



4. 제어의 흐름 이해하기

1. 조건문
2. 반복문

조건문

주어진 조건에 따라 다른 수행문이 실행되도록 프로그래밍 하는 것

if 문

```
if(조건식) {  
    수행문;  
}
```

```
if(age >= 8) {  
    System.out.println("학교에 다닙니다");  
}
```

if – else 문

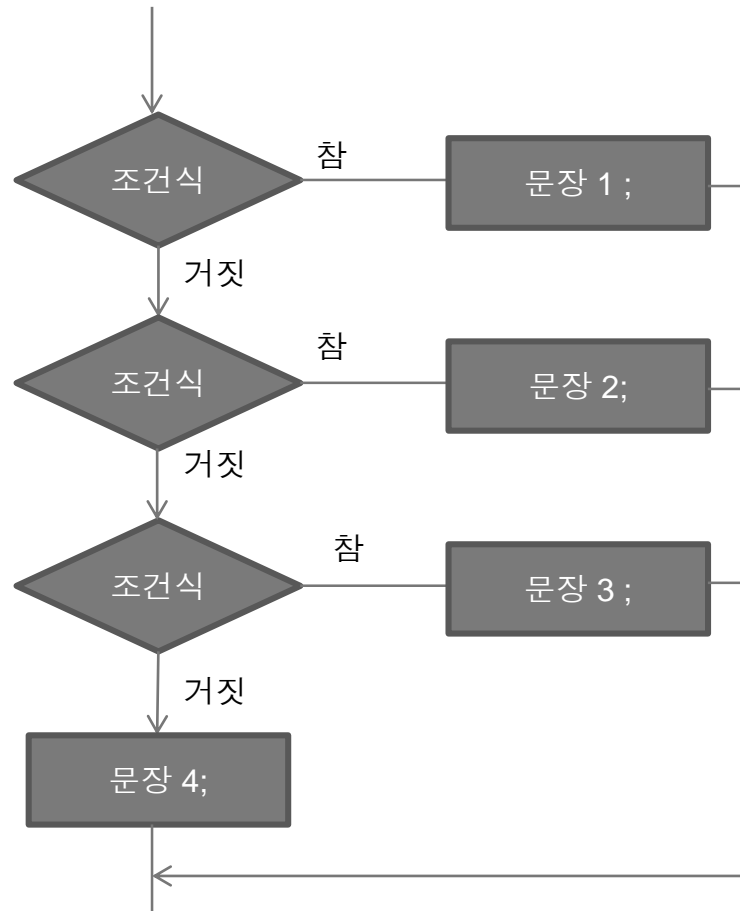
```
if(조건식) {  
    수행문1;  
}  
else {  
    수행문2;  
}
```

```
if(age >= 8) {  
    System.out.println("학교에 다닙니다");  
}  
else {  
    System.out.println("학교에 다니지 않습니다.");  
}
```

if-else if-else 문

하나의 경우에 조건이 여러 개에 해당하는 경우

```
if (조건 1)
{
    문장 1:
}
else if(조건 2)
{
    문장 2:
}
else if(조건 3)
{
    문장 3:
}
else
{
    문장 4:
}
```



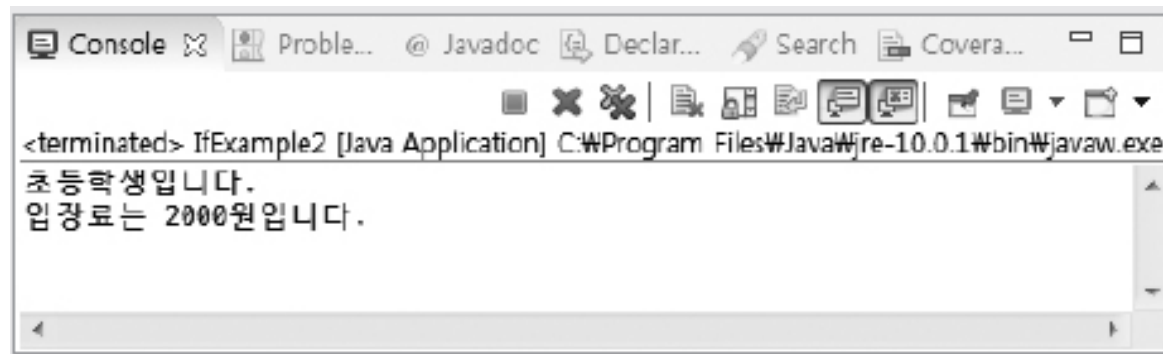
if-else if-else 문 예시

```
package ifexample;

public class IfExample2 {
    public static void main(String[] args) {
        int age = 9;
        int charge;

        if(age < 8) {
            charge = 1000;
            System.out.println("취학 전 아동입니다.");
        }
        else if(age < 14) {
            charge = 2000;
            System.out.println("초등학생입니다.");
        }
        else if(age < 20) {
            charge = 2500;
            System.out.println("중, 고등학생입니다.");
        }
        else {
            charge = 3000;
            System.out.println("일반인입니다.");
        }
        System.out.println("입장료는" + charge + "원입니다.");
    }
}
```

출력문에서 +를 사용하면 여러 단어를 연결하여 출력할 수 있습니다. '11-2 String 클래스'에서 자세히 설명합니다.



조건문과 조건 연산자

간단한 if-else조건문은 조건 연산자로 구현할 수 있음

```
if(a > b)
    max = a;
else
    max = b;
```

if-else문



```
max = (a > b) ? a : b;
```

조건 연산자

switch-case문

조건식의 결과가 정수 또는 문자열의 값이고 그 값에 따라 수행문이 결정될 때
if-else if-else문을 대신하여 **switch-case**문을 사용할 수 있습니다.

```
if(rank == 1) {  
    medalColor = 'G';  
}  
else if(rank == 2) {  
    medalColor = 'S'  
}  
else if(rank == 3) {  
    medalColor = 'B'  
}  
else {  
    medalColor = 'A'  
}
```



```
switch(rank) {  
    case 1 : medalColor = 'G';  
            break;  
    case 2 : medalColor = 'S';  
            break;  
    case 3 : medalColor = 'B';  
            break;  
    default : medalColor = 'A';  
}
```

순위에 따른 메달의 색을
정해주는 코드 switch-case 문으로
구현 가능

case문에 문자열 사용하기

자바 7부터 **switch-case** 문의 **case** 값에 문자열 사용 가능

```
public class SwitchCase2 {  
    public static void main(String[ ] args) {  
        String medal = "Gold";  
  
        switch(medal) {  
            case "Gold":  
                System.out.println("금메달입니다.");  
                break;  
            case "Silver":  
                System.out.println("은메달입니다.");  
                break;  
            case "Bronze":  
                System.out.println("동메달입니다.");  
                break;  
            default:  
                System.out.println("메달이 없습니다.");  
                break;  
        }  
    }  
}
```

반복문

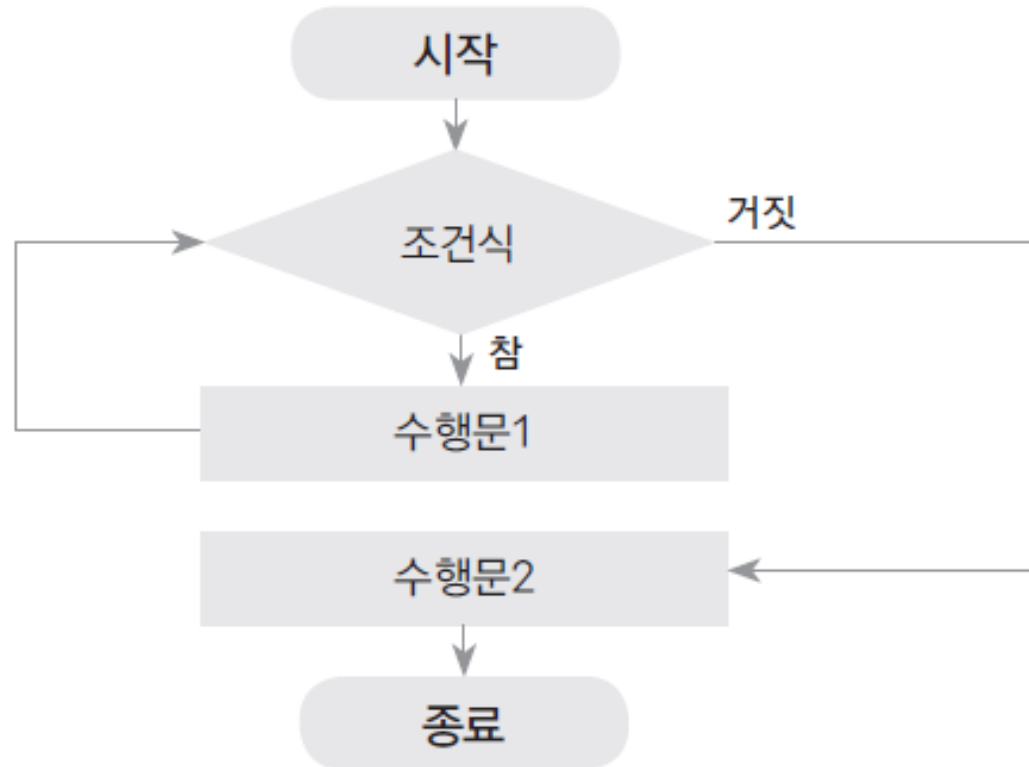
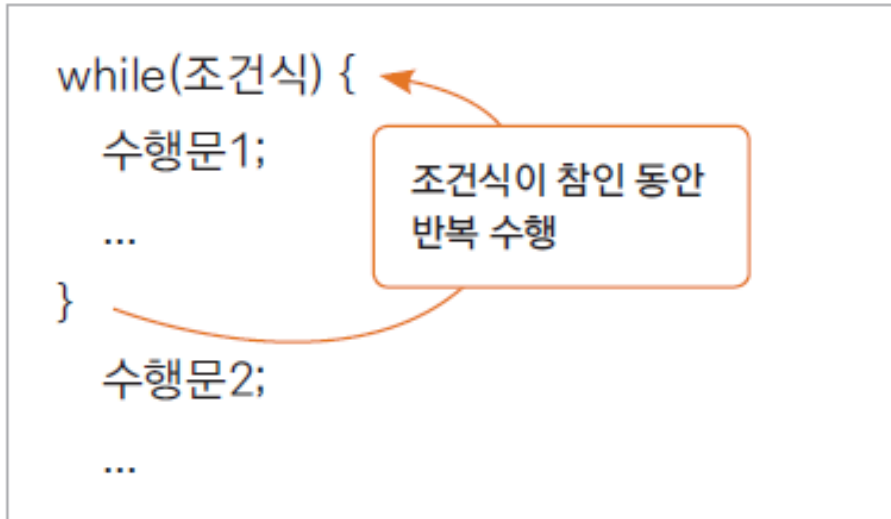
주어진 조건이 만족 할 때까지 수행문을 반복적으로 수행 함

while, do-while, for 문이 있음

조건의 만족과 반복 가능 여부에 대해 정확한 코딩을 해야 함

while 문

조건식이 참인 동안 수행문을 반복해서 수행

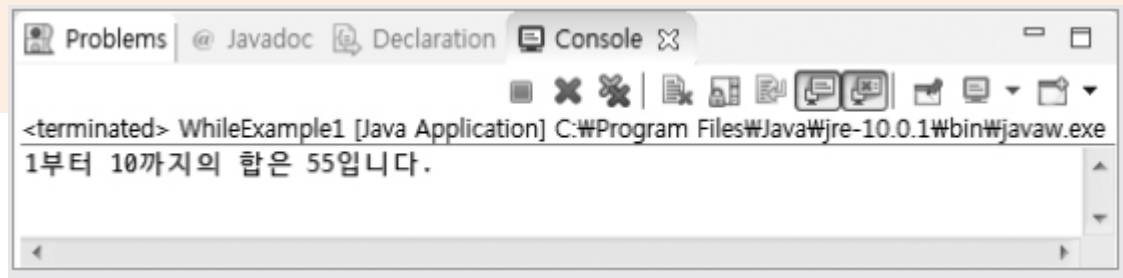


while 문 예

1부터 10까지 더하기

```
public class WhileExample1 {  
    public static void main(String[ ] args) {  
        int num = 1;  
        int sum = 0;  
  
        while(num <= 10) { // num 값이 10보다 작거나 같을 동안  
            sum += num;      // 합계를 뜻하는 sum에 num을 더하고  
            num++;           // num에 1씩 더해 나감  
        }  
        System.out.println("1부터 10까지의 합은 " + sum + "입니다.");  
    }  
}
```

조건식이 참인 동안 반복 수행

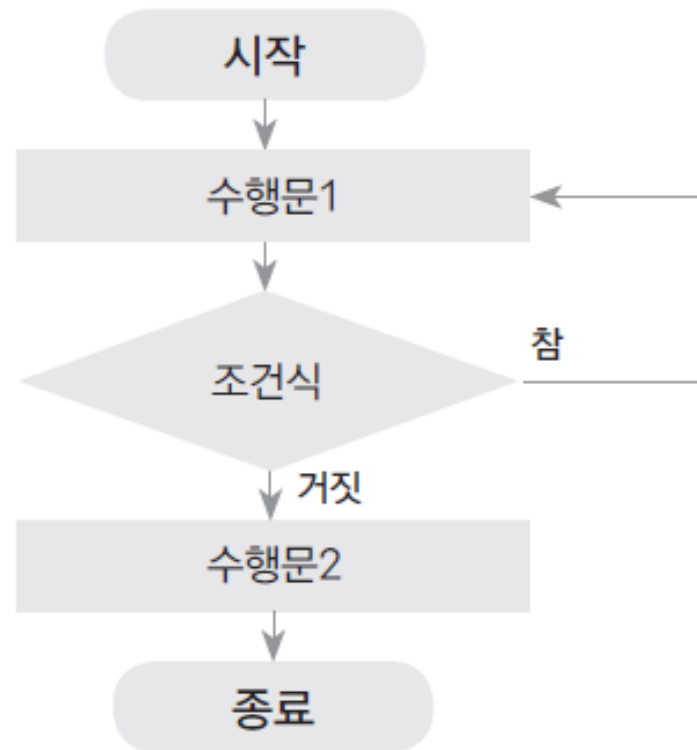


do-while 문

먼저 수행문을 한 번 수행하고 조건식 체크

수행문이 반드시 한 번 이상 수행 되어야 하는 경우 사용

```
do {  
    수행문1;  
    ...  
} while(조건식);  
    수행문2;  
    ...
```

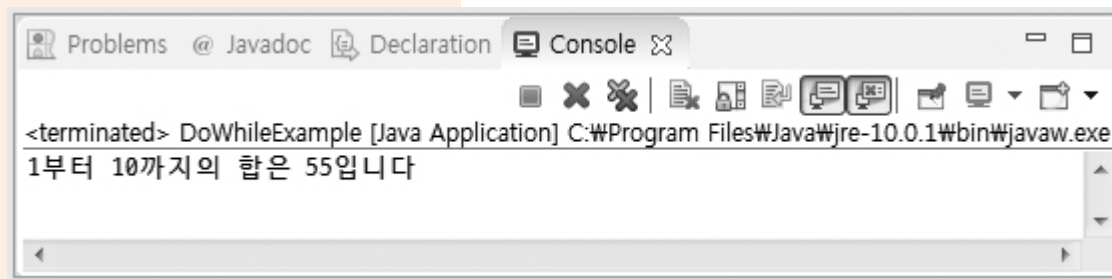


do-while 문 예

1부터 10까지 더하기

```
public class DoWhileExample {  
    public static void main(String[ ] args) {  
        int num = 1;  
        int sum = 0;  
  
        do {  
            sum += num;  
            num++;  
        } while(num <= 10);  
  
        System.out.println("1부터 10까지의 합은 " + sum + "입니다");  
    }  
}
```

조건식이 참이 아니더라도 무조건 한 번 수행함



for 문

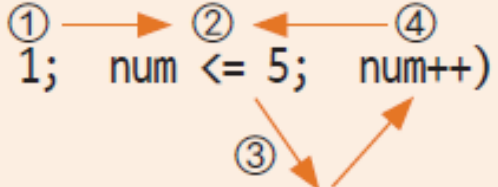
반복문 중 가장 많이 사용하는 반복문
주로 조건이 횟수인 경우에 사용
초기화식, 조건식, 증감식을 한꺼번에 작성

```
for(초기화식; 조건식; 증감식) {  
    수행문;  
}
```

for 문 수행 과정

num 이 1에서 부터 5일 때 까지 하나씩 증가하면서 출력하는 for문

```
int num;  
for(num = 1; num <= 5; num++)  
{  
    System.out.println(num);  
}
```



- ① 1초기화는 한번만 수행하고
- ② 조건식이 만족하면
- ③ 수행문 수행
- ④ 증감식 수행
- ⑤ 조건식이 만족하는지 체크
- ⑥ 조건식이 만족하면 수행문 수행
- ⑦ 조건식이 만족하지 않으면 for문 빠져 나옴

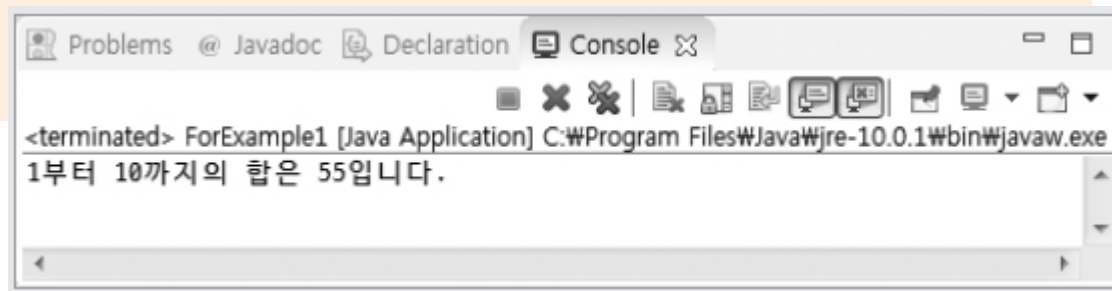
num 값	1(초기화)	2	3	4	5	6
조건식 (num <= 5)	참	참	참	참	참	거짓
출력 값	1	2	3	4	5	for문 종료
증감식	수행	수행	수행	수행	수행	x

for 문 예제

1부터 10까지 더하기

```
public class ForExample1 {  
    public static void main(String[ ] args) {  
        int i;  
        int sum;  
        for(i = 1, sum = 0; i <= 10; i++) {  
            sum += i;  
        }  
  
        System.out.println("1부터 10까지의 합은 " + sum + "입니다.");  
    }  
}
```

for문에서 가장 자주 사용하는 변수 이름은 i입니다. 주로 횟수를 표현합니다.



각 반복문의 쓰임

while 문

하나의 조건에 대해 반복 수행이 이루어질 때까지 사용
조건이 맞지 않으면 수행문이 수행되지 않음
주로 조건식이 **true, false**로 나타남

do-while문

하나의 조건에 대해 반복수행이 이루어질 때 사용
단, 수행문이 반드시 한번 이상 수행됨

for 문

수의 특정 범위, 횟수와 관련한 반복수행에서 주로 사용

무한 반복

```
while(true){  
    수행문;  
}
```

```
do{  
    수행문;  
}while(true);
```

```
for(;;){  
    수행문  
}
```

중첩된 반복문

반복문 내부에 또 반복문이 사용 됨
구구단의 예

2단부터 9단까지 반복하는 외부 반복문

```
for(dan = 2; dan <= 9; dan++) {  
    for(times = 1; times <= 9; times++) {  
        System.out.println(dan + "X" + times + "=" + dan * times);  
    }  
    System.out.println( ); // 한 줄 띄워서 출력  
}
```

각 단에서 1~9를 곱하는 내부 반복문

```
}  
}
```

continue 문

반복문과 함께 쓰이며, 반복문 내부 **continue** 문을 만나면
이후 반복되는 부분을 수행하지 않고 조건식이나 증감식을 수행함
1부터 100까지 중 홀수만 더하는 예

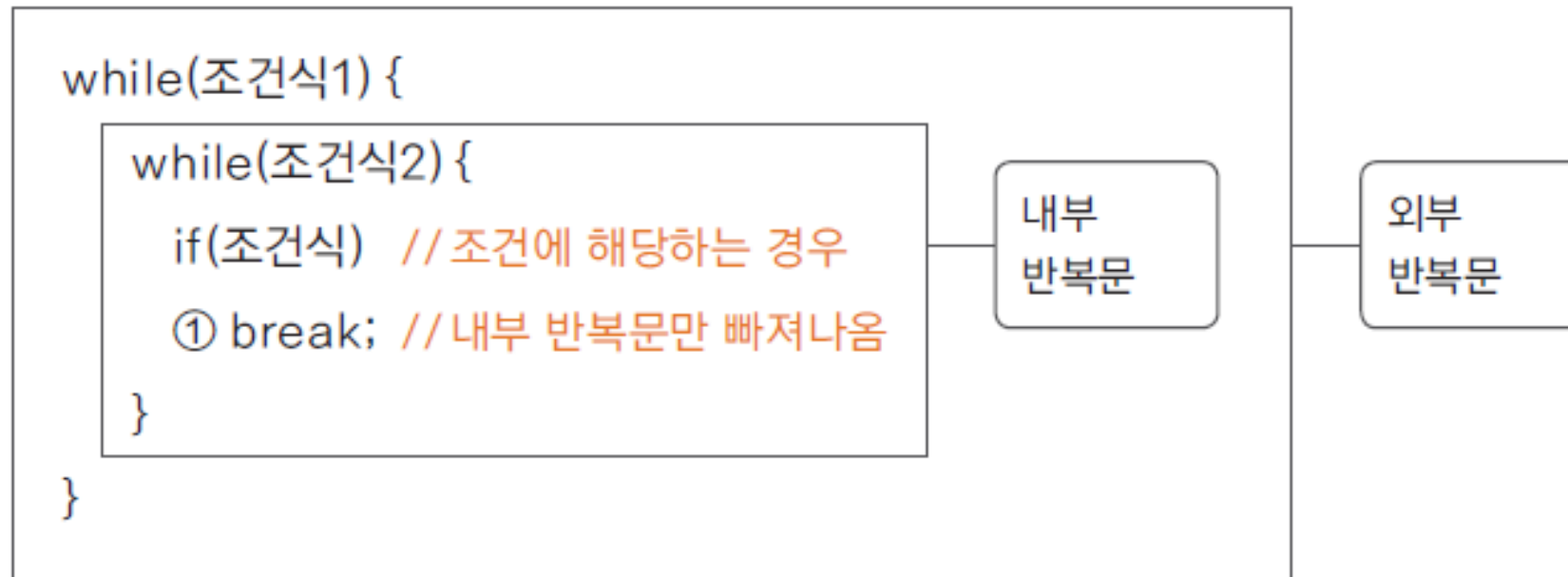
```
public class ContinueExample {  
    public static void main(String[] args) {  
        int total = 0;  
        int num;  
  
        for(num = 1; num <= 100; num++) {  
            if(num % 2 == 0)  
                continue;  
            total += num;  
        }  
        System.out.println("1부터 100까지의 홀수의 합은: " + total + "입니다.");  
    }  
}
```

// 100까지 반복
// num 값이 짝수인 경우
// 이후 수행을 생략하고 num++ 수행
// num 값이 홀수인 경우에만 수행

break 문

반복문에서 **break** 문을 만나면 더 이상 반복을 수행하지 않고 반복문을 빠져 나옴

중첩된 반복문 내부에 있는 경우 가장 가까운 반복문 하나만 빠져 나옴 아래의 경우 내부 반복문만 빠져 나옴



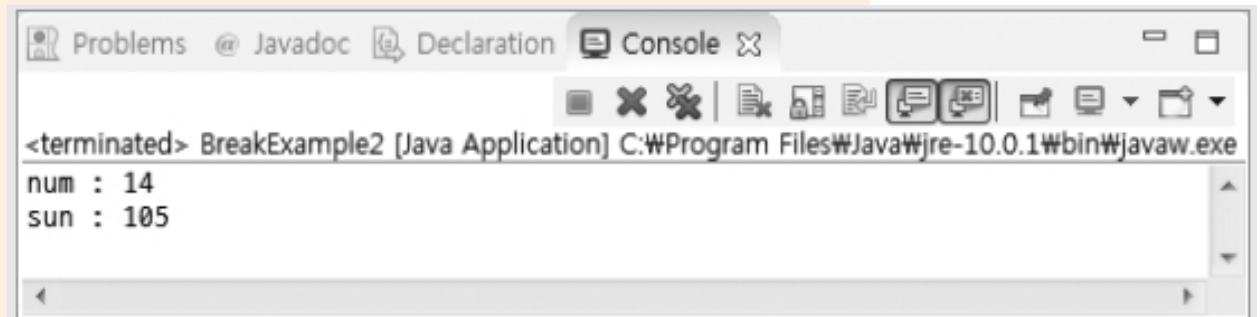
break 문 예제

0부터 시작하여 1씩 늘리며 숫자의 합이 100이 초과하는 경우 그 수와 합을 구하는 예제

```
public class BreakExample2 {  
    public static void main(String[ ] args) {  
        int sum = 0;  
        int num = 0;  
  
        for(num = 0; ; num++) {  
            sum += num;  
            if(sum >= 100)  
                break;  
        }  
        System.out.println("num : " + num);  
        System.out.println("sum : " + sum);  
    }  
}
```

조건식을 생략하는 대신 break문을
사용합니다.

// sum이 100보다 크거나 같을 때(종료 조건)
// 반복문 중단



The screenshot shows a Java IDE window with the 'Console' tab selected. The title bar indicates the application is 'BreakExample2 [Java Application]' running at 'C:\Program Files\Java\jre-10.0.1\bin\javaw.exe'. The console output displays the results of the program: 'num : 14' and 'sum : 105'. The output is preceded by a '<terminated>' status message.

```
<terminated> BreakExample2 [Java Application] C:\Program Files\Java\jre-10.0.1\bin\javaw.exe  
num : 14  
sum : 105
```