강 사 : 전 은 석
email : euns_jun@naver.com

Day 3

조건 연산자 (삼항 연산자)

하나의 조건을 정의하여 만족 시에는 '참값'을 반환하고 만족하지 못할 시에는 '거짓값'을 반환하여 단순 비교에 의해 변화를 유도하는 연산자이다.

형식

```
조건식 ? 참일때 반환값 : 거짓일때 반환값 ;
```

예제

```
int no1 = 10;
int no2 = 20;
String str1 = ( no1 > no2 ) ? ("no1이 no2보다 크다.") : ("no1이 no2 보다 작다.");
```

랜덤한 숫자 얻는 방법

```
int num = (int) Math.random() * (큰수 - 작은수 +1) + 작은수;
```

❖ 주석문

주석 종류	의미	설명
// 주석문	단행 주석처리	현재 행에서 //의 뒷문장부터 주석으로 처리된다.
/* 주석문 */	다행 주석처리	/*에서 */ 사이의 문장이 주석으로 처리된다.
/** 주석문 */	HTML 문서화 주석처리	/**에서 */ 사이의 문장이 주석으로 처리된다. 장점은 HTML 문서화로 주석이 처리되므로 API와 같은 도움말 페이지를 만들 수 있다.

키보드를 이용해서 데이터 입력받기

1. class가 만들어지기 이전에

   import java.util.*;

   이라는 명령을 이용해서 사용할 라이브러리를 등록한다.
2. 프로그램에 들어가서(우리는 주로 main 함수 안에서)

   Scanner sc = new Scanner(System.in);

   이라는 명령을 이용해서 키보드를 통해서 입력 받을 도구를 준비한다.
3. 키보드를 통해서 입력받을 필요가 생기면....

   변수 = sc.nextXXX();

   명령을 이용해서 데이터를 입력 받으면 된다.

   이때 XXX는 입력받을 데이터의 종류에 따라 달라진다.

1. sc.nextLine(); 문자열
2. sc.nextInt(); 정수
3. sc.nextFloat(); 실수

콘솔창에 출력하기

- System.out.println(내용);
  - 내용을 출력하고 줄바꿈이 된다.

- System.out.print(내용);
  - 내용을 출력하고 줄바꿈이 안된다.

- System.out.printf("형식형식형식 ... ", 내용, 내용, 내용, ...);
  - 여러 내용을 지정한 형식으로 출력한다.

형식화문자	내 용
%d	정수값을 10진수로 출력
%o	정수값을 8진수로 출력
%x	정수값을 16진수로 출력
%f	실수값을 소수 방식으로 출력
%e	실수값을 지수 방식으로 출력
%c	문자를 출력
%s	문자열을 출력
%b	논리값을 출력
%n	줄바꿈

예제

%10d

: 10진수를 10자리에 맞춰 출력

%10.4f

: 실수를 10자리에 맞게 출력하는데 소수이하는 4자리를 출력

```
System.out.printf(
        "%5d,%n%5d,%n%5d",
        168 & 245,
        168 | 245,
        168 ^ 245);
```

✥ 제어문

프로그램의 흐름에 영향을 주고 때에 따라 제어가 가능하도록 하는 것 (조건문, 반복문)

❖ 조건문

조건을 주고 조건의 일치 여부에 따라 실행을 결정하는 명령문

◆ if 문

조건이 맞을때에만 실행문을 실행할 조건문

✧ 형식

```
if ( 조건식 ) {  실행문 }
```

◆ if-else문

조건이 맞았을때와 맞지 않았을때의 실행문을 지정해 놓은 조건문

✧ 형식

```
if ( 조건식 ) {
    조건 일치시 실행문
} else {
    조건 불일치시 실행문
}
```

강 사 : 전 은 석
email : euns_jun@naver.com

◆ 다중 if(if-else if) 문

조건이 두개 이상일 경우 각 경우에 따라 실행 명령을 지정해 놓은 조건문

형식

```
if ( 조건식1 ) {
    조건식1 일치시 실행문
} else if(조건식2){
    조건식2 일치시 실행문
} else if(조건식3){
    조건식3 일치시 실행문
}
. . .
else {
    모든 조건에 맞지 않을때 실행문
}
```

강 사 : 전 은 석
email : euns_jun@naver.com

◆ switch문

인자값의 경우 따라 실행문장을 만들어두는 조건문

*** 수행문장 다음에 반드시 "break;" 문을 기술한다.

형식

```
switch ( 인자값 ) {
    case 조건값1:
        실행문;
        break;
    case 조건값2:
        실행문;
        break;
    case 조건값3:
        실행문;
        break;
    break;
    . . .
    default:
        실행문;
}
```

강 사 : 전 은 석
email : euns_jun@naver.com

❖ 반복문

실행문을 반복 수행하게 할때 사용하는 제어문

- 카운터 변수 : 반복횟수를 기록할 변수
- 조건식 : 반복을 종료할 조건식
- 증감식 : 반복될때마다 카운터 변수를 증감 해주는 명령

◆ for 반복문

카운터변수와 반복이 종료될 조건식이 있는 반복 명령

형식

```
for( 카운터변수 초기화; 조건식 ; 카운터변수 증감식){
        실행문
}
```

순서 : 1 -> 2 -> 4 -> 3 -> 2 -> 4 -> 3 -> . . . 2 -> 5

예제

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

향상된 for 명령

시퀀스 자료형(인덱스가 있는 자료형)에서 유용하게 쓸 수 있는 반복 명령
인덱스 순서로 하나씩 꺼내서 변수에 대입한다.

형식 : 매개변수에 담겨있는 데이터 타입과 동일한 타입의 변수를 만들어준다.

```
for( 데이터타입    변수이름 : 시퀀스자료 변수이름 ){
    실행문;
}
```

- 예제] 정수 자료 6개(1, 2, 3, 4, 5, 6)를 넣은 배열의 데이터를 순서대로 출력

```
int[] noArray = {1, 2, 3, 4, 5, 6};
int cnt = 0;
for(int num : noArray){
    System.out.println("index : " + cnt++ + " / 데이터 : " +  num);
}
```

강 사 : 전 은 석
email : euns_jun@naver.com

while 명령

for 반복문과 비교해서 카운터변수를 따로 만들어줘야한다는 차이점이 있다.
조건이 참이면 반복하는 명령
카운터변수의 생성과 처리는 따로 해줘야 한다.

형식

```
카운터변수 선언;
while( 반복조건 ){
    실행문;
    카운터변수 증감식;
}
```

- 예제] 정수 자료 6개(1, 2, 3, 4, 5, 6)를 넣은 배열의 데이터를 순서대로 출력

```
int[] noArray = {1, 2, 3, 4, 5, 6};
int cnt = 0;
while( cnt < noArray.length ){
    System.out.println("index : " + cnt++ + " / 데이터 : " +  num);
}
```

강 사 : 전 은 석
email : euns_jun@naver.com

do - while 명령

다른 반복문과 비교해서 최소 한번은 반드시 실행한다는 차이점이 있다.
while 문과 비교해 조건이 맨 마지막에 온다는 점이 다르다.

형식

```
카운터변수 선언;
do {
    실행문;
    카운터변수 증감식;
} while ( 반복조건 );
```

- 예제] 정수 자료 6개(1, 2, 3, 4, 5, 6)를 넣은 배열의 데이터를 순서대로 출력

```
int[] noArray = {1, 2, 3, 4, 5, 6};
int cnt = 0;
do{
    System.out.println("index : " + cnt + " / 데이터 : " +  num);
} while( cnt++ < noArray.length )
```

강 사 : 전 은 석
email : euns_jun@naver.com

break 명령

반복문과 switch문 또는 레이블이 붙은 반복문을 종료
가장 가까운 반복문을 종료

continue 명령

반복문을 다음 회차로 다시 반복
가장 가까운 반복문을 다시 실행
continue 다음에 레이블이 붙을 경우 해당 레이블이 붙은 반복문의 다음회차로 진행

형식

```
반복문( 반복조건 ){
      if( 조건식 ) {
          break( 또는 continue ) [레이블];
     }
     실행문;
}
```

강 사 : 전 은 석
email : euns_jun@naver.com

- 예제] 정수를 입력받아 그 숫자가 짝수인지 홀수인지 출력하세요.
  7번 반복하고 중간에 3이 나오면 종료하세요.

```
import java.util.*;
... 생략 ...
Scanner sc = new Scanner(System.in);
int no = 0;
int cnt = 0;


while(cnt++ < 7){
    no = sc.nextInt();
    if ( no == 3 ){
        break;
    } else {
        System.out.println(
          cnt + " 번째 입력값 : " + no + " / " + (no % 2 == 0 ? "짝수" : "홀수");
        continue;
    }
}
```

강 사 : 전 은 석
email : euns_jun@naver.com

배열

같은 자료형들끼리 모아두는 하나의 묶음

담을수 있는 자료의 타입과 길이가 먼저 정해진다.

선언 형식1

```
데이터타입[] 변수이름;
```

선언 형식2

```
데이터타입 변수이름[];
```

❖ 배열

❧ 배열 생성 형식1 : 배열 객체만 생성하는 방법

```
데이터타입 변수이름[] = new 데이터타입[길이];
```

📌 **예제] 정수 자료 6개를 넣을 배열을 만드세요.**

```
int[] noArray = new int[6];
```

❧ 배열 생성 형식2 : 데이터를 생성과 동시에 입력하는 방법

```
데이터타입 변수이름[] = {데이터1, 데이터2, 데이터3, . . . };
```

📌 **예제] 정수 자료 6개(1, 2, 3, 4, 5, 6)를 넣을 배열을 만드세요.**

```
int[] noArray = {1, 2, 3, 4, 5, 6};
```