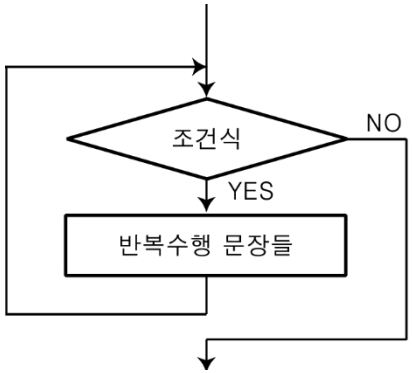
선조건 반복

- **while 문도 for 문과 똑같이 반복된 수행을 하는 경우 사용됨**
- **횟수가 정해져 있는 경우에 사용되는 for 문과는 다르게 while 문은 반복 횟수가 정해져 있는 것이 아니라 어떠한 종료 조건이 주어진 경우 사용**
- **조건식이 참인 경우 반복 수행, 거짓인 경우 반복을 벗어남**



- **특정 조건(어떤 값이 입력)이 될 때까지 반복 수행하는 전형적 구조는 다음과 같다.**

```
첫 번째 data 입력;

while(조건) {
      반복 수행할 문장들;
      다음 data 입력;
}
```

반복문

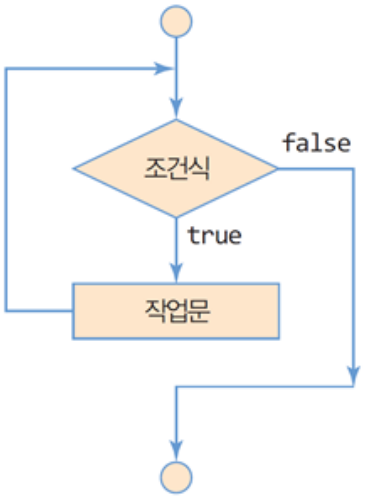
▣ while문

while문은 조건이 만족하는 동안 특정 문장을 수행시키는 기능이다. 처음부터 조건을 만족하지 않으면, 반복 문장을 한번도 수행하지 않는다. 다만 조건이 참이면 while 블록은 무한 루프에 빠지게 되므로 조건문이나 while문 블록 내에 조건을 거짓으로 만드는 문장이 있어야 한다.

◎ 형식

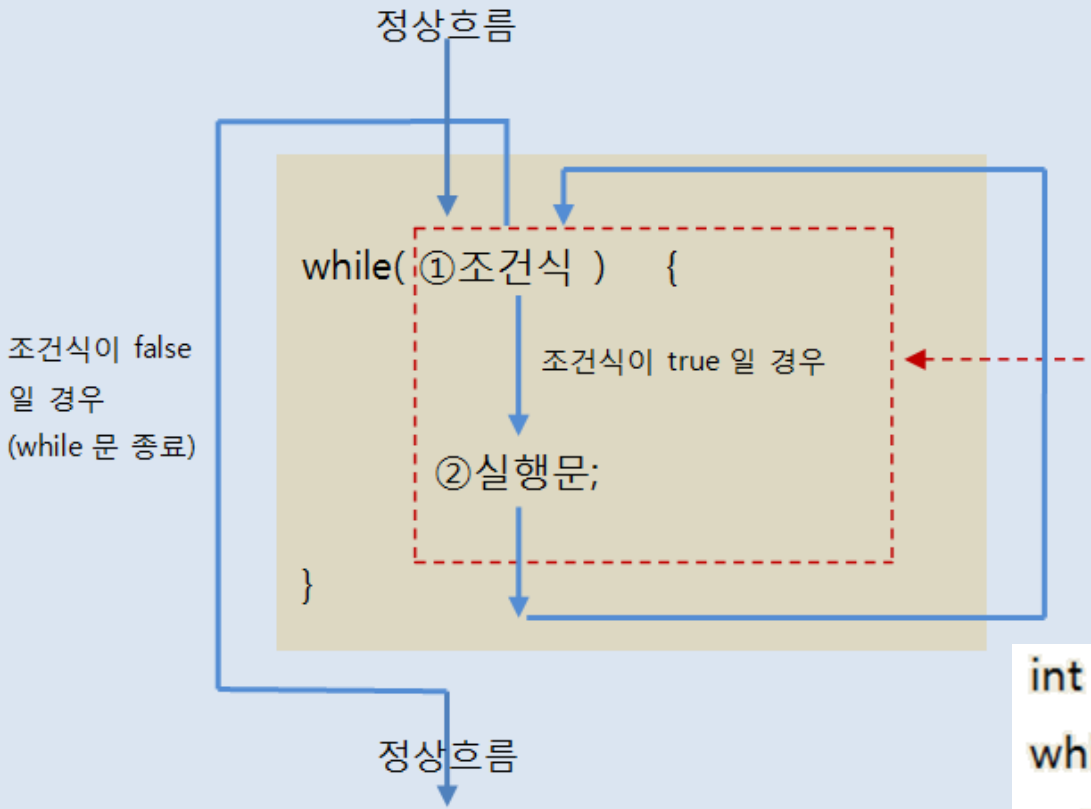
```
while(식)
    처리문1;
처리문2;
```

```
while(식) {
    처리문1;
    처리문2;
        ...
}
```



반복문(for문, while문, do-while문)

❖**while문: 조건에 따라 반복을 계속할지 결정할 때 사용**

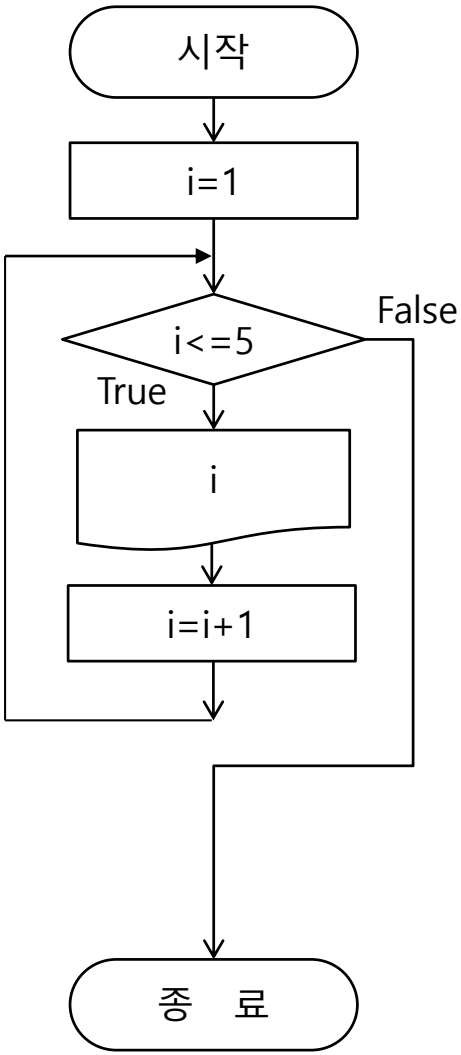



```
int i = 1;
while (i<=10) {
    System.out.println(i);
    i++;
}
```

선조건 반복 기초_예제 2

1부터 5까지 출력하는 순서도와 코드를 작성하시오.

flowchart



pseudocode

```
1     public static void main(String[] args) {
2         int i = 1;
3         while(i <= 5) {
4             System.out.print( i + " ");
5             i = i + 1;
6         }
7     }
```

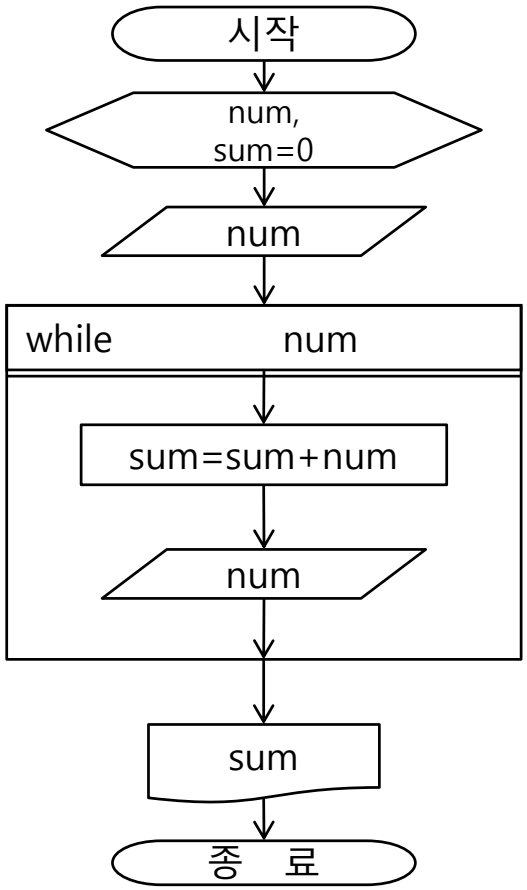
실행 예

```
1 2 3 4 5
```

선조건 반복 기초_예제 2

-1 이 입력될 때까지 정수를 입력받아 그 수들의 합을 출력하는 순서도와 슈도코드를 작성하시오.

flowchart



pseudocode

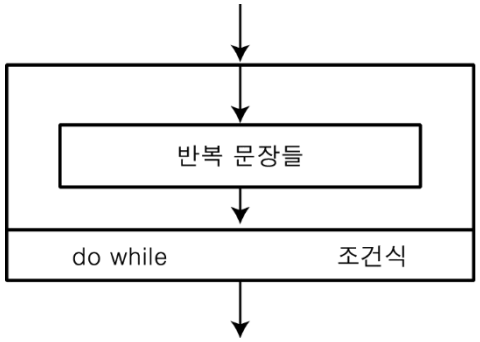
```
1     public static void main(String[] args) {
2         int num, sum = 0;
3         Scanner input = new Scanner(System.in);
4         System.out.print("수를 입력해주세요 : ");
5         num = input.nextInt();
6         while(num              ){
7             sum = sum + num;
8             System.out.print("수를 입력해주세요 : ");
9             num = input.nextInt();
10        }
11        System.out.print("합은 " + sum + "이다.");
12        input.close();
13    }
```

실행 예

```
수를 입력해주세요 : 2
수를 입력해주세요 : 5
수를 입력해주세요 : 4
수를 입력해주세요 : -1
합은 11이다.
```

반복 기초

- do - while 문은 조건 보다 먼저 수행
- 수행 후에 조건을 검사하여 참인 동안은 반복 문장을 계속 수행하고 거짓이면 do - while 문을 벗어남



①반복 문장들을 수행

②조건식을 검사

③②에서 조건식이 참이면 1부터 다시 실행

④②에서 조건식이 거짓이면 do - while 문을 벗어남

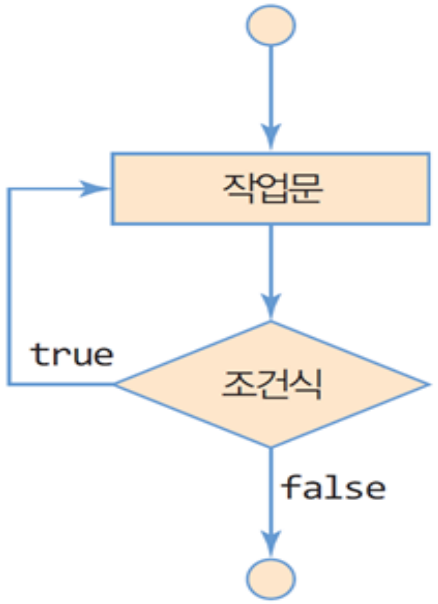
반복문

▣ do ~ while문

반복문을 수행한 후에 조건만족을 확인한다. 조건을 나중에 확인하므로, 조건이 참이던, 거짓이어도 꼭 한번은 실행한다.

◎ 형식

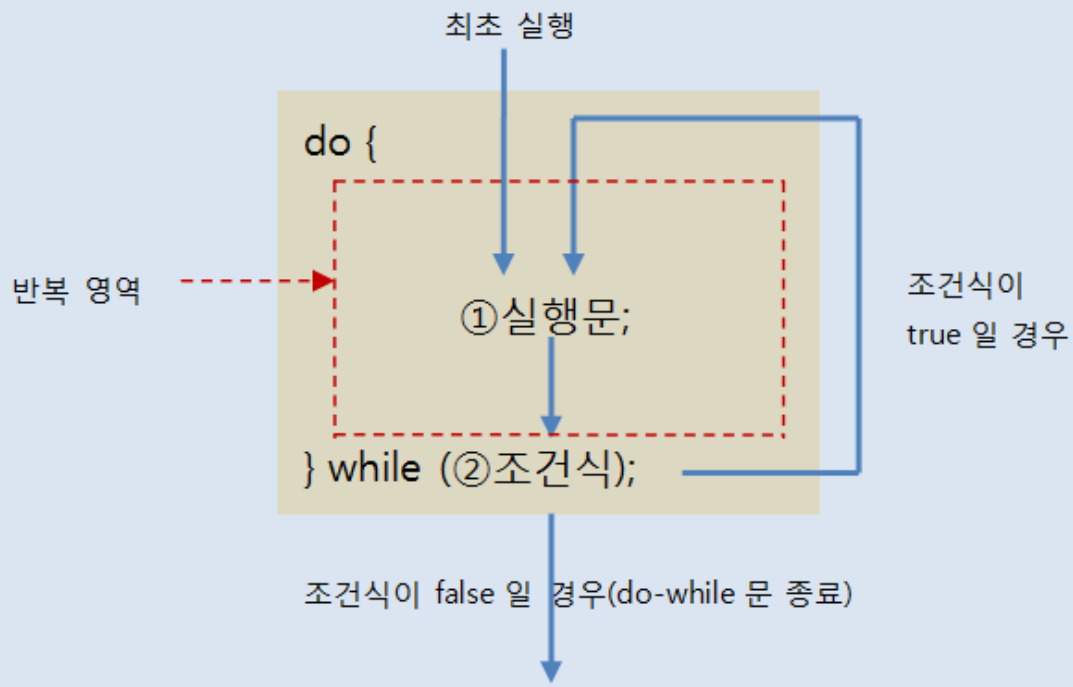
```
do {
    처리문장....
  } while(조건);
```



반복문(for문, while문, do-while문)

❖do-while문

- 조건 따라 반복 계속할지 결정할 때 사용하는 것은 while문과 동일
- 무조건 중괄호 { } 블록을 한 번 실행한 후, 조건 검사해 반복 결정



a-z까지 출력

do-while 문을 이용하여 'a'부터 'z'까지 출력하는 프로그램을 작성하시오.

```
public class DoWhileSample {
   public static void main (String[] args) {
      char c = 'a';

      do {
         System.out.print(c + " ");
         c = (char) (c + 1);
      } while (c <= 'z');
   }
}
```

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

반복 문제 1

메뉴에 있는 연산자를 선택하고 피연산자 두 개를 먼저 입력받아 사칙연산을 하는 프로그램의 코드를 작성하여라.

처리조건

① 메뉴에서 1~5사이가 아니면 다시 입력
② END가 선택될 때까지 반복

메뉴항목

1. +
2. -
3. *
4. /
5. 종료

실행예시

1. +
2. -
3. *
4. /

연산자를 선택하시오 : 2
피연산자 1 : 34
피연산자 2 : 12

34 - 12 = 22

보조 제어문(분기문)

▣ break문

반복문이나 switc문의 case를 벗어날 때 사용한다. 또한 무한 반복에서 탈출하고자 할 때도 사용한다. 즉 for문, while문, do ~ while문, switch ~ case문 구조에서 block을 빠져 나오는데 사용하는 구문이다. 단, 다중 block에서 break문을 만나면 자신을 포함하고 있는 하나의 block만을 벗어난다.

▣ continue문

for문, while문, do ~ while문의 조건부로 제어를 옮기는데 사용하는 구문이다. 즉 반복해서 처리하는 중에 특정값에 대한 처리를 제외하고자 할 때 사용한다.

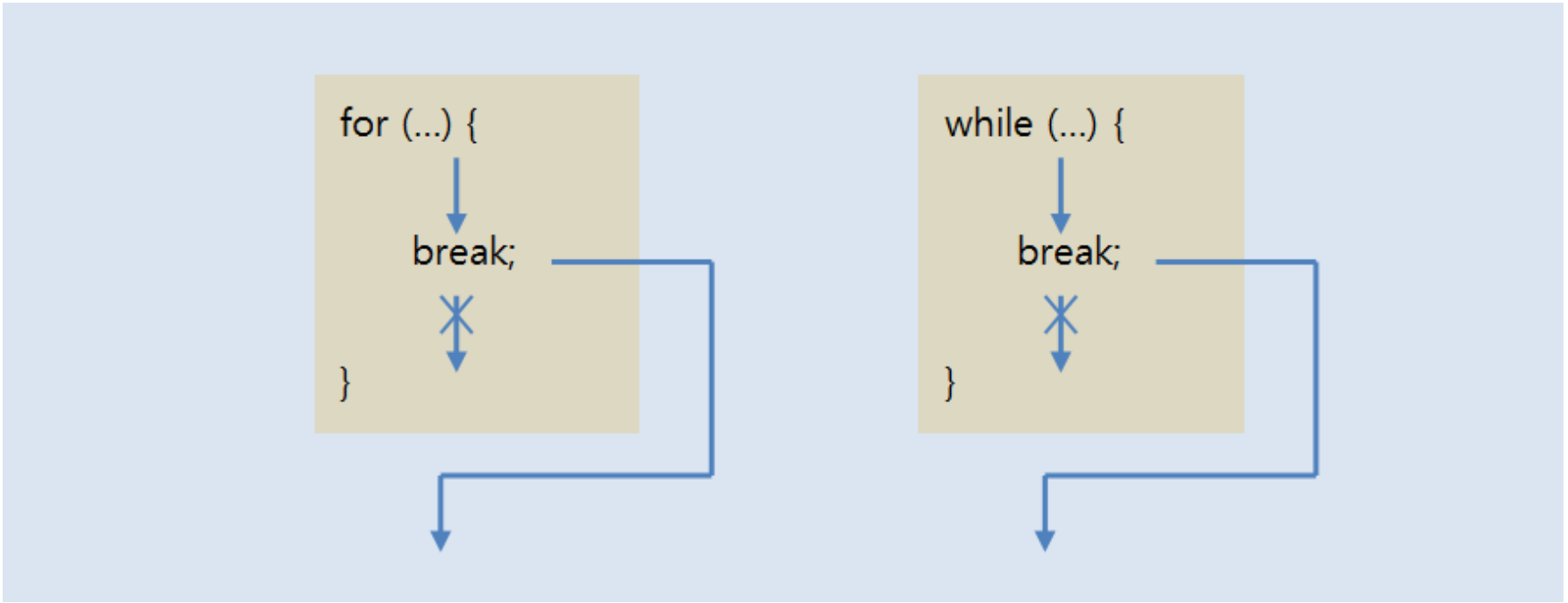
▣ return문

return문은 메서드에서 특정값을 리턴값으로 보내고 싶을 때 사용한다.

보조 제어문(break 문)

break 문

- for문, while문, do-while문 종료 (반복 취소)
- switch문 종료
- 대개 if문과 같이 사용
  if문 조건식에 따라 for문과 while문 종료할 때 사용

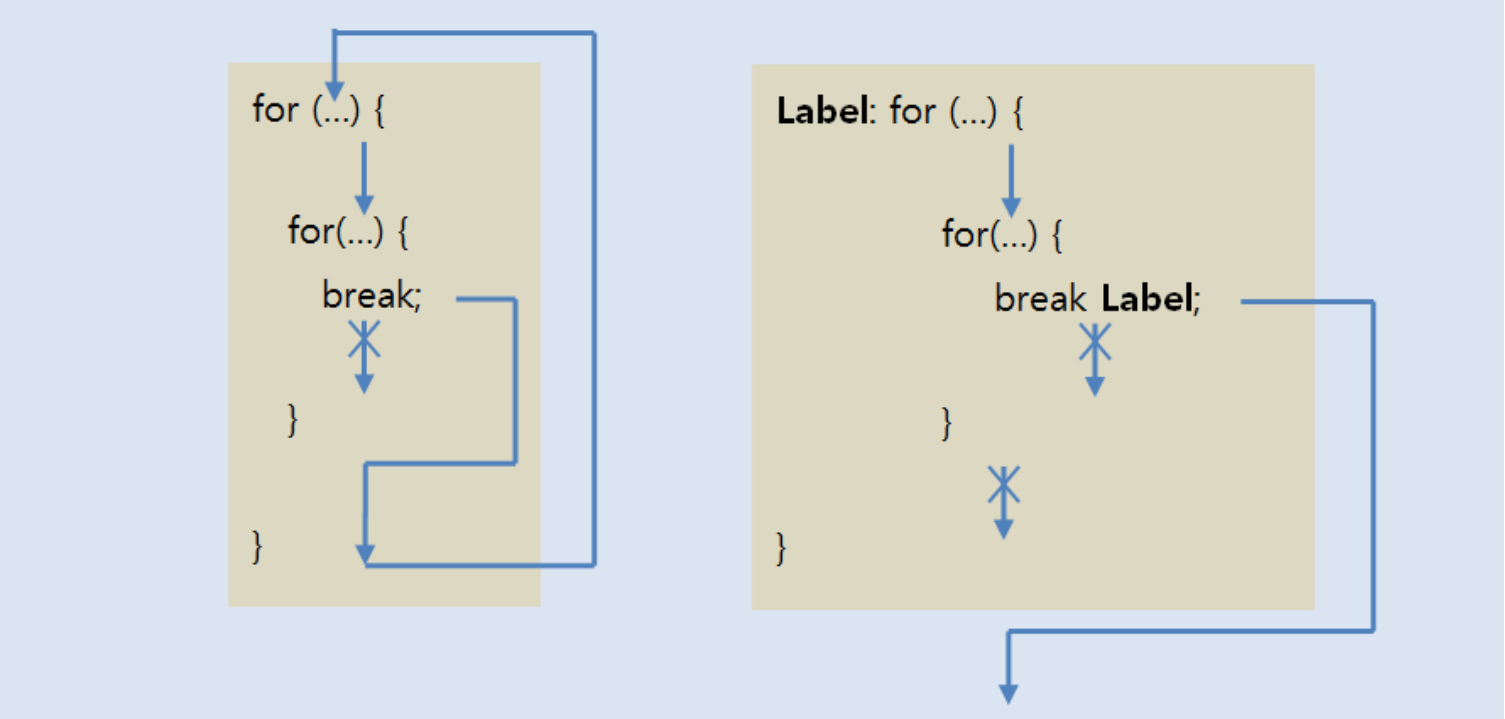


보조 제어문(break 문)

break 문

- 반복문이 중첩된 경우
  반복문이 중첩되어 있을 경우 break; 문은 가장 가까운 반복문만 종료

  바깥쪽 반복문까지 종료시키려면 반복문에 이름(라벨)을 붙이고, “break 이름;” 사용



break 문을 이용하여 while 문 벗어나기

"exit"이 입력되면 while 문을 벗어나도록 break 문을 활용하는 프로그램을 작성하라.

```
import java.util.Scanner;

public class BreakExample {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);

                System.out.println("exit을 입력하면 종료합니다.");
                while(true) {
                        System.out.print(">> ");
                        String text = scanner.nextLine();
                        if(text.equals("exit")) // "exit"이 입력되면 반복 종료
                                break; // while 문을 벗어남
                }
                System.out.println("종료합니다...");

                scanner.close();
        }
}
```

문자열 비교 시 equals()사용

```
exit을 입력하면 종료합니다.
>> java
>> 아자~~!
>> exit
프로그램 종료
```

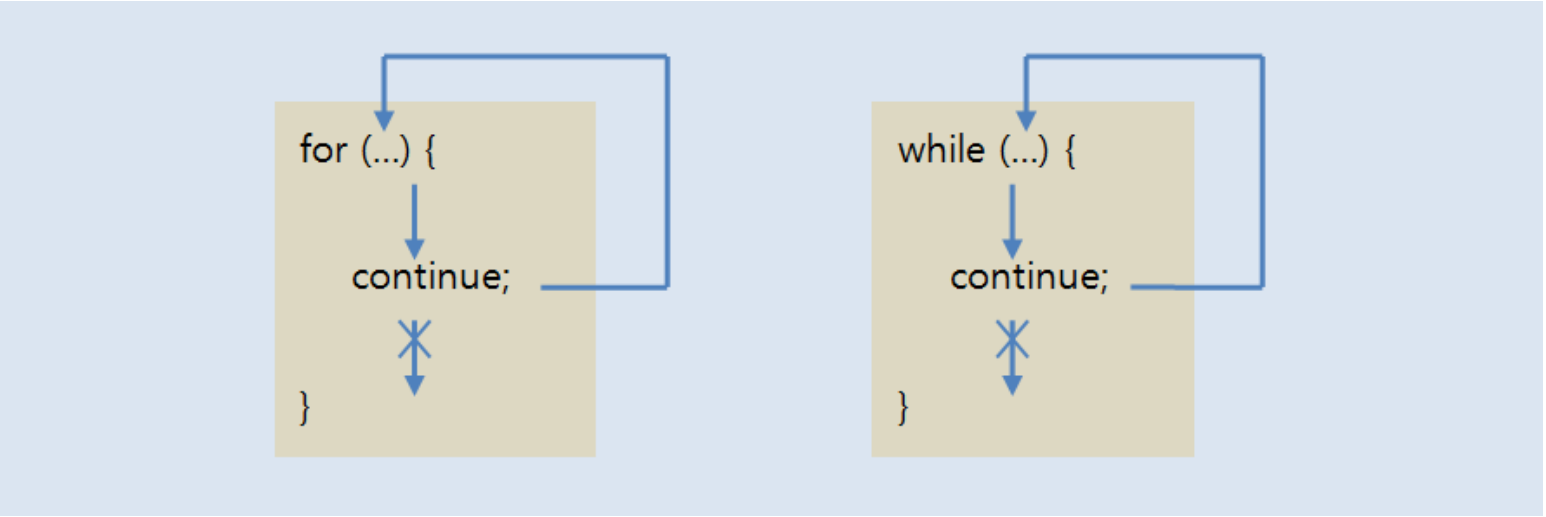
보조 제어문(분기문)

continue 문

for문, while문, do-while문에서 사용

for문: 증감식으로 이동

while문, do-while문: 조건식으로 이동




```
for(초기문; 조건식; 반복 후 작업) {
    .........
    continue;
    .........
}
```

분기

```
while(조건식) {
    .........
    continue;
    .........
}
```

조건식으로 분기

```
do {
    .........
    continue;
    .........
} while(조건식);
```

조건식으로 분기

continue 문을 이용하여 양수 합 구하기

5개의 정수를 입력 받고 그 중 양수들만 합하여 출력하는 프로그램을 작성하라.

```
import java.util.Scanner;

public class ContinueExample {
		public static void main(String[] args) {
				Scanner scanner = new Scanner(System.in);

				System.out.println("정수를 5개 입력하세요.");
				int sum=0;
				for(int i=0; i<5; i++) {
						int n = scanner.nextInt(); // 키보드에서 정수 입력
						if(n<=0) {
								continue; // 양수가 아닌 경우 다음 반복으로 진행
						} else {
								sum += n; // 양수인 경우 덧셈
						}
				}
				System.out.println("양수의 합은 " + sum);
				scanner.close();
		}
}
```

```
정수를 5개 입력하세요.
5 -2 6 10 -4
양수의 합은 21
```

함수(메서드)

(1) 함수(메서드)의 선언과 호출

함수(메서드)는 구문 실행을 위한 구문 그룹화의 구성체이다. 함수는 프로그램에서 실행하기 위해 한번 작성하여 반복적으로 사용되는 코드들을 하나의 단위로 묶어놓은 것이다. 함수는 복잡한 코드를 간단한 하나의 단위로 프로그램의 복잡성을 줄일 수 있고 코드의 재사용성을 높일 수 있다.(코드의 모듈화)

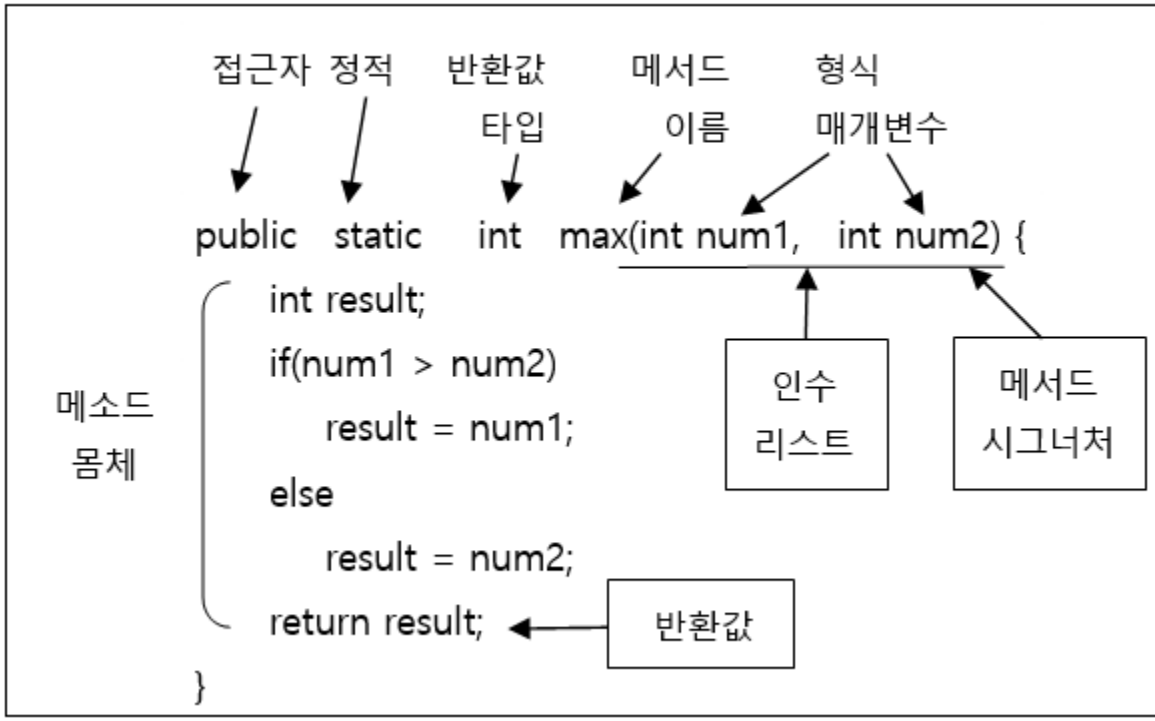
◎ 메서드 정의 형식

```
접근자 정적 return_type  functionName(type parameter, ..){
        Coding;
}
함수 호출
functionName(arguments);
```

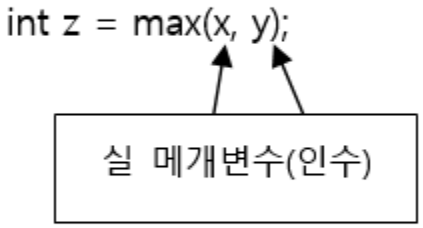
함수(메서드)는 접근자, 수정자, 반환값타입, 메서드 이름, 매개변수들로 구성된다.
접근자와 수정자의 자세한 내용은 객체와 클래스에서 살펴본다.

함수(메서드)

메소드의 정의



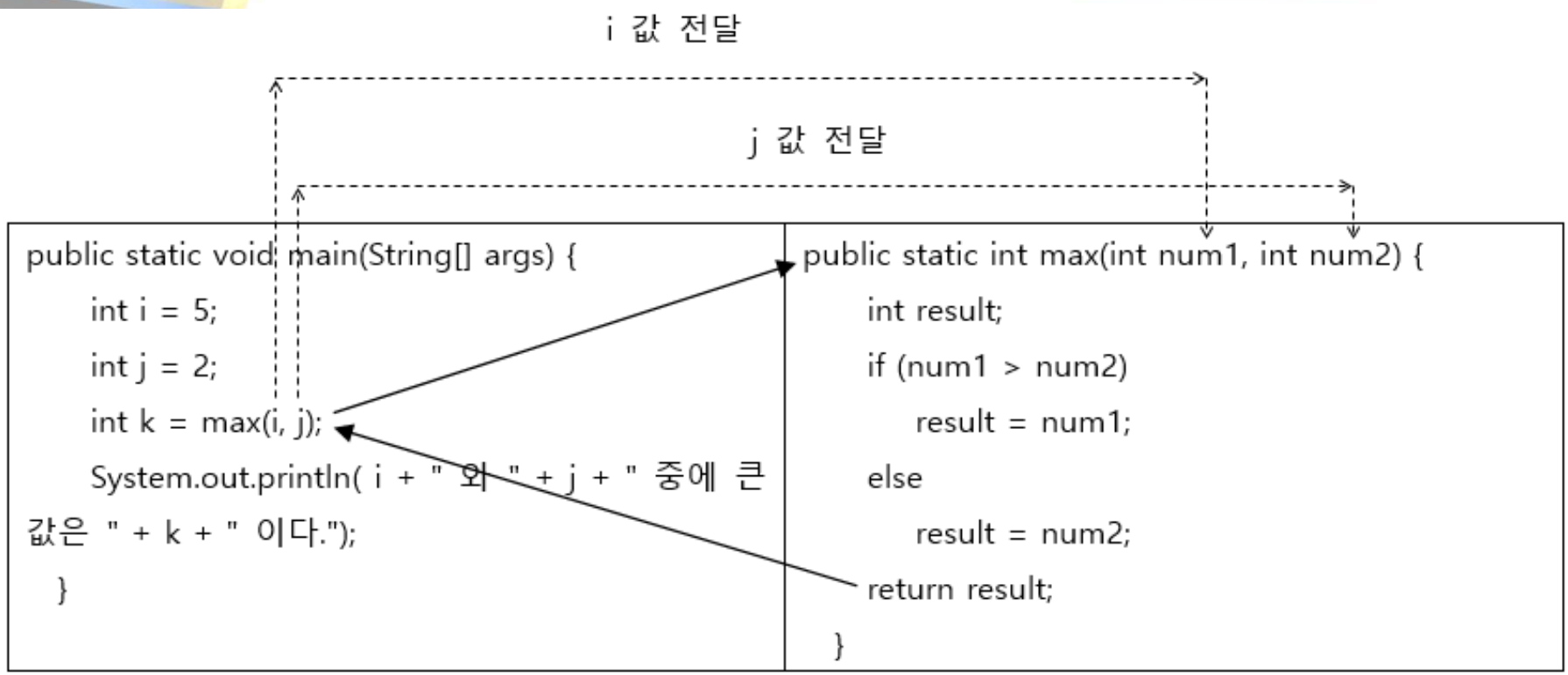
메소드 호출



예제

```
public class TestMax {
 public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println( i + " 와 " + j + " 중에 큰 값은 " + k + " 이다.");
  }
  public static int max(int num1, int num2) {
    int result;
    if (num1 > num2)
       result = num1;
    else
       result = num2;
    return result;
  }
}
```

함수(메서드)



함수(메서드) 호출시 인수는 시그너처에서 정의된 것처럼 순서, 개수, 타입이 일치해야 한다.


Thank You