



자바의 제어문과 배열



학습 목표

- * 자바의 if문과 switch문 등의 조건문의 형태에 대하여 학습합니다.
- * 자바의 for문, while문, do-while문 등의 반복문의 형태에 대하여 학습합니다.
- * 자바의 break문, continue문, return문 등의 이동문의 형태에 대하여 학습합니다.
- * 자바에서 배열 사용 구문에 대하여 학습합니다.
- * 명령행 매개변수 사용 방법에 대하여 학습합니다.



자바의 제어문과 배열

- * 자바의 조건문
- * 자바의 반복문
- * 자바의 이동문
- * 자바의 배열



자바의 조건문

- * if – else문
- * switch – case문



if – else문

* 기본 형식

```
if (boolean 타입의 표현식) ————— boolean 타입의 데이터나 연산식
    문장 또는 중괄호{}로 감싸인 여러 문장들;
else
    문장 또는 중괄호{}로 감싸인 여러 문장들;
```

boolean 타입의 표현식이 true인 경우 수행될 내용

boolean 타입의 표현식이 false인 경우 수행될 내용

* if 블록의 예

```
int count = 0;
if (count != 0) { // 라인 1.
    count = 100 / count;
    System.out.println("true state");
}
```

if-else 블록의 예

```
if((s!=null) && (s.length()>0)) {
    System.out.println("&& 참");
}
else {
    System.out.println("&& 거짓");
}
```

if – else문

* if-else if 형식





if – else문

- * 프로그램 4-1 실습
 - * IfTest.java 작성
 - * 세 수의 평균 구하고 구해진 평균에 따라 등급 구하기

```
C:\WJAVA>java IfTest  
i1, i2, i3의 평균 = 70  
평균에 따른 등급 = C
```



switch – case문

* 다중 분기문

* 기본 형식

```
switch (표현식) { ○———— 비교할 변수. byte, short, int, char 중의 하나
    case value1 :
        문장1(들); ○———— 표현식의 변수가 value1과 같으면 문장1 수행
        break; ○———— switch문 중단
    case value2 :
        문장2(들);
        break;
    ..... ○———— 여러 개의 case절 가능
    case valueN :
        문장N(들);
        break;
    default:
        문장(들); ○———— 위의 case와 일치하지 않는 나머지 경우 문장 수행
        break;
}
```




switch – case문

- * switch() 내부는 int 타입 가능
- * case 값과 일치하는 case 절 수행
- * 일치하는 값 없으면 default 수행
- * case 뒤에는 상수만 가능
- * case 는 중괄호({}) 필요없음
- * break 생략 가능



switch – case문

- * 프로그램 4-2 실습
 - * SwitchTest.java 작성
 - * 세 수의 평균 구하고 구해진 평균에 따라 등급 구하기
 - * 프로그램 4-1 변경

```
C:\WJAVA>java SwitchTest  
i1, i2, i3의 평균 = 100  
평균에 따른 등급 = A입니다.
```



자바의 반복문

- * for문
- * while문
- * do – while문



for문

* 기본 형식

```
for (반복시작문장; 반복조건식; 증감식)  
    문장 또는 중괄호{ }로 감싸인 여러 문장들;
```

* 반복시작문장

- * 반복문 수행 조건 변수 초기값 설정

* 반복조건식

- * boolean 타입의 결과가 나오는 표현식
- * 표현식 결과가 true이면 for문의 블록 내부 문장 실행

* 증감식

- * 반복 횟수 변경
- * 반복조건식 부분을 처리하여 반복 여부를 결정



for문

* 10번 반복

```
for(int i= 0 ; i < 10 ; i++){  
    System.out.println(i);  
}
```

전혀 반복하지 않음

```
for(int i = 10; i < 10 ;i++){  
    System.out.println(i);  
}
```

* 오류 발생

```
for(int i = 10; i < 10 ;i++){ //라인 1  
    System.out.println(i);  
}  
System.out.println ("i 의 최종 값은 = " + i); //라인 2. 오류 발생
```



for문

- * 프로그램 4-3 실습
 - * ForTest.java 작성
 - * 이차원 배열값 '*' 출력 반복

```
C:\WJAVA>java ForTest
```


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



while문

* 기본 형식

while (반복조건식)  boolean 타입

문장 또는 중괄호{ }로 감싸인 여러 문장들; 

while ()의 반복조건식의 결과가 true이면 while문 블록 내부 문장 실행

- * for문의 반복 초기화 문장이 while문에는 필요하지 않다는 점 제외하고 반복 동일



while문

- * 프로그램 4-4 실습
 - * WhileTest.java 작성
 - * 1부터 10까지의 합 출력

```
C:\WJAVA\java whileTest
```

```
1까지의 중간합계 = 1
```

```
2까지의 중간합계 = 3
```

```
3까지의 중간합계 = 6
```

```
4까지의 중간합계 = 10
```

```
5까지의 중간합계 = 15
```

```
6까지의 중간합계 = 21
```

```
7까지의 중간합계 = 28
```

```
8까지의 중간합계 = 36
```

```
9까지의 중간합계 = 45
```

```
10까지의 중간합계 = 55
```

```
1부터 10까지의 총합계 = 55
```




do – while문

* 기본 형식

do 문장 또는 중괄호{ }로 감싸인 여러 문장들; ○———— 반복조건식과 관계없이
최소 1번 이상은 반복될 문장
while (반복조건식); ○———— boolean 타입. true이면 do 블록 내부의 문장 계속 반복 수행

* 최소 1번 이상 반드시 수행하는 점이 while문과의 차이점



do – while문

- * 프로그램 4-5 실습
 - * DoWhileTest.java 작성
 - * 변수의 복사값 출력하는 do-while문

```
C:\WJAVA>java DoWhileTest
```

```
do while문은 최소한 1번은 실행하는 반복문입니다.
```

```
increment 변수의 복사 값 = 10
```



자바의 이동문

- * break문
- * continue문
- * return문



break문

- * 반복문이나 switch문 중단
 - * switch문 중단하고 switch{} 다음 문장 이동
 - * for, while, do-while문 등의 반복문 중단하고 반복문 {} 다음 문장 이동
 - * 중첩 반복문 구조에서 레이블이 있는 반복문 중단하고 레이블이 있는 반복문 {} 다음 문장 이동



break문

* switch문에서의 사용

```
switch (avg) {  
    case 90 :  
        grade = 'A';  
        break;  
    .....  
}  
//라인 1
```

○ avg 변수가 90과 같으면 grade 변수는 'A' 할당
○ switch문을 중단하고 라인 1의 위치로 이동

* for문에서의 사용

```
for(i = 0 ; i < 10 ; i++) {  
    if(i == 5) break;  
    System.out.println(i);  
}  
//라인 1
```

○ 반복문 수행 중 i가 5와 같으면 라인 1로 이동.
i 출력하는 반복문 중단

* 라인 1의 위치로 이동



break문

* 중첩 for문에서의 사용

```
outer: for(int i = 0 ; i < 4 ; i++) {  
    for(int j = 0; j < 3; j++) {  
  
        if(i == 2) break outer; ○———— i=2 이면 라인 1로 이동. 반복 불가능  
  
        System.out.println("i = " + i + " j = " + j);  
    }  
}
```

//라인 1

출력결과

```
i = 0 j = 0  
i = 0 j = 1  
i = 0 j = 2  
i = 1 j = 0  
i = 1 j = 1  
i = 1 j = 2
```



break문

- * 프로그램 4-6 실습
 - * BreakTest.java 작성
 - * 1부터의 합계를 구하여 합계가 30 초과하면 합계 구하는 반복 중단

```
C:\WJAVA>java BreakTest  
sum > 30이면 while문은 중단합니다.
```

```
increment = 1일 때 sum = 1  
increment = 2일 때 sum = 3  
increment = 3일 때 sum = 6  
increment = 4일 때 sum = 10  
increment = 5일 때 sum = 15  
increment = 6일 때 sum = 21  
increment = 7일 때 sum = 28
```



continue문

- * 반복문 내에서 continue문 이후 문장 수행 생략
- * 그 이후에 반복문 처음 위치로 이동
- * 현재 반복을 일시적 생략하고 반복문의 처음 위치로 이동
- * 중첩된 반복에서 레이블이 있는 반복문의 처음 위치로 이동

continue문

* 반복문에서의 사용 예

```
for(i = 0 ; i < 10 ; i++) { //라인 1
    if(i == 5) continue;
    System.out.println(i);    //라인 2
}
```

i가 5일 때 라인 2 생략하고 라인 1로 이동, 나머지 반복 계속 수행

* 중첩된 반복문에서의 사용 예

```
outer: for(int i = 0 ; i < 4 ; i++) { //라인 1
    for(int j = 0; j < 3; j++) {
        if(i == 2) continue outer;
        System.out.println("i = " + i + " j = " + j);
    }
}
```

outer 레이블이 붙은 라인 1로 이동.
i == 2인 경우 생략

i = 0	j = 0
i = 0	j = 1
i = 0	j = 2
i = 1	j = 0
i = 1	j = 1
i = 1	j = 2
i = 3	j = 0
i = 3	j = 1
i = 3	j = 2



continue문

- * 프로그램 4-7 실습
 - * ContinueTest.java 작성
 - * 1부터의 합계를 구하되 5는 제외하고 나머지 합계만 계산

```
C:\WJAVA>java ContinueTest  
increment가 5인 경우에는 for문 수행을 생략합니다.
```

```
increment = 1일 때 sum = 1  
increment = 2일 때 sum = 3  
increment = 3일 때 sum = 6  
increment = 4일 때 sum = 10  
increment = 6일 때 sum = 21  
increment = 7일 때 sum = 28  
increment = 8일 때 sum = 36  
increment = 9일 때 sum = 45  
increment = 10일 때 sum = 55
```

return문

- * 현재 수행중인 메소드 수행 중단
- * 이후에 메소드 호출 지점으로 이동

```
static void call(){  
  
    m(); ○———— m() 메소드를 호출하면 m() 메소드로 제어가 이동  
  
    System.out.println("call () 완료"); //라인 1  
}  
  
static void m(){  
    for(int j=2; j<=9; j++){  
        if( j == 5) return;  
        ○———— j == 5이면 m() 메소드 수행을 중단하고 라인 1로 이동  
        System.out.print(j + "Wt");  
    }  
    System.out.println("m 메소드 끝");  
}  
○———— m() 메소드 내에서는 j가 5가 되면 if문의 결과가 true가 되어 return문을  
만나므로 m() 메소드를 호출한 지점으로 즉시 제어 이동되어 m() 메소드  
의 나머지 모든 문장 수행 중단.
```



return문

- * 프로그램 4-8 실습
 - * ReturnTest.java 작성
 - * 1부터의 합계를 구하du 30 초과시 test 메소드 중단

```
C:\WJAVA>java ReturnTest
increment = 1일 때 sum = 1
increment = 2일 때 sum = 3
increment = 3일 때 sum = 6
increment = 4일 때 sum = 10
increment = 5일 때 sum = 15
increment = 6일 때 sum = 21
increment = 7일 때 sum = 28
main 메소드 끝
```



자바의 배열

- * 자바 배열의 특징
- * 자바 배열의 사용
- * `main()` 메서드와 명령행 매개변수

자바 배열의 특징

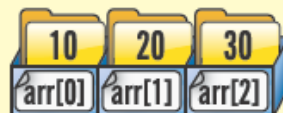
- * 하나의 이름으로 많은 데이터 사용
- * 같은 타입 데이터 저장하는 참조형 변수

자바의 배열 사용

```
int arr [] = new int[3];  
arr [0] = 10;  
arr [1] = 20;  
arr [2] = 30;
```

배열 사용 시 메모리 구조

arr





자바 배열의 사용

- * 자바 배열의 선언, 생성, 초기화
- * 배열은 변수 선언, 생성, 초기화의 순서로 사용

- * 배열 변수 선언

```
데이터타입 배열변수이름[ ];
```

- * 배열 변수 선언의 예

```
int intArray[];           //정수 배열 intArray 선언
double[] dArray, otherArray; //실수 배열 dArray와 otherArray 선언
String msgArray[];        //문자열 배열 msgArray 선언
```



자바 배열의 사용

* 배열 생성

```
배열변수이름 = new 데이터타입[배열길이];
```

* 배열 생성의 예

```
intArray = new int[5];           //정수 5개 저장 배열 intArray 생성  
dArray = new double[10];         //실수 10개 저장 배열 dArray 생성  
otherArray = new double[100];    // 실수 100개 저장 배열 otherArray 생성  
msgArray = new String[5];        //문자열 5개 저장 배열 msgArray 생성
```

* 배열의 길이

```
intArray.length    // 정수 5개 저장하므로 5  
dArray.length      // 실수 10개 저장하므로 10  
otherArray.length  // 실수 100개 저장하므로 100  
msgArray.length    // 문자열 5개 저장하므로 5
```




자바 배열의 사용

* 배열 선언과 생성 문장

```
데이터타입 배열변수이름[ ] = new 데이터타입[배열길이];
```

* 배열 선언과 생성의 예

```
int intArray[ ] = new int[5];  
double[ ] dArray = new double[10];  
double[ ] otherArray = new double[100];  
String msgArray[ ] = new String[5];
```



자바 배열의 사용

- * 배열의 자동 초기화값
 - * byte, short, int : 0
 - * long : 0L
 - * float : 0.0F
 - * double : 0.0
 - * boolean : false
 - * char : '\u0000'
 - * 모든 참조형 변수 : null



자바 배열의 사용

* 프로그램 4-9 실습

* ArrayTest.java 작성

```
C:\WJAVA>java ArrayTest
iarray 배열에 최초로 저장된 값 = 0
iarray 배열에 최초로 저장된 값 = 0
iarray 배열에 최초로 저장된 값 = 0
iarray 배열에 최초로 저장된 값 = 0
iarray 배열에 최초로 저장된 값 = 0
iarray 배열에 최초로 저장된 값 = 0
iarray 배열에 최초로 저장된 값 = 0
iarray 배열에 최초로 저장된 값 = 0
iarray 배열에 최초로 저장된 값 = 0
iarray 배열에 최초로 저장된 값 = 0
***** iarray 배열에 10부터 100까지 저장합니다 *****
iarray 배열에 최초로 저장된 값 = 10
iarray 배열에 최초로 저장된 값 = 20
iarray 배열에 최초로 저장된 값 = 30
iarray 배열에 최초로 저장된 값 = 40
iarray 배열에 최초로 저장된 값 = 50
iarray 배열에 최초로 저장된 값 = 60
iarray 배열에 최초로 저장된 값 = 70
iarray 배열에 최초로 저장된 값 = 80
iarray 배열에 최초로 저장된 값 = 90
iarray 배열에 최초로 저장된 값 = 100
```



자바 배열의 사용

* 배열 초기화 문장

```
int score[] = new int[3000];  
for (int i = 0; i < score.length; i++) {  
    score[i] = i;  
}
```

```
String[] name = new String[5];  
name[0] = "이자바";  
name[1] = "박이한";  
name[2] = "김전산";  
name[3] = "정대학";  
name[4] = "최출판";
```

* 배열 선언, 생성, 초기화 문장

```
int intArray[] = {100, 200, 300};  
float fArray[] = {1.1f, 2.2f, 3.3f, 4.4f};  
String sArray[] = {"hello", "SCJP", "exam"};
```



자바 배열의 사용

- * 다차원 배열의 사용
 - * 배열의 배열
 - * 동일 길이의 이차원 배열

```
데이터타입 배열 변수이름[][] = new 데이터타입[일차원 배열길이][이차원 배열길이];
```

- * 4*10 길이의 이차원 배열 예

```
int twoDimArray[][] = new int[4][10] ;
```

자바 배열의 사용

* 다차원 배열의 사용

* 서로 다른 길이의 이차원 배열

```
데이터타입 배열 변수이름[][] = new 데이터타입[일차원 배열길이][];  
배열변수이름[0] = new 데이터타입[이차원 배열길이1];  
배열변수이름[1] = new 데이터타입[이차원 배열길이2];
```

* 서로 다른 길이의 이차원 배열 예

```
int twoDimArray[][] = new int[4][];    // 라인 1  
  
twoDimArray[0] = new int[5];           // 라인 3  
twoDimArray[1] = new int[10];          // 라인 4  
twoDimArray[2] = new int[3];           // 라인 5  
twoDimArray[3] = new int[7];           // 라인 6  
  
for (int i = 0; i < twoDimArray.length; i++)  
    for (int j = 0; j < twoDimArray[i].length; j++)  
        twoDimArray[i][j] = 0;
```

라인 1에서는 데이터 개수가 4개인 이차원 배열을 생성

3,4,5,6에서는 이렇게 생성된 각 데이터가 필요로 하는 만큼의 배열을 생성

twoDimArray.length인 4 * twoDimArray[i].length 만큼 반복문 수행



자바 배열의 사용

- * 프로그램 4-10 실습
 - * Array2DTest.java 작성
 - * 서로 다른 길이의 이차원 배열 이용

```
C:\WJAVA>java Array2DTest
```

1	2			
2	4	6		
3	6	9	12	
4	8	12	16	20



main() 메소드와 명령행 매개변수

- * 자바 어플리케이션 실행시 JVM이 자동 호출하는 메서드
- * 정해진 형식에 따라 선언

```
public static void main(String arg[])
```
- * String 배열이 명령행 매개변수
- * 실행시에 명령행 매개변수값 입력받아 사용

main() 메서드와 명령행 매개변수

* 예문

```
class ArgTest{  
    public static void main(String arg[]){ // 라인 1
```

라인 1에 선언된 arg라는 이름의 배열에는 사용자가 입력한 문자열을 저장합니다.

```
        for(int i=0; i < arg.length; i++){ // 라인 2  
            System.out.println(arg[i]); // 라인 3
```

라인 2에서는 arg.length 즉 입력된 문자열의 개수만큼 반복하여 라인 3에서 arg[i]에 입력된 문자열 내용을 출력합니다.

```
        }
```

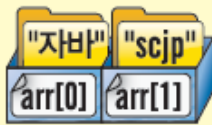
```
    }
```

```
}
```

* 실행

```
java ArgTest 자바 scjp // 라인 4
```

arg



```
java ArgTest 100 + 200 // 라인 5
```

arg





main() 메서드와 명령행 매개변수

- * 프로그램 4-11 실습
 - * ArgsTest.java 작성
 - * 명령행 매개변수 개수와 내용 출력

```
C:\WJAVA>java ArgsTest
```

```
입력한 명령행 매개변수의 개수 = 0  
입력 내용은 다음과 같습니다.
```

```
C:\WJAVA>java ArgsTest 자바 jsp servlet applet
```

```
입력한 명령행 매개변수의 개수 = 4  
입력 내용은 다음과 같습니다.
```

```
자바 : jsp : servlet : applet :
```



main() 메서드와 명령행 매개변수

- * 프로그램 4-12 실습
 - * ArgsToNumberTest.java 작성
 - * 명령행 매개변수를 정수와 실수로 변경

```
C:\WJAVA>java ArgsToNumberTest 10 20 3.14 2.56
```

네 개의 명령행 매개변수 가운데 처음의 두 변수는 int 타입,
나머지 두 변수는 double 타입으로 변환하여 덧셈 연산합니다

정수 $10 + 20 = 30$

실수 $3.14 + 2.56 = 5.7$



- * if-else문 switch문

```
if (조건식1)
    boolean 타입의 조건식1이 true인 경우 수행될 내용
[ else if(조건식2)
    boolean 타입의 조건식2가 true인 경우 수행될 내용
..... ]
[ else
    위의 조건식에 해당하지 않는 경우 수행될 내용 ]
```

```
switch (byte, short, int, char 중의 하나) {
    case value1 :
        표현식의 변수가 value1과 같은 경우 수행할 문장;
        break;    ;
    .....
    default:
        case와 일치하지 않는 나머지 경우 수행할 문장
}
```



정리

- * 자바의 반복문은 for, while, do-while 등의 3가지 반복문을 사용할 수 있습니다.

- * for문

```
for (반복시작문장; 반복조건식; 증감식)
{ 반복적으로 수행할 문장들; }
```

- * while문

```
while (boolean 타입의 반복조건식)
{ 반복적으로 수행할 문장들; }
```

- * do-while문

```
do{반복조건식과 관계없이 최소 1번 이상은 반복될 문장들;}
while(boolean 타입의 반복조건식);
```



정리

- * 자바의 이동문에는 break, continue, return문이 제공됩니다.
- * break문은 해당 제어문을 중단하며, switch문이나 반복문 내에서 사용합니다.
- * continue문은 해당 반복문 수행을 생략하고 반복을 계속합니다.
- * return문은 현재 수행 중인 메소드의 수행을 중단하고 이 메소드를 호출한 곳으로 제어를 리턴합니다.
- * 자바의 배열은 동일 타입의 데이터들의 저장소로, 배열은 선언, 생성, 초기화의 순서로 사용합니다.



정리

- * 자바 배열은 생성된 후에 length라는 변수를 자동으로 포함합니다.
- * 자바 배열의 index 범위는 0 이상이고 배열의 length 보다 작은 범위 내에 있습니다.
- * main 메소드의 매개변수는 String 배열 타입입니다.
- * main 메소드의 매개변수는 java.exe 명령어로 실행 시에 입력한 값을 저장하는 저장소로 사용합니다.