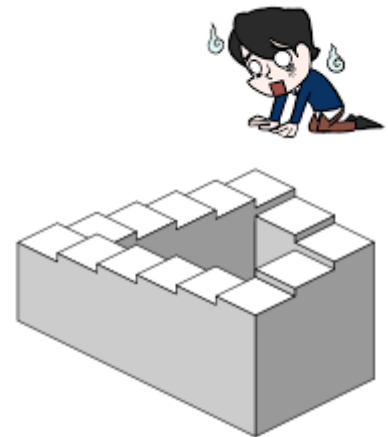


Java Web Programming 입문

Java Basic #04

오늘의 키워드

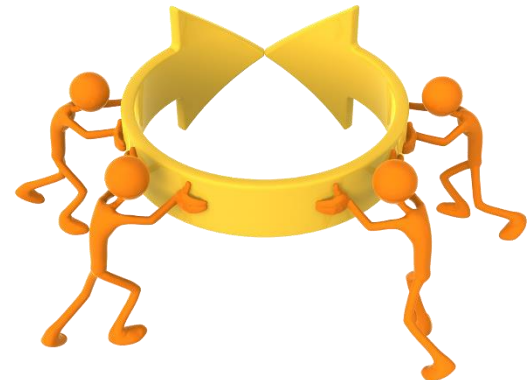
- ▶ 제어문 (control statement)
 - 반복문 (iteration statements)
 - for 반복문
 - while 반복문
 - do ~ while 반복문
 - 반복 제어문 (loop control statements)
 - Break
 - continue
- ▶ 배열 (array)
- ▶ 클래스 (class)
- ▶ 객체 (object)



제어문 (control statement)

▶ 반복문 (iteration statements)

- Loop
- 특정 코드(명령)를 반복
- 어떤 내용을 반복할 것인지?
- 몇 번 반복할 것인지?
- 언제 반복을 끝낼 것인지?

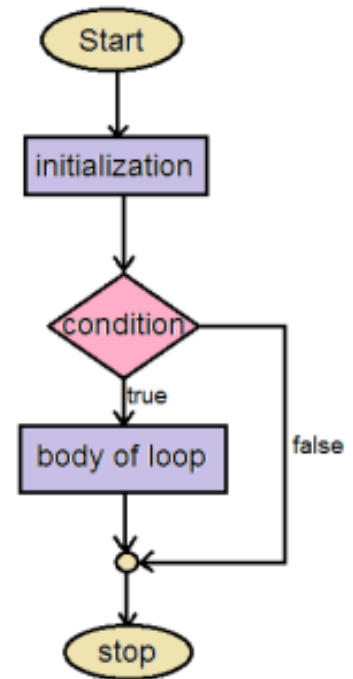


제어문 (control statement)

▶ 반복문 (iteration statements)

◦ for 반복문

- 순서대로 차근차근 반복
- 조건식이 false가 될 때까지 반복
- 일반적으로 반복횟수를 예측할 수 있을 때 사용
- 사용 빈도가 매우 높음
- 반복 기준 변수, 반복 조건식, 반복 종료를 위한 증감식

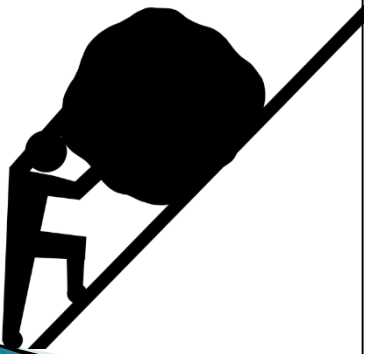
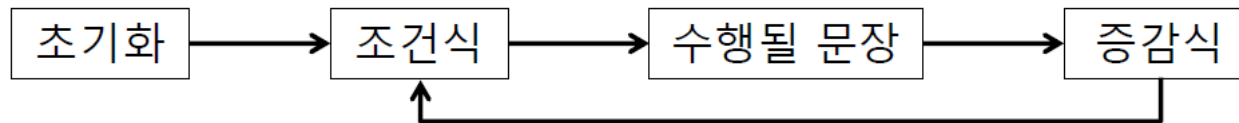


제어문 (control statement)

▶ 반복문 (iteration statements)

- for 반복문
 - 사용방법

```
for (초기화 ; 조건식 ; 증감식){  
    // 조건식이 true일때 수행  
}
```



```
public static void main(String[] args){  
    int sum = 0;  
  
    for ( int i=1 ; i <= 10 ; i++ ){  
        sum += i;    // sum = sum + i  
    }  
    System.out.println( i-1 + " 까지의 합: " + sum) ;  
}
```

제어문 (control statement)

- ▶ 반복문 (iteration statements)
 - for 반복문



```
public class ForTest01{  
    public static void main(String[] args){  
        for(int i=1 ; i <= 5; i++){  
            System.out.println("5번 출력합니다! 현재 i의 값 : " + i);  
        }  
    }  
}
```

```
public class ForTest02{  
    public static void main(String[] args){  
        for(int i=1 ; i <= 3; i++){  
            System.out.println(2 + " X " + i + " = " + 2*i);  
        }  
    }  
}
```

제어문 (control statement)

- ▶ 반복문 (iteration statements)
 - for 반복문
 - 중첩 for 반복문 (nested for loops)

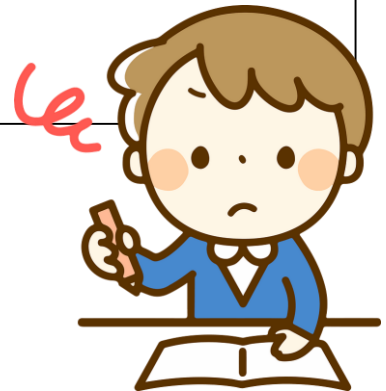
```
public class NestedForLoops{  
    public static void main(String[] args){  
        for(int i=0 ; 조건식1 ; i++){  
            ...  
            for( int j=0 ; 조건식2 ; j++){  
                ...  
                for( int k=0 ; 조건식3 ; k++ ){  
                    ...  
                }  
            }  
        }  
    }  
}
```



제어문 (control statement)

- ▶ 반복문 (iteration statements)
 - for 반복문
 - 중첩 for 반복문 (nested for loops)

```
public class ForTest03{  
    public static void main(String[] args){  
        for(int i=0 ; i<3 ; i++){  
            for( int j=0 ; j<4 ; j++){  
                System.out.println("i:" + i + ", j:" + j);  
            }  
        }  
    }  
}
```



제어문 (control statement)

- ▶ 반복문 (iteration statements)
 - for 반복문
 - 숙제
 - 자바의 for 반복문을 사용하는 방법은 한 가지가 더 있다
 - 그 방법을 찾아보자



제어문 (control statement)

▶ 반복문 (iteration statements)

◦ while 반복문

- ~ 하는 동안
- 반복횟수를 정확히 예측하긴 어렵지만, 언제 반복이 끝날지 알고 있을 때 사용
- 조건식이 시작부터 false면 단 한번도 실행되지 않음
- for문 보다 범용성이 높음
 - 하지만 for가 압도적으로 편한 경우가 있으므로, while로 대체하지는 않음
- 사용 빈도가 매우 높음
- 조건식
 - for 반복문 보다 간단해 보인다?

```
while ( 조건식 ){  
    "statement"  
}
```



제어문 (control statement)

- ▶ 반복문 (iteration statements)
 - while 반복문

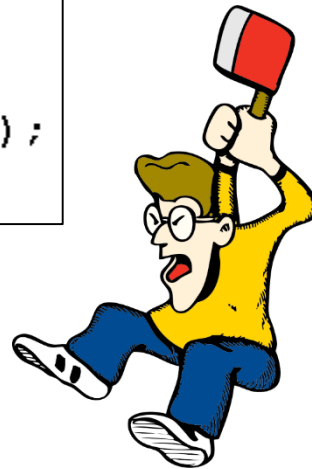
```
public class WhileTest01{  
    public static void main(String[] args){  
        int i=1;  
  
        while(i<=5){  
            System.out.println("다섯번 반복합니다. 현재 i : " + i);  
            i++;  
        }  
    }  
}
```



제어문 (control statement)

- ▶ 반복문 (iteration statements)
 - while 반복문
 - 무한 루프 (infinite loop)

```
int i = 0;  
while ( i >= 0 ) {  
    i = 10;  
    System.out.println(i--);  
}
```



제어문 (control statement)

- ▶ 반복문 (iteration statements)
 - while 반복문
 - 중첩 while 반복문 (nested while loops)

```
public class NestedWhileLoops{  
    public static void main(String[] args){  
  
        while (조건식1) {  
            ...  
            while (조건식2) {  
                ...  
                while (조건식3) {  
                    ...  
                }  
            }  
        }  
    }  
}
```

제어문 (control statement)

▶ 반복문 (iteration statements)

◦ do ~ while 반복문

- 조건식의 true 여부와 상관없이 일단 한번은 실행
 - 한번 실행한 후, 조건식을 검사

```
do{  
    "statement"  
}  
while ( 조건식 );
```

JUST DO IT.

- 많이 사용되지 않음



제어문 (control statement)

- ▶ 반복문 (iteration statements)
 - 반복 제어문 (loop control statements)
 - 반복문 내에서 반복을 제어
 - break
 - 반복횟수가 남아있어도, 즉시 반복을 중단
 - continue
 - 이번 반복주기에서의 남은 실행은 무시, 다음 반복주기로 넘어감

제어문 (control statement)

▶ 반복문 (iteration statements)

◦ 반복 제어문 (loop control statements)

- break

- 반복문이 실행되는 도중, 특정한 조건에 의해 즉시 반복을 중지시켜야할 때 사용
- 반복 횟수가 남아있는 것과 상관없이, 바로 반복문을 빠져나옴

```
반복문 {  
    ...  
    특정한조건 {  
        break;  
    }  
    ...  
}
```



```
public static void main(String[] args){  
    int sum = 0;  
    int i = 0;  
  
    while (true){  
        if ( sum > 100 )  
            break;  
        i++;  
        sum += i;  
    }  
  
    System.out.println("i=" + i);  
    System.out.println("sum=" + sum);  
}
```


제어문 (control statement)

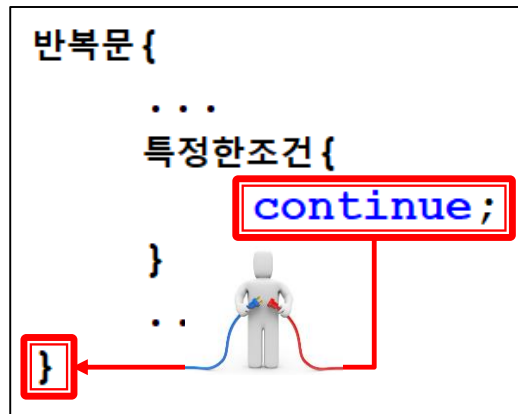
▶ 반복문 (iteration statements)

◦ 반복 제어문 (loop control statements)

- continue

- 반복문이 실행되는 도중, 특정한 조건에 의해 현재 반복 주기를 넘겨야 할 때 사용

- 사용시, 남은 코드 실행을 하지 않고, 바로 다음 반복으로 넘어감



```
public static void main(String[] args){
    for ( int i = 0 ; i <= 10 ; i++ ) {
        if (i%3 == 0)
            continue;
        System.out.println(i);
    }
}
```

제어문 (control statement)

- ▶ 반복문 (iteration statements)
 - 반복 제어문 (loop control statements)
 - Pop-quiz

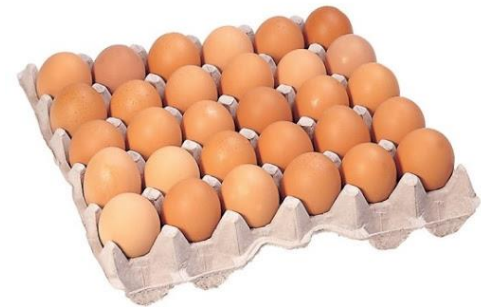
```
public class PopQuiz{  
    public static void main(String[] args){  
        int sum = 0;  
        for( int i=0 ; i < 10 ; i++ ){  
            if ( i % 2 == 0 )  
                continue;  
            if( i == 6 )  
                break;  
            sum += i;  
        }  
        System.out.println("최종 sum : " + sum);  
    }  
}
```

- 45 / 25 / 9 / 4



배열 (array)

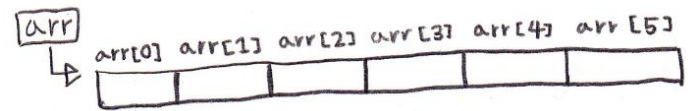
- ▶ 계란 판
- ▶ 동일한 데이터 타입의 데이터를 여러 개 저장
 - 하나의 변수를 통해
- ▶ 참조형 데이터 타입



배열 (array)

▶ 선언 방법

선언방법	선언 예
타입[] 변수이름;	int[] score; String[] name;
타입 변수이름[];	int score[]; String name[];



▶ 생성

- 배열 선언 (생성된 배열을 다루는데 사용될 참조변수 선언)

```
int[] score;
```

- 배열 생성 (5개의 int값을 저장할 수 있는 공간생성)

```
score = new int[5];
```

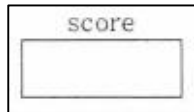
```
int[] arr = {5, 30, 110, 150};
```



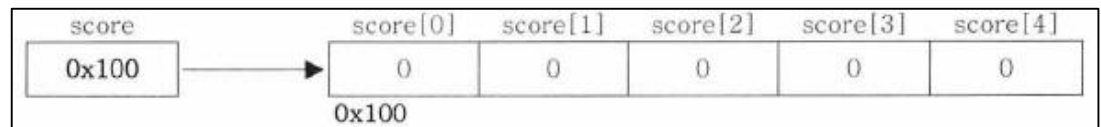
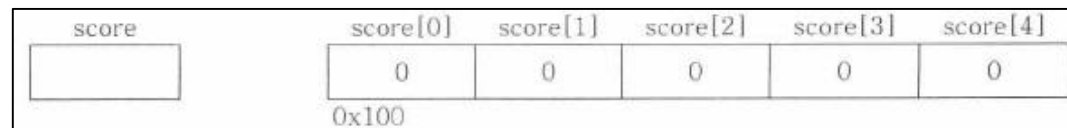
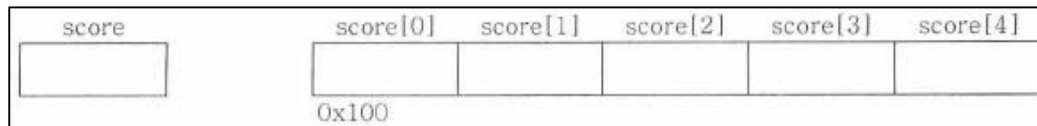
배열 (array)

▶ 자동 초기화

- `int[] score`



- `score = new int[5];`



배열 (array)

▶ 데이터 타입 별 초기화 값 (기본값)

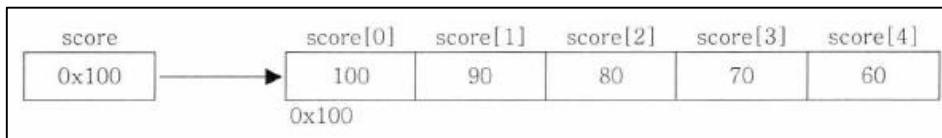
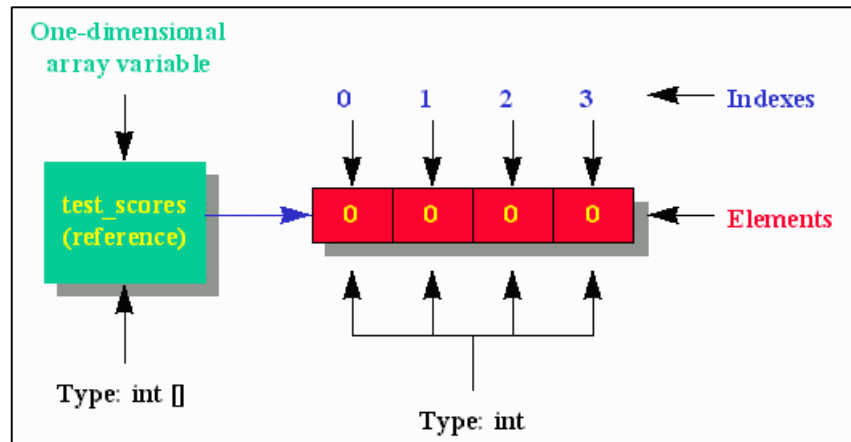
자료형	기본값
boolean	false
char	'\u0000'
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d 또는 0.0
참조형 변수	null



배열 (array)

▶ 배열에 값 넣기

```
int[] score;  
  
score = new int[5];  
  
// 크기가 5인 int형 배열을 생성  
int[] score = new int[5];  
score[0] = 100; // 각 요소에  
score[1] = 90;  // 직접 값을  
score[2] = 80;  // 저장  
score[3] = 70;  
score[4] = 60;
```



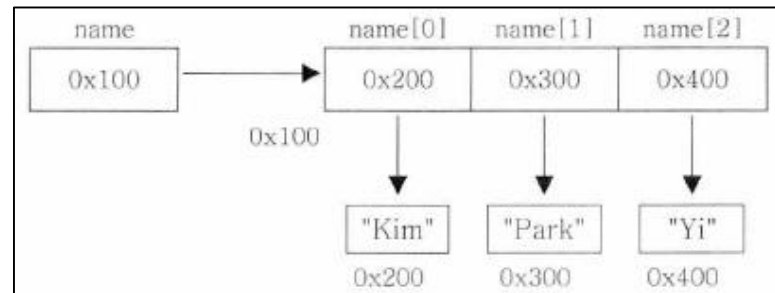
```
int[] score = { 100, 90, 80, 70, 60 };  
int[] score = new int[] { 100, 90, 80, 70, 60 };
```

배열 (array)

▶ 참조형 데이터 타입 배열

```
String[] name;           // String 타입의 참조변수 배열 선언  
name = new String[3];    // String 인스턴스의 참조변수를 담을 수 있는 배열 생성
```

```
String[] name = new String[3];  
name[0] = new String( "Kim" );  
name[1] = new String( "Park" );  
name[2] = new String( "Yi" );
```



```
String[] name = { new String( "Kim" ),  
                  new String( "Park" ),  
                  new String( "Yi" ) };  
  
String[] name = { "Kim", "Park", "Yi" };  
  
String[] name = new String[]{ new String( "Kim" ),  
                               new String( "Park" ),  
                               new String( "Yi" ) };
```


배열 (array)

- ▶ 배열과 for문은 유유상종(類類相從)

```
// score 배열의 4번째 요소에 100 값 저장  
score[3] = 100;  
  
// score 배열의 4번째 요소에 저장된 값을 value에 저장  
int value = score[3];
```

```
int[] score = { 100, 90, 80, 70, 60, 50 };  
  
for (int i=0 ; i<6 ; i++ ){  
    System.out.println(score[i]);  
}
```



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

배열 (array)

▶ 2차원 배열

선언방법	선언예
타입 [][] 변수이름;	<code>int[][] score;</code>
타입 변수이름 [][];	<code>int score[][];</code>
타입 [] 변수이름 [];	<code>int[] score[];</code>

```
// 5행 3열의 2차원 배열을 생성  
int[][] score = new int[5][3];
```



			a[1][2]
	99	85	98
row 1 →	98	57	78
	92	77	76
	94	32	11
	99	34	22
	90	46	54
	76	59	88
	92	66	89
	97	71	24
	89	29	38
			column 2

a[] []	.70 .20 .10	
	.30 .60 .10	← row 1
	.50 .10 .40	
b[] []	.80 .30 .50	column 2
	.10 .40 .10	
	.10 .30 .40	
c[] []	.59 .32 .41	
	.31 .36 .25	←
	.45 .31 .42	

$$c[1][2] = .3 * .5 + .6 * .1 + .1 * .4 = .25$$

배열 (array)

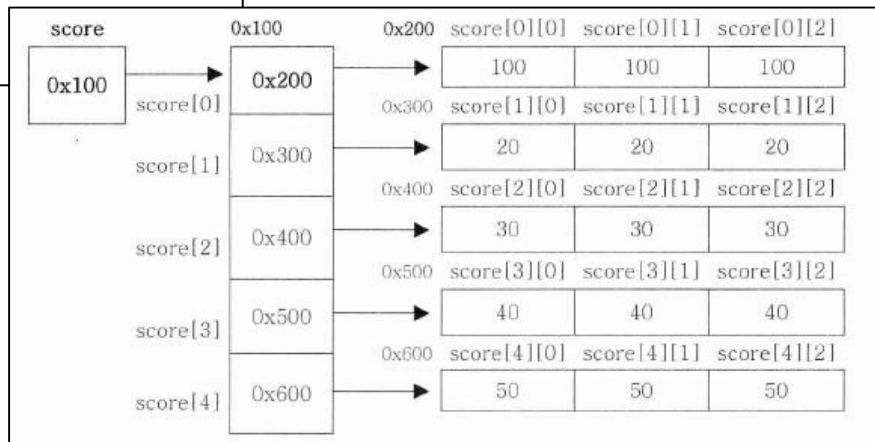
▶ 2차원 배열

	국어	영어	수학
1	100	100	100
2	20	20	20
3	30	30	30
4	40	40	40
5	50	50	50

```
score[0][0] = 100;  
score[0][1] = 100;  
score[0][2] = 100;  
score[1][0] = 20;  
score[1][1] = 20;  
...  
score[4][2] = 50;
```

```
int[][] score = {  
    { 100, 100, 100 },  
    { 20, 20, 20 },  
    { 30, 30, 30 },  
    { 40, 40, 40 },  
    { 50, 50, 50 }  
}
```

```
for (int i=0 ; score.length ; i++ ){  
    for (int j=0 ; score[i].length ; j++ ){  
        score[i][j] = 10;  
    }  
}
```



클래스 (class)

- ▶ 사용자 정의 데이터 타입 (user-defined data type)
- ▶ 참조형 데이터 타입
- ▶ 큰 구성
 - 데이터 저장소
 - 다양한 여러 종류의 데이터 타입 여러 개 포함
 - 변수(variable)
 - 실행될 기능
 - 특정한 목적을 수행하기 위한 코드 집합
 - 메서드(method)
 - 함수(function)의 개념과 유사



클래스 (class)

▶ 속성과 기능

◦ 속성 (Property)

- 멤버변수 (member Variable)
- 특성 (attribute)
- 필드 (field)
- 상태 (State)



속성(property) → 멤버변수(variable)
기능(function) → 메서드(method)

채널 → int channel
채널 높이기 → channelUp() { ... }

속성	크기, 길이, 높이, 색상, 볼륨, 채널 등
기능	켜기, 끄기, 볼륨 높이기, 볼륨 낮추기, 채널 변경하기 등

◦ 기능 (Function)

- 메서드 (method)
- 행위 (behavior)
- 함수 (function)

```
class Tv {  
    // Tv의 속성 (멤버변수)  
    String color;      // 색상  
    boolean power;    // 전원상태 (on/off)  
    int channel;      // 채널  
  
    // Tv의 기능 (메서드)  
    void power()      { power = !power; } // TV On/Off  
    void channelup()  { ++channel; } // TV채널 ▲  
    void channeldown() { --channel; } // TV채널 ▼  
}
```

클래스 (class)

▶ 객체의 생성

```
// 클래스의 객체를 참조하기 위한 참조변수 선언  
클래스명 변수명;
```

```
// 클래스의 객체를 생성 후, 객체의 주소를 참조변수에 저장  
변수명 = new 클래스명();
```

```
// Tv 클래스 타입의 참조변수 t를 선언  
Tv t;
```

```
// Tv 인스턴스를 생성한 후, 생성된 Tv인스턴스의 주소를 t에 저장  
t = new Tv();
```

▶ 객체의 사용

- . <- 요놈을 잘 사용해야 한다



객체 (object)

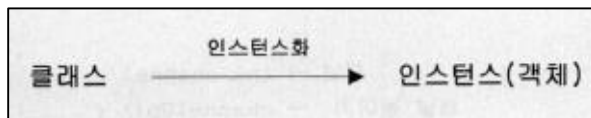
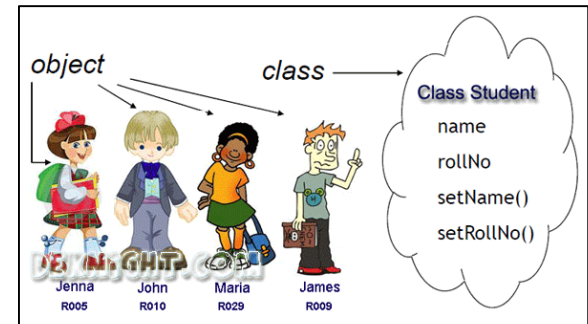
▶ 클래스와 객체의 정의/용도

○ 클래스 (class)

- 객체 (object)를 정의해 놓은 것
- 객체 (object)를 생성하는데 사용

○ 객체 (object)

- 실제로 존재하는 것, 사물 또는 개념
- 클래스 (class)를 실제로 구현해 놓은 것
- 유형의 객체 - 책상, 의자, 자동차, TV 같은 사물
- 무형의 객체 - 수학기초, 프로그램 에러 같은 논리, 개념



클래스	객체
제품설계도	제품
TV설계도	TV
붕어빵 기계	붕어빵