SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# 삼성 청년 SW 아카데미

Java

SAMSUNG SW ACADEMY FOR YOUTH

Programming Language

# Java 기본 문법

- Java 기본
- 변수와 자료형
- 연산자
- 제어문(조건문 & 반복문)

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# Java 기본

# 운영체제와 프로그램이란 ?



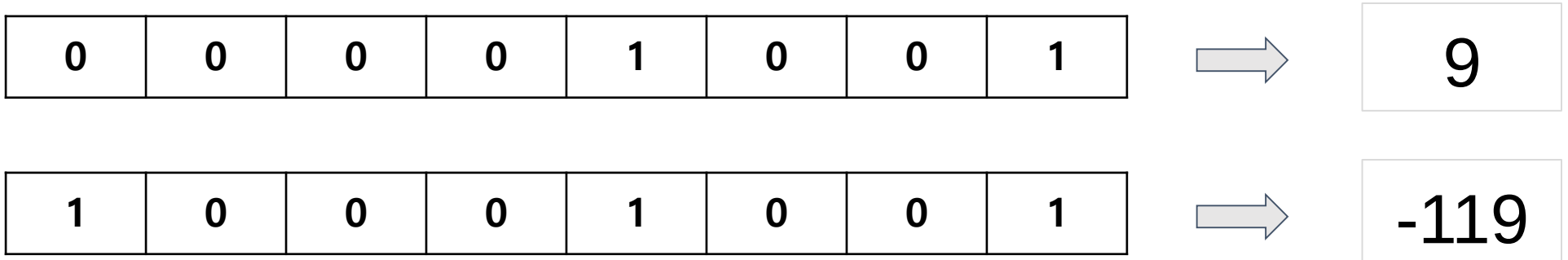
- 프로그램 (Program) :
  컴퓨터에서 실행될 때 특정 작업(specific task)을 수행하는 일련의 명령어들의 모음(집합)

- 운영체제 (Operating System, OS) :
  시스템 하드웨어를 관리할 뿐 아니라 응용 소프트웨어를 실행하기 위하여 하드웨어
  추상화 플랫폼과 공통 시스템 서비스를 제공하는 시스템 소프트웨어

# 컴퓨터의 자료표현



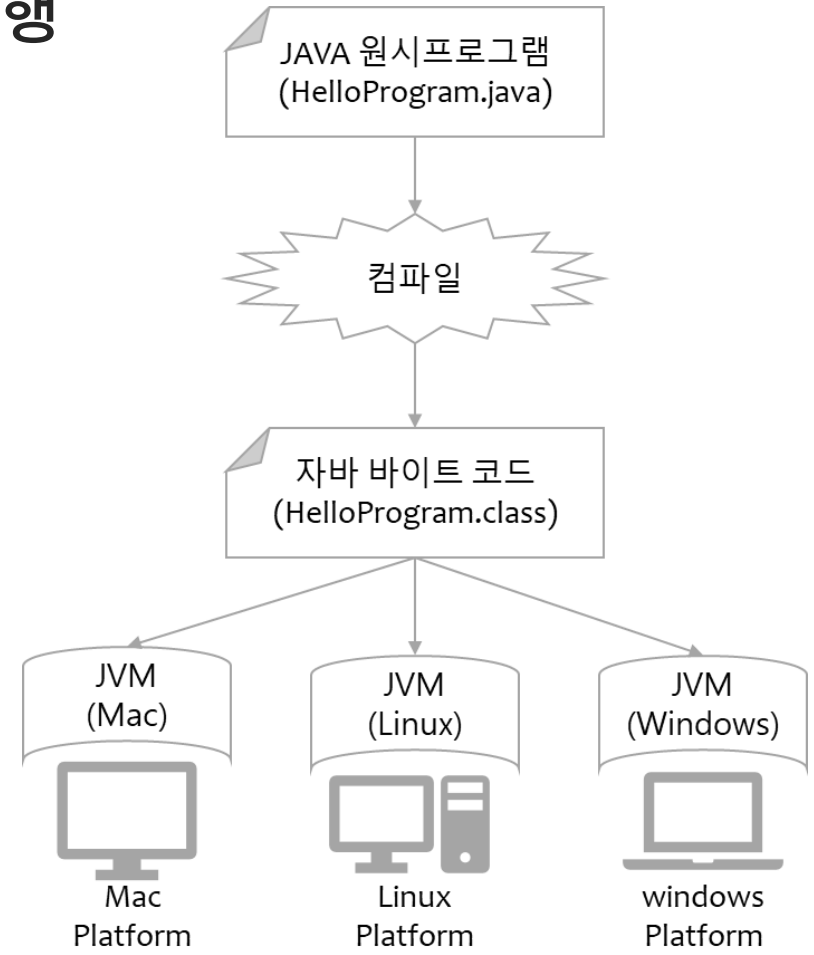
- 비트 (Bit)
- 바이트 (Byte)
- 2진수 (Binary)



# 자바 가상 머신(JVM, Java virtual machine)이란 ?



- 자바 바이트코드를 실행할 수 있는 주체
- 자바 바이트코드는 플랫폼에 독립적이며 모든 JVM은 자바 가상 머신 규격에 정의된 대로 자바 바이트코드를 실행



# Hello SSAFY



- Hello SSAFY를 출력하여 보자.

```
import java.lang.*;
public class Hello {
        public static void main(String[] args) {
                System.out.println("Hello SSAFY!!!");
        }
}
```

[Hello.java]

```
C:₩Temp>javac Hello.java
```

```
C:₩Temp>java Hello
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# Java intro

# main method



- 실행 명령인 java 를 실행 시 가장 먼저 호출 되는 부분
- 만약, Application 에서 main() 메소드가 없다면 절대로 실행 될 수 없음
- Application의 시작 → 특정 클래스의 main( ) 실행
- 형태 (고정된 형태)

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

# 주석 (Comment)



- // 내용 : 해당 기호가 있는 위치부터 그 줄 끝까지 주석처리
- /* 내용 */ : 해당 범위의 내용 주석처리
- /** 내용 */ : Documentation API를 위한 주석 처리

```
//해당 줄을 주석처리 합니다.


/*
 해당 범위의 내용을 주석처리 합니다.
 */


/**
 * Documentation API를 위한 주석처리 입니다.
 */
```

# 출력문



- print
- println
- printf
  - %d : 정수
  - %f : 실수
  - %c : 문자
  - %s : 문자열

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

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# 변수와 자료형

# 변수(Variable)

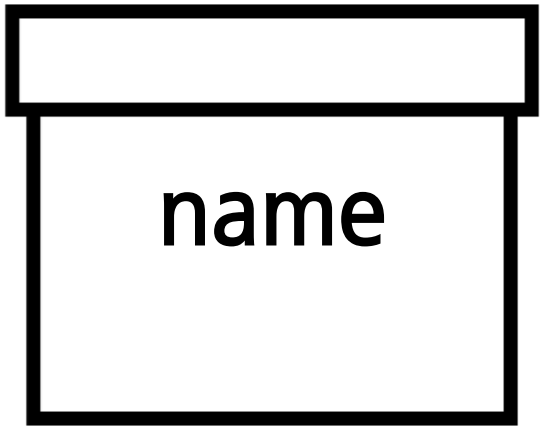


## 정의

- 데이터를 저장할 메모리의 위치를 나타내는 이름
- 메모리 상에 데이터를 보관할 수 있는 공간을 확보
- 적절한 메모리 공간을 확보하기 위해서 변수의 타입 등장
- '=' 를 통해서 CPU에게 연산작업을 의뢰

## 메모리의 단위

- 0과 1을 표현하는 bit
- 8bit = 1byte



# 변수(Variable)



- 대소문자를 구분한다.
- 공백은 허용되지 않는다.
- 숫자로 시작할 수 없다.
- '$' 와 '_' 를 변수이름에 사용할 수 있다. 이외의 특수문자는 허용하지 않는다.
- 예약어(keyword, : 자바문법을 위해서 미리 지정되어 있는 단어) 는 사용할 수 없다.
- 합성어의 경우 주로 camelCase를 활용한다.
- 한글을 이용한 변수 작명 가능

# 변수(Variable)



| 자바 예약어 목록 | | | | |
|---|---|---|---|---|
| abstract | default | if | private | throw |
| boolean | do | implements | protected | throws |
| break | double | import | public | transient |
| byte | else | instanceof | return | try |
| case | extends | int | short | void |
| catch | final | interface | static | volatile |
| char | finally | long | super | while |
| class | float | native | switch | |
| const | for | new | synchronized | |
| continue | goto | package | this | |

# 자료형 (Data Type)



## Data Type

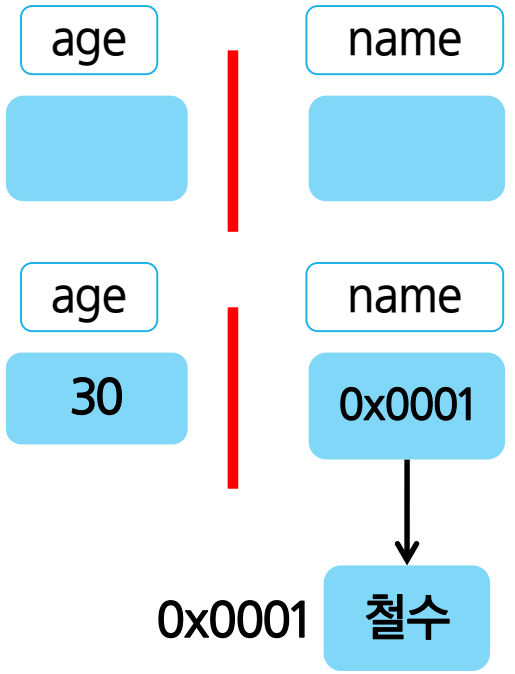
- 기본 자료형(Primitive Type)과 참조 자료형(Reference Type, 기본 자료형 8가지 외 모든 것)
- 기본 자료형 : 미리 정해진 크기의 Memory Size 표현, 변수 자체에 값 저장

| 타입 | 세부타입 | 데이터형 | 크기 | 기본값 | 값의 범위 |
|---|---|---|---|---|---|
| 논리형 | | boolean | | false | true / false |
| 문자형 | | char | 2byte | null(₩u0000) | 0 ~ 65,535 |
| 숫자형 | 정수형 | byte | 1byte | (byte)0 | -128 ~ 127 |
| | | short | 2byte | (short)0 | -32,768 ~ 32,767 |
| | | int | 4byte | 0 | -2,147,483,648 ~ 2,147,483,647 |
| | | long | 8byte | 0L | -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 |
| | 실수형 | float | 4byte | 0.0f | |
| | | double | 8byte | 0.0d | |

# 변수와 자료형



- 선언
  1. 자료형  변수명;
  2. 예) int age;  String name;  …
- 저장(할당)
  1. 변수명 = 저장할 값;
  2. 예) age = 30;  name = “철수”;
- 초기화
  1. 자료형 변수명 = 저장할 값;
  2. 예) int age = 30;



# 형 변환 (Type Casting)



- 자동 (묵시적, 암묵적) 형변환이 가능한 방향


- 데이터 형변환
  - 묵시적(암묵적) : Implicit Casting
    1. 범위가 넓은 데이터 형에 좁은 데이터 형을 대입하는 것
    2. 예> byte b = 100;  int i = b ;
  - 명시적 : Explicit Casting
    1. 범위가 좁은 데이터 형에 넓은 데이터 형을 대입하는 것
    2. 형 변환 연산자 사용 - (타입) 값;
    3. 예> int i = 100;  byte b = i ; ( X ),  byte b = (byte) i ; ( O )

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# 연산자

# 연산자 (Operator)



## 연산자 종류

| 종류 | 연산기호 | 결합 방향 | 우선 순위 |
|---|---|---|---|
| 최우선 연산자 | () . 참조연산자 [] 배열 참조 연산자 | | |
| 단항 연산자 | ++ -- +(부호) -(부호) ~ ! (type) : 형변환 | ← | 높음 |
| 산술 연산자 | * / % | → | |
| | + (덧셈) - (뺄셈) | → | |
| | << >> >>> | → | |
| 비교 연산자 | < > <= >= instanceof | → | |
| | == != | → | |
| 논리 연산자 | & | → | |
| | ^ | → | |
| | \| | → | |
| | && | → | |
| | \|\| | → | |
| 삼항 연산자 | ? : | → | |
| 대입 연산자 | = *= /= %= += -= <<= >>= >>>= &= ^= \|= | ← | 낮음 |

# 연산자 (Operator)



- 단항 연산자
  - 증감 연산자 ++, --
    - 피연산자의 값을 1 증가, 감소 시킨다.
    - 전위형(prefix) ++i
    - 후위형(postfix) i--
  - 부호 연산자 +, -
    - 숫자가 양수임을 표시 +
    - 피연산자의 부호를 반대로 변경한 결과 반환 -
  - 논리 부정 연산자 !
    - 논리 값을 반전
  - 비트 부정 연산자 ~
    - 비트 값을 반전
  - 형 변환 연산자 (type)

# 연산자 (Operator)



- 산술 연산자
  - 곱하기 연산자 *
  - 나누기 연산자 /
  - 나머지 연산자 %
  - 더하기 연산자 +
  - 빼기 연산자 -

정수와 정수의 연산 = 정수
정수와 실수의 연산 = 실수

# 연산자 (Operator)



## 비교 연산자

- 대소 비교 연산
  - > , >= , < , <=
- 동등 비교 연산
  - ==
  - !=

String 변수 비교
equals() 사용

- 객체 타입 비교 연산
  - instanceof

# 연산자 (Operator)



## 논리 연산자

- &&
  - 논리 곱 (AND) : 피연산자 모두가 true일 경우에만 true
- ||
  - 논리 합 (OR) : 피연산자 중 하나라도 true일 경우 true
- !
  - 논리 부정 (NOT) : 피연산자의 결과를 반대로 바꾼다.

효율적인 연산 가능
(short circuit evaluation)

# 연산자 (Operator)



- 삼항 연산자
  - 조건식 ? 식1 : 식2
  - 조건식이 참일 경우 식1 수행
  - 조건식이 거짓일 경우 식2 수행

- 복합 대입 연산자
  - +=, -=, *=, /= …
  - i += 1 → i = i + 1

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# 제어문(조건문)

# 조건문 (Conditional Statement)



- if 문

- switch 문

# 조건문 (Conditional Statement)



- if 문
  - 조건식의 결과에 따라 블록 실행 여부가 결정
  - 조건식 : true / false 값을 산출할 수 있는 연산식 또는 boolean 타입 변수가 올 수 있음.
  - if(조건식){
    실행할 문장1;
    실행할 문장2;
    …
    }
  - 조건식이 참일 경우 문장들을 실행하고, 거짓일 경우 실행하지 않음.
  - 실행할 문장이 하나라면 중괄호 생략 가능

# 조건문 (Conditional Statement)



- if - else 문
  - 조건식의 결과에 따라 실행할 블록 결정
  - if(조건식){
    실행할 문장1;
    실행할 문장2;
    …
    } else {
    실행할 문장a;
    …
    }
  - 조건식이 참일 경우 if 블록의 문장들을 실행하고, 거짓일 경우 else 블록의 문장들을 실행.
  - 실행할 문장이 하나라면 중괄호 생략 가능

# 조건문 (Conditional Statement)



## 중첩 if 문

- 조건문 안에 조건문을 넣어 사용함.
- if(조건식A){
  if(조건식B) {
  조건식 A, B 모두 참일 경우 수행할 문장;
  } else {
  조건식 A는 참, B는 거짓일 경우 수행할 문장;
  }
  } else {
  조건식 A가 거짓일 경우 수행할 문장;
  }
- 중첩의 횟수에는 제한이 없음.

# 조건문 (Conditional Statement)



- if - else if - else 문
  - 조건식의 결과에 따라 실행할 블록 결정
  - if(조건식){
    실행할 문장1;
    …
    } else if (조건식){
    실행할 문장a;
    …
    } … {
    } else {
    실행할 문장A;
    …
    }
  - 조건식이 참일 경우 해당 블록의 문장들을 실행하고, 거짓일 경우 다음 조건식을 확인한다.

# 조건문 (Conditional Statement)



## switch 문

- 인자로 선택변수를 받아 변수의 값에 따라서 실행문이 결정.
- switch(수식) {

```
    case 값1:
        실행문 A;
        break;
    case 값2:
        실행문 B;
        break;
    default:
        실행문 C;
}
```

주의사항

1. 수식에 올 수 있는 것
   - 1.4 버전 까지 byte, short, char, int
   - 1.5 버전 부터 enum 클래스 타입
   - 1.7 버전 부터 String 클래스 타입

2. break문 없이도 사용이 가능하다.

3. default => else의 역할과 동일하다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# 제어문(반복문)

# 반복문 (Loop)



- for 문
- while 문
- do ~ while 문

# 반복문 (Loop)



- for 문
  - for( ①초기화식 ; ②조건식 ; ④증감식 ) {
    ③반복 수행할 문장
    }
  - 초기화는 반복문이 시작될 때 한 번 실행됨.
  - 조건식이 false이면, 반복문 종료
  - 증감식은 반복문의 반복이 끝나면 실행됨.
  - 초기화식, 증감식은 (,)를 이용하여 둘 이상을 작성할 수 있음.
  - 필요하지 않은 부분은 생략할 수 있음. for( ; ; ) 무한루프
  - 반복횟수를 알고 있을 때 유용.

# 반복문 (Loop)



- 중첩 for 문
  - for문은 다른 for문을 내포할 수 있음.
  - for( 초기화식 ; 조건식 ; 증감식 ) {
    for( 초기화식 ; 조건식 ; 증감식 }
    반복 수행할 문장
    }
    }

# 반복문 (Loop)



- while 문
  - while (①조건식) {
    ②반복 수행할 문장;
    }
  - 조건식이 true일 경우에 계속해서 반복
    (조건식이 거짓이 될 때까지 문장을 반복 수행)
  - 조건식 생략 불가능

# 반복문 (Loop)



- do while 문
  - do {
    ① 반복 수행할 문장;
    } while (② 조건식);
  - 블록 내용을 먼저 수행 후 조건식 판단. (최소 한번은 수행)
  - 조건식이 true일 경우에 계속해서 반복
    (조건식이 거짓이 될 때까지 문장을 반복 수행)
  - 조건식 생략 불가능

# break & continue



- break
  - switch, while, do-while, for 문의 블록에서 빠져나오기 위해서 사용.
  - 반복문에 이름(라벨)을 붙여 한번에 빠져 나올 수 있음.

- continue
  - 반복문의 특정지점에서 제어를 반복문의 처음으로 보냄.
  - 반복문에 이름(라벨)을 붙여 제어할 수 있음.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

# 다음 방송에서 만나요!

삼성 청년 SW 아카데미