

반복문

반복문

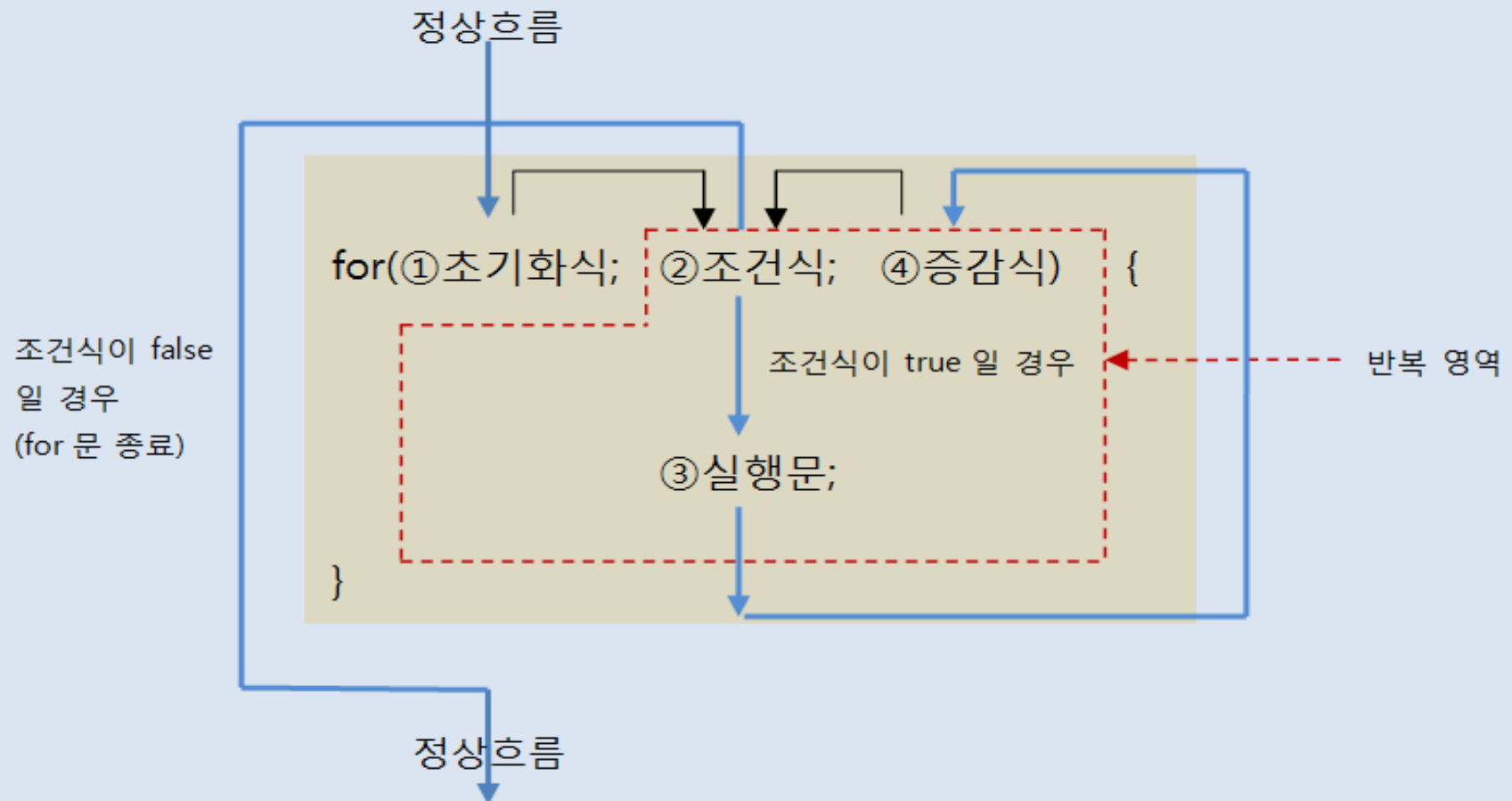
❖ 반복문

- 중괄호 블록 내용을 반복적으로 실행할 때 사용
- 종류: for문, while문, do-while문

for문

❖ for문

- 반복 횟수를 알고 있을 때 주로 사용



for문

❖ for문

- 반복 횟수를 알고 있을 때 주로 사용

```
int sum = 0;
for (int i=1; i<=100; i++) {
    sum = sum + i; ●----- 100 번 반복
}
System.out.println("1~100 까지의 합:" + sum);
```

for문

❖ 1부터 10까지 출력: ForPrintFrom1To10Example.java

```
public class ForPrintFrom1To10Example {  
    public static void main(String[] args) {  
  
        for (int i = 1; i <= 10; i++) {  
            System.out.println(i);  
        }  
    }  
}
```

for문

❖ for문 초기식

- 변수 선언 가능
 - for문 블록의 지역변수
 - for문 밖에서는 사용 불가

```
for (int i=0, j=100; i<=50 && j>=50; i++, j--) { ... }
```

초기화식 조건식 증감식

for문

❖ 1부터 100까지의 합: ForSumFrom1To100Example.java

```
public class ForSumFrom1To100Example {  
    public static void main(String[] args) {  
        int sum = 0;  
  
        for(int i=1; i<=100; i++) {  
            sum += i;  
        }  
  
        System.out.println("1~100 합 : " + sum);  
  
        System.out.println("1~" + (i - 1) + " 합 : " + sum);  
    }  
}
```

for문

❖ 반복문의 중첩 가능

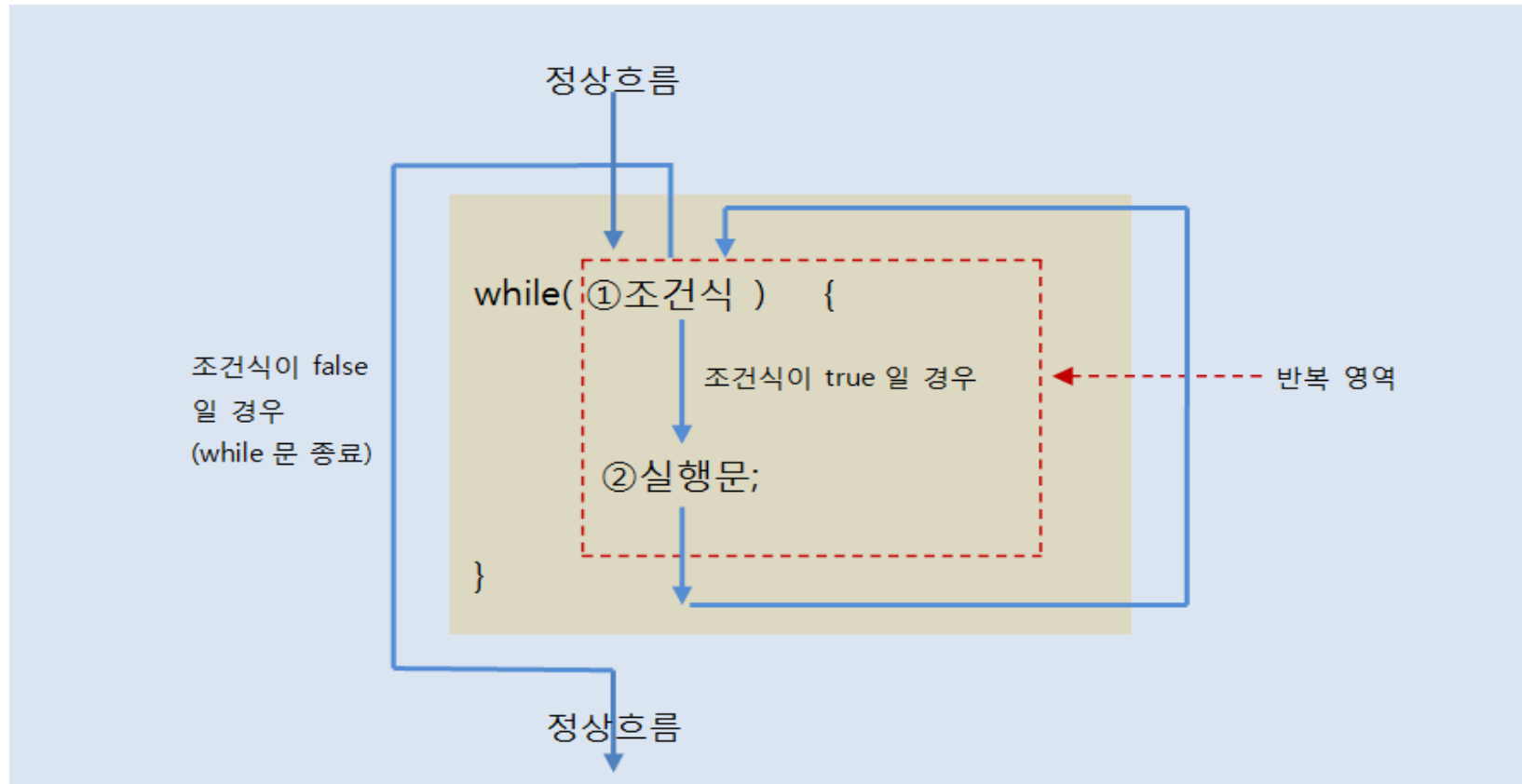
❖ 구구단 출력하기: ForMultiplicationTableExample.java

```
public class ForMultiplicationTableExample {  
    public static void main(String[] args) {  
  
        for (int m = 2; m <= 9; m++) {  
            System.out.println("*** " + m + "단 ***");  
  
            for (int n = 1; n <= 9; n++) {  
                System.out.println(m + " x " + n + " = " + (m * n));  
            }  
        }  
    }  
}
```


while문

❖ while문

- 조건에 따라 반복을 계속할지 결정할 때 사용



while문

❖ 1부터 10까지 출력: WhilePrintFrom1To10Example.java

```
public class WhilePrintFrom1To10Example {  
    public static void main(String[] args) {  
  
        int i = 1;  
        while (i <= 10) {  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

while문

❖ 1부터 100까지의 합 출력: WhileSumForm1To100Example.java

```
public class WhileSumForm1To100Example {  
    public static void main(String[] args) {  
        int sum = 0;  
        int i = 1;  
  
        while (i <= 100) {  
            sum += i;  
            i++;  
        }  
  
        System.out.println("1~" + (i - 1) + " 합 : " + sum);  
    }  
}
```

while문

❖ 키보드로부터 값 읽기

```
int keyCode = System.in.read();
```

숫자	알파벳					기능 키	방향 키
0 = 48	A = 65	N = 78	a = 97	n = 110		Backspace = 8	← = 37
1 = 49	B = 66	O = 79	b = 98	o = 111		Tab = 9	↑ = 38
2 = 50	C = 67	P = 80	c = 99	p = 112		Enter = [CR=13, LF=10]	→ = 39
3 = 51	D = 68	Q = 81	d = 100	q = 113		Shift = 16	↓ = 40
4 = 52	E = 69	R = 82	e = 101	r = 114		Ctrl = 17	
5 = 53	F = 70	S = 83	f = 102	s = 115		Alt = 18	
6 = 54	G = 71	T = 84	g = 103	t = 116		ESC = 27	
7 = 55	H = 72	U = 85	h = 104	u = 117		Space = 32	
8 = 56	I = 73	V = 86	i = 105	v = 118		PAGEUP = 33	
9 = 57	J = 74	W = 87	j = 106	w = 119		PAGEDN = 34	
	K = 75	X = 88	k = 107	x = 120			
	L = 76	Y = 89	l = 108	y = 121			
	M = 77	Z = 90	m = 109	z = 122			

while문

❖ 키보드로 while문 제어: WhileKeyControlExample.java

```
public class WhileKeyControlExample {  
  
    public static void main(String[] args) throws Exception {  
  
        boolean run = true;  
        int speed = 0;  
        int keyCode = 0;  
  
        while (run) {  
            if (keyCode != 13 && keyCode != 10) {  
                System.out.println("-----");  
                System.out.println("1.증속 | 2.감속 | 3.중지");  
                System.out.println("-----");  
                System.out.print("선택: ");  
            }  
  
            keyCode = System.in.read();  
        }  
    }  
}
```

while문

❖ 키보드로 while문 제어: WhileKeyControlExample.java

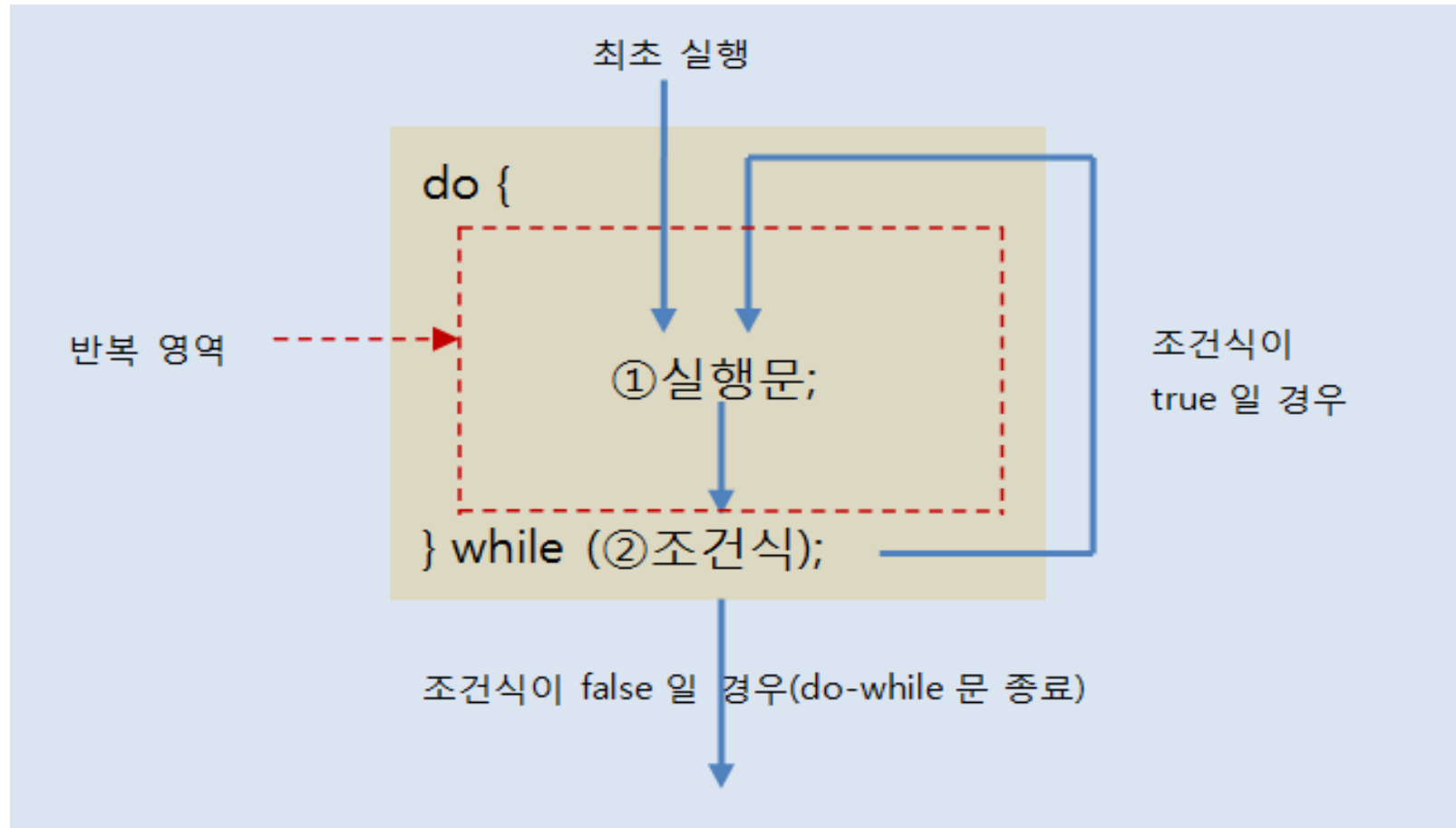
```
        if (keyCode == 49) { // 1
            speed++;
            System.out.println("현재 속도=" + speed);
        } else if (keyCode == 50) { // 2
            speed--;
            System.out.println("현재 속도=" + speed);
        } else if (keyCode == 51) { // 3
            run = false;
        }
    }

    System.out.println("프로그램 종료");
}
```

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



do-while문

❖ do-while 문: DoWhileExample.java

```
public class DoWhileExample {
    public static void main(String[] args) {
        System.out.println("메시지를 입력하세요");
        System.out.println("프로그램을 종료하려면 q를 입력하세요.");

        Scanner scanner = new Scanner(System.in); // Scanner 객체 생성
        String inputString;

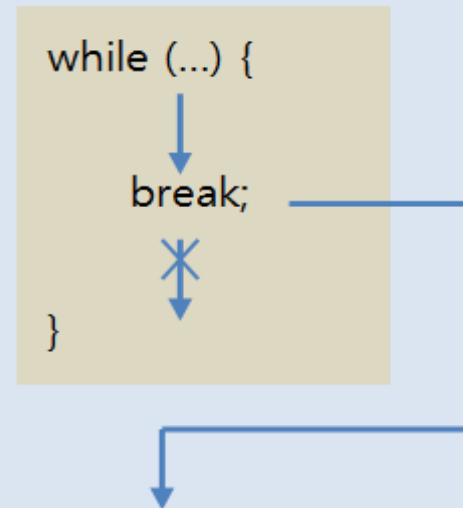
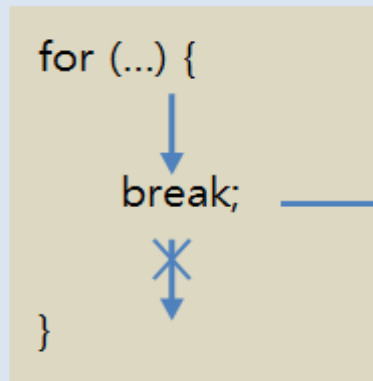
        do {
            System.out.print(">");
            inputString = scanner.nextLine(); // 키보드 입력 문자열 얻기
            System.out.println(inputString);
        } while (!inputString.equals("q")); // 문자열 비교

        System.out.println();
        System.out.println("프로그램 종료");
    }
}
```


반복문의 제어

❖ break 문

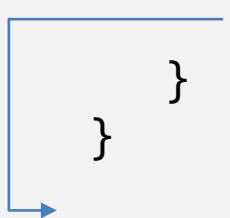
- for문, while문, do-while문 종료 (반복 취소)
- Switch문 종료
- 대개 if문과 같이 사용
 - if문 조건식에 따라 for문과 while문 종료할 때 사용



반복문의 제어

❖ break로 while 문 종료: BreakExample.java

```
public class BreakExample {  
    public static void main(String[] args) throws Exception {  
  
        while (true) {  
            int num = (int) (Math.random() * 6) + 1;  
            System.out.println(num);  
            if (num == 6) {  
                break;  
            }  
        }  
  
        System.out.println("프로그램 종료");  
    }  
}
```

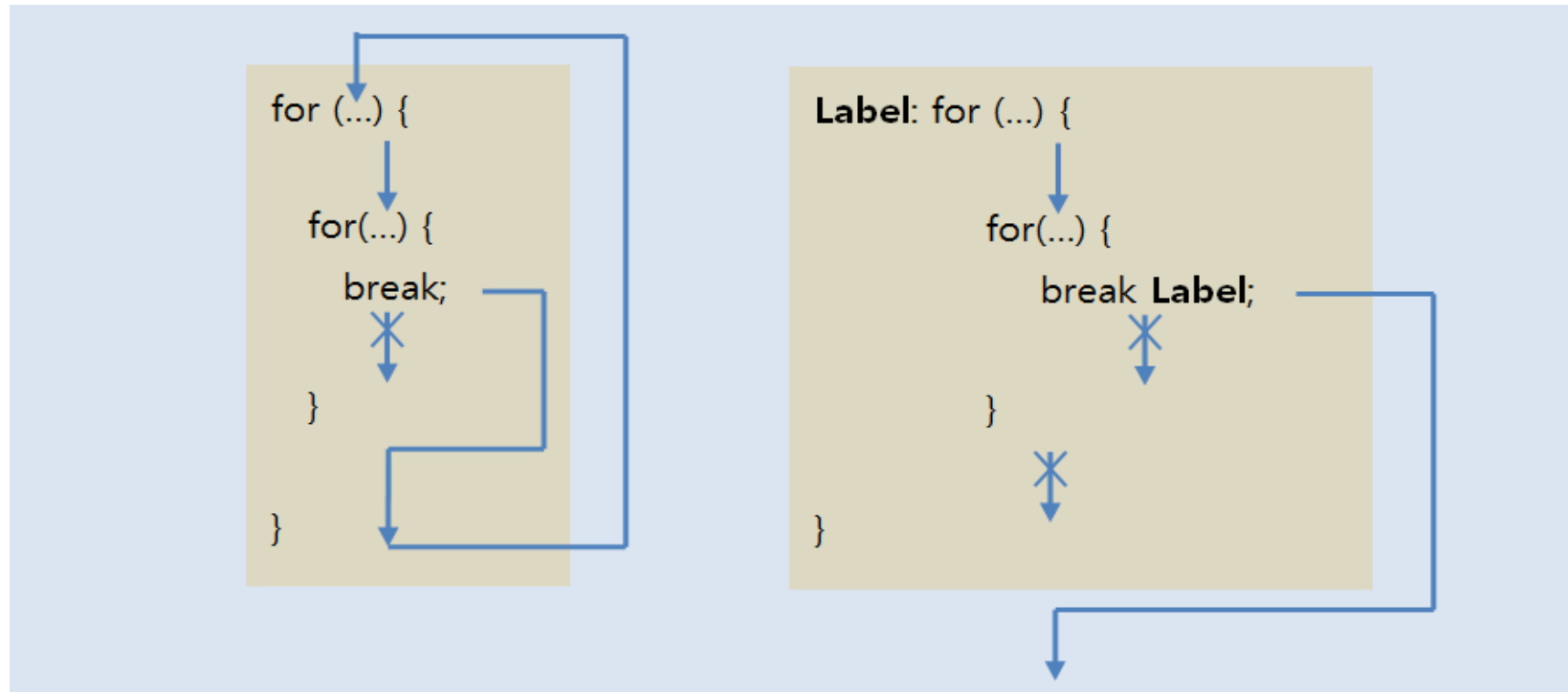


반복문의 제어

❖ break 문 (p.130~132)

○ 반복문이 중첩된 경우

- 반복문이 중첩되어 있을 경우 break; 문은 가장 가까운 반복문만 종료
- 바깥쪽 반복문까지 종료시키려면 반복문에 이름(라벨)을 붙이고, "break 레벨이름"; 사용



반복문의 제어

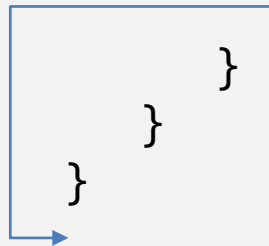
❖ 바깥쪽 반복문 종료: BreakOutterExample.java

```
public class BreakOutterExample {  
    public static void main(String[] args) throws Exception {
```

Outter:

```
        for (char upper = 'A'; upper <= 'Z'; upper++) {  
            for (char lower = 'a'; lower <= 'z'; lower++) {  
                System.out.println(upper + "-" + lower);  
                if (lower == 'g') {
```

break Outter;

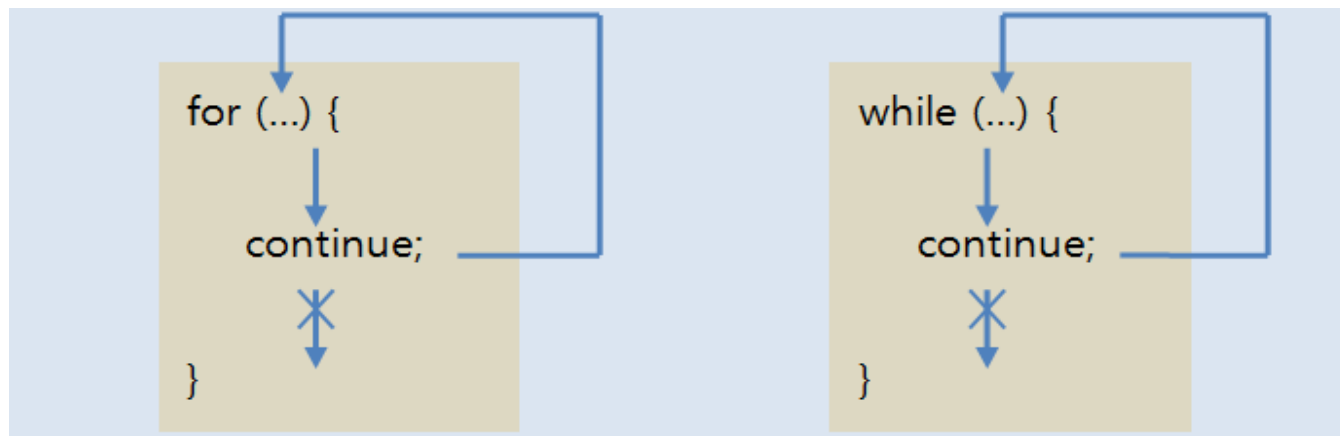


```
            }  
        }  
        System.out.println("프로그램 실행 종료");  
    }  
}
```

반복문의 제어

❖ continue 문

- for문, while문, do-while문에서 사용
 - for문: 증감식으로 이동
 - while문, do-while문: 조건식으로 이동



```
for(int i=1; i<=10; i++) {  
    if(i%2 != 0) { ●-----  
        continue;  
    }  
    System.out.println(i); ●-----  
}
```

2로 나눈 나머지가 0이 아닐 경우
즉 홀수인 경우

홀수는 실행되지 않는다.

❖ continue를 사용한 while 문: ContinueExample.java

```
public class ContinueExample {  
    public static void main(String[] args) throws Exception {  
  
        for (int i = 1; i <= 10; i++) {  
            if (i % 2 != 0) {  
                continue;  
            }  
            System.out.println(i);  
        }  
    }  
}
```