JAVA 기본 프로그래밍

자바의 기초 학습하기

# 0. 학습 목표

1. 자바의 데이터 타입 이해
2. 자바의 입출력 이해
3. 자바의 연산자 이해
4. 자바의 배열 선언 및 활용
5. 예외 개념과 자바에서의 예외 처리

# 1. 자바의 데이터 타입 이해

### 자바의 데이터 타입 기초

* **식별자** : 클래스, 변수, 상수, 메소드 등에 붙이는 이름. 대소문자 구별!
* **기본 타입의 크기는 CPU나 운영체제에 따라 변하지 않는다!**
* **변수** : 프로그램 실행 중에 값을 임시 저장하기 위한 공간.

데이터 타입에서 정한 크기 만큼, 변수를 선언할 때 메모리를 할당한다.

### 변수의 종류

* **멤버 변수(필드)** : 클래스 안에서 선언된 변수로, 인스턴스들이 개별적으로 가진다.

초기값을 지정하지 않으면 자동으로 기본값으로 초기화된다.

* **정적(static) 멤버 변수** : 모든 인스턴스들이 공통으로 이용 가능한 클래스 변수.

클래스가 메모리에 로딩될 때 초기화된다.

* **지역 변수** : 메소드 안에 선언된 변수로, **사용 전에 꼭 초기화를 해줘야 한다.**
* **레퍼런스 변수** : 클래스의 변수, 배열 변수

클래스의 인스턴스 위치를 저장한다.

메모리상의 클래스 객체나 배열의 위치를 저장한다. **(값이 아니라 위치!)**

### 리터럴

* 리터럴 : 프로그램에서 직접 표현한 값.
* 정수 리터럴 : 10진수, 8진수, 16진수, 2진수 등등… int 형으로 컴파일한다.
* 실수 리터럴 : 소수점 형태가 지수 형태로 표현한 실수. double 형으로 컴파일.
* 문자 리터럴 : 단일 인용부호 ('')로 문자 표현.
* null 리터럴 : 레퍼런스에 대입해 사용한다. 기본 타입엔 사용X!

+) 상수 선언 : final 키워드를 사용하면 초기값을 지정할 수 있고, 변경 불가능.

# 2. 자바의 입출력 이해

### 출력

* System.out.println()
* System.out.printf(“서식", 출력 대상) : **쓰려면 컴파일 버전을 1.4 넘게 해줘야 함.**

### 입력

* System.in : 키보드와 연결된 자바의 표준 입력 스트림.

입력되는 키를 바이트(문자X)로 리턴하는 저수준 스트림이다.

이걸 직접 사용하면 바이트를 문자나 숫자로 변환하는 데 많이 어렵다고 함.

* Scanner 클래스 : 읽은 바이트를 문자, 정수, 실수 등 다양한 타입으로 변환해 리턴!

> next(), nextInt(), nextDouble(), nextBoolean() 등...

# 

# 

# 3. 자바의 연산자 이용

### 비트 연산자와 시프트 연산자

* **비트 연산자** : 피 연산자의 각 비트들을 대상으로 하는 연산

|  |  |
| --- | --- |
| **비트 연산자** | **내용** |
| a & b | AND, 두 비트 모두 1일 때만 1 리턴 |
| a | b | OR, 두 비트 모두 0일 때만 0 리턴 |
| a ^ b | XOR, 두 비트가 서로 다르면 1, 같으면 0 리턴 |
| ~ a | NOT, 단항 연산자로 1을 0, 0을 1로 변환 |

* **시프트 연산자**

|  |  |
| --- | --- |
| **시프트 연산자** | **내용** |
| a >> b | 산술적 오른쪽 시프트.  a의 각 비트를 오른쪽으로 b번 시프트한다.  최상의 비트의 빈자리는 시프트 전의 최상위 비트로 다시 채운다. |
| a >>> b | 논리적 오른쪽 시프트.  a의 각 비트를 오른쪽으로 b번 시프트한다.  최상위 비트의 빈자리는 0으로 채운다. |
| a << b | 산술적 왼쪽 시프트.  a의 각 비트를 왼쪽으로 b번 시프트한다.  최하위 비트의 빈자리는 0으로 채운다. |

### 증감 연산자

* **a++** : a를 먼저 사용한 후에 a를 1만큼 증가시킨다.
* **++a** : a를 먼저 1만큼 증가시킨 다음에 a를 사용한다.

# 

# 

# 4. 자바의 배열 선언 및 활용

### 배열이란?

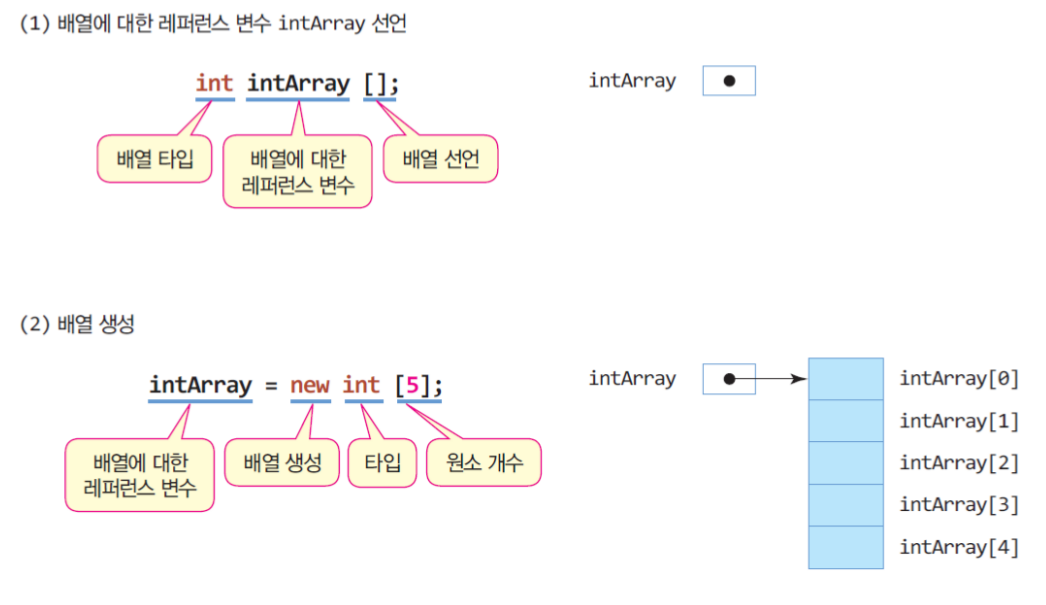
* 인덱스와 인덱스에 대응하는 데이터들로 이루어진 자료 구조
* 배열은 같은 타입의 데이터들이 순차적으로 저장되는 공간이다!
* 자바의 배열은 객체로 처리된다. 배열 객체의 필드들을 가져다 쓸 수 있다.

### 배열의 선언 및 생성

* 배열을 쓰기 위해서는 **배열 선언과 배열 생성의 두 단계가 필요하다.**

1. 선언 : int array[];
2. 생성 : array = new int[5];

+) 선언과 동시에 생성 : int array = new int[5]

* 생성과 동시에 초기화할 수도 있다.
* **반!드!시! 배열을 생성한 후에 접근해야 한다.**

### 2차원 배열 선언 및 생성

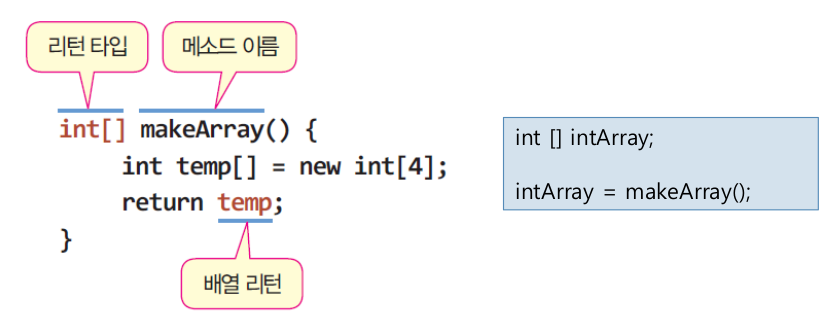
1. 선언 : int array[][];
2. 생성 : array = new int[2][5];

+) 선언과 동시에 생성 : int array[][] = new int[2][5];

### 메소드의 배열 리턴

* 배열 리턴 : 배열의 레퍼런스만 리턴한다. 배열 자체를 리턴하는 게 아니다!
* 메소드의 리턴 타입 : 리턴하는 배열 타입과 리턴 받는 배열 타입을 일치시키되,

리턴 타입에 배열의 크기를 지정하지는 않는다.



# 5. 자바의 예외 처리

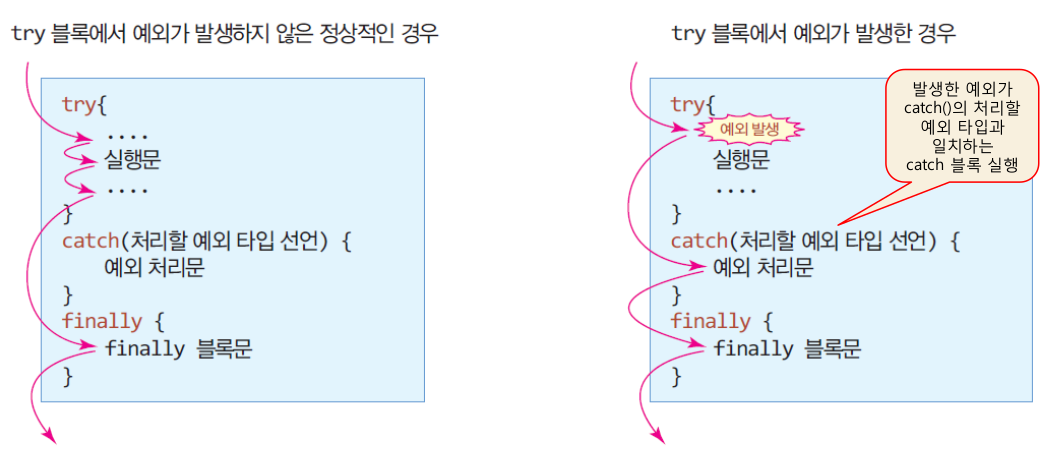
### 예외란?

* **예외** : 실행 중 오동작이나 결과에 악역향을 미치는 상황이 발생한 경우.
* 실행 중 예외가 발생하면 > 자바에서는 응용프로그램이 예외를 처리하도록 호출한다.

> 응용프로그램이 예외를 처리하지 않으면 프로그램을 강제로 종료 시킨다.

### 자바에서의 예외 처리 (try-catch-finaly)

* 발생한 예외에 대해 개발자가 작성한 프로그램 코드에서 대응하는 것.



### 자바의 예외 클래스틀

