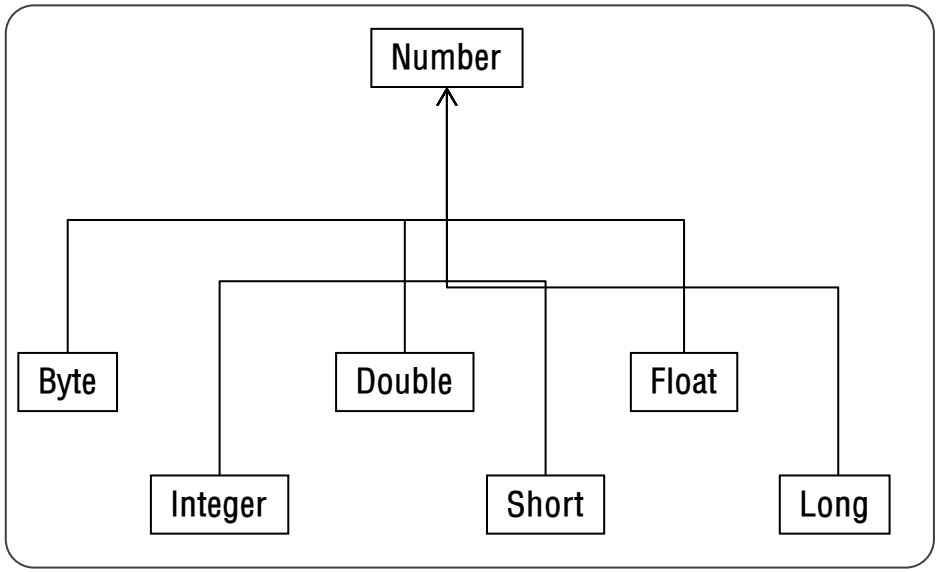
wrapper class는 기본 자료형을 포장해주는 클래스입니다.

개념 살펴 보기

자바는 기본 자료형(primitive type)과 참조 자료형(reference type)의 데이터 자료가 있습니다.
기본 자료형은 char, int, float, double, boolean 등이 있습니다.
참조 자료형은 배열과 class, interface 등이 있습니다.
기본 자료형을 객체 지향 형태로 표현하기 위하여 사용되는 개념이 wrapper(포장)입니다.
그리고, 이러한 형태로 만들어진 클래스들을 wrapper class라고 부릅니다.
기본 자료형을 내부에서 포장하고, 외부에서는 객체 형식으로 접근하는 방식입니다
기본 자료형에 대응하는 랩퍼 클래스가 각각 존재합니다.

기본 자료형	랩퍼(Wrapper) 클래스
boolean	Boolean
byte	Byte
char	Character
short	Short
int	Integer
long	Long
float	Float
double	Double

클래스 계층



주요 유틸리티 클래스

Integer 클래스

정수를 다루기 위한 Integer 클래스가 제공하는 생성자/메소드의 일부는 다음과 같습니다.

생성자	설명
Integer(int value)	value는 int 형의 값
Integer(String str) throws NumberFormatException	str은 int 형과 같은 의미의 문자열 예)"110" => int 형 110과 같은 뜻

메소드	설명
static int parseInt(String str)	str로 지정된 문자열에 해당하는 int 값을 반환합니다.
static String toString()	해당 정수를 문자열로 만들어서 출력해준다
static Integer valueOf(String str)	str로 지정된 문자열에 해당하는 Integer 객체를 반환합니다.
static Integer valueOf(String str, int radix)	str로 지정된 문자열을 radix 진법의 Integer 객체로 반환합니다.
static String toBinaryString(int i)	int 형의 정수를 2진수로 표현된 문자열을 반환합니다.
static String toHexString(int i)	int 형의 정수를 16진수로 표현된 문자열을 반환합니다.
static String toOctalString(int i)	int 형의 정수를 8진수로 표현된 문자열을 반환합니다.
static int signum(int i)	양수이면 1, 음수이면 -1, 0이면 0을 반환합니다.
static int intValue()	int 형으로 반환합니다.
static double doubleValue()	double 형으로 반환합니다.
static float floatValue()	float 형으로 반환합니다.

주요 유틸리티 클래스

Wrapper 클래스

오토 박싱과 언박싱에 대하여 살펴 봅니다.

개념 살펴 보기

기본 자료형은 stack 영역에 저장이 됩니다.
레퍼런스 변수는 stack 영역에 저장되고, 해당 인스턴스 변수(heap 영역에 저장)를 참조합니다.
stack 영역의 기본 자료형(int)이 참조 자료형(Integer)으로 변환되는 과정을 Boxing이라고 합니다.
물론 그 반대 현상은 언박싱이라고 합니다.
JDK 5.0 이후의 버전에서는 이러한 기술이 자동으로 이루어지는 데, 이것을 AutoBoxing이라고 합니다.

항목	JDK 5.0 이전 방식	JDK 5.0 이후 방식
박싱	int a = 10 ; Integer intA = new integer(a) ;	int c = 30 ; Integer intC = c ; //auto boxing
언박싱	Integer intB = new Integer(20) ; int b = intB.intValue();	Integer intD = new Integer(40) ; int d = intD; //auto unboxing

주요 유틸리티 클래스

Number 클래스

Wrapper 클래스들을 지원해주기 위한 추상 클래스입니다.

개념 살펴 보기

추상 클래스 Number는 BigDecimal, BigInteger, Byte, Double, Float, Integer, Long 및 Short 등의 클래스들을 위한 슈퍼 클래스입니다.
Character, Boolean 클래스는 제외됩니다.
하위 클래스들은 필요한 경우 Number 클래스의 추상 메소드들을 미리 구현해 두고 있습니다.(´클래스 구현´ 표 참조)

클래스 구현

메소드	설명
public byte byteValue()	객체의 값을 byte 형으로 변환해 줍니다.
public double doubleValue()	객체의 값을 double 형으로 변환해 줍니다.
public float floatValue()	객체의 값을 float 형으로 변환해 줍니다.
public int intValue()	객체의 값을 int 형으로 변환해 줍니다.
public long longValue()	객체의 값을 long 형으로 변환해 줍니다.
public short shortValue()	객체의 값을 short 형으로 변환해 줍니다.

주요 유틸리티 클래스

Character 클래스

자바의 기본 자료형 중에서 char의 값을 객체로 포장해주는 역할을 합니다.

개념 살펴 보기

문자의 형태를 판별해서 대문자/소문자/숫자/문자인 지 등등을 판별해주는 다양한 메소드를 제공하고 있습니다.

메소드	설명
public static boolean isDefined(Char ch)	ch가 유니코드이면 true 값을, 그렇지 않으면 false 값을 반환해 줍니다.
public static boolean isDigit(Char ch)	ch가 숫자이면 true 값을, 그렇지 않으면 false 값을 반환해 줍니다.
public static boolean isLetter(Char ch)	ch가 문자이면 true 값을, 그렇지 않으면 false 값을 반환해 줍니다.
public static boolean isLetterOrDigit(Char ch)	ch가 문자/숫자이면 true 값을, 그렇지 않으면 false 값을 반환해 줍니다.
public static boolean isLowerCase(Char ch)	ch가 소문자이면 true 값을, 그렇지 않으면 false 값을 반환해 줍니다.
public static boolean isSpaceChar(Char ch)	ch가 공백이면 true 값을, 그렇지 않으면 false 값을 반환해 줍니다.
public static boolean isUpperCase(Char ch)	ch가 대문자이면 true 값을, 그렇지 않으면 false 값을 반환해 줍니다.
public static char toLowerCase(Char ch)	ch를 소문자로 변형해 줍니다.
public static char toUpperCase(Char ch)	ch를 대문자로 변형해 줍니다.

주요 유틸리티 클래스

문자열 계열 클래스

문자열을 다루는 클래스로 String, StringBuffer, StringBuilder이라는 3가지 클래스가 있습니다.

개념 살펴 보기

각각의 클래스의 특징을 살펴본 후 적절한 클래스를 사용하면 됩니다.

버퍼(Bufer)는 데이터를 한 곳에서 다른 한 곳으로 전송하는 동안 일시적으로 그 데이터를 보관하는 메모리의 영역입니다.
버퍼링(buffering)이란 버퍼를 활용하는 방식 또는 버퍼를 채우는 동작을 말합니다.
다른 말로 ´큐(Queue)´라고도 표현합니다.
출처) 위키 백과

항목	String	StringBuffer	StringBuilder
버퍼 크기의 유연성	불변	가변	가변
쓰레드 동기화 지원	-	멀티 쓰레드	단일 쓰레드
처리 속도	-	느림	빠름
주 사용처	적은 양의 문자열 수정	+ 연산자를 많이 사용하는 경우	+ 연산자를 많이 사용하는 경우
단순 조회	빠름	느림	느림

주요 유틸리티 클래스

String 관련 클래스

String 클래스와 StringBuffer 클래스의 차이점에 대하여 살펴 봅니다.

요구 사항

String 클래스는 주로 문자 상수열을 나타낼 때 사용이 됩니다.

자바는 변경 가능한 문자열을 위하여 String 클래스의 대안으로 StringBuffer와 StringBuilder 클래스를 제공합니다.

String 객체의 '+' 연산은 내부적으로 계속 String 객체를 생성하게 되어 있습니다.

반복적으로 자주/많이 사용되는 문자열 연산은 되도록 StringBuffer 객체를 이용합니다.

String 객체	StringBuffer 객체

String str = new String();
String strTemp = "가나다";
 	
str += strTemp; // 가나다
str += strTemp; // 가나다가나다
	StringBuffer strBuf = new StringBuffer();
	String strTemp = "가나다";
	
	strBuf.append(strTemp); // 가나다
	strBuf.append(strTemp); // 가나다가나다

String 객체

주소 = 100	가나다	
주소 = 150	가나다	가나다

StringBuffer 객체

주소 = 100	가나다	
주소 = 100	가나다	가나다

주요 유틸리티 클래스

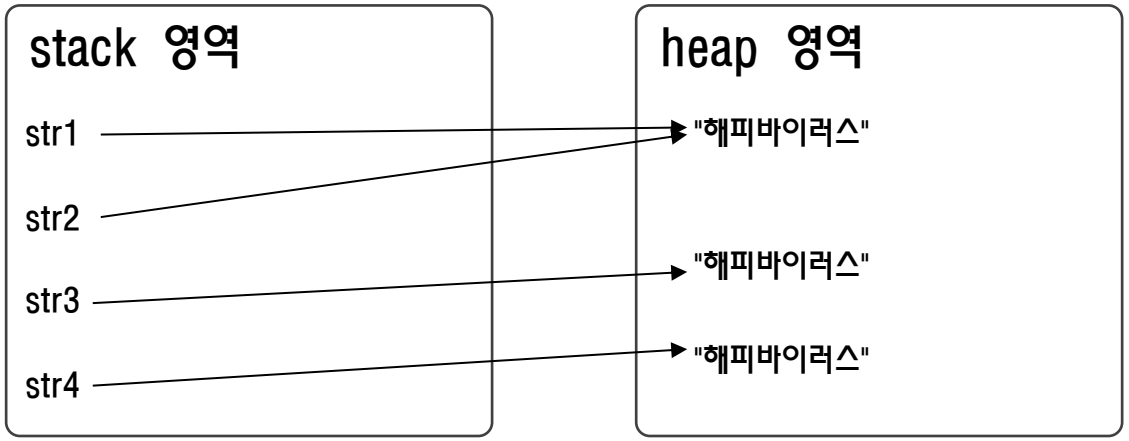
String 클래스

String 클래스와 메모리 구조에 대하여 살펴 봅니다.

개념 살펴 보기

문자열 상수는 컴파일을 수행하면 자동으로 heap 영역에 String 객체가 생성이 됩니다.
동일한 문자열 상수()이미 존재하는 문자열 상수로 다시 사용한다면이미 생성된 문자열 상수를 **공유**하게 됩니다.
객체 생성시 new 연산자를 사용하면 **별도의 기억 공간이 생성**됩니다.

```
String str1 = "해피바이러스" ;
String str2 = "해피바이러스" ;
String str3 = new String("해피바이러스)" ;
String str4 = new String("해피바이러스)" ;
```



주요 유틸리티 클래스

String 클래스의 주요 메소드

String 클래스의 주요 메소드는 다음과 같습니다.

주요 메소드	설명
String concat(String str)	문자열을 결합해 줍니다.
boolean equals(Object ob)	String 클래스의 객체를 비교(즉 문자열 비교)합니다.
boolean equalsIgnoreCase(Object ob)	대소문자를 구분하지 않고 비교합니다.
int indexOf(int ch) int indexOf(String str)	지정한 문자/문자열의 인덱스를 리턴합니다.
int length()	문자열 길이를 구해줍니다.
String substring(int start)	start 인덱스부터 문자열의 끝까지의 문자열을 추출합니다.
String substring(int start,int end)	start부터 end 직전까지 문자열을 추출합니다.
String replace(char old, char new)	문자열을 치환(교환)합니다.
String toLowerCase()	소문자로 변환해 줍니다.
String toUpperCase()	대문자로 변환해 줍니다.
char charAt(int index)	index 번째 인덱스의 문자 1개를 추출합니다.
boolean startsWith(String str)	str 문자열로 시작이 되면 true 값을 반환해 줍니다.
boolean endsWith(String str)	str 문자열로 끝나면 true 값을 반환해 줍니다.

주요 유틸리티 클래스

StringBuffer 클래스의 주요 메소드

StringBuffer 클래스의 주요 메소드는 다음과 같습니다.

주요 메소드	설명
int length()	문자열 개수를 반환해 줍니다.
int capacity()	버퍼 크기를 반환해 줍니다.
append(String str)	끝 부분에 str을 추가합니다.
insert(int offset, String str)	offset 위치에 str을 추가합니다.
String toString()	StringBuffer를 String 타입으로 반환해 줍니다.
append(boolean b)	현재 문자열 끝에 b를 추가합니다.
char charAt(int index)	index 위치에 해당 문자를 반환해 줍니다.
insert(int i , boolean b)	i 번째 인덱스 앞에 요소 b를 추가합니다.
reverse()	문자열을 역순으로 변환해 줍니다.
subSequence(int start, int end)	start부터 end 직전까지의 문자열을 추출합니다.

주요 유틸리티 클래스

StringBuilder 클래스의 주요 메소드

StringBuilder 클래스의 주요 메소드는 다음과 같습니다.

주요 메소드	설명
append(문자열)	문자열을 맨 뒤쪽에 추가합니다.
insert(인덱스, 문자열)	지정 인덱스에 문자열을 추가해 줍니다.
length	문자열의 길이를 구합니다.
charAt(인덱스)	지정 인덱스 위치를 문자를 리턴합니다.
int length()	문자열 길이를 구해 줍니다.
substring(시작인덱스, 끝인덱스+1)	시작 인덱스부터 끝 인덱스까지의 문자열을 리턴합니다.
lastIndexOf(문자열)	뒤에서부터 [문자열]을 검색하여 발견된 인덱스를 리턴합니다.
delete(시작인덱스, 끝인덱스+1)	시작 인덱스부터 끝 인덱스까지의 문자열을 삭제합니다.
replace(시작인덱스, 끝인덱스+1, 문자열)	시작 인덱스부터 끝 인덱스까지의 문자열을 지정 문자열로 변경해 줍니다.
deleteCharAt(인덱스)	지정 인덱스의 문자를 삭제합니다.
setCharAt(지정인덱스, 문자)	지정 인덱스의 문자를 변경합니다.
setLength(길이)	지정 문자열의 길이를 [길이]만큼 셋팅합니다.
reverse	문자열의 위치를 꺼꾸로 뒤집어 줍니다.

주요 유틸리티 클래스

StringTokenizer 클래스

문자열을 분석하여 토큰(Token)으로 분리 시켜 주는 기능을 제공합니다.

개념 살펴 보기

토큰 분리자(delimiter)는 문자열을 분리하고자 할때 분리 기준 문자열을 의미합니다.
기본 값으로 지정되지 않으면 공백 문자(space)를 기준으로 토큰을 분리해 줍니다.
잘라 내어진 각 문자열 각각을 token(토큰)이라고 합니다.

```
String str ;
str = "소녀시대 빅뱅 원더걸스" ;

//스페이스 바로 문자열을 분리한다
StringTokenizer st1 = new StringTokenizer(str);

while( st1.hasMoreTokens() ){
    String item = st1.nextToken() ;
    System.out.println( item );
}
```

st1

소녀시대
빅뱅
원더걸스

생성자	설명
StringTokenizer(String str)	토큰 분리자가 없는 경우 공백 문자를 토큰 분리자로 사용합니다.
StringTokenizer(String str, String delimeters)	문자열 str에서 delimeters는 토큰 분리자입니다.
StringTokenizer(String str, String delimeters, boolean returnDelims)	주어진 문자열을 위한 StringTokenizer 객체를 생성합니다. returnDelims이 true이면 분리자를 포함하여 반환합니다.

메소드	설명
int countTokens()	문자열에 존재하는 토큰의 개수를 반환합니다.
boolean hasMoreTokens()	다음 토큰이 있으면 true, 없으면 false을 반환합니다.
String nextToken()	다음 토큰 문자열을 반환해주고, 목록에서 요소가 제거 됩니다.
String nextToken(String delimeters)	다음 토큰 문자열 반환하고, 토큰 분리자를 delimeters으로 변경합니다.

주요 유틸리티 클래스

Arrays 클래스

Arrays 클래스는 배열을 쉽게 다루기 위한 유틸리티 클래스입니다.

개념 살펴 보기

Arrays 클래스는 배열을 다루기 위한 유틸리티 클래스입니다.
정렬, 탐색과 같은 메소드들을 제공합니다.
배열을 리스트 컬렉션 형태로 만들어 주는 메소드도 제공합니다.
관련 메소드(공간의 절약을 위하여 정수형 int만 명시했습니다.)

메소드	설명
static List asList(Object[] a)	주어진 배열을 고정 길이의 리스트 컬렉션으로 반환합니다.
static int binarySearch(int[] a, int key)	주어진 값을 int 형의 배열에서 이진 탐색합니다.
static int binarySearch(Object[] a, Object key)	Object 타입의 배열에서 key를 이진 탐색합니다. 예시) int 배열인덱스 = Arrays.binarySearch(배열, 찾을값); //요소가 없으면 -1
static int[] copyOf(int[] original1, int newLength)	주어진 배열을 새로운 크기의 배열에 복사합니다.
static int[] copyOfRage(int[] original1, int from, int to)	주어진 배열에서 주어진 구간의 값들을 새로운 배열에 복사합니다.
static boolean equals(int[] a, int[] a2)	두 배열의 내용 비교(equals는 배열 요소 크기와 내용이 모두 동일해야만 true)
static void fill(int[] a, int val)	주어진 값을 가지고 각각의 배열 요소에 채워 줍니다.
static void sort(int[] a)	지정된 int 형의 배열을 오름차순으로 정렬합니다.

주요 유틸리티 클래스

SimpleDateFormat 클래스

SimpleDateFormat의 Date/Time Pattern

SimpleDateFormat 클래스는 날짜에 대한 서식을 간결하게 만들어 주는 포맷 클래스입니다.

날짜와 시각에 대한 포맷은 다음과 같은 포맷 패턴 문자열을 사용하여 명시할 수 있습니다.

심벌 문자	Date or Time Component	Presentation	예시
y	Year	Year	2022; 22
Y	Week year	Year	2009; 09
M	Month in year	Month	July; Jul; 07
w	올해의 몇 번째 주인가?	Number	27
W	Week in month	Number	2
D	올해의 몇 일째인가?	Number	189
d	Day in month	Number	10
E	Day name in week(요일 이름)	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
m	Minute in hour	Number	30
s	Second in minute	Number	55
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00

주요 유틸리티 클래스

Class 클래스

이미 컴파일이 된 class 파일에 저장된 클래스나 인터페이스 정보를 읽고자 할 때 사용합니다.

Class 클래스의 필요성

수많은 클래스 중에서 클래스에 대한 분기가 필요한 경우
반환 받는 클래스가 정확히 어떤 클래스인지 불명확한 경우

Class 클래스에서 얻을 수 있는 것

특정 Class가 소유한 생성자, 변수, 메소드 정보 읽기

클래스 정보 읽어 오는 방법

- Object.getClass() 메소드를 사용하는 방법

```
String str = new String("");
Class cls = str.getClass();
```

- 클래스 파일 이름을 직접 Class 변수에 대입하는 방법

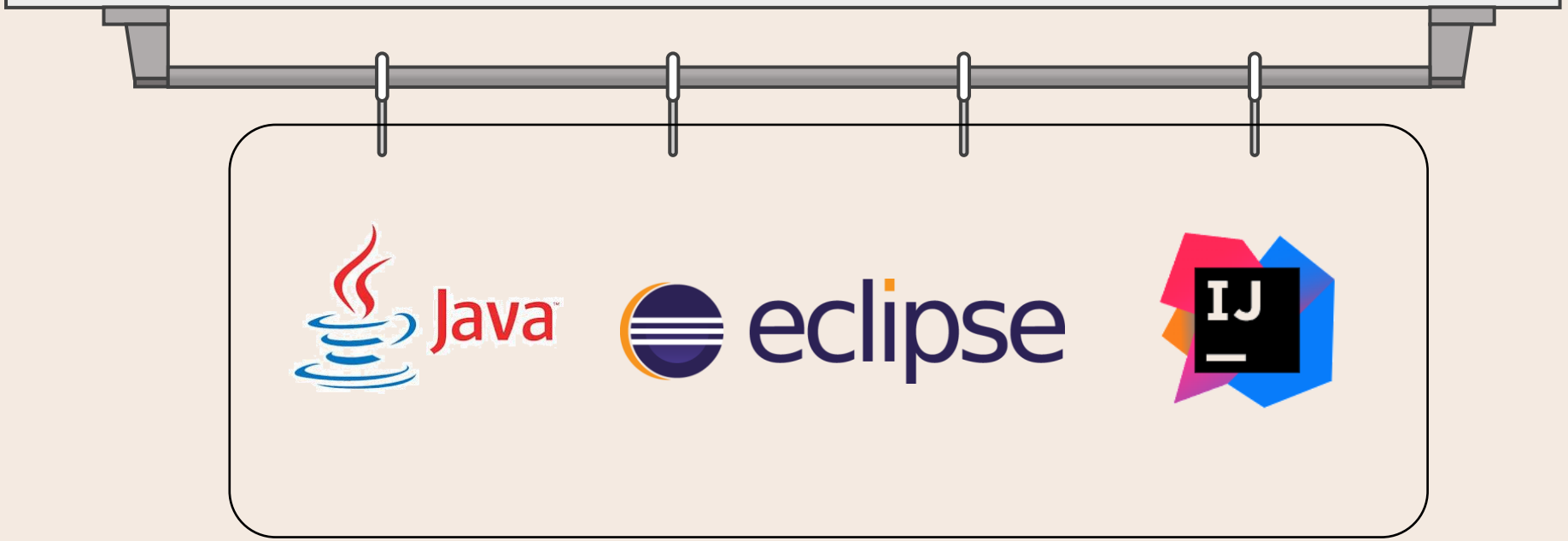
```
Class cls = String.getClass();
```

- Class.forName("") static 메소드를 사용하는 방법

```
Class cls = Class.forName("java.lang.String");
```

자바 Programming

예외(exception) 처리



예외(exception) 처리

예외 처리 개요

예외를 처리하는 방법에 대하여 살펴 보고자 합니다.

항목	항목에 대한 설명
정의	프로그램 수행을 방해하는 사건 또는 에러. 사용자의 코드 상의 부주의로 인하여 발생하는 문제점.
특징	자바에서는 여러 가지 Exception을 클래스 형태로 미리 만들어 두고 있습니다. 예시 : ArithmeticException는 [0으로 나누기를 수행시 발생하는 예외] 여러 클래스를 열거할 경우 상위 클래스는 가장 하단에 열거하도록 합니다. Unreachable catch block for ArithmeticException ... 상위 예외 클래스가 먼저 작성되어 ArithmeticException 예외는 도달할 수 없습니다.
예외 예시	0으로 나누기 : ArithmeticException 배열의 인덱스 번호 예외 : ArrayIndexOutOfBoundsException 존재 하지 않는 파일에 대한 접근 : FileNotFoundException 숫자가 아닌 문자열을 숫자형으로 바꾸고자 하는 경우 : NumberFormatException 부적절한 형 변환(문자 → 숫자, 숫자→문자)
예외 처리(Handling)	예외가 발생하지 않도록 사전에 막음 조치하는 것을 말합니다.
오류(error)	처리가 불가능한 오류 갑작스런 정전, 사용자의 부주의로 인한 리셋 버튼 누름, 시스템 오류 등등
클래스 계층도	Object - Throwable - Error - Exception - 하위Exception들
코딩 방법	변수는 try 외부에서 선언합니다. 초기화는 try 구문에서 처리하도록 합니다.

예외(exception) 처리

예외 처리 개요

예외에는 다음과 같은 종류의 것이 존재합니다.

항목	항목에 대한 설명
ArithmeticException	어떠한 수를 0으로 나누려고 하는 경우에 발생합니다.
NumberFormatException	숫자가 아닌 문자가 입력이 되었을 때 발생합니다.
NullPointerException	널 값을 가지는 어떠한 변수에 대하여 연산이 되는 경우에 발생합니다.
EmptyStackException	스택이 비어 있는 경우에 발생합니다.
FileNotFoundException	존재하지 않는 파일 이름을 제공하는 경우에 발생합니다.
ArrayIndexOutOfBoundsException	배열의 인덱스를 벗어 나는 경우 발생.(배열 인덱스 초과)
RuntimeException	실행 시간에 문제가 있으면 발생합니다.
IOException	입출력에 문제가 있으면 발생합니다.
NoSuchMethodException	메소드를 찾을 수 없을 때 발생합니다..
InterruptedException	인터럽트에 관련된 예외 처리로써 Thread.sleep() 메소드를 사용할 때 발생합니다.
ClassNotFoundException	클래스를 찾을 수 없을 때의 경우에 발생합니다.
InputMismatchException	정수 형태가 아닌 문자열을 정수 형식으로 변경하고자 하는 경우에 경우에 발생합니다.

예외(exception) 처리

객체 지향 구문에서 예외 처리 방법

객체 지향에서는 try ... catch 문으로 예외 처리를 수행합니다.

한 프로그램 내에서 try ... catch 는 여러 번 사용 가능하며, 중첩 또한 가능합니다.

구문	설명
try	예외가 발생할 가능성이 있는 코드 블록을 여기에 작성합니다. 특징 : 오류가 난 문장 이후의 내용은 수행이 되지 않습니다.
catch	해당 예외가 발생하는 경우 처리할 로직을 여기에 코딩합니다. try 구문을 사용하는 경우 catch 절은 최소 1개 이상이 필요합니다.
finally	예외 발생 여부에 상관 없이 무조건 수행되는 코드를 작성하는 영역입니다. 이 구문은 옵션 구문입니다. 사용 예시 : 파일을 열어서 작업 하다가 파일을 닫는 절차 데이터 베이스에 접속 후에 접속 종료 절차 등

사용 형식

```
try {
        //코드 작성

} catch(Exception e) {
        //예외 처리
} finally{
        //무조건 수행되는 영역
}
```

OX 문제

try ... catch 구문에서 catch는 여러 번 사용이 가능합니다.

try ... catch 구문에서 catch는 사용하지 않아도 됩니다.

try ... catch 구문은 중첩해서 사용할 수 있습니다.

예외(exception) 처리

0으로 나누기/배열 첨자 오류

다음 요구 사항대로 프로그램을 구현해 보세요.

항목	항목에 대한 설명
예외 객체	프로그램 수행 중에 예외가 발생하는 경우 해당 예외에 대한 정보를 담고 있는 객체를 말합니다. 예를 들어서 catch (Exception e){ ... } 여기서 e를 예외 객체라고 부릅니다. 객체 이름은 개발자가 마음대로 변경해도 무방하다.(일반적으로 ex 혹은 err을 많이 씁니다.)
주요 메소드	getMessage() : 예외 관련 메시지 내용을 저장하고 있습니다. toString() : 예외 객체가 가지고 있는 문자열을 출력해줍니다. printStackTrace() 스택 내용을 추적하여 프린트해줍니다. 예외 발생의 근원지를 찾아서 단계별 추적하여 보고자 할 때 사용합니다.

예외(exception) 처리

사용자 정의 예외 만들기

사용자 정의 예외를 구현하려면, throw 키워드를 사용하면 됩니다.

항목	항목에 대한 설명
용도	프로그래머가 의도적으로 예외를 발생시키고자 하는 경우에 사용합니다. 사용자 정의 예외 만든 후 이 예외를 사용자가 발생시킬 때 사용합니다.
기타	´throw´라는 단어는 ´문제를 야기 시키다.´의 의미로 해석 하면 좋을 듯 합니다.
사용 형식	throw 예외객체 또는 throw new 예외클래스(매개변수); 이때 해당 예외 클래스의 생성자가 호출됩니다. 그 다음 try … catch 절의 해당 catch 절로 이동합니다.
사용 예시	throw new EmptyStackException ; throw new Exception("사용자가 예외 발생시킴"); throw new NumberFormatException("숫자 변환 예외 발생"); throw new LessThan5("5보다 작은 수 입력은 불가능합니다.");
사용자 정의 예외 작성법	1) Exception을 상속 받는 사용자 정의 클래스를 구현합니다. 2) 구현하고자 하는데 필요한 변수/메소드/생성자/오버라이딩 등을 구현합니다. 3) 원하는 위치에 throw 구문을 이용하여 의도적으로 예외를 발생시킵니다. 4) try...catch() 절로 상황에 맞는 예외 처리를 수행해줍니다.
사용자 정의 예외 호출법	예외를 발생시키고자 하는 곳에서 throw new 예외클래스명() ; catch 절에 해당 예외를 명시합니다.

예외(exception) 처리

예외(exception) 처리 확인 사항

예외 처리와 관련하여 다음 물음에 답해 보세요.

확인해 봅시다.

모든 예외 클래스의 최상위 클래스는 ( )입니다.

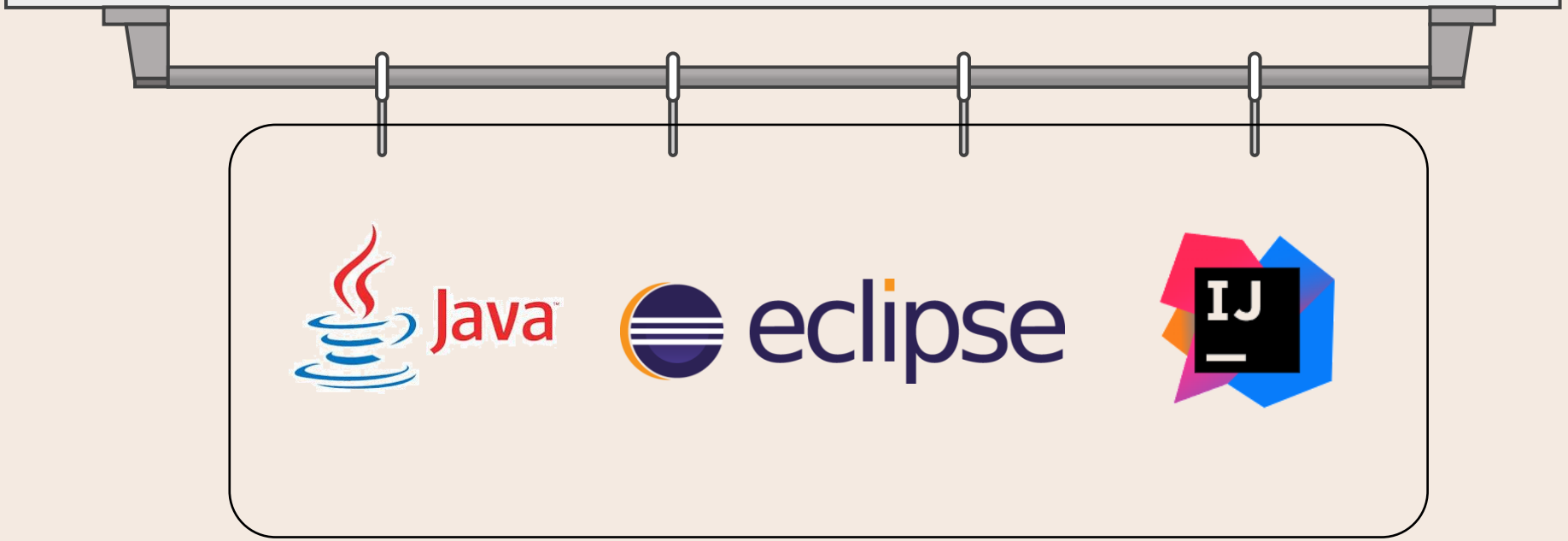
try 블록이 수행될 때, 항상 수행이 되는 블록으로 주로 파일 닫기/네트워크 리소스 등을 해제하기 위한 용도로 사용되는 블록은 ( )입니다.

예외 처리를 직접 처리하지 않고, 상위 스택으로 예외를 떠 넘기고자 할 때 사용하는 키워드는 ( )입니다.

개발자가 직접 예외를 발생시키고자 할 때 사용하는 키워드는 ( )입니다.

자바 Programming

쓰레드(thread)



쓰레드(thread)

[이론] 쓰레드 용어 정리

쓰레드와 관련된 용어를 살펴 보고자 합니다.

항목	항목에 대한 설명
파일	하드 디스크에 존재하는 프로그램.
프로세스	메모리에 적재되어 있는(현재 활성 상태의) 프로그램. 자원에 대한 할당/처리해야 할 명령. 프로세서(CPU)에 의하여 지배받는 응용 프로그램.
프로세서	물리적인 기계 장치, CPU를 말함. 1개의 CPU(단일 프로세서) / n개의 CPU(멀티 프로세서)
쓰레드	프로세스의 처리해야 할 명령어의 작업 단위. 다중 쓰레드라고 하는 것은 1개의 프로세스에 의해 수행되는 여러 개의 독립적 처리 단위가 있는 쓰레드를 말함.(어떤 명령어의 제어 흐름.) 시작점 → 진행 → 종료점을 가지는 일련된 하나의 작업 흐름
멀티스레딩의 장점	하나의 프로그램이 동시에 여러 작업을 수행하는 것이 가능 쓰레드간에 데이터의 공유가 가능하므로 시스템을 효율적으로 사용가능 인터넷 프로그램과 같이 여러 개의 태스크가 동시에 발생하는 작업을 처리하는데 유용합니다.
멀티 태스킹	하나의 CPU에서 동시에 여러 개의 프로그램을 사용하는 것.
멀티 프로세싱	한 개의 컴퓨터(n 개의 CPU)에 여러 개의 프로세서가 있는 것을 말함.

쓰레드(thread)

[이론] 쓰레드 용어 정리

쓰레드와 관련된 용어를 살펴 보고자 합니다.

항목	항목에 대한 설명
프로세스 스케줄링	프로세스간의 CPU의 점유 작업. 시간을 잘게 나누어서 프로세스들이 돌아가면서 CPU를 사용.(time sharing)
동기화	하나의 자원에 대하여 특정 시점에 오직 1개의 쓰레드가 접근 가능하도록 하는 것.
임계 영역	동시에 여러 개의 쓰레드가 접근 할 수 없는 영역.

스케줄링(Scheduling)

멀티 프로세스 시스템에서 각 프로세스는 동시에 실행 되는 것처럼 보이지만, CPU가 하나이기 때문에 실행 시간을 잘게 나누어 프로세스가 돌아가는데 각 프로세스들이 돌아가면서 CPU를 점유하는데 프로세스간 CPU 점유 작업을 말함.

쓰레드(thread)

Thread 스케줄링

쓰레드는 고정된 순위 스케줄링을 수행합니다.

즉, Priority의 숫자 값이 큰 값부터 CPU가 일정한 시간 만큼 부여해주는 방식을 취합니다.

priority는 1부터 10사이의 정수 값을 가지면 값이 클수록 우선 순위가 높습니다.

priority의 값은 getPriority(), setPriority() 메소드를 사용하여 읽거나 쓸 수 있습니다.

항목	설명
MIN_PRIORITY	최소 우선 순위(1)
NORM_PRIORITY	보통 우선 순위(5)
MAX_PRIORITY	최대 우선 순위(10)

쓰레드(thread)

쓰레드(thread)

main 메소드는 single thread입니다.

항목	설명
용어	프로그램을 실행하는 주체(논리적인 최소 단위)
특징	main 메소드도 쓰레드 중의 하나입니다. java.lang.Thread 클래스를 사용합니다. 쓰레드를 실행시키는 주체는 CPU입니다. 개발자는 start() 메소드로 cpu에게 알려 주고, cpu는 적정 시점에 run() 메소드를 찾아서 실행합니다.
싱글 쓰레드	쓰레드 1개로 운영되는 프로그램을 의미합니다. 지금까지 코딩해온 프로그램은 싱글 쓰레드입니다.
멀티 쓰레드	2개 이상의 쓰레드가 가동되는 프로그램을 의미합니다.

문제 풀기

쓰레드에서 다음 메소드의 역할을 간단히 작성해보세요.
run() :
쓰레드가 수행할 내용을 명시합니다.(하고자 하는 업무)
나중에 cpu가 적정 시점에 수행합니다.
start() :
cpu에게 쓰레드가 준비 되었음을 알리는 역할

빈 칸 채우기

자바에서는 쓰레드를 만드는 방법이 (2) 가지가 있습니다.
하나는 (Thread 클래스)를 상속 받는 것이고,
또 다른 하나는 (Runnable 인터페이스)를 상속 받는 방법이 있습니다.

쓰레드(thread)

쓰레드 클래스의 중요 생성자

Thread 클래스는 다음과 같은 생성자를 가지고 있습니다.

생성자	설명
Thread()	기본 생성자.
Thread(String name)	name은 쓰레드를 구별할 수 있는 이름.
Thread(Runnable runnable)	Runnable 인터페이스를 사용한 클래스 객체 생성합니다.
Thread(Runnable runnable, String name)	Runnable 인터페이스와 쓰레드 이름을 지정하여 생성합니다.
Thread(ThreadGroup g, String name)	쓰레드 그룹을 설정.

쓰레드(thread)

쓰레드 클래스의 중요 메소드

Thread 클래스는 다음과 같은 메소드를 가지고 있습니다.

메소드	설명
activeCount()	현재 수행되고 있는 쓰레드의 개수를 반환합니다.
currentThread()	현재 동작중인 쓰레드의 값을 객체로 리턴합니다.
setDaemon(boolean)	false 값이 기본 값으로 독립 쓰레드가 됩니다. 독립 쓰레드란 main 쓰레드가 종료 되더라도 자신이 맡은 업무를 수행하는 쓰레드를 의미합니다. true로 셋팅하면 종속 쓰레드가 되고, main 쓰레드 종료 되는 경우 같이 종료됩니다.
enumerate()	현재 동작중인 쓰레드를 배열의 형태로 리턴합니다.
getId()/getName()	쓰레드의 아이디와 쓰레드 이름을 각각 반환합니다.
join()	지정된 시간만큼 독립적으로 CPU를 실행하도록 하세요.
run()	쓰레드가 실행되는 부분을 기술하는 데 상속 받은 클래스는 반드시 OverRiding 필요합니다.
getPriority()	쓰레드의 우선 순위 값을 리턴. ↔ setPriority
interrupt()	특정 쓰레드를 인터럽트 시킵니다.
interrupted()	현재 쓰레드를 인터럽트 시킵니다.
sleep(millsecond)	millsecond만큼 쓰레드를 지연시키며, 구현시 반드시 InterruptedException에 대한 예외 처리를 해주어야 합니다.
start()	쓰레드를 시작합니다. run 메소드를 실행시킵니다.
stop()	잘 쓰이지 않습니다. 요즘은 interrupt를 많이 사용합니다.

쓰레드(thread)

쓰레드 작성 절차

Thread는 다음과 같은 순서대로 작성합니다.

구현 순서	설명
1) 사용자 정의 클래스 작성	java.lang.Thread 클래스를 상속 받습니다. class 클래스이름 extends Thread{…}
2) run() 메소드를 오버라이딩합니다.	cpu가 수행할 어떤 업무를 여기에 코딩합니다. 개발자와 cpu간의 업무에 대한 약속 장소입니다.
3) main 메소드에서 객체를 생성합니다.	무조건 실행되는 메소드에서 객체 생성이 필요합니다.
4) start() 메소드를 호출합니다.	사용자는 cpu에게 이 쓰레드의 존재를 알려 줘야합니다. 알려 주는 방법은 start() 메소드를 이용합니다. 그래야, cpu가 인식을 하기 때문입니다.

사용 예시	public static void main(){
	ThreadTest obj = new ThreadTest (); //객체 생성
	obj.start(); // start 매소드의 호출
	}

주의)	자바는 다중 상속이 안 되므로 다른 클래스를 상속 받고 있는 경우 java.lang.Runnable(인터페이스)를 상속 받아야 합니다. 만약 Runnable(인터페이스)을 상속 받는 경우는 다음과 같이 코딩합니다. class 클래스이름 extends 슈퍼클래스 implements Runnable{…} Thread th = new Thread(runnable객체) ; th.start() ;

쓰레드(thread)

동기화/임계 영역

동기화와 임계 영역에 대하여 살펴 보도록 하겠습니다.

공용 통장에서 남편은 인터넷 뱅킹을 아내는 은행을 직접 방문하여 돈을 인출하고자 하는 경우를 생각해 보겠습니다.

은행에서는 남편/아내 모두 통장에서 인출이 가능하도록 withdraw()라는 메소드를 제공하고 있습니다.

withdraw() 메소드를 이용하여 남편과 아내가 동시에 접근하여 인출할 수 없습니다.

항목	설명
메소드 동기화	공유 데이터(통장 잔액)를 동시에 edit 하지 못하게 하는 방법 은행에서 남편과 아내가 동시에 돈을 인출하고자 하는 경우
동기화의 이유	데이터의 무결성 및 신뢰도를 보장 받기 위함입니다.
동기화 방법	synchronized 키워드를 사용합니다.
임계 영역	공유하고자 하는 데이터가 사용하는 영역을 말하는 데 일반적으로 메소드의 형태로 존재합니다. 위의 예시에서는 withdraw() 메소드가 임계 영역이 됩니다.

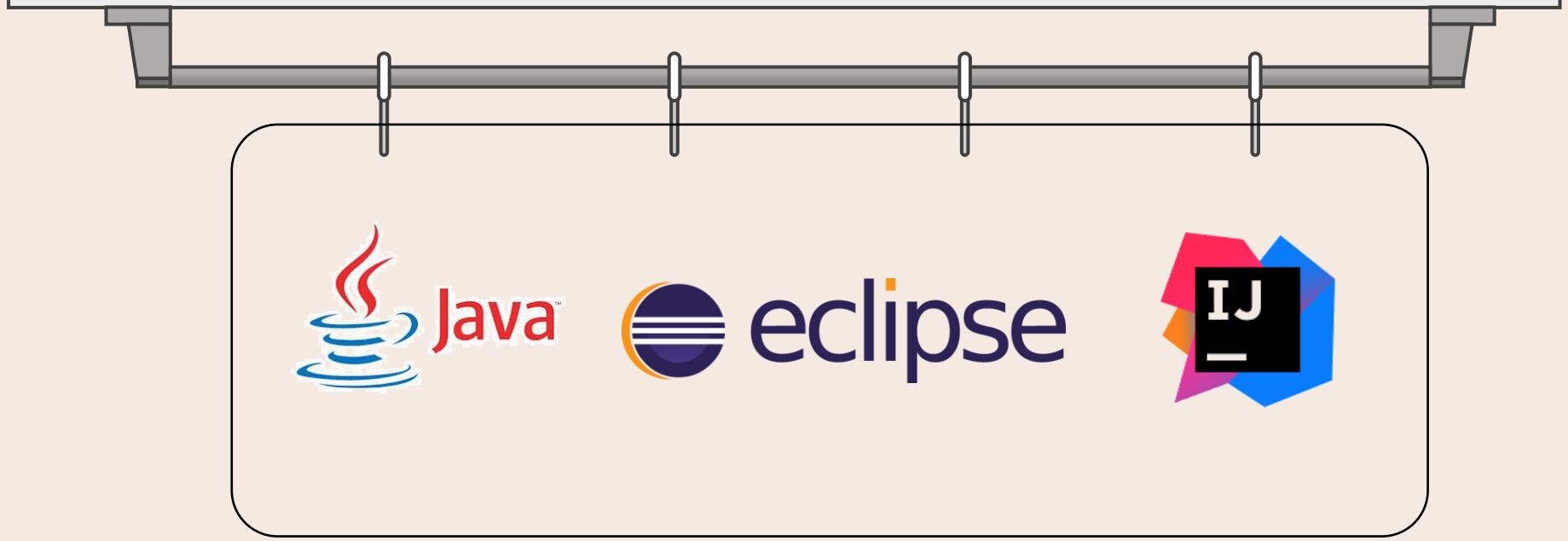
```
public synchronized void withdraw(int money)
    //인출하기 기능 구현
}
… 중략
```

키워드 synchronized를
사용하여 동기화 시키고 있습니다.

이 때 withdraw( )
메소드를 임계 영역이라고
부릅니다.

자바 Programming

컬렉션(collection)



컬렉션(collection)

컬렉션 개요

자바에서 주로 사용되는 컬렉션의 종류에는 다음과 같은 항목들이 있습니다.

컬렉션의 유형

종류	중복 여부	순서 따짐	원소 넣기	원소 제거	원소 찾기	원소 크기	구현되어 있는 클래스	예시
Set	X	X	add()	remove()	contains()	size()	HashSet/TreeSet/ LinkedHashSet	로또, (수학)집합
List	O	O	add()	remove()	get(숫자)	size()	ArrayList /Vector/Stack	기차(량)
Map	키 : X 값 : O	X	put()	remove()	containsKey containsValue get(key)	size()	HashMap/Hashtable /Properties	id=hong name=홍길동 password=1234

데이터 추출 인터페이스

종류	존재 여부 메소드	추출 메소드
Enumeration	boolean hasMoreElements()	nextElement()
Iterator	boolean hasNext()	next()
확장 for 구문	for(타입 낱개원소 : 컬렉션)	-
참고	StringTokenizer 클래스 공부	

배열과 컬렉션 비교

-	배열	컬렉션
데이터 타입	타입이 미리 정해져야 됨	기본 값은 Object 타입
요소 갯수	개수가 정해지면 수정 불가능	무한대의 요소 추가 가능

Stack : LIFO(Last Input First Output)
ArrayList, Vector : FIFO(First Input First Output)

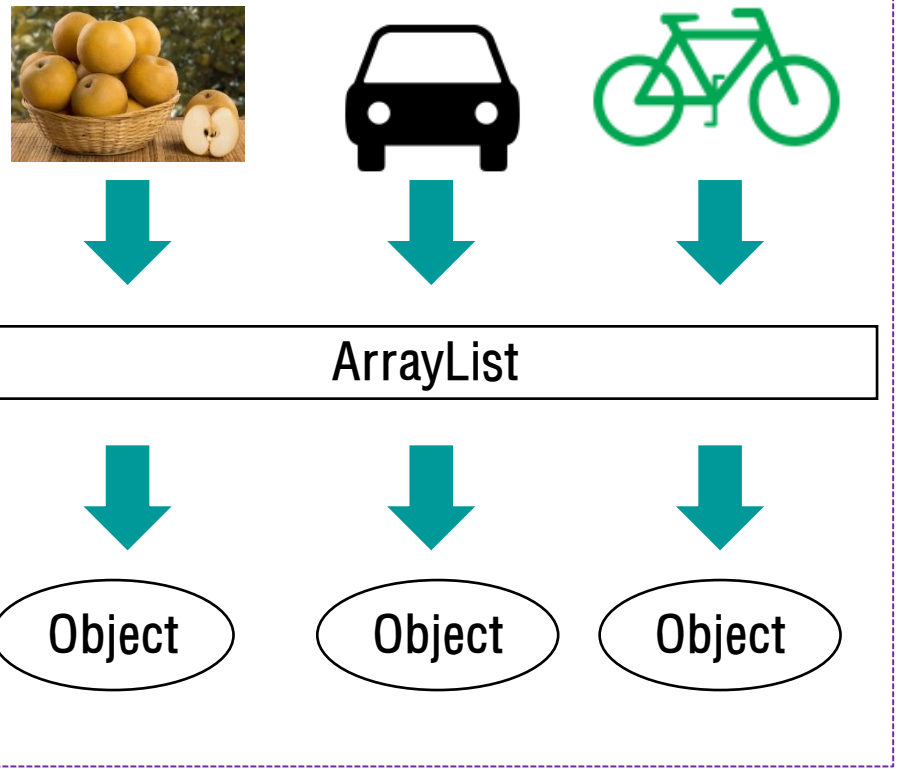
Generic(제너릭)
특정 데이터만 한정시켜 컬렉션에 저장하기 위한 기법

컬렉션(collection)

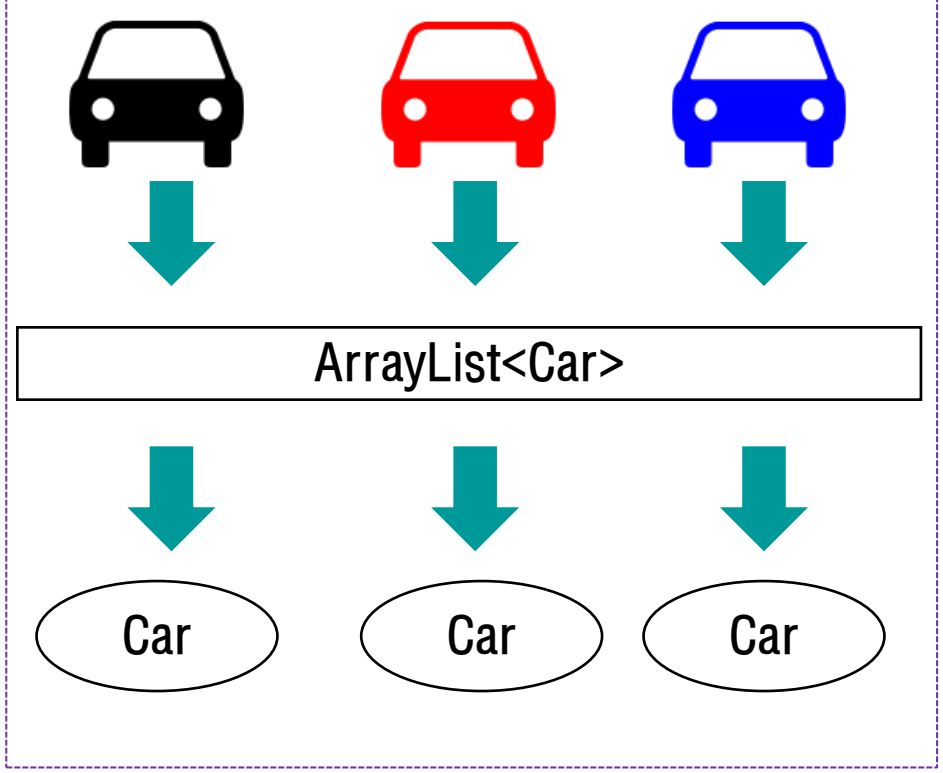
컬렉션 개요

Generic이란 클래스 내의 데이터 타입을 클래스 밖에서 지정하여 타입을 일반화하는 것입니다.

without Generics



with Generics



컬렉션(collection)

Set 계열

HashSet 클래스는 다음과 같은 생성자와 메소드를 가지고 있습니다.

생성자	설명
HashSet()	새로운 HashSet 객체를 생성하고 초기화합니다.
HashSet(collection)	명시된 collection 객체를 사용gk여 HashSet 객체를 생성합니다.

반환형	메소드 명	설명
boolean	add(E o)	제네릭 타입으로 넘어온 객체가 Set 구조에 없다면 추가하고 true를 반환해 줍니다.
boolean	addAll(E o)	합집합에 사용됩니다.
void	clear()	Set 구조에 있는 모든 요소들을 삭제합니다.
boolean	contains(Object o)	인자로 전달된 객체를 현 Collection에서 요소로 가지고 있으면 true를 반환합니다.
	isEmpty()	현 Collection에 저장된 요소가 없다면 true를 반환합니다.
Iterator<E>	iterator()	현 Set 구조의 요소들을 순서대로 처리하기 위해 Iterator 객체로 반환합니다.
boolean	remove(Object o)	현 Set 구조에 인자로 전달된 객체를 삭제합니다. 이때 삭제에 성공하면 true를 반환합니다.
int	size()	현 Set 구조에 저장된 요소의 수를 반환합니다.
boolean	retainAll(E o)	교집합에 사용됩니다.
String	toString()	요소들을 콤마를 이용하여 출력합니다.
Object[]	toArray()	배열로 만들어 줍니다.
boolean	removeAll(E o)	차집합에 사용됩니다.

컬렉션(collection)

Collections 클래스

Collections 클래스는 여러 가지 유용한 알고리즘을 구현해 놓은 메소드를 제공합니다.

이 메소드들은 Generic 기법을 이용하여 static 메소드의 형태로 구현되어 있습니다.

메소드 명	설명
sort(컬렉션)	데이터를 오름차순으로 정렬시켜 줍니다.
sort(컬렉션, Comparator)	Comparator를 이용하면 역순으로 정렬이 가능합니다. Collections.sort(목록, Collection.reverseOrder());
shuffle(컬렉션)	목록을 랜덤하게 섞어 줍니다.
shuffle(컬렉션, Random)	Random 객체를 사용하여 목록을 섞어 줍니다.
binarySearch(컬렉션, key)	목록에서 해당 key를 이용하여 이진 탐색한 결과를 정수 값으로 리턴합니다. 만약 항목이 4번째에서 발견이 되면 숫자 3을 리턴합니다.
min(), max()	목록에서 최대 값을 최소 값을 찾아 줍니다.
reverse()	목록 요소들의 순서를 반대로 합니다.(꺼꾸로 뒤집기)
fill()	지정된 값으로 목록을 채워 줍니다.
copy()	destination 목록과 source 목록을 받아서 복사합니다.(source → destination)
swap(collection, i, j)	목록 collection에서 지정된 위치 i번째와 j 번째의 원소들을 서로 맞바꿔 줍니다.
addAll()	컬렉션 안의 지정된 모든 원소들을 추가합니다.
frequency(collection, finddata)	지정된 collection에서 지정된 원소 finddata가 얼마나 많이 등장하는 지를 정수 값으로 반환합니다.
disjoint()	두 개의 컬렉션이 겹치지 않는 지를 검사합니다.

컬렉션(collection)

풀어 봅시다.

다음 물음에 대한 답을 작성해 보세요.

문제01

동일한 데이터에 대하여 indexOf() 메소드와 lastIndexOf() 메소드의 반환 값이 동일하다는 것은 (빈칸)을 의미합니다.

문제02

다음은 mylist 리스트 요소 내의 임의의 위치에 신규 요소 "호호호"를 1개 추가하는 방법입니다.

빈칸을 채워 넣으세요.

mylist.add(빈칸, "호호호");

문제03

리스트 내의 요소 중에서 2번째 요소와 5번째 요소를 서로 맞바꾸려면 Collections.빈칸 메소드를 사용하면 됩니다.

빈칸에 들어갈 코드는 ?

컬렉션(collection)

Map 계열

Map은 키(Key)와 값(Value)으로 구성된 데이터 구조입니다.

HashTable : (키, 값) 으로 구성된 하나의 쌍 구조

key와 value를 매핑하는 객체

key는 중복이 불가능하고, 1개의 value만 매핑합니다.

Enumeration 인터페이스

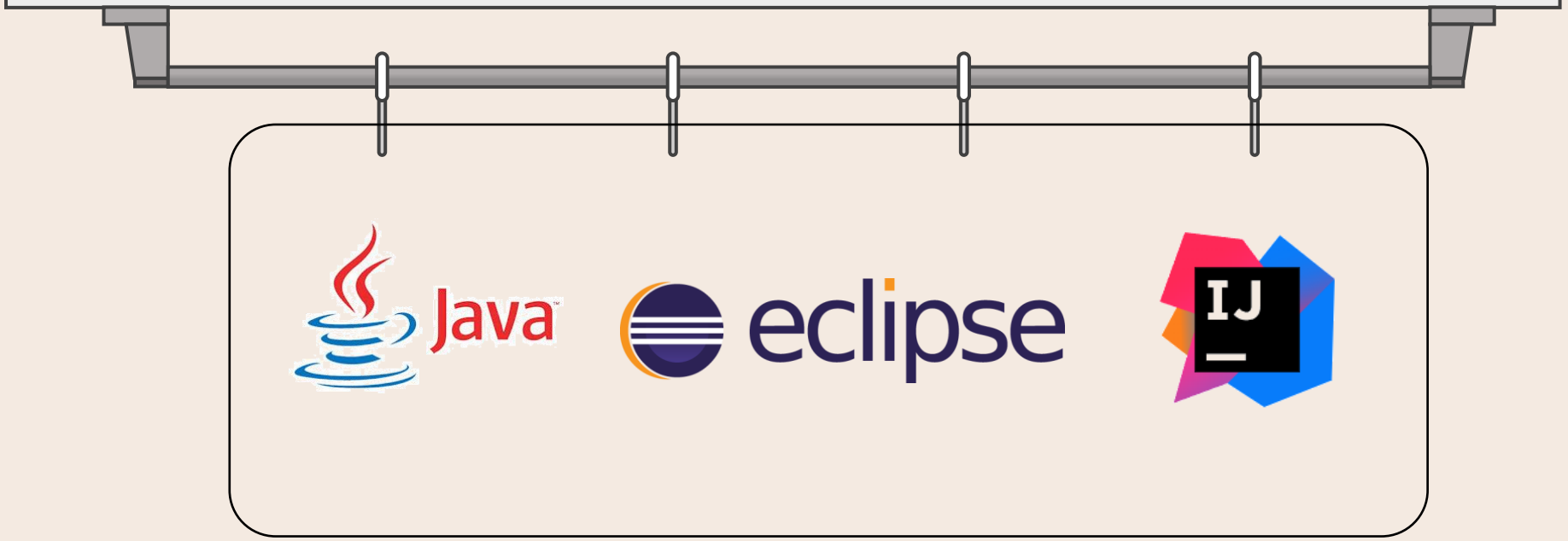
hasMoreElements() : 다음 항목이 존재하는가 ?

nextElement() : 다음 항목을 추출합니다.

메소드 명	설명
containsKey(키)	컬렉션에 키가 존재하는가?
containsValue(값)	컬렉션에 값이 존재하는가?
get(키)	키를 토대로 get 메소드를 이용하여 값을 얻어 내기
keys	목록들을 enumeration형 배열로 리턴
put(키, 값)	항목을 추가하기

자바 Programming

스트림(stream)

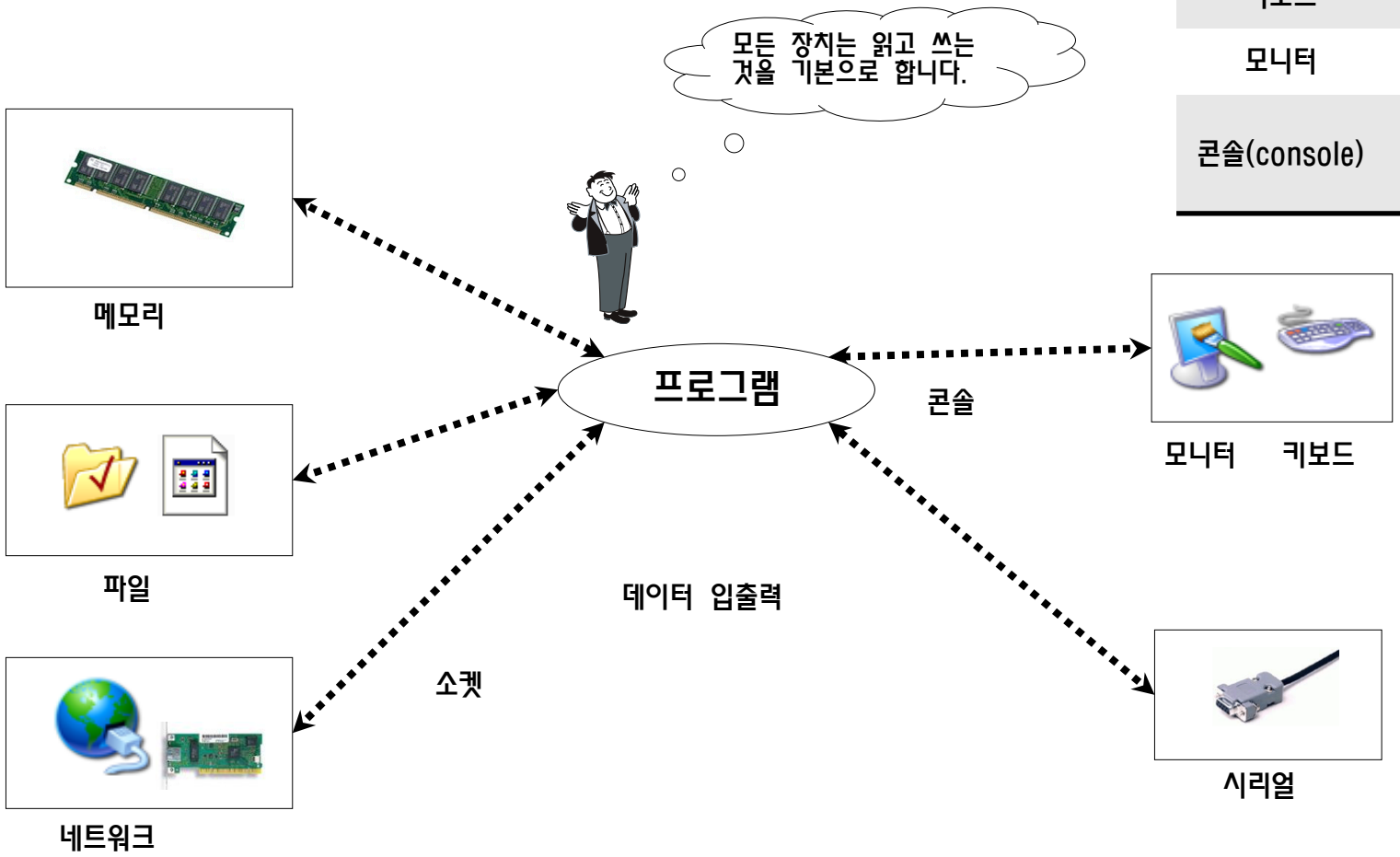


스트림(stream)

스트림(Stream)

stream의 사전적 의미는 물이 흘러 가는 개울을 의미하는데, 프로그래밍에서는 순서가 있는 데이터의 흐름을 의미합니다.

항목	설명
키보드	표준 입력 스트림(System.in)
모니터	표준 출력 스트림(System.out)
콘솔(console)	일반적으로 모니터와 키보드를 합쳐서 부르는 이름입니다.



스트림(stream)

스트림(Stream)

스트림은 서로 다른 장치들 사이의 입출력을 위한 표준 방법을 제시합니다.

항목	설명
정의	사용자와 입출력 장치(Device)를 연결해 주는 가상적인 다리(Bridge) 역할을 합니다. 데이터를 송수신 하기 위한 일종의 가교 역할을 수행합니다.
특징	FIFO 구조(선입 선출 구조)입이다. 읽기, 쓰기가 동시에 진행 되지 않는 단방향 흐름입니다. 즉, 파일 내용을 읽어 오다가 쓰기 작업을 할 수 없습니다. 네트워크로 인한 트래픽 지연 현상이 발생할 수 있습니다.
스트림을 사용하는 이유	서로 다른 장치들 사이의 입출력을 위한 표준 방법을 제시합니다. 데이터가 존재하는 장소가 다릅니다. 장치(Device)에 따라서 데이터를 읽고 쓰는 방법이 다릅니다. Read, Write. 장치에 데이터를 읽고 기록하기가 쉽습니다. 입출력 대행자(장치에 대하여 알 필요 없습니다.) 역할을 합니다.
관련 패키지	import java.io.* ; // io : input/output의 약자(abbr)

항목	표준 스트림	비표준 스트림
생성/소멸	automatic	개발자가 직접 생성/소멸해야 합니다.
소속	System 클래스	거의 대부분이 java.io.* 패키지 내에 존재함
특징	static, final	클래스마다 적절한 예외 처리 구현이 필요함.

스트림(stream)

스트림(Stream)

스트림으로 가능한 업무는 다음과 같은 항목들이 있습니다.

가능한 일

키보드로 입력 받아서 모니터에 출력하기

키보드로 입력 받아서 파일에 기록하기

파일 입력(읽음) → 모니터 출력

파일 입력(읽음) → 파일 출력(쓰기)

특정 폴더 내

    파일 개수 및 디렉토리 개수를 파악할 수 있습니다.

    파일의 속성을 확인(생성 및 수정 일자, 크기 등등)

2개의 파일을 병합하기

인터넷으로 그림 파일을 다운로드하는 경우(byte 스트림을 읽어 들인다.)

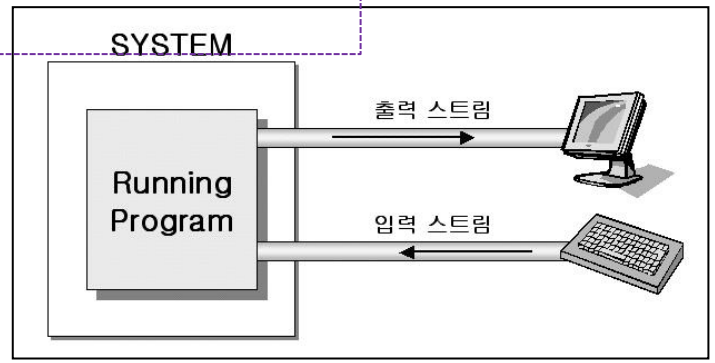
채팅(소켓을 이용한 byte 스트림의 교환)

스트림(stream)

표준 스트림

스트림으로 가능한 업무는 다음과 같은 항목들이 있습니다.

사용자가 입출력을 손쉽게 할 수 있도록 미리 만들어 둔 스트림으로써 크게 in과 out가 있습니다.
표준 스트림은 자동으로 생성이 되고, 자동으로 소멸이 됩니다.
둘 다 System 클래스에 속해 있고, 클래스 이름으로 접근 가능하도록 static 키워드를 사용하고 있고, 편집이 불가능하게 하기 위하여 final 키워드를 사용하고 있습니다.
콘솔(console)이란 [키보드]와 [모니터]를 합친 용어이다.



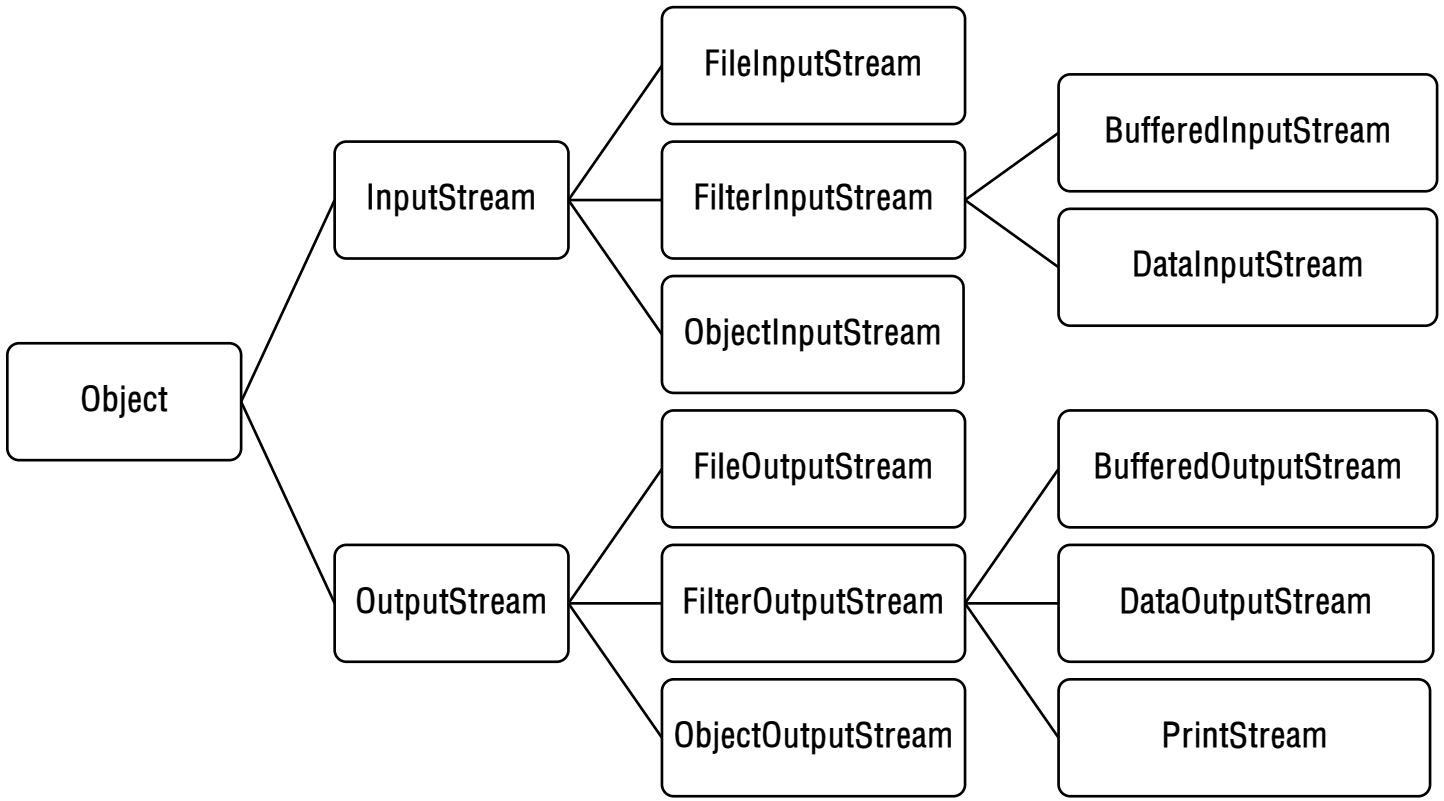
항목	설명
표준 입력 객체 키보드 System.in.read();	표준 입력(키보드) 스트림의 in 객체 버퍼 입력 스트림 객체의 read() 메소드. read() 메소드 : 1바이트씩 읽기, 더 이상 값이 없으면 -1을 리턴. System.in 객체는 InputStream 클래스이기 때문에 다음과 같이 선언이 가능합니다. * InputStream myIn = System.in;
표준 출력 객체 모니터 System.out.print ();	표준 출력(모니터) 스트림의 out 객체. 프린트 스트림 객체의 print() 메소드 System.out 객체는 PrintStream 클래스로 선언이 되었고, PrintStream은 OutputStream 클래스의 하위 클래스이다. * OutputStream myOut = System.out;

스트림(stream)

바이트 스트림

바이트 스트림은 1 byte를 입출력하는 스트림입니다.

일반적으로 바이트로 구성된 파일은 다음과 같습니다.
동영상 파일 / 이미지 파일 / 음악 파일 처리 등등

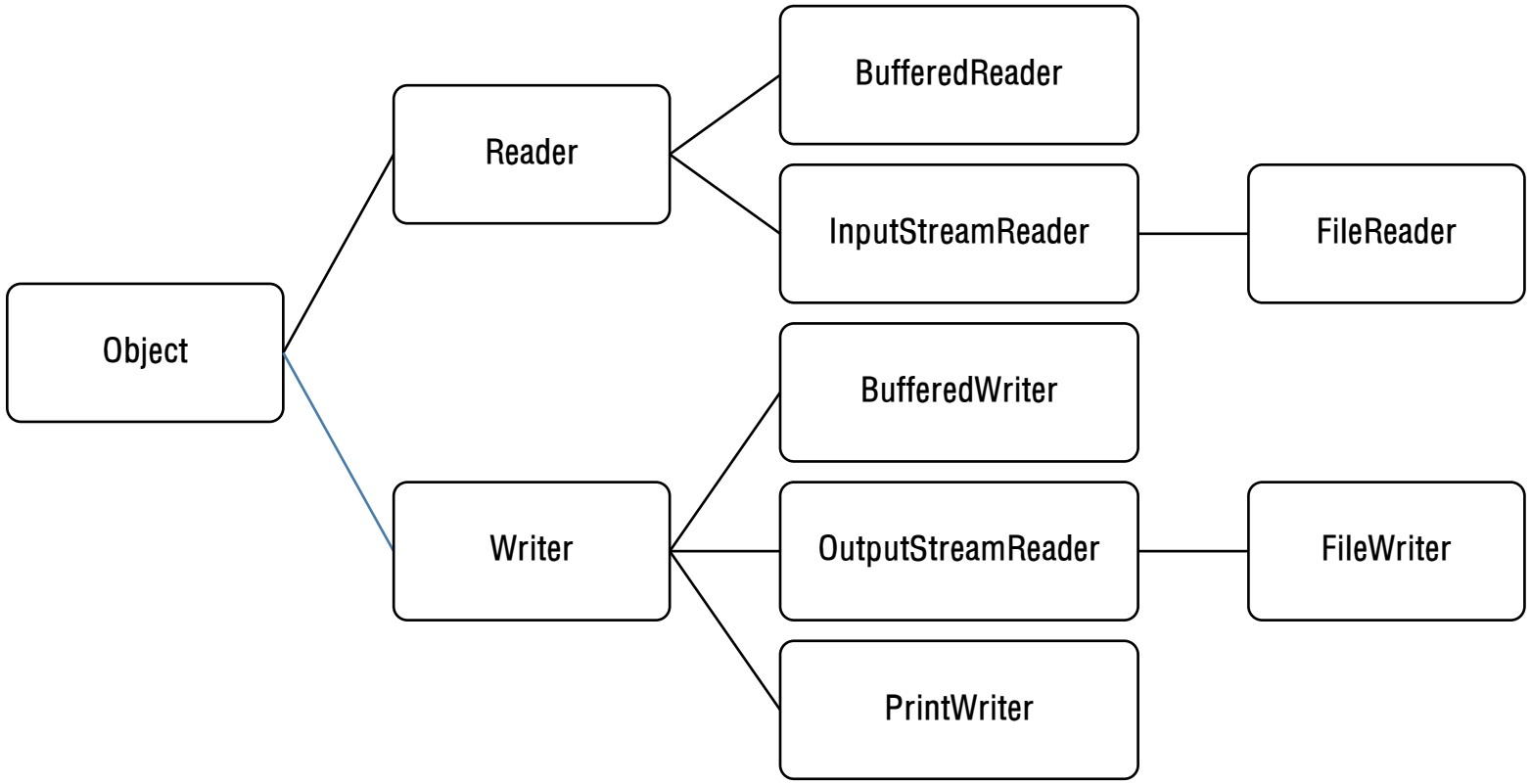


스트림(stream)

문자 스트림

문자열 스트림은 문자열을 입력/출력해주는 스트림입니다.

문자 입력 스트림은 Reader의 하위 클래스로 구현합니다.
문자 출력 스트림은 Writer의 하위 클래스로 구현합니다.



스트림(stream)

File 클래스

파일이나 폴더(디렉토리)를 자바 프로그램의 객체로 만들어서 처리합니다.

운영 체제의 디렉토리/파일을 제어하는 클래스입니다.
파일과 폴더(디렉토리)를 다루기 위한 File 클래스 제공
파일의 삭제, 이름 변경, 존재 확인 등등이 가능합니다.
일반적으로 파일 이름으로 File 클래스 생성이 가능합니다.

항목	설명
File(String pathname)	pathname 디렉토리로 파일 객체를 생성합니다. File file1 = new File("c:\\java\\")
File(String pathname, String filename)	pathname 디렉토리에 존재하는 파일 filename으로 객체를 생성합니다. File file2 = new File("c:\\java\\", "batch.bat")
File(File dir, String filename)	디렉토리 dir에 존재하는 filename 이름의 File 객체를 생성합니다. File file3 = new File(file1, "batch.bat")

스트림(stream)

File 클래스

파일이나 폴더(디렉토리)를 자바 프로그램의 객체로 만들어서 처리합니다.

File 클래스의 주요 메소드는 다음과 같은 항목들이 있습니다.
메소드 이름에 can으로 시작하면 가능성 여부를 나타내고, 메소드 이름에 is으로 시작하면 boolean 타입을 반환해 줍니다.

항목	설명
String getName()	디렉토리 이름을 제외한 파일 이름을 반환.
String getPath()	파일의 상대 경로를 반환.
String getAbsolutePath()	파일의 절대 경로를 반환.
String getCanonicalPath()	파일에 대한 정규 경로를 반환
String getParent()	파일의 상위 경로를 반환.
boolean canRead()	파일이 읽기 가능하면 true, 아니면 false를 반환
boolean canWrite()	파일이 쓰기 가능하면 true, 아니면 false를 반환
boolean delete()	파일을 삭제하고 true를 반환, 파일을 삭제할 수 없으면 false를 반환
boolean equals(Object obj)	현재의 객체와 obj로 지정된 객체가 같은 파일을 가지고 있으면 true, 아니면 false
boolean exists()	파일이 존재하면 true, 아니면 false를 반환
boolean isAbsolute()	경로가 절대 경로이면 true, 아니면 false를 반환
boolean isDirectory()	현재의 객체가 디렉토리이면 true, 아니면 false를 반환
boolean isFile()	현재의 객체가 파일이면 true, 아니면 false를 반환
long lastModified()	1970/1/1(GMT)부터 파일이 마지막으로 수정된 날자까지의 시간을 밀리 초로 반환

스트림(stream)

File 클래스

파일이나 폴더(디렉토리)를 자바 프로그램의 객체로 만들어서 처리합니다.

항목	설명
long length()	파일의 바이트 수를 반환
String[] list()	디렉토리에서 파일의 이름을 반환
boolean mkdir()	디렉토리를 생성. 경로로 지정된 모든 부모 디렉토리가 존재하여야 합니다. 지정한 디렉토리가 생성되면 true를 반환하고, 아니면 false를 반환
boolean mkdirs()	디렉토리를 생성. 경로로 지정된 디렉토리가 존재하지 않으면 생성한 다음 지정한 디렉토리를 생성. 디렉토리가 생성되면 true를 아니면 false를 반환
boolean renameTo(File newName)	파일이나 디렉토리의 이름을 newName으로 변경한 다음 true를 반환. 이름을 변경하지 못하면 false를 반환
File[] listFiles()	해당 폴더/파일을 File 타입의 배열로 리턴합니다.
boolean createNewFile()	새로운 파일을 생성합니다.