CURSED JAVA

이거는 말이G...

개요

Java의 개요

▼ JAVA의 특징

1. 객체지향프로그래밍 언어
2. 분산 네트워크 기술 지원
3. 간단한 코드 작성 가능
4. 다중 Thread 지원(웹에선 사용하지않는다고하는데 볼 필요가 있음 ++ 따로공부)
5. 보안기능 지원
6. 플랫폼 독립적 (운영체제와 상관없이 실행 through JVM)

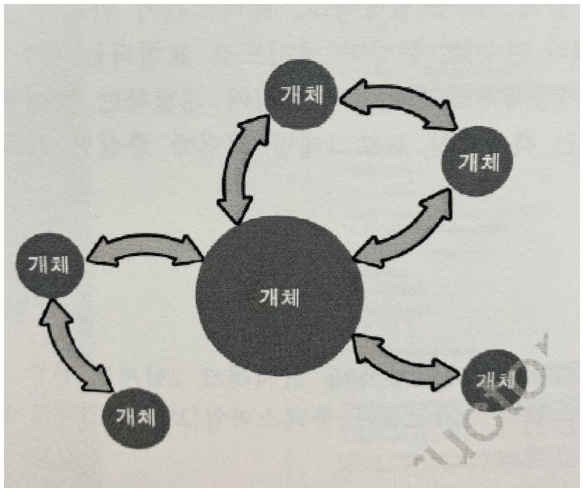
▼ JVM(Java Vritual Machine)

1. 실제 하드웨어에 소프트웨어가 설치되어 구현된 가상 CPU
2. 플랫폼에 독립적인 코드인 바이트코드를 실행

3. 소프트웨어나 하드웨어 형태로 구현
4. JDK 또는 브라우저에서 지원

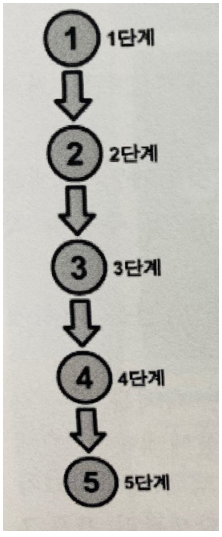
▼ 객체지향 프로그래밍

- 다른 객체와 상호작용 할 수 있는 재사용 가능한 객체를 이용하여 문제를 해결하는 방법



▼ ↔ 절차적 프로그래밍

순서에 중점을 두는 개발방법, 코드가 짜여진대로 순서있게 진행 (구조나 특징이 변경되면 전체코드에 영향을 줌)



식별자(Identifier)

개발자가 키보드를 사용하여 입력시킨 이름(클래스이름, 변수이름, 메서드 이름등을 지정할 때 사용)

▼ 규칙

첫 문자는 영문자 (_ , $ 는 예외)

길이제한X

대소문자 구분!

▼ 종류

1. 시스템 정의 식별자
    - 시스템이 필요에 의해 먼저 정의한 식별자, ==예약어 / 키워드
    - 개발자 지정 식별자로 사용 불가

분류	예약어
기본 데이터 타입	boolean, byte, char, short, int, long, float, double
접근 지정자	private, protected, public
클래스와 관련된 것	class, abstract, interface, extends, implements, enum
객체와 관련된 것	new, instanceof, this, super, null
메소드와 관련된 것	void, return
제어문과 관련된 것	if, else, switch, case, default, for, do, while, break, continue
논리값	true, false
예외 처리와 관련된 것	try, catch, finally, throw, throws
기타	transient, volatile, package, import, synchronized, native, final, static, strictfp, assert

2. 사용자 정의 식별자
    - 개발자의 필요에 의해 정의한 식별자
    - 클래스명, 변수명, 메소드명, 상수 사용시 지정 가능

구분	정의 규칙	사용 예

클래스	❑첫 문자는 항상 대문자로 표현
	❑하나 이상의 단어가 합쳐질 때는 각 단어의 첫 문자들만 대문자로 표현 (Camel 표기법)
	❑의미있는 명사형으로 지정.
		class JavaTest{
		...;
		}

변수와 메서드	❑첫 문자는 항상 소문자로 표현
	❑하나 이상의 단어가 합쳐질 때는 두 번째부터 오는 단어의 첫 문자들만 대문자로 표현 (Camel 표기법)
	❑ 변수는 의미있는 명사형으로 , 메소드는 의미있는동사형으로 지정.
		String itLand;
		public void getTest(){
		...;
		}

상수	❑모든 문자를 대문자로 표현 ❑하나 이상의 단어가 합쳐질 때 공백 필요 시 under score(_)를 사용하여 연결한다. ❑ 의미있는 명사형으로 지정.	int javaTest= 10; final int JAVA_TEST = 20;

데이터형(Data Type)

자바언어가 처리할 수 있는 데이터의 종류

▼ 종류

1. 기본 데이터형(Primitive Date Type : PDT)

▼ 논리형 - boolean

- 참과 거짓 표현
- 무조건 소문자

▼ 예시

```
boolean result = false;
boolean flag = true;
```

▼ 문자형 - char

- 하나의 문자
- 16비트 유니코드 값을 표현
- 정수형에 속함
- Single quotes ' ' 사용

▼ 예시

```java
char c = 'A';
char c1 = '홍'
```

```java
// 문자데이터가 정수형으로 변경되어 연산되는 경우
char c = 'A';
System.out.println(c+1); //  A = 65, 값 66

char c1 = 'a';
System.out.println(c1+1); // a = 97, 값 98
```


String name = "홍길동"; 처럼 문자열은 Double quotes(" ") 사용

▼ 정수형 - byte, short, int, long

- 기본형은 int
- long은 대문자 L · 소문자 l 을 정수값 뒤에 붙여서 int 와 구별

▼ 예시

```java
int age = 24;
long num = 100L;
```

▼ 실수형 - float, double

- 기본형은 double
- 소수점을 가진 숫자
- float사용시 대문자 F ·소문자 f 를 실수값 뒤에 붙여 double과 구별

▼ 예시

```java
float xyz = 3.12f;
float x1 =  3.15F;
double abc = 43.2;
```

[표 2-5] 기본 자료형의 종류

자료형	키워드	크기	기본값	표현 범위
논리형	boolean	1bit	false	true 또는 false(0과 1이 아니다)
문자형	char	2byte	\u0000	0 ~ 65,535
정수형	byte	1byte	0	-128 ~ 127
	short	2byte	0	-32,768 ~ 32,767
	int	4byte	0	-2,147,483,648 ~ 2,147,483,647
	long	8byte	0	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
실수형	float	4byte	0.0	-3.4E38 ~ +3.4E38
	double	8byte	0.0	-1.7E308 ~ +1.7E308

2. 참조 데이터형(Reference Data Type : RDT)
    - 위 8개의 기본형에 속하지 않는 모든 데이터
    - ex) 클래스, 배열, 인터페이스

```
String name = "홍길동";
int [] arr = {1, 2, 3, 4};
Student stu = new Student("유관순", 17, "서울");
```

리터럴(Literal)

자바언어가 처리하는 실제 데이터

▼ 종류

▼ 문자형

- 하나의 문자데이터 Single quotes ' ' 사용
- 유니코드 표현시
- escape 문자 사용시

```java
//escape문자
System.out.println("aaa\tbbb"); //tab 기능 4칸 띄어쓰기
System.out.println("aaa\nbbb"); //계행 다음줄
System.out.println("aaa\'bbb"); // 작은따옴표 '
System.out.println("aaa\"bbb"); // 큰따옴표 "
System.out.println("aaa\\bbb"); // \역슬래쉬
```

이스케이프 문자	용도	유니코드
'\t'	수평 탭	0x0009
'\n'	줄 바꿈	0x000a
'\r'	리턴	0x000d
'\"'	" (큰따옴표)	0x0022
'\''	' (작은따옴표)	0x0027
'\\'	\	0x005c

▼ 정수형

- 정수데이터
- 10진수, 8진수, 16진수로 표현 가능

▼ 실수형

- 소수점을 가진 실수데이터
- 10진수, 8진수, 16진수로 표현 가능

```
3.14 : 일반적인 표현방식
6.02E23 : 지수를 이용한 표현방식 (큰 실수데이터사용시)
2.718F : 간단한 float형
123.4E + 306D : double형을 명시한 큰 실수데이터
```

▼ 논리형

- 참 or 거짓 표현시 사용

변수(Variables)

Literal을 저장하기 위한 용도

▼ 특징

1. 복수개의 값이 아닌 단, 하나의 값만 저장 가능(복수값 → 배열, 컬렉션 사용)
2. 한가지 타입만 저장 가능

▼ 문법

- 동일한 변수이름으로 중복 선언 불가능

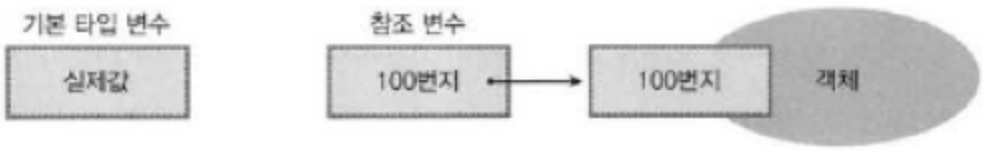
```
//기본문법

  변수명 = 값;

int sum ; //기본형 변수
String name; //참조형 변수
int age = 10,height,weight; //권장 안함
```

▼ 사용법·예시

```
//홍길동 20 서울 성별(남) 결혼여부(false) 키(185.63) 체중(78.25)

//1. 변수 선언
String name;
int Age;
String address;
char gender;
boolean isMarried;
float height;
double weight;


//2. 변수에 값 초기화(값을 맨 처음 설정하는 작업) 사용은 반드시 초기화 후
name = "홍길동";
age = 20;
address = "서울";
gender = '남';
isMarried = false;
height = 185.63f;
weight = 78.25;

//3. 사용- 출력
System.out.println(name);
System.out.println(age);
System.out.println(address);
System.out.println(gender);
System.out.println(isMarried);
```

```
System.out.println(height);
System.out.println(weight);
```

- 기본형 변수와 참조형 변수의 차이점



연산자(Operator)

연산을 수행하는 기호

▼ 종류 및 우선순위

▼ 증감연산자

피연산자에 저장된 값을 1 증가 또는 감소

```
/*
증가 연산자 ( ++ ) : 피연산자의 값을 1증가 시킴
감소 연산자 ( -- ) : 피연산자의 값을 1감소 시킴

전위형 : 값이 참조되기 전 증가 -> j = ++i;
후위형 : 값이 참조 된 후 증가 -> j = i++;

**
'++i' 또는 'i++'처럼 증감연산자가 수식이나 메소드호출에 포함되지않고
독립적인 하나의 문장으로 쓰인 경우 전위형/후위형 차이가 없음.

*/
```

▼ 예시

```
int i = 5;
int j = 5;

System.out.println(i++);  // i = 5 -> 이후 출력시 6
```

```
System.out.println(++j); // j = 6

System.out.println("i = " + i + ", j = " + j); // i = 6, j = 6
```

▼ 산술연산자

사칙연산자와 나머지 연산자

▼ 예시

```
System.out.println(10+5); // 15
System.out.println(5-2); // 3
System.out.println(4*3); // 12
System.out.println(14/3); // 4
System.out.println(14/3.0); // 4.66
System.out.println(3.2/2); // 1.6
System.out.println(5%3); // 2
```

▼ 시프트연산자

아직 몰라요!

▼ 비교연산자

두 연산자의 비교

▼ 예시

```
int a = 10;
int b = 5;

System.out.println(a == b); // false
System.out.println(a > b); // true
System.out.println(a >= b); // true
System.out.println(a < b); // false
System.out.println(a <= b); // false
System.out.println(a != b); // true
```

▼ 비트연산자

아직 몰라요!

▼ 논리연산자

boolean → true or false 반환

▼ 예시

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


System.out.println(a && a); // true
System.out.println(a && b); // false
System.out.println(b && a); // false
System.out.println(b && b); // flasse

System.out.println(a || a); // true
System.out.println(a || b); // true
System.out.println(b || a); // true
System.out.println(b || b); // false

System.out.println(!a); // false
System.out.println(!b); // true
```

▼ 조건(3항)연산자

▼ 기본문법

```
변수 = (조건식)? 값1 : 값2

**중첩 가능
```

▼ 예시

```java
//a, b, c를 비교하는 3항연산을 한문장으로 하시오.
int a = 10;
int b = 20;
int c = 30;

int result = (a > b)? ((a > c)? a : c) : ((b > c)? b : c);
System.out.println(result); // result = 30;
```

▼ 대입연산자

변수에 새로운 값을 저장하는 경우 사용 됨

▼ 예시

```java
//a, b, c를 비교하는 3항연산을 한문장으로 하시오.
int a = 10;
int b = a;
System.out.println(b); // b = 10

b += a;
System.out.println(b); // b = b + a -> 20

b -= a;
System.out.println(b); // b = b - a -> 10

b *= a;
System.out.println(b); // b = b * a -> 100

b /= a;
System.out.println(b); // b = b / a -> 10

b %= a;
System.out.println(b); // b = b % a -> 0
```

[표 2-8] 연산자의 종류와 우선순위

종류	연산자	우선순위
증감 연산자	++, --	1순위
산술 연산자	+, -, *, /, %	2순위
시프트 연산자	>>, <<, >>>	3순위
비교 연산자	>, <, >=, <=, == , !=	4순위
비트 연산자	&, |, ^, ~	~만 1순위, 나머지는 5순위
논리 연산자	&&, ||, !	!만 1순위, 나머지는 6순위
조건(삼항) 연산자	?, :	7순위
대입 연산자	=, *=, /=, %=, +=, -=	8순위

▼ Short cut circuit 연산자

[표 2-13] 논리 연산자의 종류 (2)

연산자	설명
&&	선조건이 true일 때만 후조건을 실행하며 선조건이 false이면 후조건을 실행하지 않는다.
||	선조건이 true이면 후조건을 실행하지 않으며 선조건이 false일 때만 후조건을 실행한다.

분기문

특정조건에 따라서 실행되는 문

▼ 종류

▼ IF

주어진 조건에 만족하는 경우에만 특정문장을 수행

▼ 기본문법

```
문장 1;
if(조건식){
  문장 2;
}
문장 3;

**실행문장이 하나일 경우
문장 1;
if(조건식)
  문장 2;
문장 3
```

▼ 예시

```
int n = 10; // 문장 1
if(n > 5){ // 조건식
  System.out.println(n + "은 5보다 크다."); // 문장 2
}
System.out.println("프로그램 종료"); // 문장 3
```

▼ IF

조건에 따라 실행되어야 하는 문장이 서로 다른 경우 사용

▼ 기본문법

```
문장 1;
if(조건식){
  문장 2;
}else{
  문장 3;
}
[문장 4;]
```

▼ 예시

```
int n = 9; // 문장 1
if(n % 2 == 1){ // 조건식
  System.out.println(n + "은 홀수이다."); // 문장 2
}else{
  System.out.println(n + "은 짝수이다."); // 문장 3
}
```

▼ 다중 IF-ELSE문

여러번 조건을 비교해야하는 문장에 사용

▼ 기본문법

```
if(조건식 1){
  문장 1;
}else if(조건식 2){
  문장 2;
}else if(조건식 3){
  문장 3;
}else{
  문장 n;
}
```

▼ 예시

```
int score = 88;
char grade;
```

```java
if(score >= 90 ){
  grade = 'A';
}else if (score >= 80){
  grade = 'B';
}else if (score >= 70){
  grade = 'C';
}else if (score >= 60){
  grade = 'D';
}else {
  grade = 'F';
}
System.out.println("학점은 " + grade);
```

▼ Switch문

일치하는 값 비교시 사용

▼ 기본문법

```java
switch(표현식){ //** 지정가능한 타입 (byte, short, int, char, String, enum)
  case 값1 : 문장1; break;
  case 값2 : 문장2; break;
  case 값3 : 문장3; break;
  case 값n : 문장n; break;
  default : 문장 n+1;
}
```

▼ 예시

```java
int score = 88;
char grade;

switch(score){
case 90 : grade = 'A'; break;
case 80 : grade = 'B'; break;
case 70 : grade = 'C'; break;
case 60 : grade = 'D'; break;
default : grade = 'F';
}
System.out.println("학점은 " + grade);
```

```java
String month = "10월";

switch(month){
case "1월" : month = "겨울"; break;
case "3월" : month = "봄"; break;
```

```
case "7월" : month = "여름"; break;
case "10월" : month = "가을"; break;
}
```