java 제어문

1. Java 기초 – 제어문

1. Java 는 크게 두 가지의 제어문을 가지고 있습니다

조건문

주어진 조건에 대해 분기를 한다

```
if(조건) { 코드1; } else if (조건2) { 코드2; } else { 코드3; }
switch(파라미터){
  case 값 : 코드 ; break; }
```

반복문

조건에 따라 반복문 블록 안의 코드 반복적으로 처리한다

```
for(조건) { 반복할 코드; }
do { 반복할 코드; } while (조건)
    또는 while(조건) { 코드; }
```

예외처리

예외처리를 제어문의 한 종류로 분류한 책도 있지만
여기서는 따로 분류하겠습니다

2. Java 기초 – 조건문 - if

1. if 는 주어진 조건을 비교하여 실행할 코드를 결정하는데 아래와 같은 세 가지 형태로 사용할 수 있습니다

```
if(  a > b ) {  // a 가  b 보다 클 경우 코드1 실행
    코드1;
}
```

```
if( a == b ) { // a 가 b 와 같을 경우 코드 1 실행
    코드1;
} else {         // 위의 조건(a==b)을 만족하지 않을 경우 코드 2 실행
    코드2;
}
```

```
if( a >= b ) {          // a 가 b 보다 크거나 같을 경우 코드1 실행
   코드 1;
} else if ( b <= c ) {  // b 가 c 보다 작거나 같을 경우 코드 2 실행
   코드 2;
} else {                // 위의 두 조건을 모두 만족하지 않을 경우 코드 3 실행
   코드 3
}
```

3. Java 기초 – 조건문 - switch

1. 또 다른 형태의 조건문으로 switch 가 있습니다.
   두 가지는 사용하는 경우가 다른데 if 문은 모든 경우에 사용할 수 있지만 switch 문은 특정 값을 지정할 때만 사용할 수 있습니다

```
int num = 20;
switch(num){
   case 10: 코드1; break;
   case 20: 코드2; break; // num의 값이 20이기 때문에 코드2가 실행된다.
   default: 코드3;          // 매칭되는 값이 없을 경우 default 에 있는 코드3실행
}
// switch는 위의 코드에서 처럼 범위는 사용할 수 없다
```

- break 의 중요성

```
int num = 20;
switch(num){
   case 20: 코드1;          // num 의 값과 20이 같기 때문에 코드1을 실행하지만
                              break 가 없기 때문에 다음 case 도 실행한다
   case 25: 코드2; break;  // break 가 있기 때문에 default 는 실행하지 않는다
   default: 코드3;
}
```

4. Java 기초 – 반복문 - for

1. for 문은 시작, 종료, 증감값을 가지고 코드를 반복해 줍니다.

```
for ( 시작값 ; 종료값 ; 증감값) {
    반복할 코드;
}
```

- 아래는 시작 값인 a 변수에 0을 넣고, 시작한 후에 a 값이 5보다 작거나 같을 때 까지 코드를 반복합니다. (증감 값 자리의 a++ 는 a = a+1 과 같습니다)

```
int result = 20;
for(int a=0; a<=5; a++){
    result += a;
    System.out.print("result의 값은 = " + result);
}
```

```
for( ; ; ){
      // 경우에 따라 시작값, 종료값, 증감값을 생략하고 사용할 수 있다
      // 의도적인 무한루프
}
```

4. Java 기초 – 반복문 - for

2. for 중첩 – 각각의 제어문은 중첩해서 사용할 수 있습니다.

```
for( int i=0 ; i < 10; i++) {
    for( int j=0 ; j < 15 ; j++) {
        for(int k=0 ; k < 20 ; k++) {
            // 반복 코드
        }
    }
}
```

- 중첩코드 출력하기

```
for(int i=1 ; i<=2 ; i++) {
    for(int j=1 ; j<=3 ; j++) {
        System.out.println("i=" + i + ", j=" + j);
    }
}
```

예제 출력결과

```
i=1, j=1
i=1, j=2
i=1, j=3
i=2, j=1
i=2, j=2
i=2, j=3
```

5. Java 기초 – 반복문 - while

1. while문은 조건이 참일 동안 블럭 안의 코드를 반복 실행합니다. 반복이 가능한 if 문이라고 생각하면 쉽게 접근할 수 있습니다

```
while(조건){
  // 코드;
}
```

2. 2. do while 문은 코드를 한번 실행 하고 조건을 비교한다

```
do{
    // 여기 코드를 무조건 1번은 실행한다
} while ( 조건 );

// while 의 조건이 참일 경우 do 블록을 다시 실행한다
// 끝에 ; (세미콜론)을 붙여야한다
```

6. Java 기초 – break, continue

1. break
   break 는 switch 또는 반복문을 종료시켜준다

- switch 에서의 break

```
switch(param){
   case 10:      //param 이 10이면
      코드1;     // 코드1을 실행하고
      break;     // break문을 만났기 때문에 switch 문이 종료된다.
   case 20:      // break 문이 없으면 case 20 의 코드2도 실행된다
      코드2;
}
```

- 반복문에서의 break

```
for ( int i=0 ; i < 10 ; i++ ) {
   // 코드 1
   break;   // break 명령어를 만나면 코드 2를 실행하지 않고 for 문이 종료된다
   // 코드 2
}
```

6. Java 기초 – break, continue

2. continue
continue는 반복문에서 실행순서를 반복문 블록의 끝으로 이동시킵니다

```
for(int i=1 ; i<=10 ; i++){
      if(i <= 5) continue;     // continue 시 아래로직은 무시된다
      System.out.println("i의 값은 = " + i);
}
```

예제 출력결과
i의 값은 = 6
i의 값은 = 7
i의 값은 = 8
i의 값은 = 9
i의 값은 = 10

조건에서 i 가 5보다 작거나 같을경우
continue
명령을 실행하기 때문에
System.out 을 실행하지 않고
for 문의 끝으로 이동 후
다음 조건을 실행한다.

6. Java 기초 – break, continue

3. 중첩된 반복문에서 break와 continue

- 중첩된 반복문에서의 continue

```
for(int i=1 ; i<=3 ; i++) {
    for(int j=1 ; j<=2 ; j++) {
        if(i<=2) continue;
        System.out.println(i+" "+j);
    }
    System.out.println("end j");
}
```

중첩된 반복문에서 continue는
해당 블럭에만 영향을 미친다
따라서 예제의 continue 는
내부에 있는 for 문의 끝으로 실행순서를 이동시킨다

- 중첩된 반복문에서의 break;

```
for(int i=1 ; i<=3 ; i++) {
    for(int j=1 ; j<=2 ; j++) {
        if(i<=2) break;
        System.out.println(i+" "+j);
    }
    System.out.println("end j");
}
```

중첩된 반복문에서 break 또한
해당 블럭에만 영향을 미친다
따라서 내부의 for 문이 종료된다

7. Java 기초 – 배열

프로그래밍 언어의 변수에는 하나의 값만 담을 수 있지만 변수를 배열이라는 형태로 선언하면 여러 개의 값을 하나의 변수에 담을 수 있습니다.

1. 배열 초기화

```
// 값으로 초기화 하기
int numbers[ ] = {1, 2, 3, 4, 5};  // 5개의 값이 담긴 배열로 초기화된다

// 입력할 값의 개수로 초기화 하기
int numbers[ ] = new int[7]     // 7 개의 값을 담을 수 있는 배열로 초기화된다
```

2. 값 넣고 빼기 : 0번 부터 시작하는 index(위치)로 값을 넣고 뺄수 있다

```
int numbers[ ] = {1, 2, 3, 4, 5};  // 5개의 값이 담긴 배열로 초기화된다

numbers[2] = 15; // numbers 변수의 세 번째 값인 3이 15로 변경된다
                 배열은 위치(index)가 0 부터 시작한다
System.out.println(numbers[0]);  // 0번 index 값인 1이 출력된다
```

7. Java 기초 – 배열

3. 반복문으로 String 배열 출력하기

```
String days[ ] = {"월", "화", "수", "목", "금", "토", "일"};

// 1. 일반적인 반복문
int daysLength = days.length; // 배열의 크기를 가져와서 변수에 담는다
for(int i=0; i<daysLength; i++) { // 배열의 크기만큼 반복한다
    System.out.println(days[i]); // 월 화 수 목 금 토 일 이 출력된다
}

// 2. 향상된 반복문
for(String day : days) { // days배열에 담겨있는 값을
                         // 개수만큼 반복하면서 day 변수에 담는다
    System.out.println(day); // 반복문 영역 안에서 day 변수를 사용할 수 있다
                             // 월 화 수 목 금 토 일 이 출력된다
}
```