1. 데이터 출력

```
System.out.print(         ); // 여러 데이터를 출력가능
System.out.println(         ); // 하나의 데이터만 출력가능

- 위에 ( )안에 데이터를 넣는 두 가지 방법
1) "",'' 안에 문자를 넣는다.
System.out.println("Hello World!!!");

2) 변수명을 적는다.
int age = 23;
System.out.println(age + "살 입니다.");
```

2.변수

- 변수란, 데이터를 담아두는 상자

변수 선언

- int => 자료형(Type)
- age => 변수명(Name)

변수 값 부여

- 변수명 = 값

변수 선언과 값 부여 동시에 하기(변수의 초기화)

- 자료형(Type) 변수명(Name) = 값

변수명 규칙

1. 영문자, 숫자, _ 한글 조합으로 변수명을 짓는다. Ex) thisYear, man1, man2, 변수
2. 반드시 첫글자는 영문자 선호한다.
3. 예약어(Reserved Word)를 변수명으로 사용할 수 없다. Ex) in, if, double
4. 대소문자를 구분해서 사용한다.
5. 변수명은 128자까지 인식한다.

형변환

3. 연산자(Operator)

- 연산자는 특정 연산을 나타내는 기호를 의미

연산자와 피연산자

- 3.14(피연산자) *(연산자) radius(피연산자)

피연산자(operand, 연산 대상)의 갯수 분류

1) 단항 연산자(Unary)

- 피연산자가 1개 있는 연산자.
- b = -a // 부호변환연산자

2) 이항 연산자(Binary)

- 피연산자가 2개 있는 연산자
- v = v1 + v2
- x = (3 + (5*2));

3) 삼항 연산자(Thrnary)

- 피연산자가 3개 있는 연산자
- 결과 = 조건? 참 : 거짓

연산자의 종류

1) 계산/산술(Arithmatic) 연산자

```
연산자기호 | 연산자 | 의미
    +        덧셈     값을 더한다.
    -        뺄셈     값을 뺀다.
    *        곱셈     값을 곱한다.
    /       나눗셈    값을 나눈다.
    %       나머지    값을 나눌 때의 나머지 값.

++ : 증감연산자
ex> a++/++a
의미 : a = a + 1 / a에 1을 더해서 a에 대입하라.

후행(Postfix) 연산자     |     선행(Prefix) 연산자
        a++                          ++a
ex>                          ex>
a = 3;                       a = 3;
b = a++;                     b = ++a;

-> 연산과정                  -> 연산과정
a = 3;                       a = 3;
b = a;                       a = a+1;
a = a+1;                     b = a
a = 4, b = 3                  a = 4, b = 4

연산자 우선순위
++ > *,/,% > +,-
```

2) 비교(Relative) 연산자

- 결과값은 true 혹은 false로 출력된다.

```
> | 크다
< | 작다
>= | 크거나 같다
<= | 작거나 같다
== | 같다
!= | 같지 않다

ex>
int age = 23;
boolean adult = age > 19;
if(adult == true) {
    System.out.println(age + "살은 성인");
}else{
    System.out.println(age + "살은 미성년자");
}
출력: 23살은 성인
```

3) 논리 연산자

- 결과값은 true(1) 혹은 false(0)로 출력된다.

```
연산자 기호 | 의미
    &&        and : 둘 중에 하나라도 거짓이면 거짓이다.
    ||        or : 둘 중에 하나라도 참이면 참이다.
    !         not
```

4) 비트 연산자

- 데이터를 bit단위로 변환하여 연산하는 연사자. 특수한 연산(시스템 연산)에 사용된다.

```
연산자 기호     | 의미
    -             bit not
    &             bit and
    ^             bit xor
    |             bit or

ex>
a = 3; > 0 000 0011
b = ~a; > 1 111 1100 : -4

5) 할당(Assign) 연산자
연산자 기호     |   의미
    +=              a += 3 >> a = a + 3
    -=              a -= 3 >> a = a - 3
    *=              a *= 3 >> a = a * 3
```

```
/=              a /= 3 >> a = a / 3
%=              a %= 3 >> a = a % 3
```

---

4. 조건문

- (!) 복수의 문장을 실행할 경우 {} 블럭 안에 문장들을 넣어 준다.

1) if ~ else 문

```
if(조건식){
    조건식이 true일 때 실행할 문장1;
    조건식이 true일 때 실행할 문장2;
}else{
    조건식이 false일 때 실행할 문장;
}

2) switch ~ case문
switch(조건){
    case 조건값1 : 조건값 1일때 실행할 문장; break;

    case 조건값2 : 조건값2일때 실행할 문장; break;

    case 조건값3 : 조건값3일때 실행할 문장; break;

    default : 앞선 case 조건값에 해당하지 않는 경우, 실행할 문장; break;
}
```

---

5. 반복문

1)for문

```
for(초기식; 조건식; 증감식;)
      반복하고자 하는 문장;
```

2) while문

```
while(조건식){
    반복하고자 하는 문장;
}
```

3) do ~ while문

```
do{
    반복하고자 하는 문장;
} while(조건식);
```

4) break와 continue

```
break
반복 루프를 종료하는 명령어.

continue
현재의 반복을 건너뛰게 하는 명령어.
```

---

6. 배열

```
사용방법
1)배열 참조 변수 선언하기
자료형[](Type)   변수명(Name);

2)배열 객체 생성하기
배열명(Name) = new 연산자  자료형[배열수];

3)배열 선언과 생성 동시에 하기(변수의 초기화)
자료형(Type)[] = 배열명(Name)new 연산자  자료형[배열수];
```

7.클래스와 객체

객체지향 프로그래밍이란

- 코드의 길이가 길어지면서 코드의 수정/보안을 쉽게 하기 위해 같은 속성의 기능별로 묶어 그 기능이 상호 작용하게 만든 개발 형태.

클래스(Class)란

- 객체를 찍어내기위한 형틀(template).

객체(Object)란

- 같은 속성의 데이터를 모아둔 부품. 이를 인스턴스(Instance)라고도 부른다.

필드(Field)란

- 객체의 속성(상태/동작)을 나타내는 변수.

메소드(Method)란

- 객체의 동작을 수행하는 함수.

```
*코드*

Class Car{
    //Field
    String color;
    int speed;
    int gear;
    int tire;

    //Method
    void run(){
        System.out.println( color +" 색 자동차가" + speed " km로 달리고 있다.");
    }
}


public class  CarTest{

         public static void main(String[] args){

                Car myCar = new Car(); //myCar라는 이름의 객체 생성

                myCar.color = "blue"; //myCar라는 이름의 객체에 색상 속성 부여

                myCar.speed = 70; //myCar라는 이름의 객체에 스피드 속성 부여

                myCar.gear = 1; //myCar라는 이름의 객체에 기어 속성 부여

                myCar.tite = 4; //myCar라는 이름의 객체에 타이어 속성 부여

                myCar.run(); //myCar라는 이름의 객체의 달리기 메소드 호출

        }

}
```