반복문

while, for

while문

- while(){...} 구조로 if문 구조가 똑같다.
- ()안의 조건을 만족하는 동안 {}블록 안을 **반복**한다.

```
while(true) {
    System.out.println("안녕하세요");
}
```

while문

- while(false){...} 은 컴파일 에러! (수행될 일이 없는 코드)
- () 항상 true일 경우 **무한반복** 되기때문에 **조건 설정 중요!**
- 특정 횟수만큼 반복하고 싶다면 변수를 선언하여 반복횟수를 설정할 용도로 사용해야한다.

```
int i=0;

while(i<10) {
    System.out.println("안녕하세요");

    i++;
}
```

for문

- while문의 산만한 코드를 깔끔하게 만든 형태 (int i=0; i++)
- for(초기화; 조건식; 증감식){...} 의 구조를 가진다.
- 초기화, 조건식에는 (;)이 들어가지만 증감식에는 없다!

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

while과 for

- for문의 i는 반복이 끝나면 사라지지만, while에서 사용한 i는 반복문 위쪽에 선언했기 때문에 사라지지 않는다.
- while문은 반복횟수를 알 수 없을 때,

  for문은 반복횟수를 알 수 있을 때 주로 사용한다.
- while과 for문은 서로 대체 가능하지만, 분명히 위와 같은 이유로 쓰임새가 나뉘는 상황이 반드시 온다! (둘다 기억하자)

break

- **반복문을 중단**하고 싶을 때 사용
- 반복도중 특정 조건을 만족하면 반복문을 빠져나오고 싶은 경우

```
int i=0;
while(i<10) {

    if(i == 5) break;

    System.out.println("안녕하세요");
    i++;
}
```

```
for(int i=0; i<10; i++) {

    if(i == 5) break;

    System.out.println("안녕하세요");
}
```

continue

- 특정 조건에서 아래 실행문을 **건너뛰고 싶은 경우** 사용
- 반복문을 빠져나가는 것이 아니라 **건너뛰고 계속 반복**한다.

```
int i = -1;
while(i<9) {
    i++;

    if(i == 5) continue;

    System.out.println(i + " 안녕하세요");
}
```

```
for(int i=0; i<10; i++) {

    if(i == 5) continue;

    System.out.println(i + " 안녕하세요");
}
```

중첩 반복문

- 반복문 안에서 다시 반복문이 나타날 수 있다.
- 마치 시계의 시,분,초 처럼 동작한다.

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