SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

7기 입학을 축하합니다!

Start Camp

Start Camp

1. 웹 개발 Overview
2. Java 기본
3. Firebase Realtime Database를 활용한 채팅시스템 개발


SAMSUNG
SW
ACADEMY
FOR
YOUTH

웹 개발 Overview

HTML

HTML is

- Hyper Text Markup Language
- Markup Language, Not Programming Language
- Markup Tag의 집합
- W3C(World Wide Web Consortium)에서 표준 관리

참고 URL
https://www.w3schools.com/
https://developer.mozilla.org/ko/docs/Web/HTML

Web Editor
https://codepen.io/pen/

HTML

Element(요소)

- content의 type을 지정
- Element_name은 표준으로 정의

Attribute(속성)

- 생략 가능
- Element의 속성 지정
- attribute_name과 value set을 표준으로 정의

Comments(주석)

- <!---->
- 브라우저에선 표시되지 않음
- 더 읽고 이해하기 쉬움

HTML

Block level element

- 대부분 Inline 요소와 Text 요소를 포함
- 일부 요소는 Block 요소를 포함
- 새로운 행에 표시
- Ex) : , <p>, <h1>, <form>, <table>, <li>...

Inline level element

- 오직 Inline 요소와 Text 요소만 포함
- Text처럼 취급됨
- 새로운 행에 표시되지 않음
- Ex) : , <a>, <img>, <input>, <label> ...

https://www.w3schools.com/html/html_blocks.asp

HTML

<pre>

- Preformatted text
- Block level element
- 공백문자와 줄바꿈 문자를 보존

- Ordered List
- Block level element
- Attributes
    - start = number
    - type = {1 | A | a | I | i}

HTML

- Unordered List
- Block level element

- List item
- 또는 안에서 하나의 Item을표현
- Attributes
    - value = number, only within

HTML

- Block level element
- Grouping block level elements
- No visual changes

HTML

<a>

- Anchor
- Hyper link to another page
- Inline level element
- Attributes
- href= URL
- target = { _blank | _parent | _self | _top | framename}
- hreflang= language_code

HTML

- Inline level element
- Grouping inline level elements
- No visual changes

HTML

- Defines an image
- Inline level element
- Attributes
- src=URL
- alt=text, 대체 text
- height=pixels or %
- width=pixels or %
- usemap=#map_name
- ismap=ismap

CSS

Cascading Style Sheets

- (x)HTML Element(Markup)의 시각적 표현(Appearance) 정의

- CSS Levels
  - CSS, Earliest Draftin May, 1995
  - CSS Level1, W3C Official Recommendation in Dec, 1996
  - CSS Level2, W3C Official Recommendation in May, 1998
  - CSS Level3, Working Draft(Not yet Recommendation)

- 장점
  - 구조와 표현의 분리, SementicMarkup
  - FileSize 감소
  - 효율적이고 정교한 디자인 제어
  - Browser 호환성에 대처 용이

CSS

CSS를 사용하는 3가지 방법

- **External Stylesheet, <link>**

```
<head>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

- **Internal StyleSheet, <style>**

```
<head>
    <style type="text/css">
        body { margin: 0; padding: 0}
        p      { color : red; }
    </style>
</head>
```

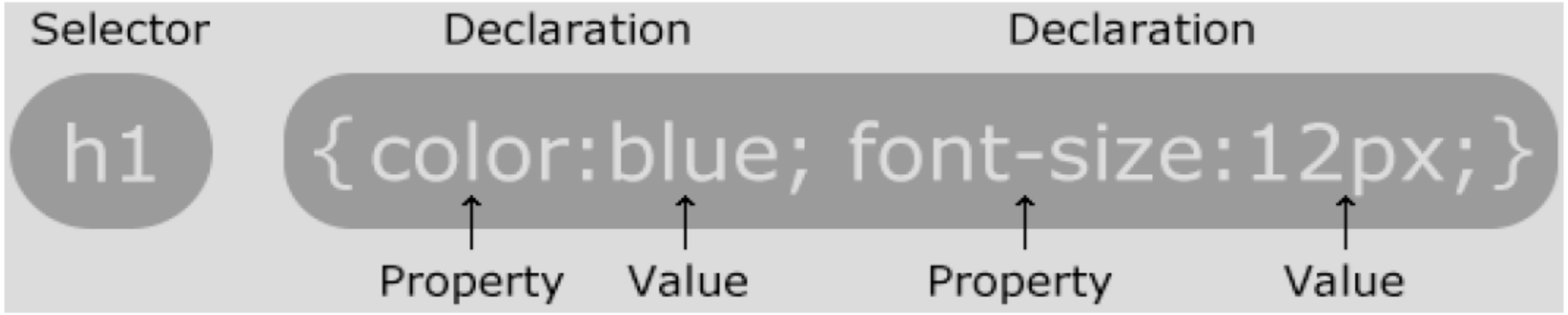
- **Inline Styles**

```
<p style="color:red; border:1px solid #000;">This is a paragraph. </p>
```

CSS

CSS Syntax

- Selector, Property, Value



- 주석

```
<head>
    <style type="text/css">
        /*  This is a comment */
        p{ color:red; text-align:center; }
    </style>
</head>
```

Selector(선택자)

스타일을 지정할 대상 요소를 선택하는데 사용하는 패턴 표기법

- Universal Selector

Selector	Example	Description
*	*	모든 요소 선택

- Type Selector

Selector	Example	Description
element	div	모든 요소 선택

CSS

- ID Selector

Selector	Example	Description
#id	#notice	id="notice"인 요소 선택

- Class Selector

Selector	Example	Description
.class	.head	class="head" 인 요소 선택

- Attribute Selector

Selector	Example	Description
[attribute]	[type]	type 속성을 갖는 모든 요소
[attribute=value]	[type=text]	type="text" 속성을 갖는 모든 요소
[attribute~=value]	[type~=text]	type 속성의 값이 "text" 단어를 포함하는 모든 요소
[attibute|=value]	[type|=text]	Type 속성의 값이 "text" 단어로 시작하는 모든 요소

자기소개 페이지 만들기

- 자기소개 페이지 만들기

참조 : https://www.w3schools.com/

https://codepen.io/pen/
http://jsbin.com/

자기소개 페이지 발표

재미로 보는 웹/안드로이드 통계

URL

stackoverflow
https://insights.stackoverflow.com/survey/2020

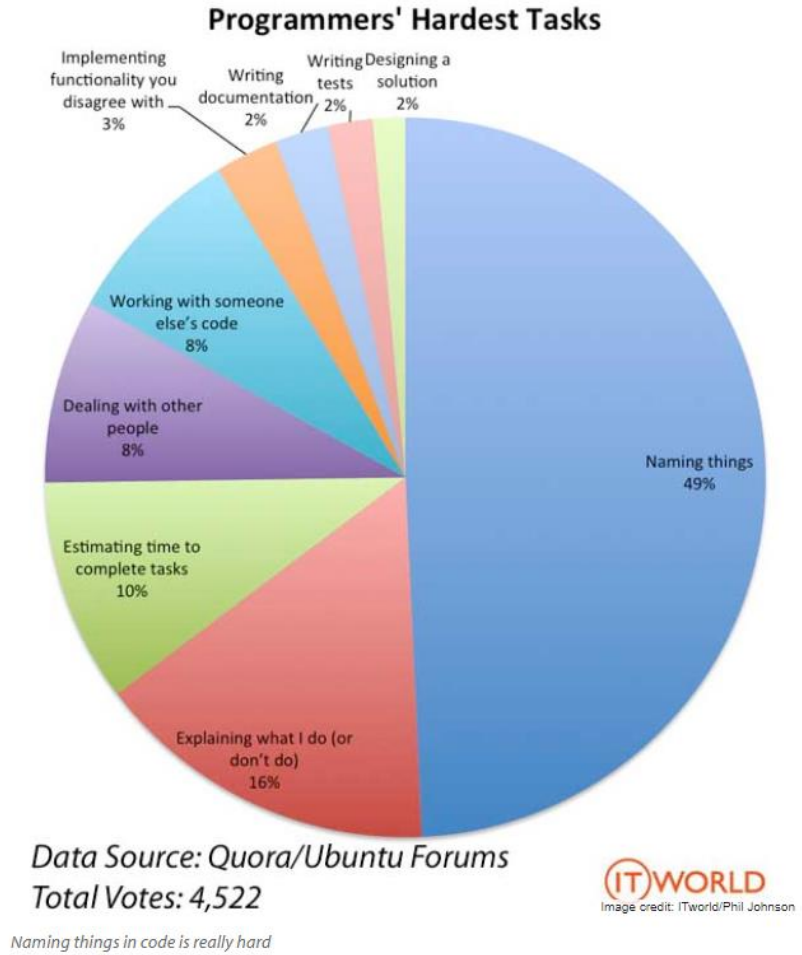
App Annie
https://www.appannie.com/kr/go/state-of-mobile-2021/

statcounter
https://gs.statcounter.com/

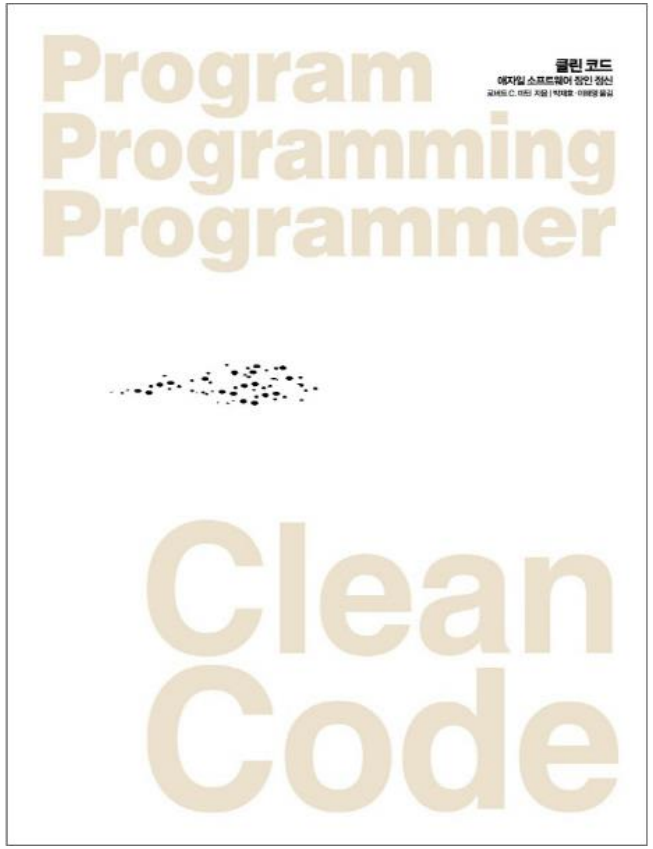
Programmers Hardest Tasks

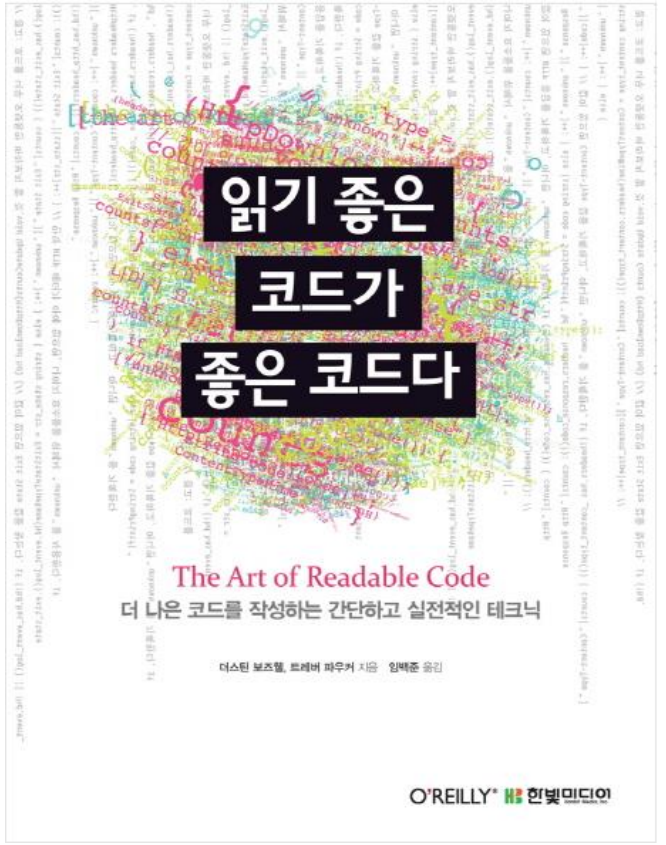
https://www.cio.com/article/3404204/the-9-hardest-things-programmers-have-to-do.html

https://www.itworld.co.kr/slideshow/85296



좋은코드 만들기


Program
Programming
Programmer
클린 코드
애자일 소프트웨어 장인 정신
Clean
Code


읽기 좋은
코드가
좋은 코드다
The Art of Readable Code
더 나은 코드를 작성하는 간단하고 실전적인 테크닉
O'REILLY
한빛미디어

어떤프로그래머가 훌륭한 프로그래머일까요?

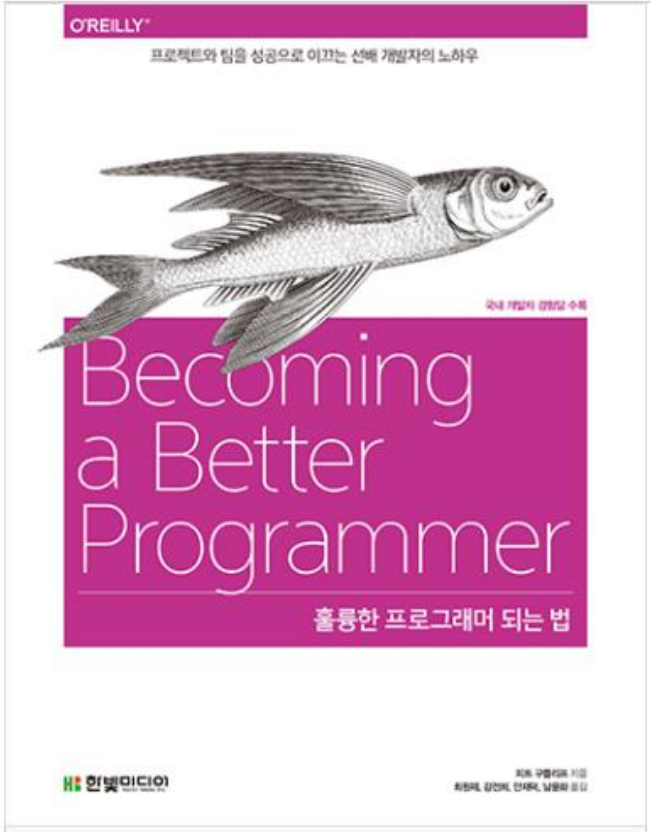
제 1의 조건 :
코드에 신경쓰고 귀 기울이라.
단지 작동하는 코드를 거부하라.
의도가 드러나는 코드를 작성하라.
유지보수가 가능하게 하라.
협업하라.

기본강령 :
어떤 코드를 만지더라도 만지기 전보다 더 나아지도록 해야 한다.

코드 만들기 :
규칙 : 모호하게 구두로 전해지는 팀의 규칙에 의존하지 마라. 명백하게 만들어라.

중복코드 금지. copy and paste 금지.

불필요한 코드를 만들지 마라. : YAGNI (You Aren't Gonna Need It !)
당장 필요없다면 만들지 마라.



어떤프로그래머가 훌륭한 프로그래머일까요?

기술부채 : Technical Debt
임시방편 처리기술은 빚이다. 추후 반드시 수정(지불)해야 한다.

콘웨이 법칙 : 개발팀이 4조직이면 코드모양도 4가지, 컴파일러도 네가지가 만들어진다.
의사소통의 구조에 따라 코드의 형태가 결정된다.

간단하게 만들라.
KISS : Keep It Simple, Stupid !

작업은 작게 나누고, 상태가 보이게 관리하라.

만들고 난 후 :
코드는 배포하라.

변하지 않는 것은 없다.
- 용감한 수정이 필요한가? 수정에 필요한 것은 무모함이 아니라 용기와 기술이다.

어떤프로그래머가 훌륭한 프로그래머일까요?

만들어진 코드는 :
리펙토링 : 작동은 그대로 유지한 채 소스코드를 바꾸는 행위.

코드는 항상 정리정돈해야 한다. 정리정돈 과 변경을 동시에 하지 마라.!

개발하는 과정에서 사장된(죽은) 코드는 튀어 나오기 마련이다.
불리지 않는 메소드, 사용되지 않는 클래스, 변수. 수행하지 않는 if문 else문...
남아 있는 불필요한 코드들 : 당장 제거하라.
미래에 필요할지 모르는 기능이다... 필요할 때, 그때 찾으라.

소프트웨어를 개선하는 최고의 방법 가운데 하나는 코드를 제거하는 것이다.
불필요한 코드를 제거하라.

오래된 코드를 보고 스스로 반성하라.

어떤프로그래머가 훌륭한 프로그래머일까요?

똥덩어리를 다루는 전략.
일부러 그렇게 만든사람은 없다. 냉정하게 분석하라.
- 고쳐야 할까?
- 달아나야 할까?
- 바꿔야 할까?
보이스카우트 규칙 : 캠핑지역은 캠핑전보다 더 깨끗하게 정리하고 떠나자.
--> 간단한 수정이라도 좋다. 코드의 외관부터 정리한다. 변수명부터 바꾸기 시작하라.
거대한 함수를 더 작게 자르고, 적절히 명명되게 나누라.
조급하지 말고, 조금씩 진행하라.
이런과정을 낭비라고 생각하지 마라. 그리고, 나쁜코드에 비난을 퍼부을 필요는 없다.

간단히 만들고, 테스트 코드를 작성하라.

어떤프로그래머가 훌륭한 프로그래머일까요?

협업하기 :
QA를 적으로 생각하지 마라. 도움에 감사하라.
- 오류는 처음부터 개발자의 잘못이다.
- Developer create, Testers break. 개발자는 만들고, 테스터는 부순다.
  테스트는 최종 산출물을 개선하려는 목적으로 업무를 수행하는 것이다.
- QA와 개발자를 한팀으로 배치하여 협업하게 하라.
- 코드를 고치려면 팀의 구조를 개선하라.

Code freeze라는게 가능한가?
- 동결은 기능동결, 코드동결등이 있다.
- Feature Creep
  제품에 여러가지 기능, 기술을 지속적으로 추가하다가 결국 아무도 원치 않는 결과가 나오는 현상
  (필요없는 기능이 주렁주렁 붙는 현상)

더 낫게 만들 수도 있는 소프트웨어를 그냥 배포하는 것이 잘못된 일인가?
- 심각하지 않다면 기술부채를 안는 것으로 기대하라.

심각한 영향을 미치는 변경을 가해야 한다면? freezing을 풀어야 한다.

어떤프로그래머가 훌륭한 프로그래머일까요?

업무 외적으로 성장하기 :
지속적으로 배우는 상태를 유지하라.
배움을 즐기는 것을 배우라.
배우기 위하여 가르치라.
배우기 위하여 실천하라.

도전을 즐기고, 정체에 주의하라.
- 더나은 프로그래머를 추구하는 과정은 안락한 삶을 의미하지 않는다.

현명하게 재 사용하라.
- 윤리적 프로그래머.
- 허용하는 범위내에서 있는 코드를 가져다 써라. 더 중요한 일에 시간을 투자하라.

모든 언어를 사랑하라.

어떤프로그래머가 훌륭한 프로그래머일까요?

그리고, 무엇보다도 중요한
커뮤니케이션.
기계와의 대화, 팀원과의 대화, 도구와의 대화, 고객과의 대화

태도가 핵심이다.
당신의 코드가 좋은 것인지 당신이 함께 일하기 좋은 사람인지를 결정하는 기준은
바로 당신의 태도이다.

Java 기본 지식 복습 및 확인

SAMSUNG SW ACADEMY FOR YOUTH

목차

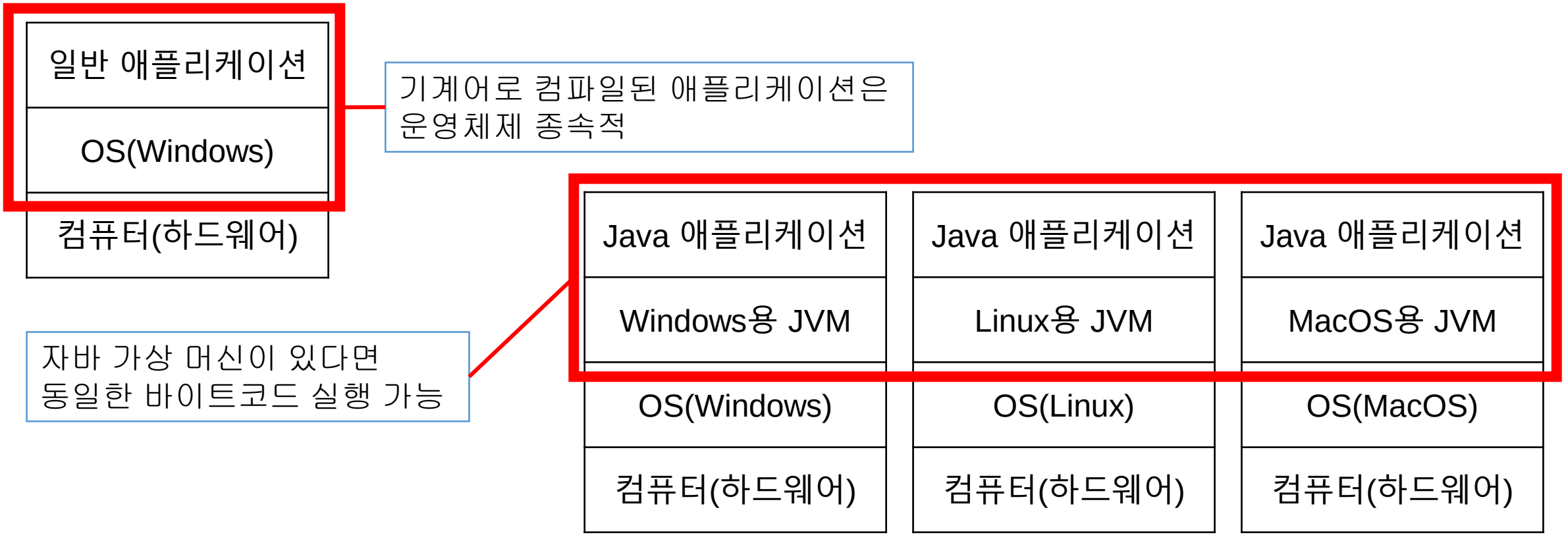
1. Java 기본 지식 복습 및 확인

- 자바의 특징
- 변수
- 데이터 타입
- 연산자
- 조건문
- 반복문
- 배열

자바의 특징

기본 문제 #01

- **Write Once, Run Anywhere**는 **Java**의 어떤 특징을 설명하는지 기술하세요.



기본 문제 #02

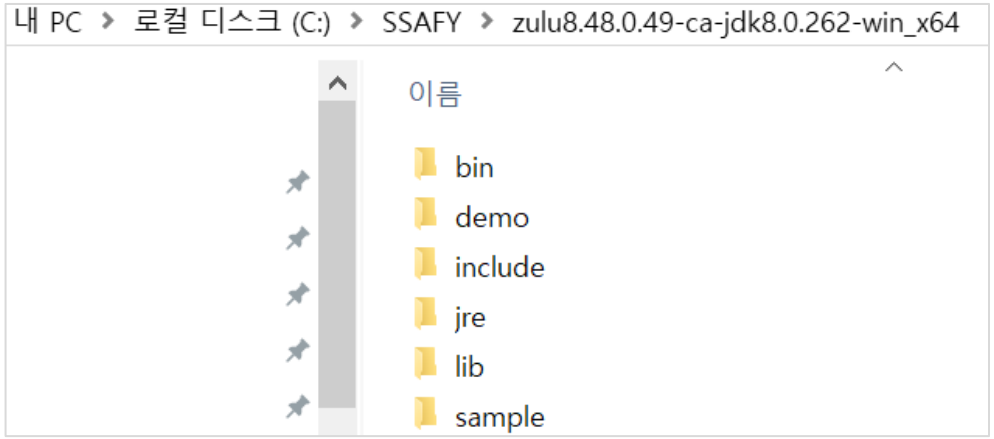
- 자바의 특징 중 하나로 더 이상 사용하지 않는 메모리를 자동으로 정리하는 기능을 무엇이라고 하는가?

```
String hello = new String("Hello");

System.out.println(hello);

hello = null;
```

Garbage Collection

기본 문제 #03

- 다음은 **Windows 10** 기준 **Zulu Open JDK 1.8** 을 설치한 **Folder** 구조이다. 환경변수 **JAVA_HOME** 에 맞는 값은?



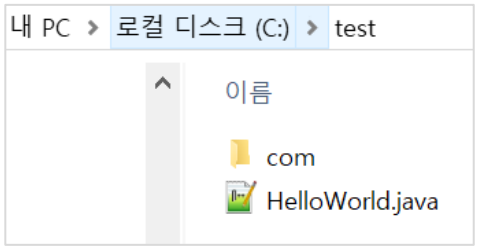
1. C:\SSAFY
2. C:\SSAFY\zulu8.33.0.1-jdk8.0.192-win_x64
3. C:\SSAFY\zulu8.33.0.1-jdk8.0.192-win_x64\bin
4. C:\SSAFY\zulu8.33.0.1-jdk8.0.192-win_x64\jre

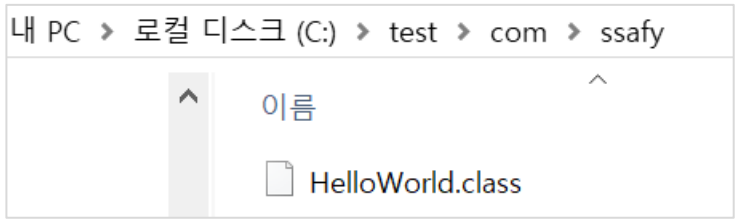
2. C:\SSAFY\zulu8.33.0.1-jdk8.0.192-win_x64

기본 문제 #04

- 다음 **Java Code**는 **"Hello, World!"** 를 출력하도록 **C:\test** 폴더에 작성되었다. 동일 폴더에 **compile** 한 **.class** 파일을 **package** 구조로 만들고자 한다. **CLI compile & run** 을 수행하시오.

```java
package com.ssafy;

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello, World!!");
    }
}
```





1. compile : C:\test>
2. run : C:\test>

```
C:Wtest>javac -d . HelloWorld.java
```

```
C:Wtest>java com.ssafy.HelloWorld
Hello, World!!
```

CLI 실습 - HelloWorld.java

- 메모장
- 코드 : 기본 문제 #04

기본 문제 #05

- **Java** 프로그램을 개발할 때, 일반 **Editor(notepad, notepad++** 등) 사용하면, 어떤 점이 불편한지 기술하고, 이를 개선하기 위해 **eclipse** 같은 **IDE**를 사용하면 어떤 장점이 있는지 기술하세요.

IDE (Integrated Development Environment)

예) Eclipse, Spring Tools Suite, Visual Studio Code, IntelliJ 등

코드 편집	코드 Assistant	소스 버전 관리	…
	단위테스트 툴	프로젝트 관리	
	코드 리펙토링	개발도구 관리	

실수가 줄어들고
개발 생산성이 향상된다!

IDE 실습 - HelloWorld.java

- Eclipse
  - Encoding & Perspective & Build path
  - eclipse.ini (-Dfile.encoding=UTF-8)
- 클래스 구성
  - public class
  - public static void main()
  - System.out.println()
- Java API
  - https://docs.oracle.com/javase/8/docs/api/index.html?overview-summary.html

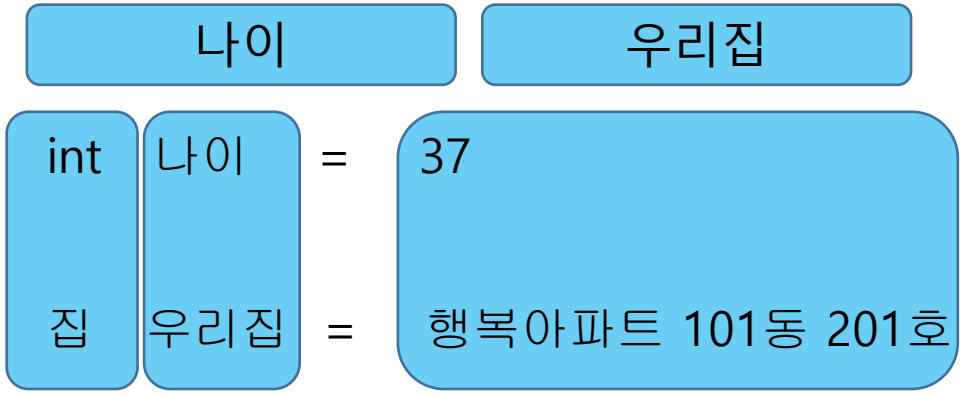
변수, 데이터 타입

Variable 이란?

- 수학에서는 변하는 수
- 컴퓨터에서는 메모리 공간, 그릇
- 메모리 공간에 값**(value)**을 할당**(assign)** 후 사용
- 공간의 크기는 타입별로 달라짐

Type 이란?

- 변수에 저장되는 데이터의 종류
- **Primitive Type (기본형)**
  - **미리 정해진 크기의 Memory Size로 표현**
  - **변수 자체에 값 저장**
- **Reference Type (참조형)**
  - **미리 정해질 수 없는 데이터의 표현**
  - 변수에는 실제 값을 참조할 수 있는 주소만 저장



long	l, L	long value = 19873343L;
float	f, F	float f = 13.579F;
double	d, D	double d = 13.579d; // 생략 가능

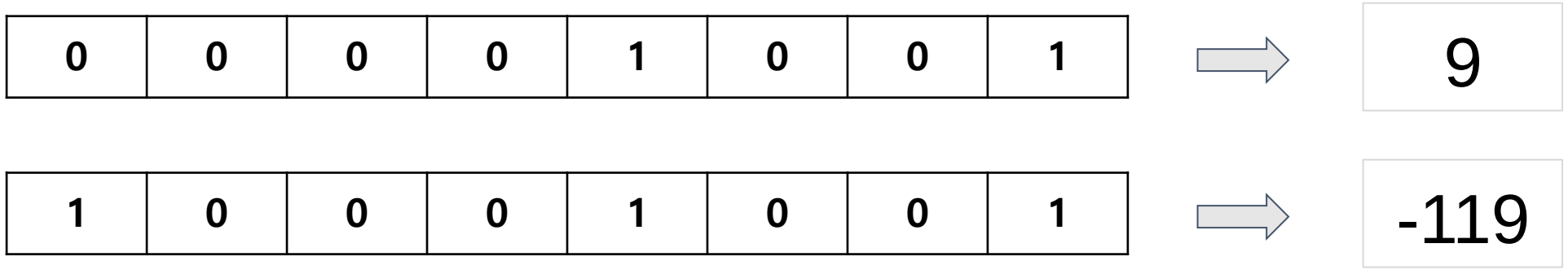
기본 문제 #06

- **Java 의 Primitive Type 을 모두 기술하세요.**

구분	Type	bit 수	값
논리형	**boolean**	1	true / false
정수형	**byte**	8	-2^7 ~ 2^7-1 (-128 ~ 127)
	short	16	-2^15 ~ 2^15-1 (-32768 ~ 32767)
	int	32	-2^31 ~ 2^31-1 (-2147483648 ~ 2147483647)
	long	64	-2^63 ~ 2^63-1 (-9223372036854775808 ~ 9223372036854775807)
실수형	**float**	32	
	double	64	
문자형	**char**	16	₩u0000 ~ ₩uffff (0 ~ 2^16-1) 아스키코드

기본 문제 #07

- 다음은 **byte type** 의 어떤 수에 대한 메모리 표현이다. 이 수의 값은?



- 128 + 9
- Java Uses Signed Number Representation → First Bit 1 means Negative

기본 문제 #08

- 다음 코드는 어떤 문제가 있는지 설명하세요.

```
package com.ssafy;

public class Test {
    public static void main(String[] args) {
        int i;

        System.out.println(i);
    }
}
```

이후, 문제 실습 코드는
Test Class 를 이용하세요!

- 선언 후 초기화 되지 않고 사용되었음.
- local variables 는 사용 전에 값을 부여 받아야 함.

기본 문제 #09

- 다음 코드는 어떤 문제가 있는지 설명하세요.

```
package com.ssafy;

public class Test {
    public static void main(String[] args) {
        {
            int i = 0;
            System.out.println(i);
        }

        System.out.println(i);
    }
}
```

- 두 번째 println(i) 의 i 는 유효하지 않음.
- local variable 은 선언된 { } 에서 유효함.

기본 문제 #10

- 다음 코드는 어떤 문제가 있는지 설명하세요.

```
package com.ssafy;

public class Test {
    public static void main(String[] args) {

        final int i = 0;
        System.out.println(i);

        i = 10;
        System.out.println(i);
    }
}
```

- 상수 (Constant) 는 값이 한번 정해지면 변경할 수 없음.

기본 문제 #11

- 다음 코드는 어떤 문제가 있는지 설명하세요.

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

    // A
    {
        int i = 10;
        byte b = i;
    }

    // B
    {
        byte b = 10;
        int i = b;
    }

}
```

- byte b = i;
- 큰 type 의 변수는 작은 type 의 변수에 바로 할당 할 수 없음.
- 명시적으로 형변환 (type-casting) 할 수 있으나, 값이 손실 될 수 있음.

```
byte b = (byte) i;
```

형변환 (Type-Casting) 이란?

- 변수의 타입을 다른 타입으로 변환하는 것
  - **char <- -> int**
- **primitive**는 **primitive**끼리, **reference**는 **reference**끼리 형 변환 가능
  - **Boolean**은 다른 기본 타입과 호환되지 않음
  - 기본 타입과 참조형의 형 변환을 위해서 **Wrapper** 클래스 사용

- 형 변환 방법
  - 형 변환 연산자(괄호) 사용

```
double d = 100.5;
int result = (int)d;
```

result = ?
d = ?

JAVA 기본 문법, 응용

- 기본형의 형 변환 진행
  - 작은 집에서 큰 집으로 이동할 땐 값 손실이 없음
  - 큰 집에서 작은 집으로 이동할 땐 값 손실이 있음

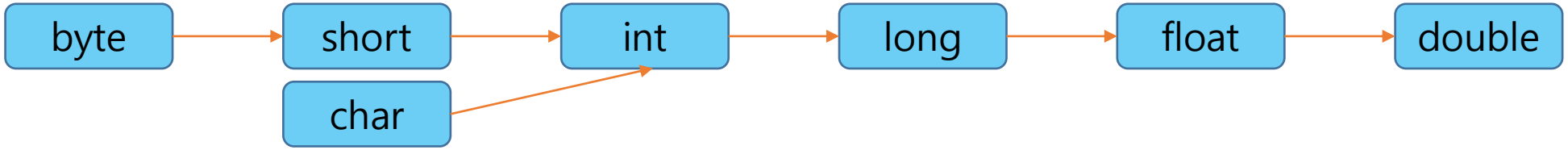
- 묵시적 형 변환

```
byte b = 10;
int i = (int)b;
int i2 = b;
```

명시적 형 변환

```
int i = 300;
byte b = (byte)i;
```

- 값의 크기, 타입의 크기가 아닌 타입의 표현 범위가 커지는 방향으로 할당할 경우는 묵시적 형 변환 발생



- 명시적 형 변환은 값 손실이 발생할 수 있으므로 프로그래머 책임 하에 형 변환 진행
- 묵시적 형 변환은 자료의 손실 걱정이 없으므로 **JVM**이 서비스 해줌

기본 문제 #12

- **Java** 에서 문자열 **"Hello"**를 변수 **s** 에 할당하려고 한다. 맞는 것은?

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

    string s1 = "Hello";

    String s2 = "Hello";

    String s3 = new String("Hello");

    char[] s4 = "Hello";

}
```

- Primitive Type 이 아닌 Reference Type String Class 를 이용.
- java.lang.String
- 상수 문자열 객체를 이용하는 것이 일반 String 객체를 이용하는 방법보다 효율적

기본 문제 #13

- 다음 중 **Java** 의 **comment** 로 올바르지 않은 것은**?**

1. /* 코드 */
2. -- 코드
3. // 코드
4. /** 코드 */

2. -- 코드

- Java 의 inline comment는 //, -- 는 comment 가 아님

연산자

기본 문제 #14

- 다음 코드의 실행 결과는?

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

    int k = 66;
    char c = (char) k;
    System.out.println(c);

    c = 'A';
    k = c;
    System.out.println(k);

    int i = 10 / 3;
    System.out.println(i);

    float f = 10 / 3;
    System.out.println(f);

    float f2 = 10f / 3F;
    System.out.println(f2);

    double d = 10d / 3D;
    System.out.println(d);

    System.out.println( ( 10 / 3 ) * 3 );
}
```

```
B
65
3
3.0
3.3333333
3.3333333333333335
9
```

기본 문제 #15

- 다음 코드의 (1) ~ (4) 의 실행 결과는?

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

    int i = 10;

    System.out.println( ( i-- ) % 2 );  // ( 1 )

    System.out.println( --i );          // ( 2 )

    System.out.println( i++ );          // ( 3 )

    System.out.println( ++( i - 2 ) );  // ( 4 )

}
```

```
0
8
8
// 컴파일 에러
```

연산자 (Operator)

- 어떤 기능을 수행하는 기호 (+, -, *, / …)
- 연산자 종류와 우선순위 및 결합 방향에 유의해야 함

구분	연산자
더하기	+
빼기	-
곱하기	*
나누기 몫	/
나눈 나머지	%

구분	연산자
값 증가, 감소	++, --
논리 부정	!
값 비교	== , !=
대소	>, <, >=, <=
논리 AND, OR	&&, ||, &, |
Bit	<<, >>, >>>, ~, &, |, ^
삼항	? :

기본 문제 #16

- 다음 코드의 실행 결과는?

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

    // bit and
    System.out.println("3 & 4 = " + (3 & 4));

    // bit or
    System.out.println("3 | 4 = " + (3 | 4));

    // bit exclusive
    System.out.println("3 ^ 4 = " + (3 ^ 4));

    // bit not
    System.out.println(" ~ 4 = " + (~ 4));

}
```

```
0000 0000 0000 0000 0000 0000 0000 0100
1111 1111 1111 1111 1111 1111 1111 1011



0   0   0   0   0   1   0   0
1   1   1   1   1   0   1   1
   64      16      4       1
128    32       8       2
```

```
3 & 4 = 0
3 | 4 = 7
3 ^ 4 = 7
 ~ 4 = -5
```

```
//   0011
// & 0100
// ------
//   0000

//   0011
// | 0100
// ------
//   0111

//   0011
// ^ 0100
// ------
//   0111

//   0100
// ~
// ------
//   1011
```

기본 문제 #17

- 다음 코드의 실행 결과는?

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

    // <<
    System.out.println("1 << 2 = " + (1 << 2));
    System.out.println("3 << 3 = " + (3 << 3));

    // >>
    System.out.println("1 >> 2 = " + (1 >> 2));
    System.out.println("-16 >> 2 = " + (-16 >> 2));

    // >>>
    System.out.println("7 >>> 2 = " + (7 >>> 2));
    System.out.println("-5 >>> 24 = " + (-5 >>> 24));

}
```

```
1 << 2 = 4
3 << 3 = 24
1 >> 2 = 0
-16 >> 2 = -4
7 >>> 2 = 1
-5 >>> 24 = 255
```

```
//    .... 0000 0001
// << .... 0000 0100

//    .... 0000 0011
// << .... 0001 1000

//    .... 0000 0001
// >> .... 0000 0000

//    .... 1111 0000
// >> .... 1111 1100

//     0000 .... 0000 0111
// >>> 0000 .... 0000 0001

//     1111 1111 1111 1111 1111 1111 1111 1001
// >>> 0000 0000 0000 0000 0000 0000 1111 1111
```

비트 이동연산자(쉬프트 연산자)

- 비트(**bit**) 단위로 왼쪽 또는 오른쪽으로 이동
- **<<: 앞의 피연산자 비트 열을 뒤 피연산자 만큼 왼쪽으로 이동, 빈 공간은 0으로 채움**
- **>>:** 앞의 피연산자 비트 열을 뒤 피연산자 만큼 오른쪽으로 이동, 빈 공간은 부호 비트
  예) 음수는 **1**, 양수는 **0**으로 채움
- **>>>: 앞의 피연산자 비트 열을 뒤 피연산자 만큼 오른쪽으로 이동, 빈 공간은 0으로 채움**
- **<<** 연산자는 곱하기 **2**와 동일 **(* 2)**
- **>>** 연산자는 나누기 **2**와 동일 **(/ 2)**
- ***, /** 연산자에 비해, 처리 속도가 훨씬 빠르다.

Random 수 구현하기

구분	코드	
Math.random()	Math.random()	0.0 <= ? < 1.0
	Math.random() * N	0.xxx <= N-1.xxx
	(int) (Math.random() * N)	0 <= ... <= N-1
	(int) (Math.random() * N) + 1	1 <= ... <= N
java.util.random.nextInt()	java.util.Random.nextInt(N)	0 <= ? <= N-1
	java.util.Random.nextInt(N) + 1	1 <= ? <= N
% Operator	충분히 큰 Random 수 % N	0 <= ? <= N-1
	충분히 큰 Random 수 % N + 1	1 <= ? <= N

기본 문제 #18

- 주사위를 던져서 나올 수 있는 경우의 수 **(1~ 6)** 를 시뮬레이션 하려고 한다. **Random** 수를 이용하여 코드를 작성하세요.

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

    int N = 6;

    // Math.random()
    System.out.printf( "%3d", (int) (Math.random()*N) + 1 );

    // java.util.Random
    java.util.Random generator = new java.util.Random();
    System.out.printf( "%3d", generator.nextInt(N) + 1 );

    // %
    System.out.printf( "%3d", ( (int) (Math.random()*100) % N ) + 1 );

}
```

```
  3   5   4
```

조건문

기본 문제 #19

- 기본 문제 #18 주사위를 이용하여, 1~6 의 랜덤한 결과를 얻고, 그 결과에 따라 각각 다르게 처리하는 구조를 if 문을 이용하여 만드시오.
  (단, 실제 처리 코드는 무시)

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

    int N = 6;
    int result = (int) (Math.random()*N) + 1;

    if( result == 1 ) {
        // do something
    }else if( result == 2 ) {
        // do something
    }else if( result == 3 ) {
        // do something
    }else if( result == 4 ) {
        // do something
    }else if( result == 5 ) {
        // do something
    }else if( result == 6 ) {
        // do something
    }

}
```

기본 문제 #20

- 기본 문제 **#19** 를 **if** 대신 **switch**를 사용하는 코드로 변경하시오.

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

    int N = 6;
    int result = (int) (Math.random()*N) + 1;

    switch( result) {
        case 1 : // do something
                 break;
        case 2 : // do something
                 break;
        case 3 : // do something
                 break;
        case 4 : // do something
                 break;
        case 5 : // do something
                 break;
        case 6 : // do something
                 break;
        default : // do something
    }

}
```

기본 문제 #21

- 다음 코드의 **Local Variables** 중 **switch() X** 에 사용할 수 없는 것은?

```
public static void main(String[] args) {
    int I = 3;
    byte B = 3;
    short S = 3;
    char C = 'C';
    double D = 3.0d;
    String str = "STR";

    switch( x ) {

    }
}
```

```
double D = 3.0d;
```

조건문 (Conditional Statement)

if (_____)

변수	boolean b;
비교식	x >= y
Method Call	isEven()

switch (_____)

변수	byte, short, char, int x;
Enum	Day.MONDAY
Class Object	Byte, Short, Character, Integer, String(java 7)
Method Call	getNumber()

기본 문제 #22

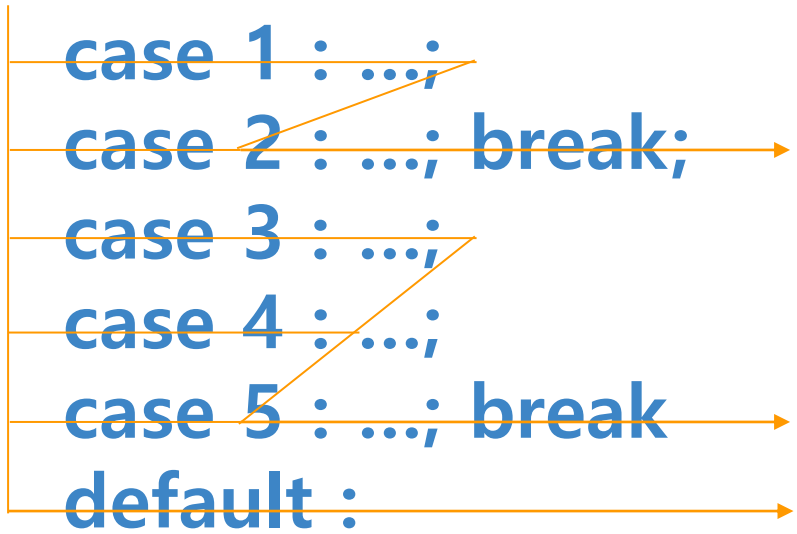
- 다음 코드의 실행 결과는?

```
public static void main(String[] args) {
    int num = 3;

    switch( num ) {
        case 1 : System.out.println(num);
        case 2 : System.out.println(num);
        case 3 : System.out.println(num);
        case 4 : System.out.println(num);
        case 5 : break;
        case 6 : break;
        default : System.out.println(num);
    }
}
```

```
3
3
```

switch - break, default

```
switch ( value )
```



기본 문제 #23

- 다음 코드의 오류가 발생한다. 그 이유는?

```
public static void main(String[] args) {
    char C  = 'A';

    switch( C ) {
        case 'A' :  // do nothing
                    break;
        case 'B' :  // do nothing
                    break;
        case 65  :  // do nothing
                    break;
    }
}
```

Duplicate Case

기본 문제 #24

- 다음 코드의 실행 결과는?

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

    int num = 4;

    if( num == 3 & isEven(num) ) {
        System.out.println("3 !!");
    }
}

static boolean isEven(int num) {
    if( num % 2 == 0 ) {
        System.out.println("Even !!");
        return true;
    }else {
        return false;
    }
}
```

```
Even !!
```

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

    int num = 4;

    if( num == 3 && isEven(num) ) {
        System.out.println("3 !!");
    }
}

static boolean isEven(int num) {
    if( num % 2 == 0 ) {
        System.out.println("Even !!");
        return true;
    }else {
        return false;
    }
}
```

기본 문제 #25

- 다음 코드의 결과는?

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

    int N = 6;

    boolean isEven = ( N % 2 == 0 ) ? true : false;

    N = ( ! isEven ) ? 10 : 20;

    System.out.println(N);
}
```

```
20
```

기본 문제 #26 -1

- 기본 문제 #18 주사위를 이용하여, 주사위를 두 번 던져서 연속적으로 짝수 또는 홀수가 나오면, **"A"**를 그렇지 않으면 **"B"** 를 출력하는 코드를 작성하시오.

Nested If 사용

```
int N = 6;

int result = (int) (Math.random()*N) + 1;
System.out.println(result);
boolean isEven = false;
if ( result % 2 == 0 ) {
    isEven = true;
}

result = (int) (Math.random()*N) + 1;
System.out.println(result);
if ( result % 2 == 0 ) {
    if( isEven ) {
        System.out.println("A");
    }else {
        System.out.println("B");
    }
}else if ( result % 2 == 1 ) {
    if( isEven ) {
        System.out.println("B");
    }else {
        System.out.println("A");
    }
}
```

```
2  2
2  1
A  B
```

기본 문제 #26 -2

- 기본 문제 #18 주사위를 이용하여, 주사위를 두 번 던져서 연속적으로 짝수 또는 홀수가 나오면, **"A"**를 그렇지 않으면 **"B"** 를 출력하는 코드를 작성하시오.

```java
int N = 6;

int result = (int) (Math.random()*N) + 1;
System.out.println(result);
boolean isEven = ( result % 2 == 0 ) ? true : false;

result = (int) (Math.random()*N) + 1;
System.out.println(result);
boolean isSame = ( ( result % 2 == 0 ) == isEven ) ? true : false;

if( isSame ) {
    System.out.println("A");
}else {
    System.out.println("B");
}
```

Java 언어의 이해에 의한 개선

2	2
2	1
A	B

기본 문제 #26 -3

- 기본 문제 #18 주사위를 이용하여, 주사위를 두 번 던져서 연속적으로 짝수 또는 홀수가 나오면, **"A"**를 그렇지 않으면 **"B"** 를 출력하는 코드를 작성하시오.

```
int N = 6;

int result1 = (int) (Math.random()*N) + 1;
System.out.println(result1);

int result2 = (int) (Math.random()*N) + 1;
System.out.println(result2);

if( ( result1 + result2 ) % 2 == 0 ) {
    System.out.println("A");
}else {
    System.out.println("B");
}
```

```
2 2
2 1
A B
```

수학적 개념에 의한 개선

반복문

기본 문제 #27

- 프로그램은 반복적으로 어떤 작업을 수행해야 할 수 있다.
  기본 문제 #18 주사위를 이용하여, 100 번 주사위를 던진, 결과의 합과 평균값을 출력하는 코드를 for 문을 이용하여 구현하시오.

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

    int N = 6;

    float result = 0;

    for( int i=0; i<100; i++ ) {
        result += (int) (Math.random()*N) + 1;
    }

    System.out.println(result);
    System.out.println(result / 100);
}
```

```
348.0
3.48
```

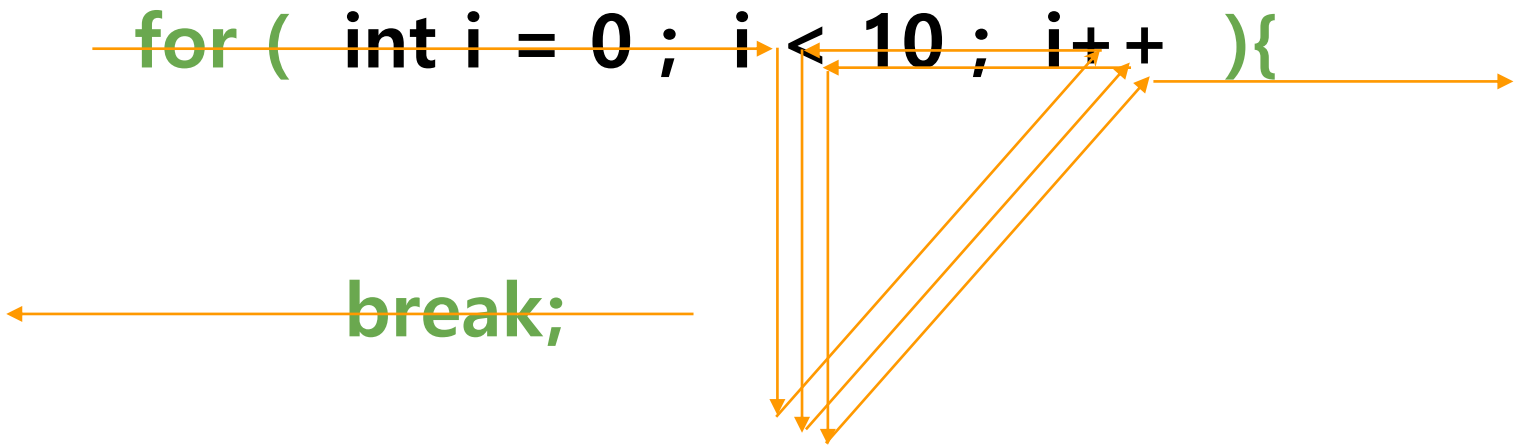
반복문 for 구성

```
for (    ___    ;    ___    ;    ___    )
```

변수 초기화	반복 조건	증감식
int i = 0;	i < 10;	i++

반복문 for 실행 및 종료

```
for ( int i = 0 ; i < 10 ; i++ ){

        break;

}
```



기본 문제 #28

- 기본 문제 #27 을 **while** 및 **do-while** 문을 이용하여 만드시오.

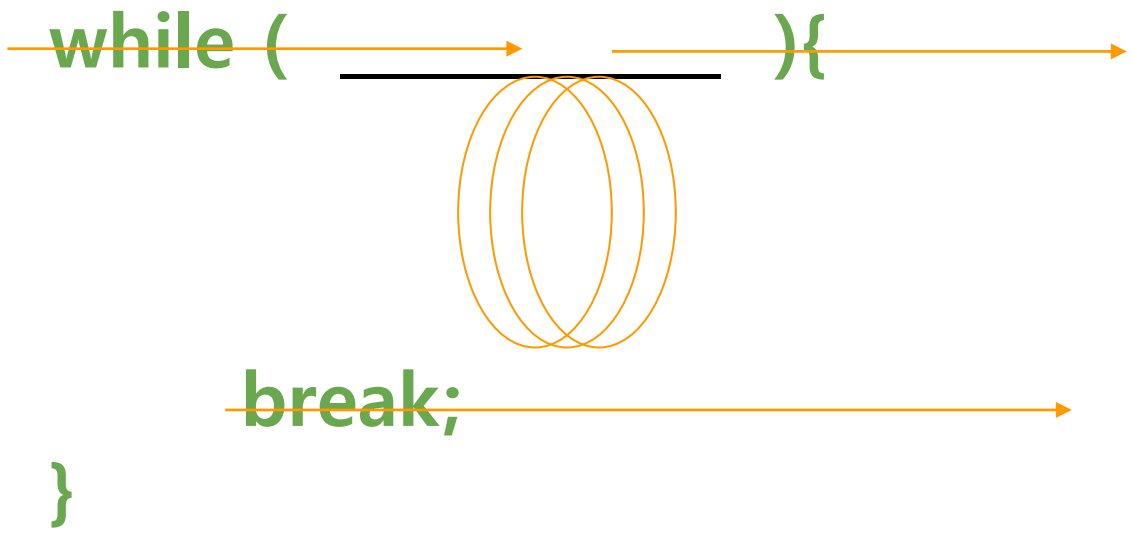
```
public static void main(String[] args) {

    int N = 6;

    float result = 0;

    int i=0;

    while(  i<100 ) {
        result += (int) (Math.random()*N) + 1;
        i++;
    }

    System.out.println(result);
    System.out.println(result / 100);
}
```

```
348.0
3.48
```

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

    int N = 6;

    float result = 0;

    int i=0;

    do {
        result += (int) (Math.random()*N) + 1;
        i++;
    }while(  i<100 );

    System.out.println(result);
    System.out.println(result / 100);
}
```

```
337.0
3.37
```

반복문 while 실행 및 종료



반복문 while 3가지 유형

```
while ( true ){
    …
    if ( … ) break;
}
```

```
while ( X ){
    … X …
    if ( X … ) break;
}
```

```
while ( m() ){
    …
}

boolean m(){
    … X …
    return true/false;
}
```

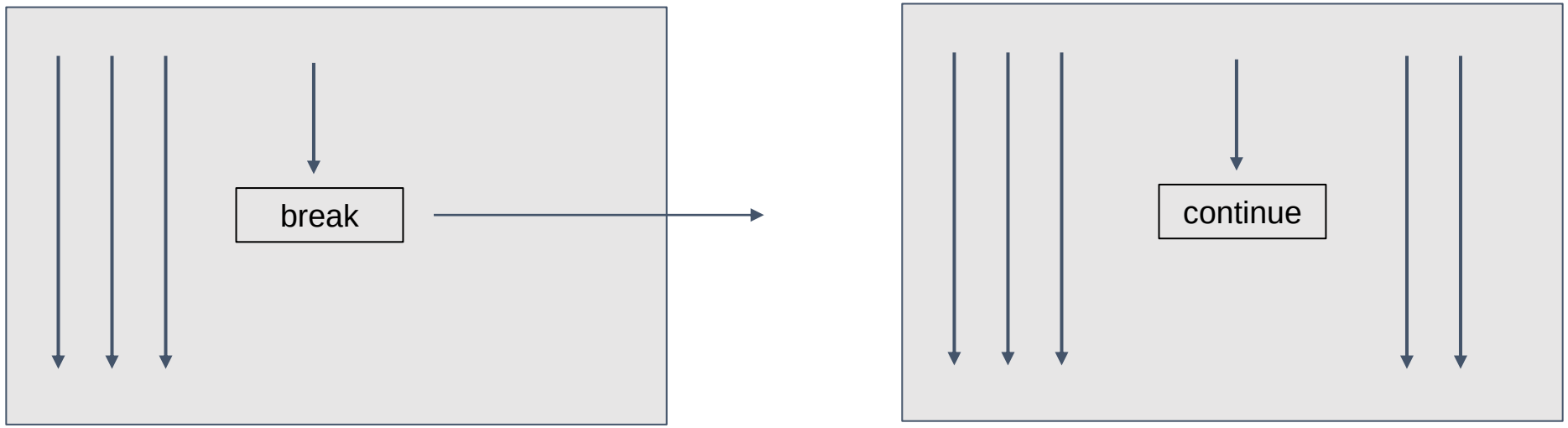
기본 문제 #29

- 프로그램은 반복문을 수행하는 도중, 특정 조건에 따라, 반복문을 중단하거나, 그 조건의 경우에만 수행하지 않고, 계속 반복문을 이어서 진행하는 경우가 있다. 이에 해당하는 **Java Keyword** 는?

1. exit, continue
2. stop, skip
3. break, skip
4. break, continue

4. break, continue

break vs continue



기본 문제 #30

- 다음 코드의 결과는?

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

    for( int i=0; i<2; i++ ) {

        for( int j=0; j<10; j++ ) {

            if( j==5 ) break;
            if( j==3 ) continue;

            System.out.println(j);
        }
    }
}
```

```
0
1
2
4
0
1
2
4
```

Nested Loop #1

독립적인 index

i j	0	1	2	3	4	5	6...
0	**0**	**1**	**2**	continue	**4**	break	해당 없음
1	**0**	**1**	**2**	continue	**4**	break	

기본 문제 #31

- 다음 코드의 결과는?

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

    for( int i=0; i<5; i++ ) {

        for( int j=0; j<i; j++ ) {

            if( j==3 ) break;
            if( j==2 ) continue;

            System.out.println(j);
        }
    }
}
```

```
0
0
1
0
1
0
1
```

Nested Loop #2

index 간섭 : j < i

i \ j	0	1	2	3	4
0					
1	0				
2	0	1			
3	0	1	continue		
4	0	1	continue	break	

해당 없음

기본 문제 #32

- 기본 문제 #18 주사위를 이용하여, 주사위를 계속 던지고 나온 값을 합을 계산한다. 단, 합이 100 을 넘으면 중지하고 그 시점의 합과 주사위를 던진 횟수를 출력하는 코드를 작성하시오.

Unexpected Break Condition

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

    int N = 6;

    int result = 0;
    int count = 0;

    while(true) {
        count++;
        result += (int)(Math.random()*N) + 1;
        if( result > 100 ) break;
    }

    System.out.println(count);
    System.out.println(result);
}
```

```
28
101
```

for vs while

for	- 예측 가능한 반복 - index 의 증감 활용
while	- 예측 가변적인 반복 - index 보다는 break, continue 활용

for-each 문은 Array 학습 이후 !!

기본 문제 #33 - 1

- **for** 반복문을 사용하여 * 문자가 아래와 같이 출력되도록 코드를 작성하세요.

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

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

    for( int i=0; i<5; i++ ) {

        for( int j=0; j<5; j++ ) {

            if( j<=i) {
                System.out.print("*");
            }
        }

        System.out.println();
    }
}
```

기본 문제 #33 - 2

- for 반복문을 사용하여 * 문자가 아래와 같이 출력되도록 코드를 작성하세요.

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

index 활용을 통한 개선

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

    for( int i=0; i<5; i++ ) {

        for( int j=0; j<=i; j++ ) {

            System.out.print("*");
        }

        System.out.println();
    }
}
```

기본 문제 #34

- 다음 코드의 결과는?

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

    for (int i = 0; i < 5; i++) {

        if( i % 2  == 0 ) {
            System.out.println( i );
        }

        if( i == 3 ) break;
    }

    for (int i = 0; i < 5; i++) {

        switch( i % 2 ) {
            case 0 : System.out.println( i );
        }

        switch( i ) {
            case 3 : break;
        }
    }
}
```

```
0
2
0
2
4
```

배열

배열 문제 #01-1

- 주사위를 이용하여, **5**번 던져서 각각의 결과 값을 저장한 후, 필요할 때 사용하려고 한다. 우선, 각각의 값을 출력하는 코드를 작성하시오. 단, **Array**를 사용하지 않고 작성하세요.

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

    int N = 6;

    int result1 = (int)(Math.random()*N) + 1;
    int result2 = (int)(Math.random()*N) + 1;
    int result3 = (int)(Math.random()*N) + 1;
    int result4 = (int)(Math.random()*N) + 1;
    int result5 = (int)(Math.random()*N) + 1;

    System.out.println(result1);
    System.out.println(result2);
    System.out.println(result3);
    System.out.println(result4);
    System.out.println(result5);
}
```

```
6
6
4
1
1
```

동일한 Type의 변수를 여러 개 사용

- 변수의 수 증가
- 코드의 길이 증가
- 반복문 적용 불가
- 변수의 수가 동적으로 결정될 경우, 사용 불가

배열(Array)로 동일 Type 변수 대신하기

- 배열이란? 동일한 타입의 데이터 여러 개를 하나의 연속된 메모리 공간에서 관리하는 것
- 요소에 접근하는 속도가 매우 빠르며 크기 변경 불가

Array 만들기 #1

- **int Type** 기준으로 배열 **(Array)** 만들기

선언	생성	개별 요소 값 할당
int[] arr;	arr = new int[5];	arr[0] = 3;
int arr[];	arr = new int[10];	arr[2] = 7;

- 변수의 타입과 저장하는 데이터의 타입?

```
int a;
int[] arr;
```

변수 a의 타입은?
변수 arr의 타입은?
arr이 저장하는 데이터의 타입은?

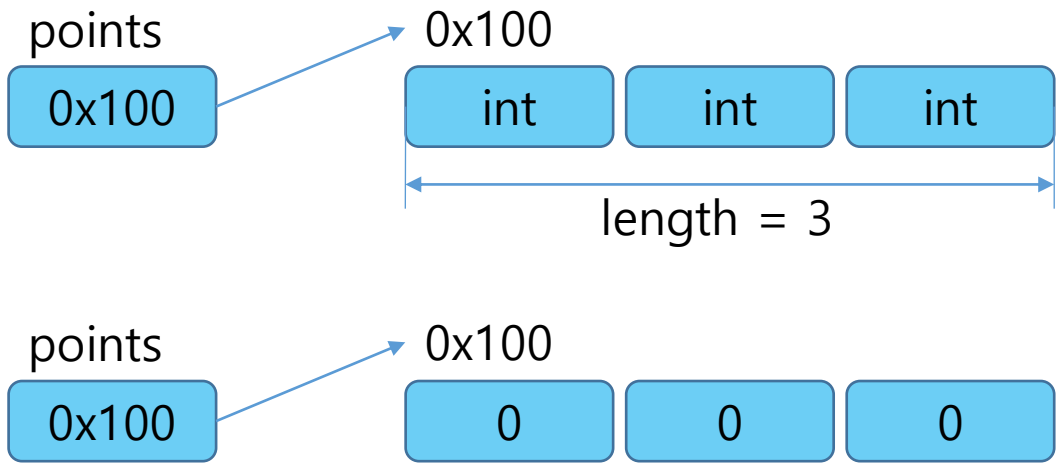
JAVA 기본 문법, 응용

배열의 생성과 초기화

- 생성
  - **'new' 키워드와 함께 배열의 데이터 타입 및 크기 지정: new data_type[length]**
  - new int[3]; int 타입의 자료 3개를 저장할 수 있는 배열을 메모리에 생성
  - int[] points = new int[3]; 생성된 배열을 points라는 변수에 할당
  - points는 메모리에 있는 배열을 가리키는 reference 타입 변수

- 초기화
  - 배열을 생성과 동시에 자료형에 대한 default 초기화 진행

자료형	기본값	비고
boolean	False	
char	'₩u0000'	공백문자
byte, short, int	0	
long	0L	
float	0.0f	
double	0.0	
참조형 변수	null	아무것도 참조하지 않음



배열 문제 #01-2

- 주사위를 이용하여, **5**번 던져서 각각의 결과 값을 저장한 후, 필요할 때 사용하려고 한다. 우선, 각각의 값을 출력하는 코드를 **반복문 없이 Array**를 사용하여 작성하시오.

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

    int N = 6;

    int [] resultArray = new int[5];
    resultArray[0] = (int)(Math.random()*N) + 1;
    resultArray[1] = (int)(Math.random()*N) + 1;
    resultArray[2] = (int)(Math.random()*N) + 1;
    resultArray[3] = (int)(Math.random()*N) + 1;
    resultArray[4] = (int)(Math.random()*N) + 1;

    System.out.println(resultArray[0]);
    System.out.println(resultArray[1]);
    System.out.println(resultArray[2]);
    System.out.println(resultArray[3]);
    System.out.println(resultArray[4]);
}
```

```
6
6
4
1
1
```

배열 문제 #01-3

- 주사위를 이용하여, **5**번 던져서 그 각각의 결과 값을 저장한 후, 필요할 때 사용하려고 한다.
  우선, 각각의 값을 출력하는 코드를 **반복문과 Array**를 사용하여 작성하시오.

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

    int N = 6;

    int [] resultArray = new int[5];

    for( int i=0; i<5; i++ ) {
        resultArray[i] = (int)(Math.random()*N) + 1;
    }

    for( int i=0; i<5; i++ ) {
        System.out.println(resultArray[i]);
    }
}
```

```
6
6
4
1
1
```

배열 문제 #02-1

- String "SSAFY" 를 이용하여, char 배열 ssafyArray를 만들고, 출력하는 코드를 작성하세요.

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

    String ssafyStr = "SSAFY";

    char[] ssafyArray = new char[ssafyStr.length()];

    for (int i = 0; i < ssafyArray.length; i++) {
        ssafyArray[i] = ssafyStr.charAt(i);
    }

    for (int i = 0; i < ssafyArray.length; i++) {
        System.out.print(ssafyArray[i]);
    }
}
```

```
SSAFY
```

Array 출력을 편리하게

- for 문을 통한 출력대신 Arrays.toString()

toString

```
public static String toString(char[] a)
```

Returns a string representation of the contents of the specified array. The string representation consists of a list of the array's elements, enclosed in square brackets ("[]"). Adjacent elements are separated by the characters ", " (a comma followed by a space). Elements are converted to strings as by String.valueOf(char). Returns "null" if a is null.

Parameters:

a - the array whose string representation to return

Returns:

a string representation of a

Since:

1.5

java 8 api documents

Array 만들기 #2

- 생성과 동시에 할당한 값으로 초기화
  - int[] b = {1, 3, 5, 6, 8};
  - int[] c = new int[] {1, 3, 5, 6, 8};
- 선언 후 생성 시 초기화 주의
  - int[] points;

    points = {1, 3, 5, 6, 8}; // 컴파일 오류
  - int[] points;

    points = new int[] {1, 3, 5, 6, 8}; // 선언할 때는 배열의 크기를 알 수 없을 때

배열 문제 #03

- 다음은 배열의 다양한 생성 코드이다. 잘못된 것은?

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

    // 1
    int [] intArray = new int[3];
    intArray[2] = 10;

    // 2
    char charArray[] = { 'S','S','A','F','Y' };

    // 3
    String [] strArray = { "S","S","A","F","Y" };

    // 4
    int[] intArray2;
    intArray2 = { 1, 2, 3, 4, 5 };
}
```

4.
int[] intArray2;
intArray2 = { 1, 2, 3, 4, 5 };

{} 를 통한 배열 생성은 변수 선언과 동시에 !!

배열 .length, default value

- **Array** 객체의 **length Attribute** 로 길이를 표현

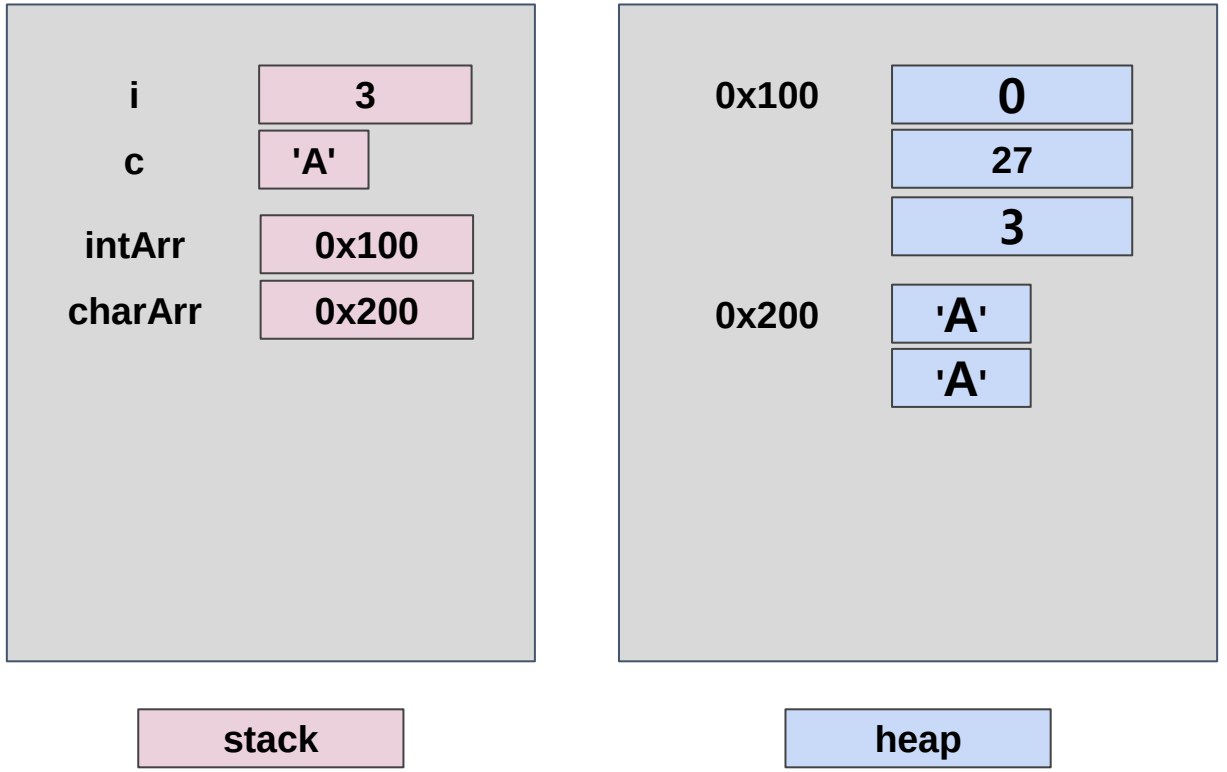
```
int [] intArray = { -3, -1, 1, 3, 5 };
System.out.println( intArray.length );
```

- **Array** 요소 중 값을 할당받지 않은 요소는 **default value** 값을 가진다.

```
int intArray [] = new int[3];
System.out.println( intArray[0] );
```

Local Variables vs Array in Memory

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

    int i = 3;
    char c = 'A';

    int[] intArr = new int[3];
    intArr[2] = i;
    intArr[1] = 27;

    char[] charArr = new char[2];
    charArr[0] = 'A';
    charArr[1] = c;
}
```



for-each with Array

- **Java 5.0** 부터 지원
- 가독성이 개선된 반복문으로, 배열 및 **Collections** 에서 사용
- **index** 대신 직접 요소**(elements)**에 접근하는 변수를 제공
- **naturally read only (copied value)**
- **:** 사용

```
int intArray [] = { 1, 3, 5, 7, 9 };

for( int x : intArray ){
    System.out.println( x );
}
```

배열 문제 #04-1

- 기본 문제 #18 주사위를 이용하여 5번 던져서 각각의 결과 값을 저장한 후, 필요할 때 사용하려고 한다.
  우선, 각각의 값을 출력하는 코드를 Array와 length 속성 및 for-each 구문을 통해 작성하시오.

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

    int N = 6;

    int [] resultArray = new int[5];

    for( int i=0; i<resultArray.length; i++ ) {
        resultArray[i] = (int)(Math.random()*N) + 1;
    }

    for( int x : resultArray ) {
        System.out.println(x);
    }
}
```

```
6
6
4
1
1
```

배열 문제 #04-2

- 기본 문제 #18 주사위를 이용하여 5번 던져서 각각의 결과 값을 저장한 후, 필요할 때 사용하려고 한다.
  우선, 각각의 값을 출력하는 코드를 **Array**와 **length** 속성 및 **for-each** 구문을 통해 작성하시오.

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

    int N = 6;

    int [] resultArray = new int[5];

    for( int x : resultArray ) {
        x = (int)(Math.random()*N) + 1;
    }

    for( int x : resultArray ) {
        System.out.println(x);
    }
}
```

```
0
0
0
0
0
```

read only (copied value)

배열 문제 #05

- **SAFFY**의 어떤 반 학생들의 이름이 아래와 같다. 아래 이름을 **Array**로 순차적으로 저장하고, **for-each**로 출력하는 코드를 작성하시오.

1	홍길동
2	박사피
3	윤멀티
4	나자바

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

    String [] students = { "홍길동", "박사피", "윤멀티", "나자바" };

    for( String student : students ) {
        System.out.println(student);
    }
}
```

```
홍길동
박사피
윤멀티
나자바
```

배열 문제 #06

- 배열 문제 #04 코드에 이어서 '박사피' 와 '윤멀티' 의 순서를 바꾸고 다시 **for-each**로 이름을 출력하는 코드를 작성하시오.

1	홍길동
2	박사피
3	윤멀티
4	나자바

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

    String [] students = { "홍길동", "박사피", "윤멀티", "나자바" };

    for( String student : students ) {
        System.out.println(student);
    }

    String temp = students[1];
    students[1] = students[2];
    students[2] = temp;

    for( String student : students ) {
        System.out.println(student);
    }
}
```

```
홍길동
박사피
윤멀티
나자바
홍길동
윤멀티
박사피
나자바
```

배열 문제 #07

- 배열 문제 #05 코드에 이어서 새로운 학생 '신자바'를 추가하는 코드 중 틀린 것은?

1	홍길동
2	박사피
3	윤멀티
4	나안드
5	신자바

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

    String [] students = { "홍길동", "박사피", "윤멀티", "나안드" };

    System.out.println(Arrays.toString(students));

    // #1
    students = { "홍길동", "박사피", "윤멀티", "나안드", "신자바" };

    // #2
    String [] students2 = { "홍길동", "박사피", "윤멀티", "나안드", "신자바" };


    // #3
    String [] students3 = new String[5];
    System.arraycopy(students, 0, students3, 0, 4);
    students3[4] = "신자바";

}
```

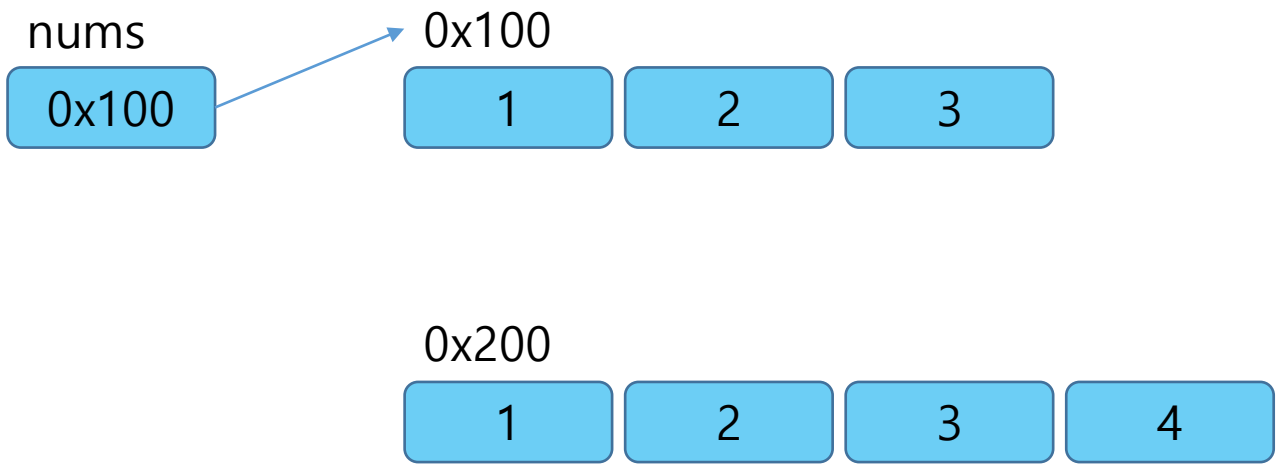
1. `students = { "홍길동", "박사피", "윤멀티", "나안드", "신자바" };`

Array Constant

Array is Immutable

- 최초 메모리 할당 이후, 변경할 수 없음.
- 개별 요소는 다른 값으로 변경이 가능하나, 삭제할 수는 없음.
- 크기를 늘리거나 줄일 수 없음.
- 변경이 필요한 경우, 새로 작성하는 것이 일반적으로 유리함.

```
int[] nums = {1, 2, 3};

nums[1] = 100;

nums = new int[] {1, 2, 3, 4};
```



System.arraycopy

- api 제공하는 배열 복사 method

arraycopy

```
public static void arraycopy(Object src,
                             int srcPos,
                             Object dest,
                             int destPos,
                             int length)
```

Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array. A subsequence of array components are copied from the source array referenced by `src` to the destination array referenced by `dest`. The number of components copied is equal to the `length` argument. The components at positions `srcPos` through `srcPos+length-1` in the source array are copied into positions `destPos` through `destPos+length-1`, respectively, of the destination array.

System.arraycopy

- api 제공하는 배열 복사 method

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

    int[] srcArray = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int[] tgtArray = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

    System.arraycopy(srcArray, 2, tgtArray, 1, 3);

    System.out.println(Arrays.toString(tgtArray));
}
```

배열 문제 #08-1

- 절대 값이 1000을 넘지 않는 정수로 구성된 intArray 배열이 아래와 같이 주어졌을 때, 최대값, 최소값을 출력하는 프로그램을 작성하세요.
- int[] intArray =
  { 3, 27, 13, 8, 235, 7, 22, 9, 435, 31, 54 };

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

    int[] intArray = { 3, 27, 13, 8, 235, 7, 22, 9, 435, 31, 54 };

    int min = 999;  // Integer.MAX_VALUE
    int max = 0;    // Integer.MIN_VALUE

    for (int i = 0; i < intArray.length; i++) {
        if( intArray[i] < min ) {
            min = intArray[i];
        }
        if( intArray[i] > max ) {
            max = intArray[i];
        }
    }

    System.out.println("Min : " + min + " Max : " + max);
}
```

```
Min : 3 Max : 435
```

배열 문제 #08-2

- 문제 08-1의 코드를
  Integer, Math Class를 이용하여
  Refactoring 해보세요.
- int[] intArray =
  {3, 27, 13, 8, 235, 7, 22, 9, 435, 31, 54};

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

    int[] intArray = { 3, 27, 13, 8, 235, 7, 22, 9, 435, 31, 54 };

    int min = Integer.MAX_VALUE;
    int max = Integer.MIN_VALUE;

    for (int i = 0; i < intArray.length; i++) {
        min = Math.min(min, intArray[i]);
        max = Math.max(max, intArray[i]);
    }

    System.out.println("Min : " + min + " Max : " + max);
}
```

```
Min : 3 Max : 435
```

충분히 큰 수 → Integer.MAX_VALUE
충분히 작은 수 → Integer.Min_VALUE
Math Class 비교 method 활용

배열 문제 #09

- intArray 배열이 아래와 같이 주어졌을 때, 각 숫자가 몇 번 사용되었는지 숫자별로 사용 횟수를 출력하세요.
  사용 안된 숫자는 0으로 출력하세요.
  (단, 숫자는 0-9 까지만 사용)

  int[] intArray =
  {3, 7, 2, 5, 7, 7, 9, 2, 8, 1, 1, 5, 3};

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

    int[] intArray = { 3, 7, 2, 5, 7, 7, 9, 2, 8, 1, 1, 5, 3 };
    int[] used = new int[10];

    for (int i = 0; i < intArray.length; i++) {
        used[intArray[i]]++;
    }

    System.out.println(Arrays.toString(used));
}
```

```
[0, 2, 2, 2, 0, 2, 0, 3, 1, 1]
```

default value : 0
배열 숫자가 used 배열의 index 로 활용

배열 문제 #10

- 1~20까지의 숫자를 사용한 intArray 배열이 아래와 같이 주어졌을 때, 사용되지 않은 숫자를 출력하세요.

  int[] intArray =
  {1, 3, 4, 7, 8, 10, 12, 15, 16, 17, 18};

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

    int[] intArray = {
        1, 3, 4, 7, 8, 10, 12, 15, 16, 17, 18
    };

    int[] used = new int[21];   // 0 : dummy

    for (int i = 0; i < intArray.length; i++) {
        used[intArray[i]]++;
    }

    for (int i = 1; i < used.length; i++) {
        if( used[i] == 0 ) {
            System.out.print(i + " ");
        }
    }
}
```

```
2 5 6 9 11 13 14 19 20
```

2차원 배열

배열 문제 #11

- 아래 표와 같이 **4x3 = 12** 개의 영역으로 나누어 **'A', 'B', 'C'** 로 등급을 나누려고 한다. 이를 **Array**를 이용하여 표현하고, 표 대로 출력하는 코드를 작성하시오.

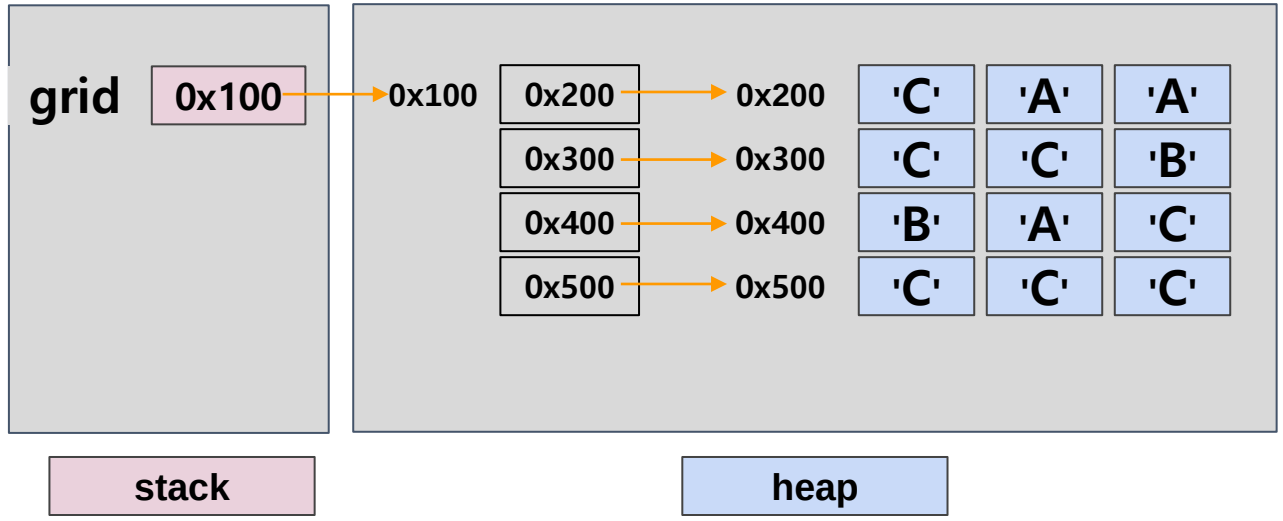
C	A	A
C	C	B
B	A	B
C	C	C

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

    char[][] grid = new char[4][3];

    grid[0][0] = 'C'; grid[0][1] = 'A'; grid[0][2] = 'A';
    grid[1][0] = 'C'; grid[1][1] = 'C'; grid[1][2] = 'B';
    grid[2][0] = 'B'; grid[2][1] = 'A'; grid[2][2] = 'C';
    grid[3][0] = 'C'; grid[3][1] = 'C'; grid[3][2] = 'C';

    for( int i=0; i<grid.length; i++ ) {
        for( int j=0; j<grid[i].length; j++ ) {
            System.out.print(grid[i][j]);
        }
        System.out.println();
    }
}
```

```
CAA
CCB
BAC
CCC
```

문제 #11 Array in Memory

```
char[][] grid = new char[4][3];

grid[0][0] = 'C'; grid[0][1] = 'A'; grid[0][2] = 'A';
grid[1][0] = 'C'; grid[1][1] = 'C'; grid[1][2] = 'B';
grid[2][0] = 'B'; grid[2][1] = 'A'; grid[2][2] = 'C';
grid[3][0] = 'C'; grid[3][1] = 'C'; grid[3][2] = 'C';
```



배열 문제 #12

- 다음은 **4x3** 의 **2**차원 배열을 만드는 방법이다. 올바르지 않은 것은?

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

    int  intArray[][] = new int [4][3];
    int [] intArray2[] = new int [4][3];
    int [][] intArray3 = new int [4][3];
    int [][] intArray4 = new int [4]{1,2,3};
    int [][] intArray5 = new int[][] {{1,2,3},{1,2,3},{1,2,3},{1,2,3}};
}
```

```
int [][] intArray4 = new int [4]{1,2,3};
```

Type mismatch: cannot convert from int[] to int[][]

1 quick fix available:

Change type of 'intArray4' to 'int[]'

Press 'F2' for focus

생략하면 Array Constants

2차원 Array 만들기 #1

- **int Type 기준으로 4x3 배열 (Array) 만들기**

선언	생성	할당
int [][] intArray;	intArray = new int[4][3];	intArray[0][2] = 3;
int intArray [][];		
int [] intArray [];		

- **1차 Array 및 2차 Array 모두 Object**

2차원 Array 만들기 #2

- **int Type** 기준으로 **4x3** 배열 **(Array)** 과 값을 동시에 만들기

선언, 생성, 할당 동시에
int [][] intArray = { { 0, 1 ,2 }, { 0, 1, 2 }, { 0, 1, 2 }, { 0, 1, 2 } }; // int intArray [][], int [] intArray []

- **Array** 는 **{ }** 안에 **,** 과 **{ }** 을 이용해서 선언과 동시에 값을 할당
- **Array Constants** 는 **동일 변수에 새로운 배열 할당 불가**

2차원 Array 만들기 #3

- **int Type 기준으로 4x? 배열 (Array) 만들기**

1,2차 선언 / 1차 생성
int [][] intArray = new int[4][]; // int intArray [][], int [] intArray []

- **1차 Array 만 생성 후, 필요에 따라 2차 배열을 생성함**

2차 생성

intArray[1] = new int[2];
intArray[0] = new int[4];
intArray[2] = {1,2,3}; (**X**)

배열 문제 #13

- 다음은 **5x5** 의 **2**차원 배열을 나타낸다. 각 항목의 숫자 중 **3**의 배수의 수와 합을 구하는 코드를 작성하시오.

2	3	1	4	7
8	13	3	33	1
7	4	5	80	12
17	9	11	5	4
4	5	91	27	7

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

    int[][] grid = {
            {2, 3, 1, 4, 7},
            {8,13, 3,33, 1},
            {7, 4, 5,80,12},
            {17,9,11, 5, 4},
            {4, 5,91,27, 7}
    };

    int count = 0;
    int sum = 0;

    for( int i=0; i<grid.length; i++ ) {
        for( int j=0; j<grid[i].length; j++ ) {
            if( grid[i][j] % 3 == 0 ) {
                count++;
                sum += grid[i][j];
            }
        }
    }

    System.out.println(count);
    System.out.println(sum);
}
```

```
6
87
```

java.util.Scanner

- API Doc 로 살펴보기

https://docs.oracle.com/javase/8/docs/api/index.html?overview-summary.html

- .hasNext()
- .next()
- .hasNextXXX()
- .nextXXX()
- .nextLine()
- .nextChar() (X) ← .next().charAt(0)

InputStreamReader 와
BufferedReader 를 이용하면 더욱 빠르게
사용자의 입력을 처리할 수 있습니다.

알고리즘 수업시간에 좀 더 자세히
학습해 보세요.

배열 문제 #14

- 다음은 **4x4** 의 **2**차원 배열을 나타낸다. **Scanner** 를 이용해서 사용자의 입력으로 아래의 배열을 구성하고 출력하는 코드를 작성하시오. 사용자의 입력은 아래와 같이 제공된다.

2	3	1	4
1	X	3	2
3	4	X	X
X	4	1	5

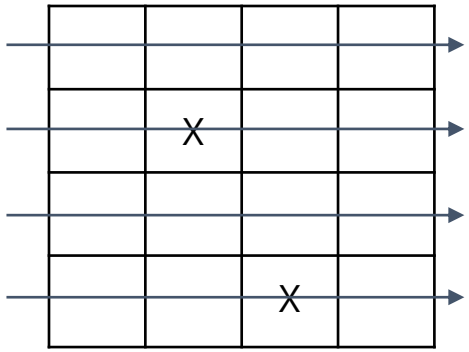
```
2 3 1 4
1 X 3 2
3 4 X X
X 4 1 5
```

사용자 입력

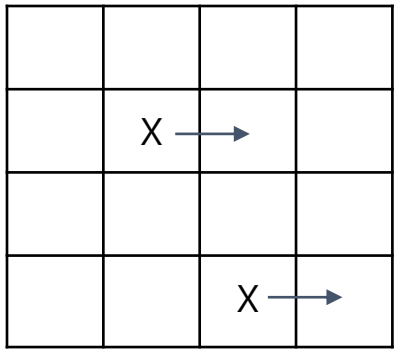
```
public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);
    char[][] grid = new char[4][4];

    for( int i=0; i<4; i++ )
        for( int j=0; j<4; j++ )
            grid[i][j] = sc.next().charAt(0);

    for( int i=0; i<4; i++ ) {
        for( int j=0; j<4; j++ ) {
            System.out.print(grid[i][j]);
        }
        System.out.println();
    }
    sc.close();
}
```

```
2 3 1 4
1 X 3 2
3 4 X X
X 4 1 5
2314
1X32
34XX
X415
```

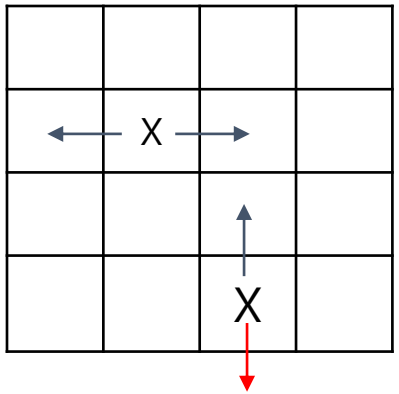
Array 순회 / 탐색



전체 중 X를 만나면



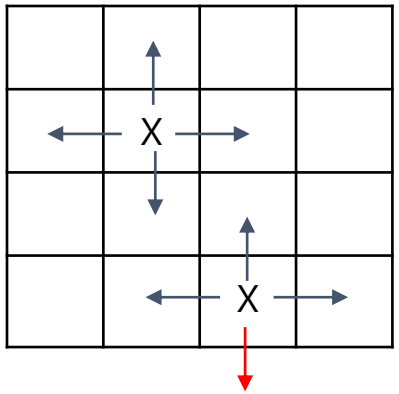
X가 움직이면서



X 주변 탐색

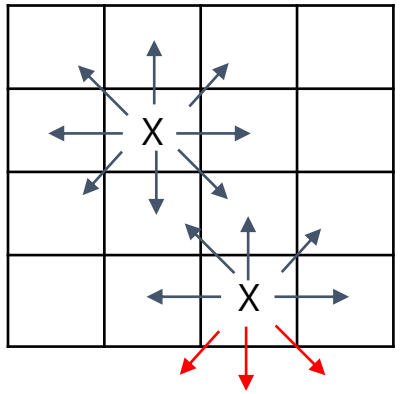
좌/우

상/하



X 주변 탐색

4방



X 주변 탐색

8방

특정 좌표로부터 주변을 탐색하는 경우, 배열의 범위를 벗어나지 않기 위한 코드 필요

배열 문제 #15

- 다음은 **4x4** 의 **2**차원 배열을 나타낸다. **X** 로 표시된 항목의 좌우 숫자의 합을 구하는 코드를 작성하시오.

2	3	1	4
1	**X**	3	2
3	4	**X**	**X**
X	4	1	5

```
2 3 1 4
1 X 3 2
3 4 X X
X 4 1 5
12
```

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

    Scanner sc = new Scanner(System.in);
    char[][] grid = new char[4][4];

    int sum = 0;

    for( int i=0; i<4; i++ )
        for( int j=0; j<4; j++ )
            grid[i][j] = sc.next().charAt(0);

    for( int i=0; i<4; i++ )
        for( int j=0; j<4; j++ )
            if( grid[i][j] == 'X') {
                if( j-1 >= 0 && grid[i][j-1] != 'X' ) sum += grid[i][j-1] - '0';
                if( j+1 <  4 && grid[i][j+1] != 'X' ) sum += grid[i][j+1] - '0';
            }

    System.out.println(sum);
    sc.close();
}
```

배열 문제 #16

- 다음은 **4x4** 의 **2**차원 배열을 나타낸다. **X** 로 표시된 항목의 **상하**좌우 숫자의 합을 구하는 코드를 작성하시오. **단, 합을 구할 때, 이미 사용된 숫자는 다시 사용하지 않음**

위의 X 와 우측의 X 에 모두 인접하지만, 한번만 더함.

2	3	1	4
1	**X**	3	2
3	4	**X**	**X**
X	4	1	5

배열 문제 #16 - 코드

2	3	1	4
1	X	3	2
3	4	X	X
X	4	1	5

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

    Scanner sc = new Scanner(System.in);
    char[][] grid = new char[4][4];
    boolean[][] used = new boolean[4][4];

    int sum = 0;

    for( int i=0; i<4; i++ )
        for( int j=0; j<4; j++ )
            grid[i][j] = sc.next().charAt(0);
```

```
    for( int i=0; i<4; i++ )
        for( int j=0; j<4; j++ )
            if( grid[i][j] == 'X') {
                if( i-1 >= 0 && grid[i-1][j] != 'X' && ! used[i-1][j] ) {
                    sum += grid[i-1][j] - '0';
                    used[i-1][j] = true;
                }
                if( i+1 < 4 && grid[i+1][j] != 'X' && ! used[i+1][j] ) {
                    sum += grid[i+1][j] - '0';
                    used[i+1][j] = true;
                }
                if( j-1 >= 0 && grid[i][j-1] != 'X' && ! used[i][j-1] ) {
                    sum += grid[i][j-1] - '0';
                    used[i][j-1] = true;
                }
                if( j+1 < 4 && grid[i][j+1] != 'X' && ! used[i][j+1] ) {
                    sum += grid[i][j+1] - '0';
                    used[i][j+1] = true;
                }
            }

    System.out.println(sum);
    sc.close();
}
```

사용 여부 관리, 사방 탐색

배열 문제 #17

- 배열은 **index** 로 요소**(element)**에 접근한다. 만약, **index** 의 범위가 벗어나면 발생되는 예외**(Exception)**은**?**

java.lang.ArrayIndexOutOfBoundsException

api doc 에서 확인해 보세요

Version Control Overview

Version Control System

버전 관리(version control, revision control),
소스 관리(source control),
소스 코드 관리(source code management, SCM)
와 동일한 용어

여러 사람이 작업할 경우 코드 관리 및 변경사항을 추적하기 위한 용도

주요 Tool

VSS(Visual Source Safe), CVS (Concurrent Versions System), SVN (Subversion)

주요 용어

Repository,
master/slave, main/secondary,
branch, merge, conflict
checkout,
checkin, commit

GIT Overview

컴퓨터 파일의 변경사항을 추적하고
여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한
분산 버전 관리 시스템

2005년 리누스 토발즈가 개발

GIT Overview

setup git bash

gitlab repository project 생성

git init

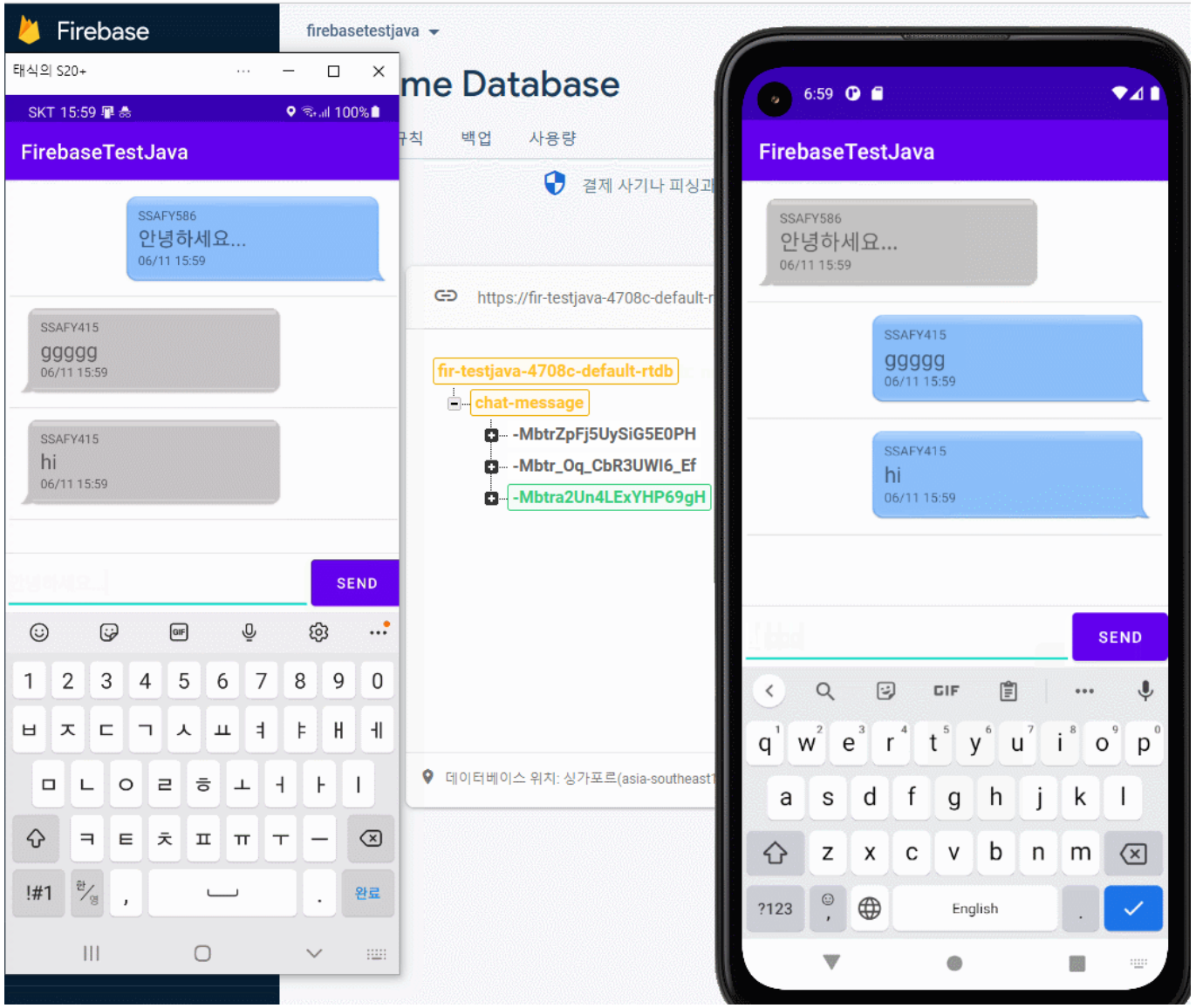
git clone, commit, pull, push

이클립스를 이용한 git 관리 실습

간단한 code conflict 실습 및 해결

Firebase RDBMS를 활용한 단톡방 만들기

Firebase RDBMS를 활용한 단톡방 만들기



Start Camp


SAMSUNG
SW
ACADEMY
FOR
YOUTH