

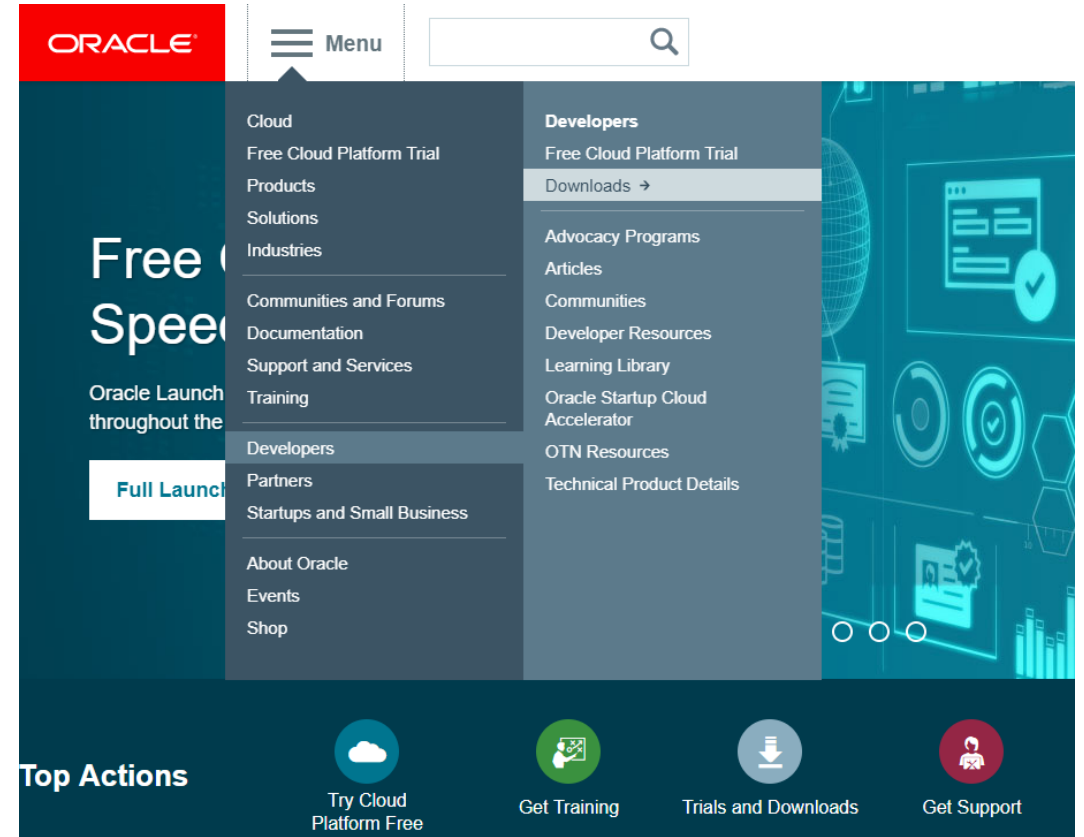
JAVA

0.설치

download

www.oracle.com

Developers -> Downloads
-> Java -> Java(JDK) for Developers



0.설치

download

JVM (Java Virtual Machine)

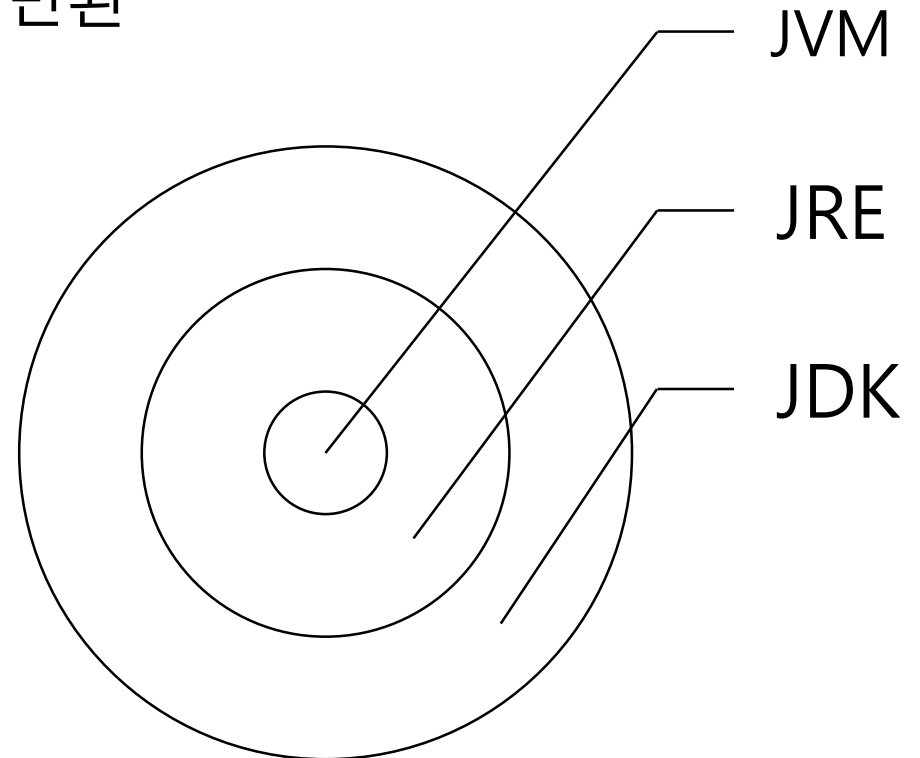
- java 실행 프로그램. os에 맞게 java code 변환

JRE(Java Runtime Environment)

- 자바 실행 환경.

JDK(Jva Development Kit)

- JRE + 개발도구



install

1 : package

2 : class

3 : member

1

```
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.annotation
java.lang.instrument
java.lang.invoke
java.lang.management
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio
java.nio.channels
java.nio.channels.spi
java.nio.charset
java.nio.charset.spi
java.nio.file
Thread.UncaughtExceptionHandler
```

Classes

```
Boolean
Byte
Character
Character.Subset
Character.UnicodeBlock
Class
ClassLoader
ClassValue
Compiler
Double
Enum
Float
InheritableThreadLocal
Integer
Long
Math
Number
Object
Package
Process
StringBuilder
StringBuilder.Redirect
Runtime
RuntimePermission
SecurityManager
Short
StackTraceElement
StrictMath
```

Allocates a new string that contains the sequence of characters currently contained in the string builder argument.

Method Summary

| All Methods | Static Methods | Instance Methods | Concrete Methods | Deprecated Methods |
|--------------------------|----------------|-------------------------------|--|---|
| Modifier and Type | | Method and Description | | |
| | char | | charAt(int index) | Returns the char value at the specified index. |
| | int | | codePointAt(int index) | Returns the character (Unicode code point) at the specified index. |
| | int | | codePointBefore(int index) | Returns the character (Unicode code point) before the specified index. |
| | int | | codePointCount(int beginIndex, int endIndex) | Returns the number of Unicode code points in the specified text range of this String. |
| | int | | compareTo(String anotherString) | Compares two strings lexicographically. |
| | int | | compareToIgnoreCase(String str) | Compares two strings lexicographically, ignoring case differences. |
| | String | | concat(String str) | Concatenates the specified string to the end of this string. |
| | boolean | | contains(CharSequence s) | Returns true if and only if this string contains the specified sequence of char values. |
| | boolean | | contentEquals(CharSequence cs) | Compares this string to the specified CharSequence. |
| | boolean | | contentEquals(StringBuffer sb) | Compares this string to the specified StringBuffer. |
| | static String | | copyValueOf(char[] data) | Equivalent to valueOf(char[]) . |
| | static String | | copyValueOf(char[] data, int offset, int count) | |

3

2

0.설치

install

1) 내 컴퓨터 -> 속성 -> 고급 시스템 설정
-> 환경변수 -> 시스템 변수 -> 새로 만들기

변수 이름 : JAVA_HOME

변수 값: JDK 설치 경로 폴더 (C:\ProgramFiles\Java\jdk1.8.0_131)

0.설치

install

2) Path(시스템 변수) -> 편집

-> 새로 만들기 -> %JAVA_HOME%\bin\ 추가

3) cmd(명령 프롬프트) -> java

cmd -> java -version

cmd -> javac

0.설치

compiler

cmd -> cd.. -> cd test -> javac MyTest.java -> java MyTest

C:\> 명령 프롬프트

```
C:\test>javac MyTest.java
```

```
C:\test>java MyTest  
Hello, World!
```

```
C:\test>
```

이름

MyTest.class
MyTest

유형

CLASS 파일
JAVA 파일

0.설치

compiler

java의 complier : jre -> javac.exe -> java.exe

- 컴퓨터가 알아들을 수 있는 언어로 한 번에 변환하여 실행

*interpreter : 한 줄씩 변환 및 실행

0.설치

eclipse ide

<http://www.eclipse.org/>

Eclipse IDE (Integrated Development Environment)

- 통합 개발 환경
- java 뿐 만 아니라 다른 언어들도 추가하여 사용할 수 있다.
- java 개발 시 가장 많이 사용

0.설치

class, member, ...

class : 설계도

instance : 실제 물건

field : 속성

member : class가 가진 것

method : 기능

constructor : 생성자

```
public class UserClass {
```

```
    // field (member variable, 전역변수)
```

```
    String s01 = "Hello";           // instance variable
```

```
    static String s02 = "World";    // class variable
```

```
    // method
```

```
    public void prn() {
```

```
        // 지역변수
```

```
        String s03 = s01 + ", " + s02;
```

```
    }
```

```
    public static void main(String[] args) { //parameter(argument)
```

```
    }
```

```
    // constructor
```

```
    public UserClass() {
```

```
    }
```

```
}
```

1.Type

변수

- 변수 선언 방법

`type 변수 = 값;`

- 메모리에 값을 할당하는 것.
- 값 자체를 literal 이라고 한다.
- java의 type은 기본타입과 참조타입으로 나뉜다.

* 기본타입 : call by value, immutable (변경 불가)

참조타입 : call by reference, mutable (변경 가능)

1.Type

기본타입

* 1 byte = 8 bit

| 정수형 | | | | 실수형 | | 문자형 | 논리형 |
|-------|-------|-------|-------|-------|--------|-------|---------|
| byte | short | int | long | float | double | char | boolean |
| 1byte | 2byte | 4byte | 8byte | 4byte | 8byte | 2byte | 1byte |

1.Type

참조타입

object 형이라고도 한다.

메모리에 할당된 주소를 참조한 값을 사용한다.

*모든 클래스는 object를 상속받는다.

Class Class<T>

```
java.lang.Object  
    java.lang.Class<T>
```

Type Parameters:

T - the type of the class modeled by this Class object. For example, the type of String.class is Class<String>. Use Class<?> if the class being modeled is unknown.

All Implemented Interfaces:

Serializable, AnnotatedElement, GenericDeclaration, Type

1.Type

System.out.print

print("") : console에 출력하는 기본 명령

println("") : 출력 후 줄바꿈(enter)

printf("% %",arg1, arg2..) : args 등을 받아서 출력

*arg type - %s : String - %f : float

 - %d : digit - %n : new line (\r\n)

*escape 문자 - \n : 줄바꿈 - \t : tab(가로)

 - \' : 작은 따옴표 - \" : 큰 따옴표

 - \r : 줄의 가장 앞자리로

- \\ : \를 표현

- \b : backspace

2.Method

method란

- 어떤 작업을 수행하기 위한 명령문들의 집합
- 하나의 이름으로 여러 개의 명령을 일괄처리 하는 작은 모듈
- function

- 선언 방법

```
접근제한자 메모리영역 리턴타입 메소드명(파라미터){  
    명령문;  
}
```

2.Method

접근제한자

멤버필드나 메소드의 접근을 제어

| 접근제한자 | 특징 |
|--------------|---|
| public(+) | 어디서나 접근/참조 가능 |
| protected(#) | 상속일 경우, 상속된 어디서나 접근/참조 가능 상속이 아닐 경우 같은 패키지 내에서만 접근/참조 가능 |
| (default) | 같은 패키지 내에서만 접근/참조 가능 |
| private(-) | 해당 클래스 내에서만 접근/참조 가능 |

2.Method

메모리영역

| static | non-static |
|--|---|
| application이 실행되면 모두 메모리에 할당되고, 종료되면 삭제된다. | 클래스의 인스턴스를 생성할 때 만들어진다. |
| 객체를 생성하지 않고 클래스명.메서드명 으로 호출 ex) Test01.myTest(); | 객체를 생성하여 객체명.메서드명 으로 호출 Test01 t = new Test01(); t.myTest(); |

2.Method

return type

method를 호출하면 반환해주는 값의 type
(반환값이 없을 땐 void)

ex)

```
public static String myPhone(){  
    String phoneNum = "010-1234-5678";  
    return phoneNum;  
}
```

2.Method

param/arg

parameter와 argument의 차이점

```
public static void test(int x){  
    int res = x+20;  
}
```

...

```
checkValue = test(10);
```

//parameter : method의 내부에서 사용하기 위함

//argument : method에 값을 전달하기 위함

3.Operator

사칙/대입연산자

사칙연산 : +, -, *, /(몫), %(나머지)

대입연산

$a += 10;$ \rightarrow $a = a + 10;$

$a -= 10;$ \rightarrow $a = a - 10;$

$a *= 10;$ \rightarrow $a = a * 10;$

$a /= 10;$ \rightarrow $a = a / 10;$

$a \% = 10;$ \rightarrow $a = a \% 10;$

3.Operator

증감연산자

`i++` : 선 처리 후 증가

`++i` : 선 증가 후 처리

`j--` : 선 처리 후 감소

`--j` : 선 감소 후 처리

3.Operator

논리연산자

| 연산자 | 뜻 | 설명 |
|-----|-----------|-------------------|
| ! | 논리부정(not) | 논리식의 진위를 반대로 만든다. |
| & | 논리곱(and) | 두 논리식이 모두 참이어야 참 |
| | 논리합(or) | 두 논리식 중 하나만 참이면 참 |

<, >, <=, >=, ==, != 등의 식도 참(true)/거짓(false)으로 반환

&&, || (short circuit)

- 두 논리식 중 앞의 논리식이 조건에 맞을 때 뒤의 논리식을 확인한다.

3.Operator

삼항연산자

type 변수명 = (조건) ? A : B

- 조건이 true면 A 반환, false면 B 반환
- A와 B를 반환 받을 변수의 type은 같아야 한다.
- 반환 받을 변수는 꼭 있어야 한다.

ex)

```
int i = (a>b)?1:0;
```

3.Operator

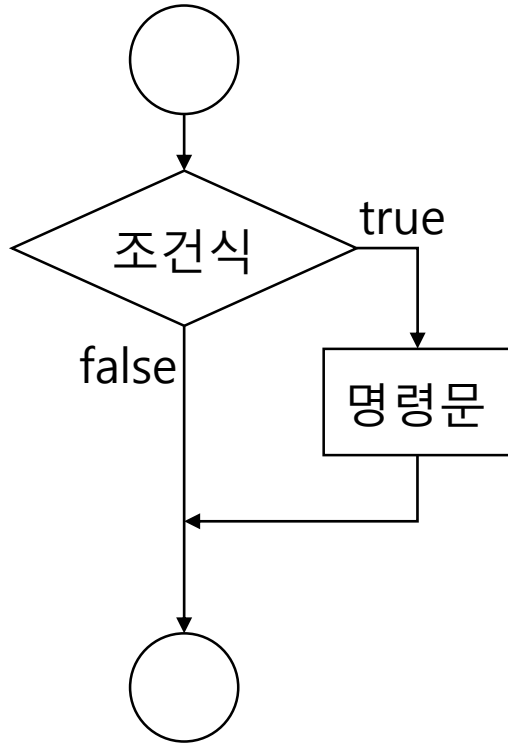
연산자 우선순위

| 우선순위 | 연산자 | 종류 |
|------|-------|-----------------------------------|
| 1 | 단항연산자 | [] ++ -- + - ~ ! new(type) |
| 2 | 산술연산자 | * / % |
| 3 | | + - |
| 4 | | << >> |
| 5 | 비교연산자 | < <= > >= instanceof |
| 6 | | == != |
| 7 | 논리연산자 | & |
| 8 | | ^ |
| 9 | | |
| 10 | | && |
| 11 | | |
| 12 | 삼항연산자 | ? : |
| 13 | 대입연산자 | = *= /= %= += -= <<= >>= &= ^= = |

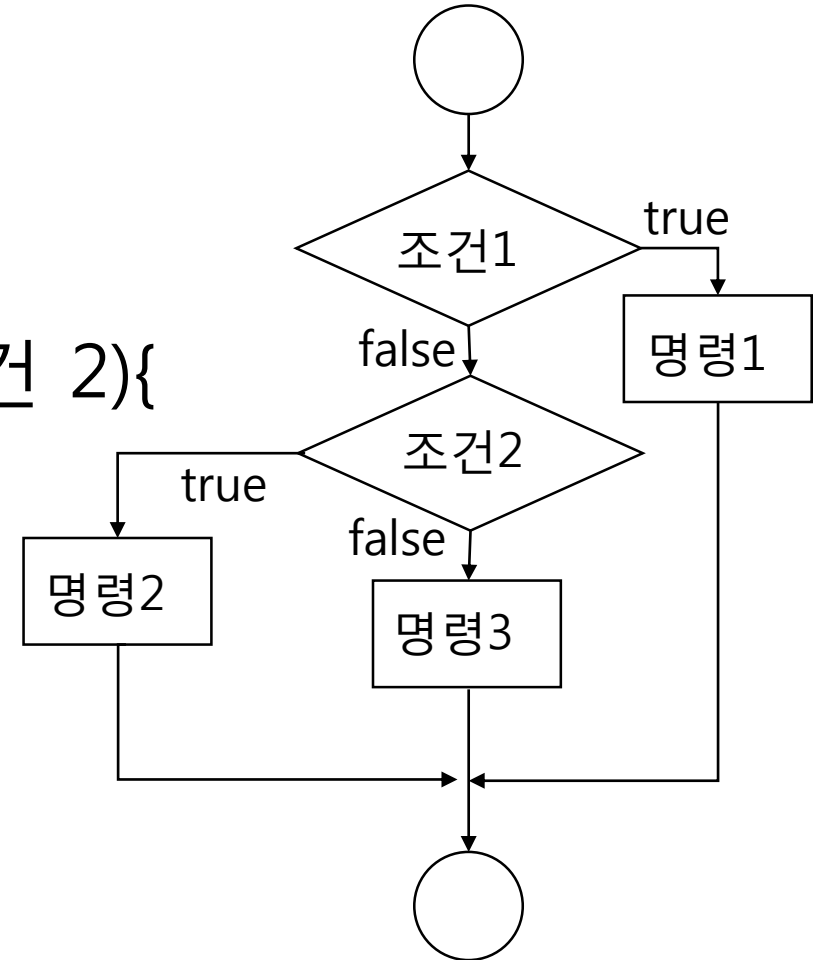
4.제어문

if - else if - else

```
if(조건){  
    명령문;  
    ...  
}
```



```
if(조건1){  
    명령문1;  
    ...  
}else if(조건 2){  
    명령문2;  
    ...  
}else{  
    명령문3;  
}
```



4.제어문

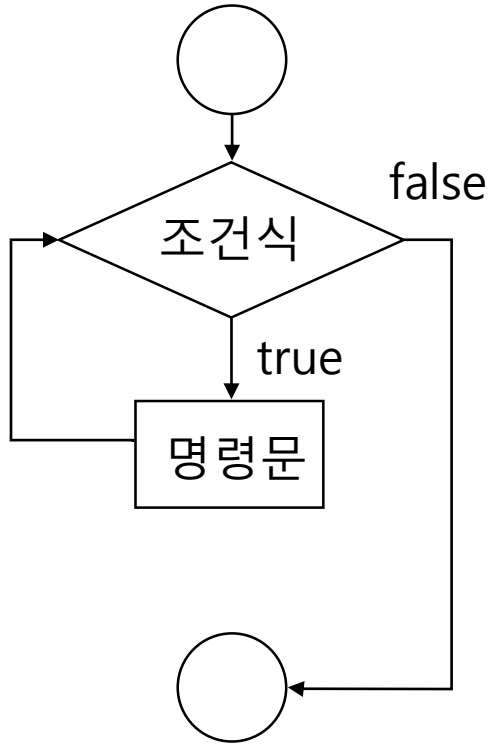
switch

```
switch(변수){  
case 값1:  
    명령문; break;  
case 값2:  
    명령문; break;  
...  
default :  
    명령;  
}
```

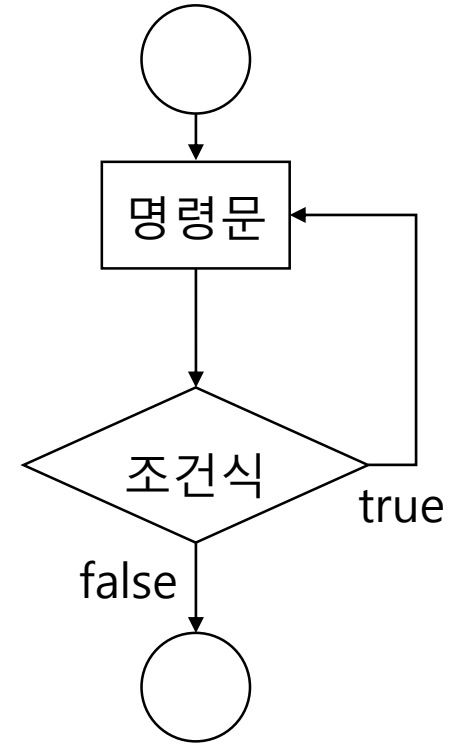
4.제어문

while

```
while(조건){  
    명령문;  
    ...  
}
```



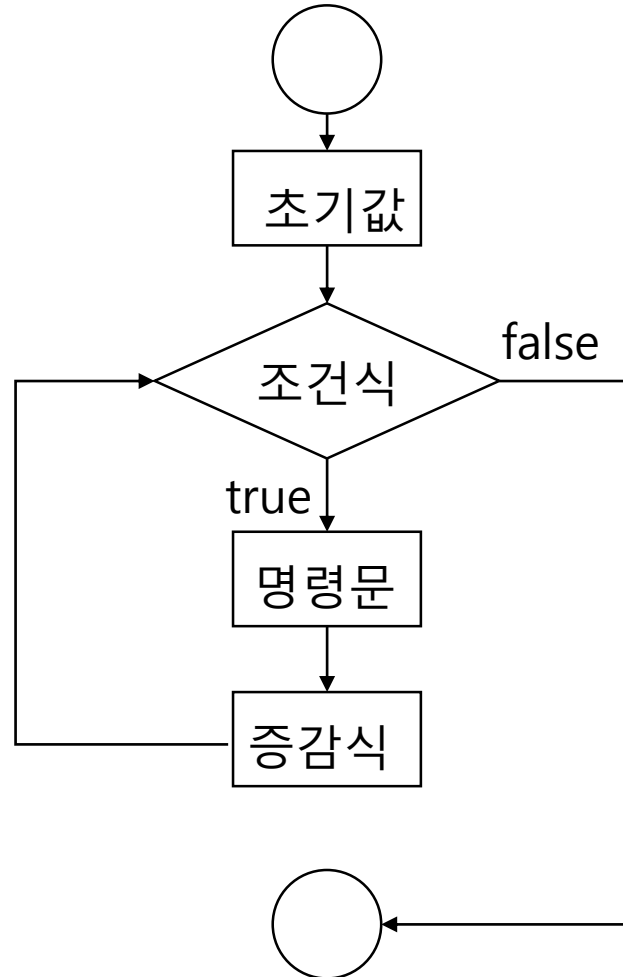
```
do{  
    명령문;  
    ...  
}while(조건);
```



4.제어문

for

```
for(초기값 ; 조건식 ; 증감식){  
    명령문;  
}
```



4.제어문

break/continue

* break와 continue의 차이점

break : 해당 블록{ } 을 탈출한다.

- 코드의 아랫부분으로 내려간다.

continue : 처음으로 돌아간다.

- 코드의 윗부분으로 올라간다.

5.Scanner

console 입력

```
import java.util.Scanner;  
Scanner sc = new Scanner(System.in);
```

| method | 설명 |
|---------------------|----------------------|
| String next() | 다음 아이템을 문자열 타입으로 리턴 |
| String nextLine() | 한 라인 전체를 문자열 타입으로 리턴 |
| long nextLong() | long 타입으로 리턴 |
| int nextInt() | int 타입으로 리턴 |
| double nextDouble() | double 타입으로 리턴 |

6.Array

배열 선언

*배열 -> 참조 타입(mutable) : 주소값으로 변경

*new 를 사용하지 않아도 객체 생성 가능!

데이터타입 [] 변수 = {,,};

선언 방법

1.선언; 정의; 초기화;

```
//int[]a; a = new int[]; a[n]=m;
```

2.선언 정의 초기화;

```
//int[]b = new int[]{m};
```

3.선언 초기화;

```
//int[]c = {m};
```

6.Array

다차원배열

차원에 따라 [], [], 다차원이 존재한다.

데이터타입 [][] 변수 = {{},{},{}};

//int[][] d = {{1,2,3,4},{5},{6,7},{8}};

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | | | |
| 6 | 7 | | |
| 8 | | | |

6.Array

복사

shallow copy : 주소값 전달. (복사본을 변경하면 원본도 변경)

deep copy : 객체를 복사하여 새로운 객체 연결값을 넘김
(복사본을 변경해도 원본에 영향 없음)

7.String

String class

문자열을 관리하는 클래스.

char[] 대신 "" 상수를 관리

한 번 생성된 문자열은 CRUD(create, read, update, delete) 불가.

7.String

memory

heap 영역에 있는 string pool에 리터럴 생성

```
String a = "abc";  
String b = "abc";  
String c = new String("abc");  
System.out.println(a==b);           // true  
System.out.println(a==c);           // false  
System.out.println(a.equals(b));     // true  
System.out.println(a.equals(c));     // true
```

7.String

특징

- 참조타입 중 유일하게 기본타입의 특징을 가지고 있다.
- immutable (대입하기 전에는 값이 변하지 않는다.)
- 값이 변하면 새로운 객체를 생성한다.
(hash코드가 변하는 것을 볼 수 있다.)

*String Concatenation : 문자열을 만나면 문자열이 된다.

```
// String s = "Hello";
```

```
1) s = 1 + 2 + s;           "3Hello"
```

```
2) s = s + 1 + 2;          "Hello12"
```

7.String

StringBuffer/StringBuilder

- mutable (대입하지 않아도 값이 변한다.)
- 값을 변경해도 hash 코드는 변하지 않는다.
(=하나의 객체가 가진 값이 변한다.)

7.String

StringBuffer/StringBuilder

*공통점

- Create(생성자), Read(tostring), Update(append,insert), Delete(delete)

*차이점

- StringBuffer 클래스는 스레드에 안전하게(ThreadSafe) 설계되어 있으므로, 여러 개의 스레드에서 하나의 StringBuffer 객체를 처리해도 전혀 문제가 되지 않는 특징을 가진다.
- StringBuilder는 단일 스레드에서 안정성만을 보장한다. (동기화가 없다)
여러 개의 스레드에서 하나의 StringBuilder 객체를 처리하면 문제가 발생.

7.String

문자열 자르기

1. substr (java.lang.String.substring)

– index로 자르기

2. split (java.lang.String.split)

– 정규식으로 자르기 (배열로 리턴)

3. StringTokenizer(java.util.StringTokenizer)

– 일정한 token으로 자르기 (토큰으로 리턴)

8.Class

OOP

Object Oriented Programming : 객체를 기능별로 나눈다.

- 추상화(abstraction) : 여러 객체들을 하나의 추상적인 큰 개념으로 묶는다.
- 캡슐화(encapsulation) : 객체의 기능을 사용만 한다. (코드는 은닉)
- 상속(inheritance) : 상위 개념을 하위 객체가 물려받는다.
- 다형성(polymorphism) : 같은 이름의 기능이지만, 다른 행위를 하는 것

8.Class

inheritance

1. 하위클래스(자식)와 상위클래스(부모) 간의 확장 : extends
 - 다중상속 불가능. 자식 : this / 부모 : super
2. 클래스와 인터페이스 간의 확장 : implements
 - 다중상속 가능, 다형성
3. 인터페이스와 인터페이스 간의 확장 : extends
 - 다중상속 가능

8.Class

memory

특징

- 객체를 생성할 때 부모객체를 먼저 생성
- 부모의 메소드가 먼저 생성되고 자식의 메소드가 생성
- 생성된 주소는 부모의 주소
- 메모리에 공개되어 있는 메소드만 사용 가능

| static area | stack area | heap area |
|---------------------------|-------------------------|--------------------|
| class, (static) method | local variable, 연산 | 객체, 생성자, 필드, 멤버 |
| application 실행 시 가장 먼저 할당 | lifo(last in first out) | garbage collection |
| 딱 1번만 메모리에 적재 | | |

8.Class

polymorphism

*전제조건 : 상속과 오버로딩이 되어야 한다.

여러가지 타입을 가질 수 있다.

- 부모(선조)의 타입으로 자식(후손) 생성

```
//Parent p = new Child();
```

- 부모의 이름으로 자식 대입

```
//Child c = new Child();
```

```
//Parent p = c;
```

- 부모의 메서드를 자식의 메서드로 호출

8.Class

overload override

overloading

- 전제조건 : 내 클래스 내에서만 작동
- 동일 기능의 메서드이지만, 파라미터가 다를 때(개수, 타입 등).
- 리턴 타입도 다를 수 있다.(이름은 같아야 함)

overriding

- 전제조건 : 상속(is a 관계) implements, extends 에서 작동
- 부모의 메서드를 자식이 변경 또는 확장
- 동일 이름, 동일 파라미터, 동일 리턴타입

8.Class

abstract

추상 클래스 : 하나 이상의 추상 메소드를 가진 클래스

- new 사용 불가
- 상속 강요 (상속받는 곳에서 반드시 구현해야 한다.)
- 클래스 내에 abstract method 가 한 개 이상 있으면 반드시 abstract class가 되어야 한다.
- 반드시 상속을 통해서 사용해야 할 클래스의 경우 abstract Class로 선언을 한다
ex) 전자정부프레임웍(eGov) 의 eGvoAbstract Class 등

8.Class

interface

인터페이스 : 모든 메소드가 추상 메소드

- 변수는 자동으로 상수화
- private, protected 사용 불가 (public 또는 default사용)
- 선언만 한다 : 바디({}) 를 붙이지 않는다. (메서드 이름을 나열)

9.Collection

비교

collection : 자료(데이터, 객체)를 보다 쉽게 관리하기 위한 클래스

*collection과 array의 차이점 : generic(type 관리), capacity(가변 용량)

| | List<t> | Set<t> | Map<k,v> |
|-----|---------|--------|----------------|
| 순 서 | O | X | X |
| 중 복 | O | X | k : X v : O |

9.Collection

generic

형 변환을 하지 않고 클래스를 사용할 수 있는 기능.

(프로그래머가 만들고자 하는 의도대로 특정한 타입만 들어갈 수 있도록 강제함.)

```
// List<String> al= new ArrayList<>();  
    al.add("string type");
```

*<T> : Type

<K> : Key

<V> : Value

10.Singleton

design pattern

지정한 클래스의 인스턴스를 단 1개만 존재하도록
(= 메모리 구조에서 heap 영역에 단 한 개의 인스턴스만 가지도록)

*singleton pattern 생성 방법

1) 생성자를 private으로 선언

```
// private singleton(){};
```

2) 객체를 확인할 주소(reference) 객체명 선언

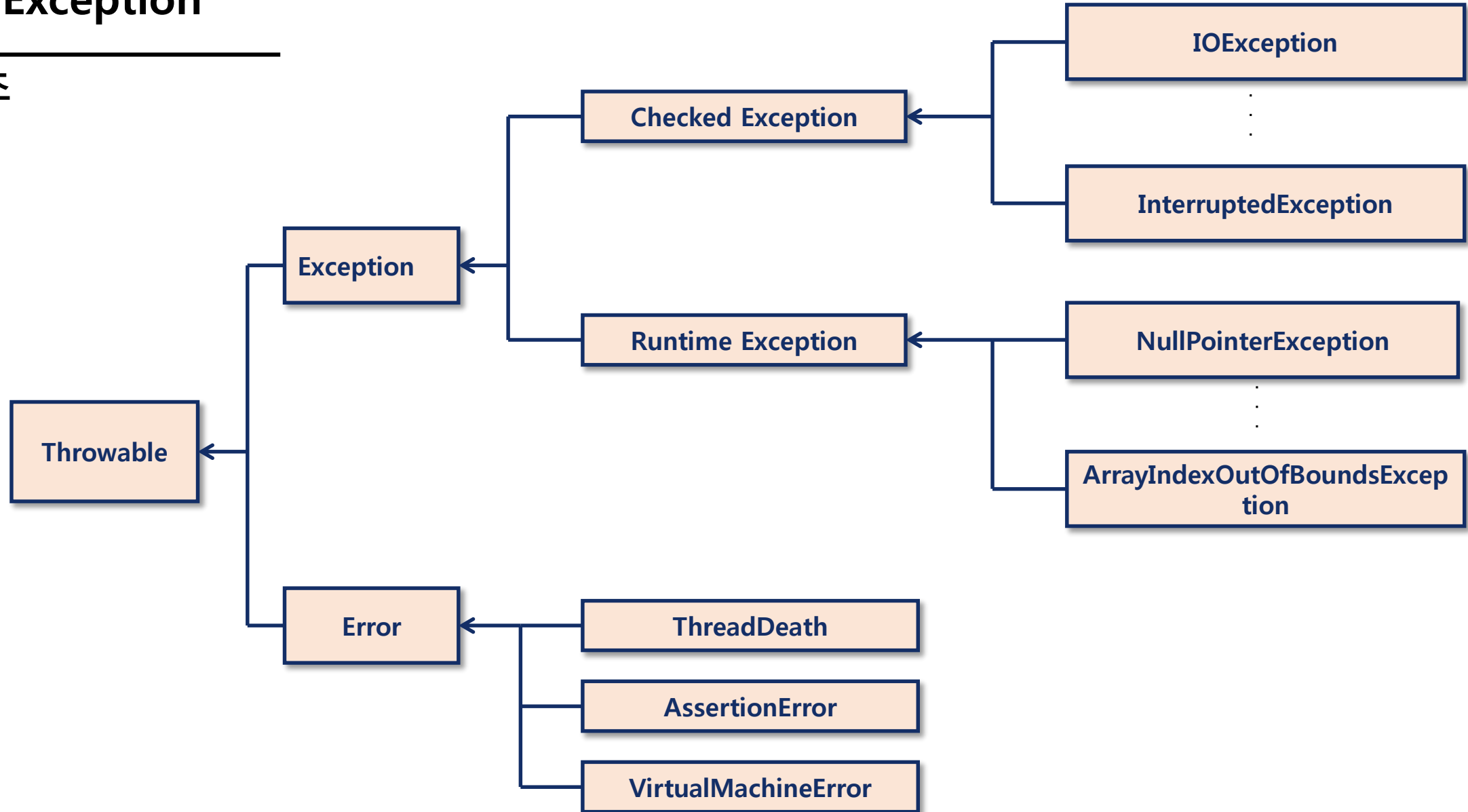
```
// private static Singleton singleton;
```

3) 외부에서 접근 가능한 메서드 선언 (자신의 클래스를 리턴타입으로)

```
// public static Singleton getInstance(){};
```

11.Exception

구조



11.Exception

error와 exception

error는 기본적으로 jvm에서.. (= 우리가 건드릴 수 없다.)

exception은 해당 코드에서 발생. 예측하거나 처리할 수 있다.

- checked exception : 컴파일 단계에서 확인된다. 꼭 처리해야 함.
- runtime exception : 실행 단계에서 확인.

프로그래머의 부주의로 발생하는 것이 대부분

11.Exception

예외처리 방법

```
try{  
    // 예외가 발생할 수 있는 명령;  
} catch( 발생할 수 있는 예외 ){  
    // 예외가 발생했을 때 실행할 명령;  
} finally{  
    // 예외와 상관없이 마지막에 무조건 실행해야 하는 명령;  
}
```

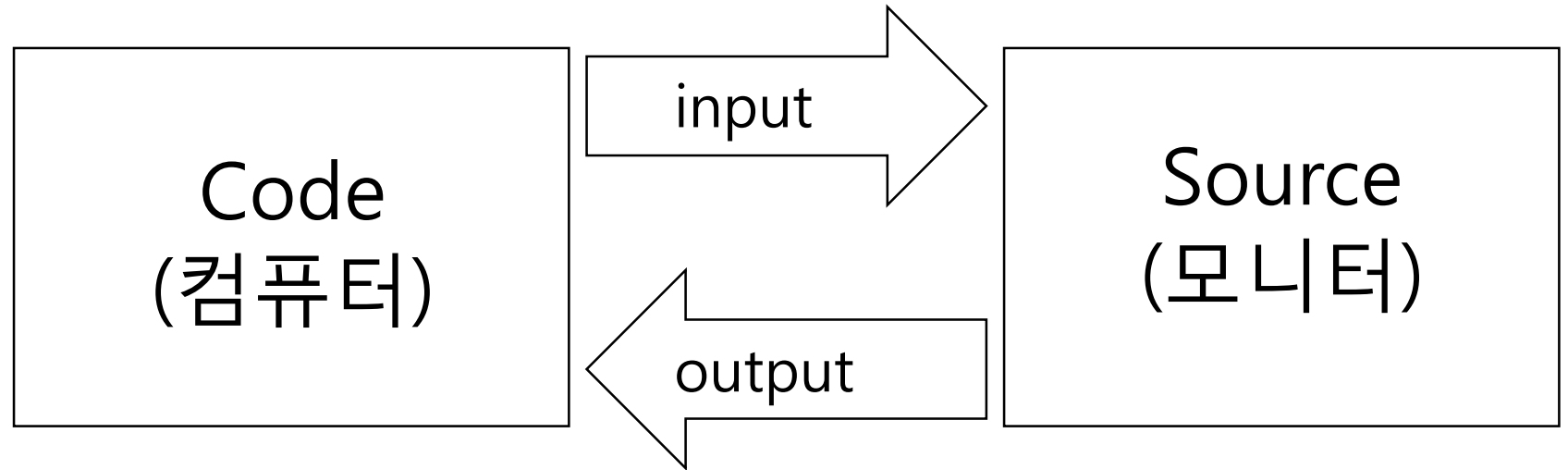
12.IO

input, output

Stream

- 연속적인 데이터의 흐름.
- 전용 단위는 byte

*input : read(); //출력
output : write(); //입력



13.Thread

process

Process : 응용프로그램 단위로 실행되는 프로그램

- 실행 중인 프로그램을 뜻한다.
- 하나의 cpu(프로세서)는 하나의 프로세스를 실행시킨다.
- 스케줄링의 대상이 되는 작업(task)이라고도 한다.

13.Thread

thread

Thread : 하나의 process에서 실행되는 작업 단위

- 메서드 단위로 실행되는 모듈
- java의 main() : java applicatio을 실행했을 때 가장 먼저 시작되는 main thread

*thread 작성 방법

- 1) Runnable 을 implements
- 2) Thread를 extends

14.URI

uri url urn

URI (Uniform Resource Identifier) : 자원 식별자

URL (Uniform Resource Locator) : 자원 위치

URN (Uniform Resource Name) : 위치에 상관이 없는 자원 이름

*URI = URL + URN

15.Network

socket 통신

TCP : data를 packet으로 쪼개서, 받았다는 응답을 받을 때 까지 보냄

UDP : data block을 보냄, 응답을 상관하지 않기 때문에 빠름

- tcp server : 소켓 생성, 연결 기다림, 응답

tcp client : 소켓 생성, 연결

- udp server : 소켓 생성, 데이터 송수신

udp client : 소켓 생성, 데이터 송수신

