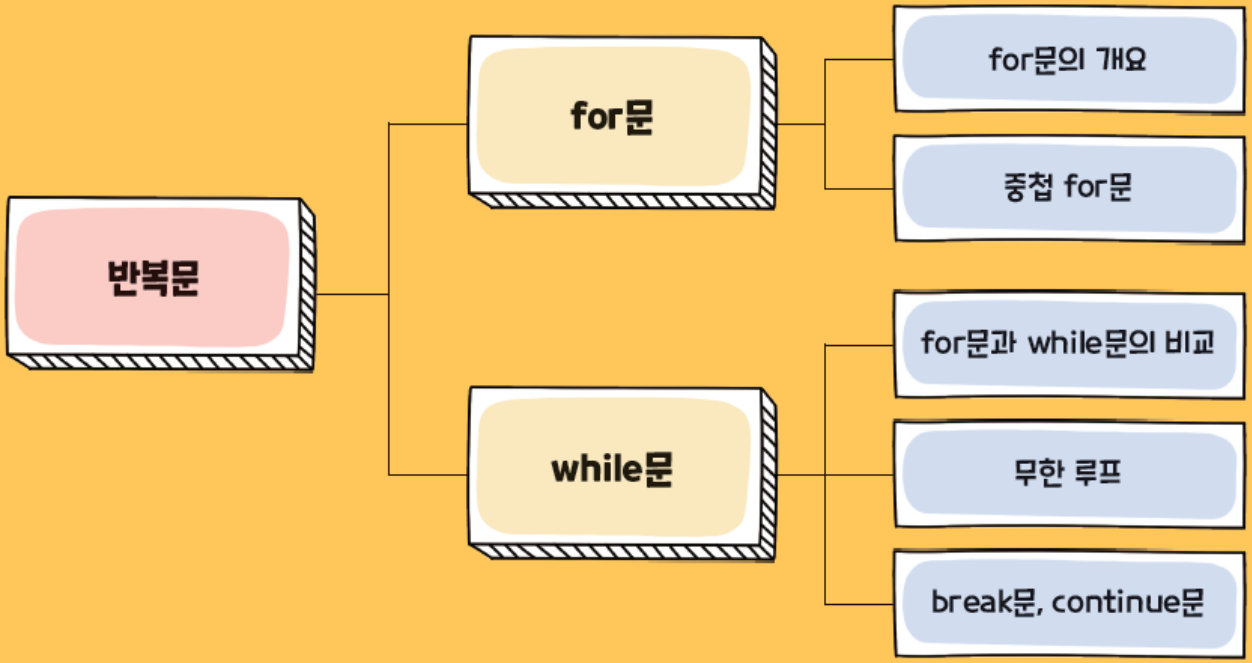
bantaehak

Chapter 06
반복문

목차

Preview



학습목표

- 반복문의 필요성을 이해합니다.
- for문 사용법을 익힙니다.
- while문과 for문의 차이를 이해하고 while문 사용법을 익힙니다.
- break문과 continue문 사용법을 익힙니다.

Section 01
반복문의 개요

1. 반복문의 필요성

반복문의 필요성

- 다음 결과를 출력하는 코드를 작성해보기

```
난생처음 자바는 재미있습니다. ^^
난생처음 자바는 재미있습니다. ^^
난생처음 자바는 재미있습니다. ^^
```

- 방법1) System.out.println 3번 작성

코드 6-1 Code06_01.java

```
01  public class Code06_01 {
02      public static void main(String[] args) {
03          System.out.println("난생처음 자바는 재미있습니다. ^^");
04          System.out.println("난생처음 자바는 재미있습니다. ^^");
05          System.out.println("난생처음 자바는 재미있습니다. ^^");
06      }
07  }
```

1. 반복문의 필요성

■ 반복문의 필요성

- 다음 결과를 출력하는 코드를 작성해보기

```
난생처음 자바는 재미있습니다. ^^
난생처음 자바는 재미있습니다. ^^
난생처음 자바는 재미있습니다. ^^
```

- 방법2) 반복문 사용

  → 20줄, 50줄, 100줄을 반복해야 할 때 더욱 유용해짐

코드 6-2 Code06_02.java

```
01  public class Code06_02 {
02      public static void main(String[] args) {
03          for (int i=0 ; i<3 ; i++) {
04              System.out.println("난생처음 자바는 재미있습니다. ^^");
05          }
06      }
07  }
```

2. for문의 개요

- for문의 형식과 동작 원리

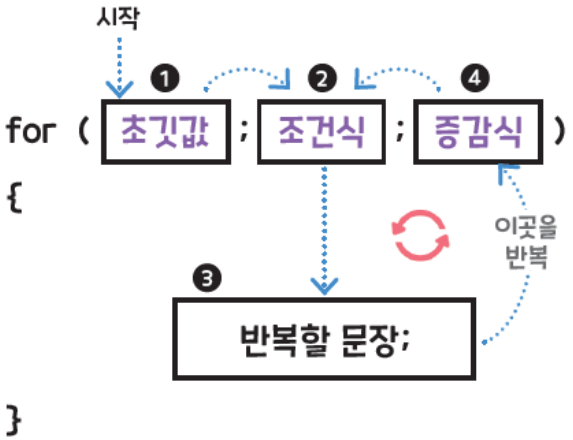


그림 6-1 for문의 동작 원리

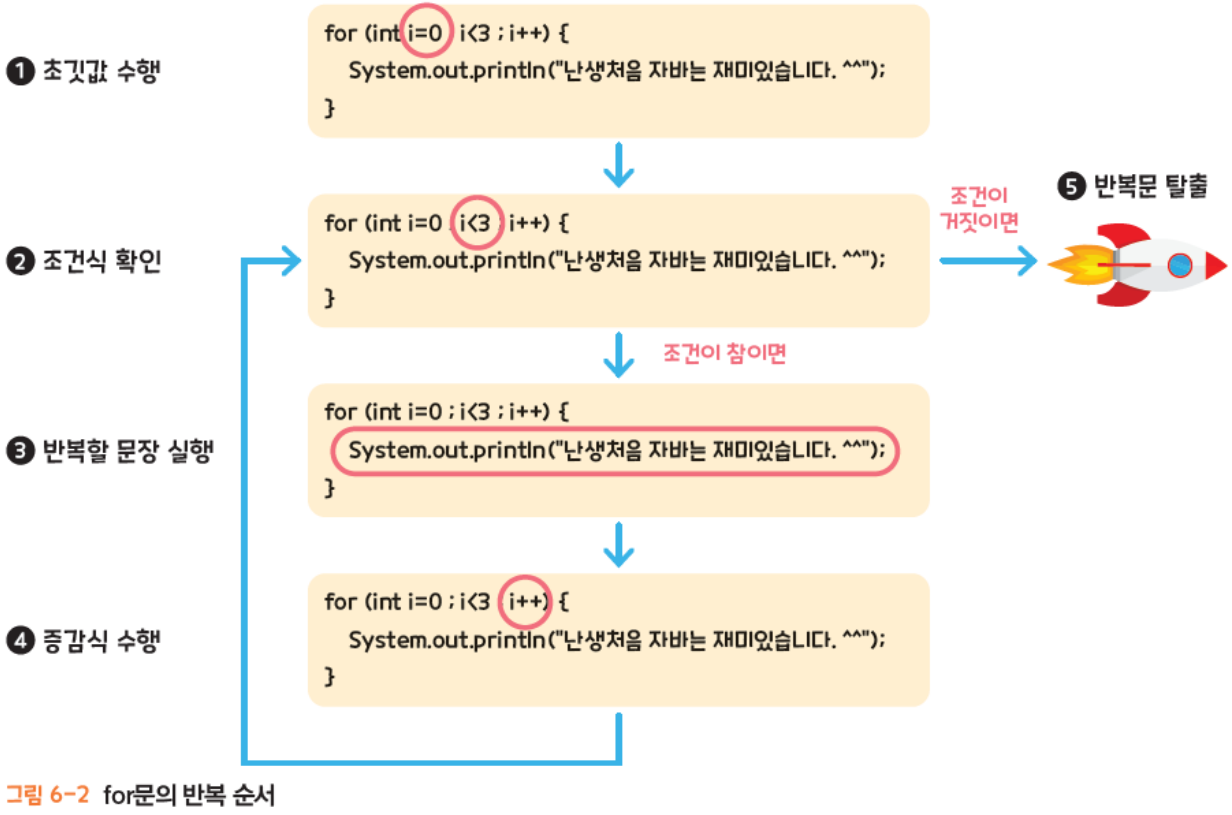


그림 6-2 for문의 반복 순서

2. for문의 개요

- for문 예시
  - 조건식에 사용된 i를 코드 내부에서도 사용해보기

코드 6-3 Code06_03.java

```
01   public class Code06_03 {
02       public static void main(String[] args) {
03           for (int i=0 ; i<3 ; i++) {
04               System.out.println(i + ": 난생처음 자바는 재미있습니다. ^^");
05           }
06       }
07   }
```

```
0 : 난생처음 자바는 재미있습니다. ^^
1 : 난생처음 자바는 재미있습니다. ^^
2 : 난생처음 자바는 재미있습니다. ^^
```

2. for문의 개요

- for문 예시
  - 숫자 1~10 출력하기

코드 6-4 Code06_04.java

```
01   public class Code06_04 {
02       public static void main(String[] args) {
03           for (int i=1 ; i<=10 ; i++) {
04               System.out.print(i + " ");
05           }
06       }
07   }
```

```
1  2  3  4  5  6  7  8  9  10
```

2. for문의 개요

확인문제

다음과 같은 결과가 출력되도록 코드의 빈칸을 채우시오.

```
for (int          ) {
    System.out.print(i + " ");
}
```

```
2  3  4  5
```

정답

Click!

[LAB] 학생들 줄 세우기

다섯 명의 학생에게 도시락을 나눠주려고 합니다.
이들이 도시락을 받기 위해 순서대로 줄을 서는 경우의 수는 모두 몇 가지일까요?
이는 팩토리얼(factorial)을 사용하면 쉽게 알아낼 수 있습니다.
팩토리얼은 1부터 n까지의 곱을 의미하며, 기호 !로 표시합니다.
예를 들어 5!은 1×2×3×4×5=120입니다.

실행 결과

```
A, B, C, D, E 학생들을 순서대로 세우는 경우의 수 : 120
```

팩토리얼이란 '수를 단계적으로 곱한다.'는 뜻으로 숫자 옆에 !표를 붙어서 표현한다.
팩토리얼을 구하는 것은 아래와 같이 표현할 수 있다.

3!=3*2*1 =6
4!=4*3*2*1 = 24
5!=5*4*3*2*1= 120

[LAB] 학생들 줄 세우기

1. Lab06_01.java 파일을 만들고, A~E 학생들이 줄을 서는 경우의 수를 계산하는 데 필요한 변수를 선언하기
   - 팩토리얼은 곱셈이므로 팩토리얼을 저장할 변수 fact의 초깃값을 1로 해야 한다는 것을 주의해야 함
   - 만약 초깃값을 0으로 하면 모든 곱셈이 0이 되기 때문

```
int fact = 1;
int friends_num = 5;
```

2. 1부터 5까지 1씩 증가하는 i를 fact에 곱하기 위해 for문을 활용함

```
for (int i = 1; i <= friends_num; i++) {
    fact = fact * i;
}

System.out.println("A, B, C, D, E 학생들을 순서대로 세우는 경우의 수 :
" + fact);
```

Section 02
for문

1. for문의 활용

- for문 활용 예제
  - 1~10의 합계 구하기 – 의사 코드

```
1부터 10까지 합계를 누적할 변수 hap 준비

for(변수 i가 1을 시작으로 10까지 1씩 증가)
    hap 값에 i 값을 더함

hap 값을 출력
```

[하나 더 알기] 의사 코드

프로그램 코드가 아니라 코드와 비슷하게 글로 적은 것을 의사 코드(pseudocode)라고 합니다. 의사 코드는 진짜로 작동하지는 않지만 프로그램의 흐름을 파악하는 데 도움이 되며, 자바뿐 아니라 다른 프로그래밍 언어로 변환할 때도 사용됩니다. 컴퓨터 프로그래밍 분야에서 종종 쓰이는 용어이니 기억해두세요

1. for문의 활용

- for문 활용 예제
  - 1~10의 합계 구하기 – 자바 코드

    → 오류 발생: hap가 갑자기 튀어나온 변수이기 때문

코드 6-6 Code06_06.java

```
01  public class Code06_06 {
02      public static void main(String[] args) {
03          int hap;
04          for (int i = 1 ; i <= 10 ; i++) {
05              hap = hap + i;
06          }
07          System.out.println("1부터 10까지의 합계 : " + hap);
08      }
09  }
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
        The local variable hap may not have been initialized
        ~~~생략~~~
```

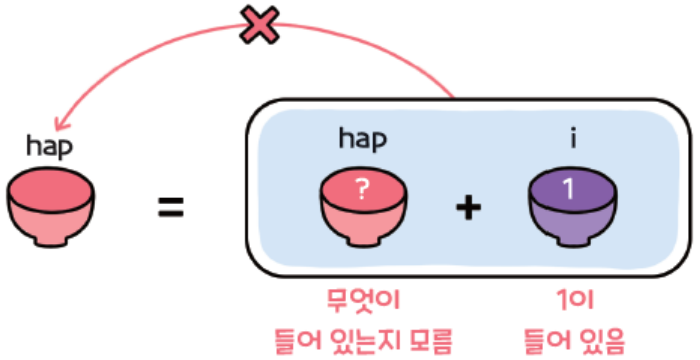


그림 6-3 [코드 6-6]의 변수 hap

1. for문의 활용

- for문 활용 예제
  - 1~10의 합계 구하기 – 자바 코드
    - → 오류 발생: hap가 갑자기 튀어나온 변수이기 때문
    - → 오류 해결: hap을 0으로 초기화(3행)

```
int hap = 0;
```

```
1부터 10까지의 합계 : 55
```

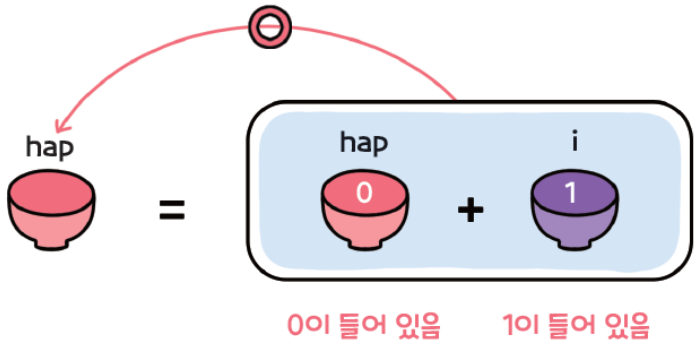


그림 6-4 수정된 [코드 6-6]의 변수 hap

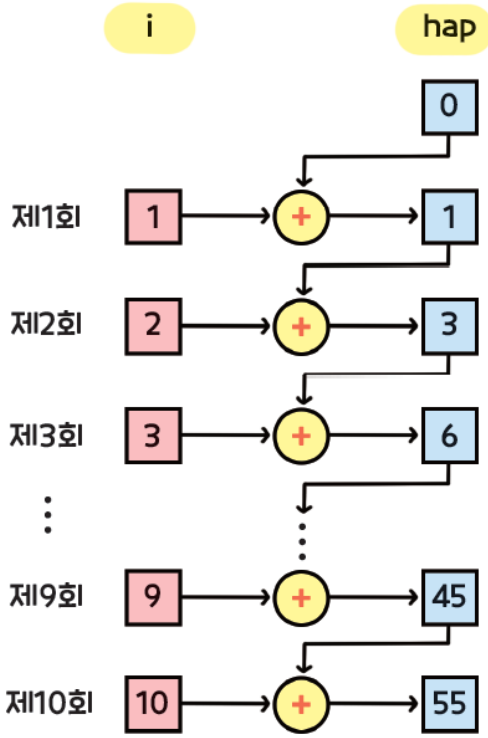


그림 6-5 변수 i와 hap의 변화

1. for문의 활용

- for문 활용 예제
  - 1~1,000의 합계 구하기

코드 6-7 Code06_07.java

```
01   public class Code06_07 {
02       public static void main(String[] args) {
03           int hap = 0;
04           for (int i=1 ; i<=1000 ; i++) {
05               hap = hap + i;
06           }
07           System.out.println("1부터 1000까지의 합계 : " + hap);
08       }
09   }
```

```
1부터 1000까지의 합계 : 500500
```

1. for문의 활용

- for문 활용 예제
  - 1000~2000 사이에 있는 홀수의 합 구하기

코드 6-8 Code06_08.java

```
01   public class Code06_08 {
02       public static void main(String[] args) {
03           int hap = 0;
04           for (int i=1001 ; i<=2000 ; i+=2) {
05               hap += i;
06           }
07           System.out.println("1000부터 2000까지 홀수의 합계 : " + hap);
08       }
09   }
```

```
1000부터 2000까지 홀수의 합계 : 750000
```

hap += i와 hap = hap+i는 동일한 코드입니다.

1. for문의 활용

확인문제

500~1000 사이에 있는 짝수의 합을 구하는 코드의 빈칸을 채우시오.

```
int hap = 0;
for (int                    ) {
    hap += i;
}
System.out.println("500부터 1000까지 짝수의 합계 : " + hap);
```

```
500부터 1000까지 짝수의 합계 : 188250
```

정답

Click!

2. 중첩 for문

- 중첩 for문
  - for문 안에 또 for문을 사용하는 것을 의미함
  - 중첩 for문의 실행 횟수 = 외부 for문 반복 횟수×내부 for문 반복 횟수

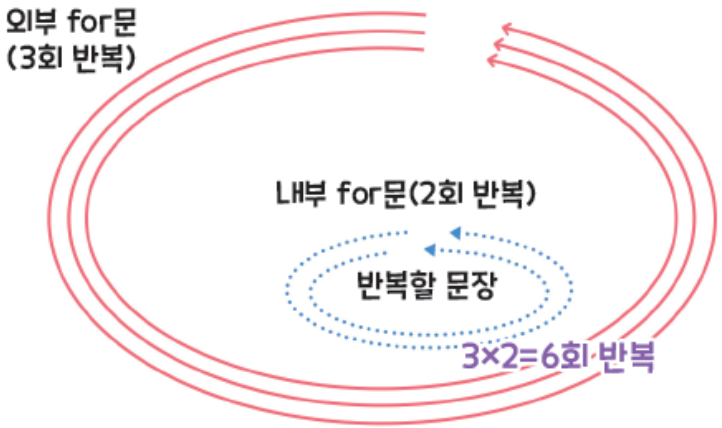


그림 6-6 중첩 for문의 동작 원리

2. 중첩 for문

■ 중첩 for문 예제

코드 6-9 Code06_09.java

```
01  public class Code06_09 {
02      public static void main(String[] args) {
03
04          for (int i=0 ; i<3 ; i++) {
05              for (int k=0 ; k<2 ; k++) {
06                  System.out.println("난생처음은 쉽습니다.^^"+"(i:"+i+",k:"+k+")");
07              }
08          }
09
10      }
11  }
```

```
난생처음은 쉽습니다.^^(i:0,k:0)
난생처음은 쉽습니다.^^(i:0,k:1)
난생처음은 쉽습니다.^^(i:1,k:0)
난생처음은 쉽습니다.^^(i:1,k:1)
난생처음은 쉽습니다.^^(i:2,k:0)
난생처음은 쉽습니다.^^(i:2,k:1)
```

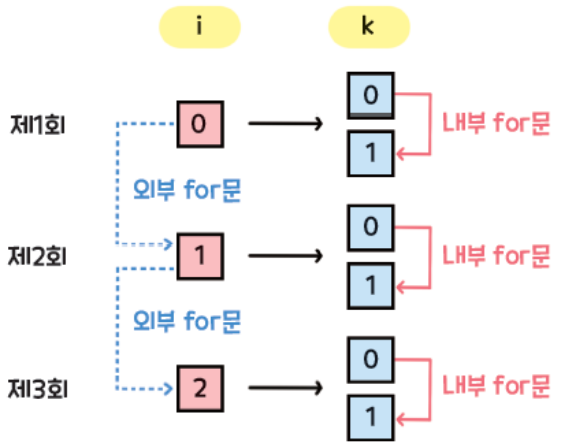


그림 6-7 중첩 for문에서 i와 k 값의 변화

2. 중첩 for문

확인문제

다음 코드를 실행하면 "**중첩 for문입니다.**"가 총 몇 번 출력되나요?

```
for (int i=0 ; i<2 ; i++) {
    for (int k=0 ; k<3 ; k++) {
        System.out.println("중첩 for문입니다.");
    }
}
```

정답

Click!

[LAB] 구구단 계산기 만들기

2단부터 9단까지 구구단을 출력하는 구구단 계산기를 만들어봅시다.

이를 위해 for문을 사용하는데, 이때 2×1, 2×2, 2×3, ..., 2×9와 같이 곱하는 숫자가 1~9로 바뀌기 때문에 곱하는 숫자에도 for문을 사용해야 합니다.

즉 외부 for문에는 2단, 3단..., 9단을 사용하고, 내부 for문에는 곱하는 숫자인 1, 2, ..., 9를 사용하므로 중첩 for문입니다.

실행 결과

```
2 X 1 =2
2 X 2 =4
2 X 3 =6
2 X 4 =8
2 X 5 =10
2 X 6 =12
2 X 7 =14
2 X 8 =16
2 X 9 =18
3 X 1 =3
3 X 2 =6
~~~~생략~~~~
```

구구단의 생성 규칙

X	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

[LAB] 구구단 계산기 만들기

1. Lab06_02.java 파일을 만들고, main() 메서드 내부에서 단으로 사용할 변수 dan과 곱할 숫자로 사용할 변수 num을 선언하기

```
int dan, num;
```

2. 외부 for문의 경우 2단부터 9단까지이므로 여덟 번 반복하고, 내부 for문의 경우 1부터 9까지이므로 아홉 번 반복함. 그리고 내부 for문에서 dan와 num를 곱하여 구구단을 출력하기

```
for (dan=2 ; dan<=9 ; dan++) {
    for (num=1 ; num<=9 ; num++) {
        System.out.println(dan+"x"+num+"="+(dan*num));
    }
}
```

Section 03
while문

1. for문과 while문의 비교

- for문과 while문의 형식 비교
  - for문: 사용할 때는 대개 반복할 횟수를 결정한 후 그 횟수만큼 반복
  - while문: 반복 횟수를 미리 결정하기보다는 조건식이 참일 때 반복

```
for(초깃값; 조건식; 증감식) {
    반복할 문장
}
```

```
while(조건식) {
    반복할 문장
}
```

1. for문과 while문의 비교

- while문의 형식
  - 조건식을 확인하여 이 값이 참이면 반복할 문장을 수행
  - 반복할 문장이 끝나는 곳에서 다시 조건식으로 돌아와 같은 작업을 반복

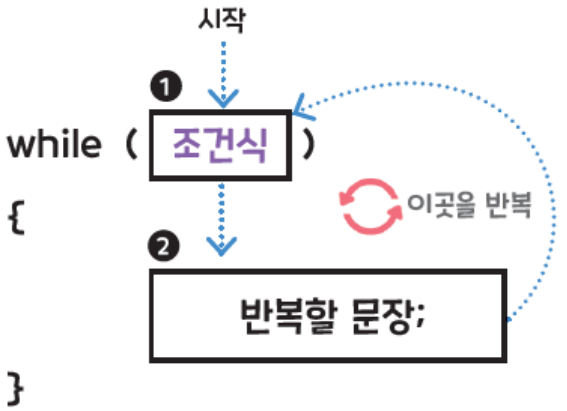


그림 6-8 while문의 동작 원리

코드 6-10 Code06_10.java

```
01   public class Code06_10 {
02       public static void main(String[] args) {
03           int i=0;
04           while(i<3) {
05               System.out.println(i + ": 난생처음 자바는 재미있습니다. ^^");
06               i++;
07           }
08       }
09   }
```

```
0 : 난생처음 자바는 재미있습니다. ^^
1 : 난생처음 자바는 재미있습니다. ^^
2 : 난생처음 자바는 재미있습니다. ^^
```

1. for문과 while문의 비교

확인문제

다음 코드를 실행하면 "**즐거운 난생처음 ^^**"이 총 몇 번 출력되나요?

```
int i=0;
while(i<5) {
    System.out.println("즐거운 난생처음 ^^");
    i++;
}
```

정답

Click!

2. 무한 루프를 위한 while문

무한 루프(무한 반복)

- 반복문을 빠져나올 수 없어 무한히 반복되는 것을 의미함
  → 이를 while문에 사용할 수 있음
- while문에 무한 루프를 적용하려면 조건식을 true로 지정해야 함

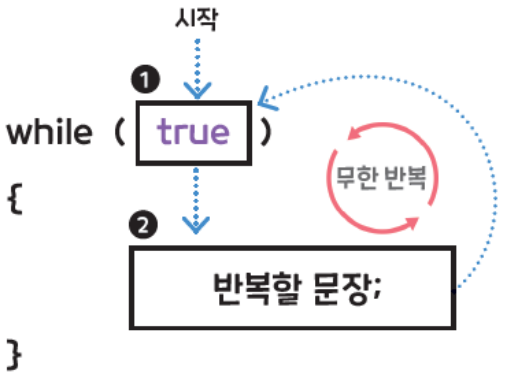


그림 6-9 while을 이용한 무한 루프

코드 6-11 Code06_11.java

```
01  public class Code06_11 {
02      public static void main(String[] args) {
03          while(true) {
04              System.out.print("ㅎ");
05          }
06      }
07  }
```

```
ㅎㅎㅎㅎㅎㅎㅎㅎㅎㅎ ~~~무한 반복~~~
```

이클립스에서 무한 반복을 종료하려면 [Console] 창의 'Terminate' 아이콘(빨간색 네모)을 클릭합니다.

2. 무한 루프를 위한 while문

무한 루프 예제

- 프로그램을 종료하기 전까지 두 숫자의 합계를 반복 계산하기

코드 6-12 Code06_12.java

```
01  import java.util.Scanner;
02  public class Code06_12 {
03      public static void main(String[] args) {
04          Scanner s = new Scanner(System.in);
05          int hap = 0;
06          int num1, num2;
07
08          while(true) {
09              System.out.print("숫자1 ==> ");
10              num1 = s.nextInt();
11              System.out.print("숫자2 ==> ");
12              num2 = s.nextInt();
13
14              hap = num1 + num2;
15              System.out.println(num1 + " + " + num2 + " = " + hap);
16          }
17      }
18  }
```

```
숫자1 ==> 30
숫자2 ==> 77
30 + 77 = 107
숫자1 ==> 8
숫자2 ==> 12345
8 + 12345 = 12353
숫자1 ==>
```

사용자가 입력

사용자가 입력

'Terminate' 아이콘을 클릭하여 종료

2. 무한 루프를 위한 while문

[하나 더 알기] 무조건 한 번은 실행하는 do~while문

반복문 중에 do~while문도 있는데, 이는 for문이나 while문보다 사용 빈도가 낮습니다. do~while문의 중요한 특징은 반복할 문장이 무조건 한 번은 실행된다는 것입니다. for문 및 while문과 비교해보기 위해 먼저 다음 코드에서 System.out.println()이 몇 번 실행되는지 생각해봅시다.

```
int num = 10;
for (int i=num; i<9; i++) {
    System.out.println("for문이 실행됐어요. ^^");
}

while (num<9) {
    System.out.println("while문이 실행됐어요. ^^");
}
```

```
아무것도 출력되지 않음
```

위의 for문과 while문에서는 System.out.println()이 한 번도 실행되지 않습니다. for문에서는 i가 10부터 시작되어 조건식 i<9가 처음부터 false이므로 실행되지 않고, while문도 마찬가지로 10<9여서 처음부터 false이므로 실행되지 않습니다. 다시 말해 for문과 while문은 조건식이 false이면 반복할 문장이 한 번도 실행되지 않고 반복문이 종료될 수도 있습니다.

2. 무한 루프를 위한 while문

[하나 더 알기] 무조건 한 번은 실행하는 do~while문

그럼 앞의 코드에 do~while문을 사용하면 어떻게 될까요?

```
int num = 10;

do {
    System.out.println("do~while문이 실행됐어요. ^^");
} while (num < 9);
```

```
do~while문이 실행됐어요. ^^
```

do~while문의 경우 무조건 중괄호 안의 내용을 실행한 후 while(조건식)에서 조건식을 확인합니다. 따라서 System.out.println()이 한 번 실행된 후 조건식 10<9가 false이므로 반복문이 종료되었습니다.

3. break문과 continue문

■ 반복문을 탈출시키는 break문

- 반복문 안에 break문을 사용하면 break문을 만났을 때 무조건 반복문을 빠져나감

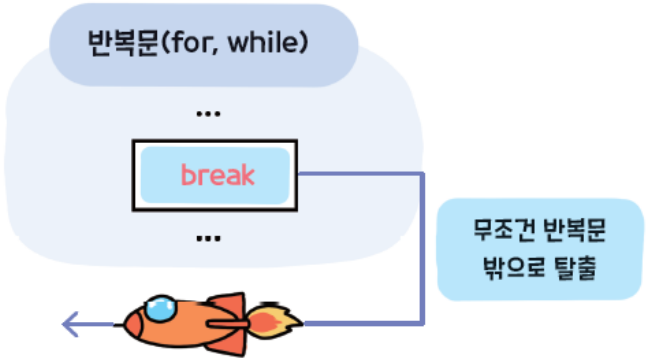


그림 6-10 break문의 동작 원리

코드 6-13 Code06_13.java

```
01  public class Code06_13 {
02      public static void main(String[] args) {
03
04          for (int i=0; i<1000; i++) {
05              System.out.println(i + " : 반복문을 실행합니다");
06              break;
07          }
08
09      }
10  }
```

```
0 : 반복문을 실행합니다
```

3. break문과 continue문

break문 예제

- 'Terminate' 아이콘을 클릭할 때까지 계속 두 숫자의 합계를 구하는 [코드 6-12]에 break문을 적용해보기

코드 6-14 Code06_14.java

```
01  import java.util.Scanner;
02  public class Code06_14 {
03      public static void main(String[] args) {
04          Scanner s = new Scanner(System.in);
05          int hap = 0;
06          int num1, num2;
07
08          while(true) {
09              System.out.print("숫자1 ==> ");
10              num1 = s.nextInt();
11              if (num1 == 0)
12                  break;
13              System.out.print("숫자2 ==> ");
14              num2 = s.nextInt();
15
16              hap = num1 + num2;
17              System.out.println(num1 + " + " + num2 + " = " + hap);
18          }
19
20          System.out.println("0을 입력해서 계산을 종료합니다");
21          s.close();
22      }
23  }
```

```
숫자1 ==> 333
숫자2 ==> 444
333 + 444 = 777
숫자1 ==> 0
0을 입력해서 계산을 종료합니다
```

사용자가 입력

사용자가 입력

3. break문과 continue문

- 반복문의 처음으로 돌아가게 하는 continue문
  - continue문은 반복문의 남은 부분을 모두 건너뛰고 반복문의 처음으로 돌아가게 함

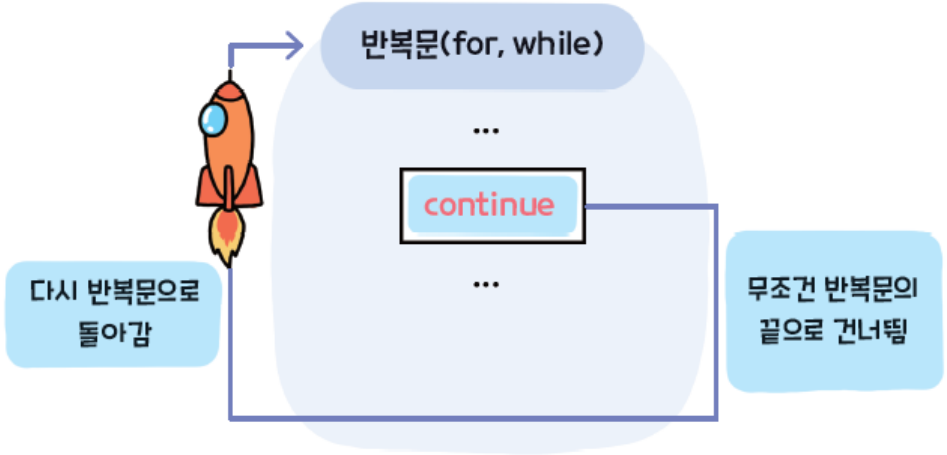


그림 6-11 continue문의 동작 원리

3. break문과 continue문

- continue문 예제

코드 6-15 Code06_15.java

```
01  public class Code06_15 {
02      public static void main(String[] args) {
03          int hap = 0;
04          for (int i=1 ; i<=100 ; i++) {
05              if (i%4 == 0) {
06                  continue;
07              }
08              hap += i;
09          }
10          System.out.println("1~100의 합계(4의 배수 제외) : " + hap);
11      }
12  }
```

```
1~100의 합계(4의 배수 제외) : 3750
```

3. break문과 continue문

- continue문 예제

확인문제

다음 빈칸에 알맞은 말을 넣으시오.

- 무한 반복이 되게 하려면 `while(조건식)`의 조건식을 [㉠]로 지정해야 한다.
- while문이나 for문을 탈출하기 위한 키워드는 [㉡]이다.
- while문이나 for문의 남은 부분을 건너뛰고 반복문의 처음으로 돌아가기 위한 키워드는 [㉢]이다.

정답

Click!

[LAB] 같은 숫자가 나올 때까지 주사위 던지기

주사위 세 개를 동시에 던져서 모두 같은 숫자가 나와야만 반복문을 탈출하는 게임을 만들어봅시다.

몇 번을 던져야 주사위 세 개의 숫자가 같은지는 알 수가 없으니 일단 주사위 세 개 던지기를 무한 루프로 작성하고, 주사위 세 개의 숫자가 같으면 무한 루프를 빠져나가도록 조건문을 추가합니다.

실행 결과

```
3개 주사위는 모두 2입니다.
같은 숫자가 나올 때까지 91번 던졌습니다.
```

[LAB] 같은 숫자가 나올 때까지 주사위 던지기

1. Lab06_03.java 파일을 만들고, 주사위를 던진 횟수를 저장할 변수와 주사위 세 개의 숫자를 저장할 변수를 선언하기

```
int count = 0;
int dice1, dice2, dice3;
```

2. 몇 번을 던져야 주사위 세 개의 숫자가 같은지 알 수 없으므로 무한 반복하기
   - 주사위 세개 던지기를 반복할 때마다 던진 횟수를 1씩 증가시킴

```
while(true) {
    count ++;
}
```

[LAB] 같은 숫자가 나올 때까지 주사위 던지기

3. 무한 반복문 안에 다음 코드를 추가함
   - Math.random()은 0.0000~0.9999 중 임의의 숫자를 반환하므로 6을 곱하면 0.0000~5.9999이며, 여기에 1을 더하고 정수로 변경하면 주사위의 숫자 1~6 중 하나가 됨
   - 그리고 주사위 세 개의 숫자가 같으면 무한 반복이 종료되도록 break문을 넣기

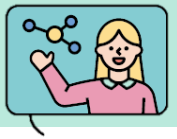
```
dice1 = (int)(Math.random()*6 + 1);
~~~dice2, dice3 생략~~~

if ((dice1 == dice2) && (dice2 == dice3))
    break;
```

4. 무한 반복을 빠져나오면 주사위 세 개의 숫자와 주사위를 던진 횟수를 출력하기

```
System.out.printf("3개 주사위는 모두 %d입니다.\n", dice1);
System.out.printf("같은 숫자가 나올 때까지 %d번 던졌습니다.\n", count);
```

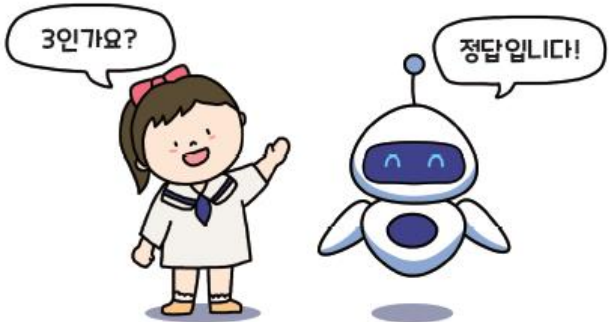
[LAB] 컴퓨터와 숫자 맞히기 게임하기



컴퓨터가 1~5 중 하나를 생각하고 사람이 그 숫자를 맞히는 게임을 프로그램으로 구현해 봅시다.

이세돌 9단과 알파고의 대국만큼 거창하지는 않지만 컴퓨터와 사람이 겨루는 대결입니다.

열 번 안에 맞혀야 하며, 숫자를 맞히면 break문으로 반복문을 빠져나가고 숫자를 맞히지 못하면 반복문의 처음으로 돌아갑니다.



실행 결과

```
게임 1회 : 컴퓨터가 생각한 숫자는? 3 ●——— 사용자가 입력
 아까워요. 4였는데요. 다시 해보세요. ㅠ
게임 2회 : 컴퓨터가 생각한 숫자는? 2 ●——— 사용자가 입력
 아까워요. 3였는데요. 다시 해보세요. ㅠ
게임 3회 : 컴퓨터가 생각한 숫자는? 4 ●——— 사용자가 입력
 아까워요. 3였는데요. 다시 해보세요. ㅠ
게임 4회 : 컴퓨터가 생각한 숫자는? 5 ●——— 사용자가 입력
 맞혔네요. 축하합니다!!
게임을 마칩니다.
```

[LAB] 컴퓨터와 숫자 맞히기 게임하기

1. Lab06_04.java 파일을 만들고, 키보드로 값을 입력받기 위해 Scanner 클래스를 사용할 수 있도록 준비하기

```
import java.util.Scanner;

Scanner s = new Scanner(System.in);

s.close();
```

2. 컴퓨터가 고른 숫자와 사용자가 고른 숫자를 저장할 변수를 선언하기

```
int computer, user;
```

3. 컴퓨터가 1~5 중 임의의 숫자를 고르고, 게임의 현재 횟수를 출력하면서 사용자로부터 숫자를 입력받기. 그리고 이 과정을 10회 반복하기

```
for (int i=1; i<=10; i++) {
    computer = (int)(Math.random()*5 + 1);
    System.out.printf("게임 %d회 : ", i);
    System.out.printf("컴퓨터가 생각한 숫자는? ");
    user = s.nextInt();

}
```

[LAB] 컴퓨터와 숫자 맞히기 게임하기

4. 컴퓨터가 고른 숫자와 사용자가 고른 숫자가 같으면 정답 메시지를 출력하고 for문을 빠져나감. 반대로 숫자가 다르면 오답 메시지를 출력하고 반복문의 처음으로 돌아감

```
if (computer == user) {
    System.out.println(" 맞혔네요. 축하합니다!! ");
    break;
} else {
    System.out.printf(" 아까워요. %d였는데요. 다시 해보세요. ㅠ\n",
computer);
    continue;
}
```

5. 컴퓨터가 고른 숫자와 사용자가 고른 숫자가 같거나 열 번의 기회가 끝나면 게임 종료 메시지를 출력하기

```
System.out.println("게임을 마칩니다." );
23 s.close();
```

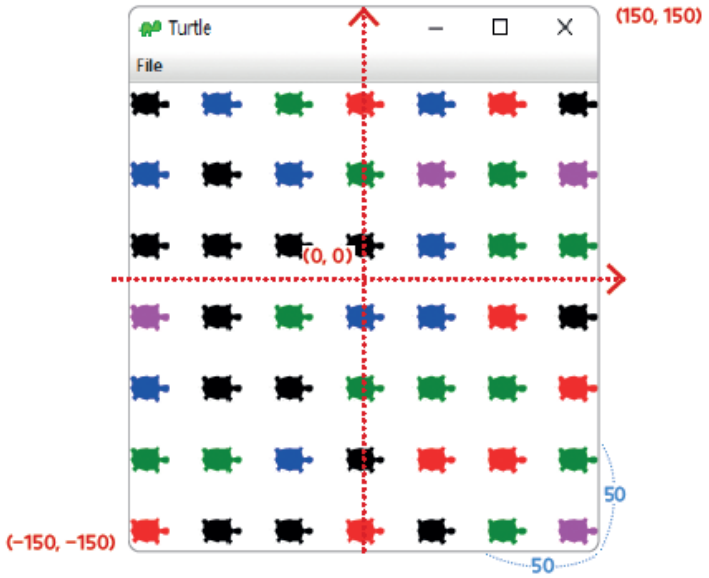
[실전 예제]
거북이 무늬 벽지 만들기

[실전 예제] 거북이 무늬 벽지 만들기

[문제]

중첩 for문을 활용하여 흰 벽지에 거북이 모양의 무늬를 그려봅시다.

벽지의 크기를 가로 300, 세로 300으로 지정하고, 다양한 색상의 거북이를 50만큼 거리를 두고 배치합니다.



[실전 예제] 거북이 무늬 벽지 만들기

[해결]

Ex06_01.java

```
01  public class Ex06_01 {
02      public static void main(String[] args) {
03          Turtle turtle = new Turtle();
04          int x, y;
05          String[] colors = {"red", "green", "magenta", "blue", "black"};
06
07          turtle.speed(100);
08          turtle.outlineColor("white");
09          turtle.setCanvasSize(330, 330);
10          turtle.up();
11
12          for (int i=0; i<7; i++) {
13              for (int k=0; k<7; k++) {
14                  x = i*50 - 150;
15                  y = k*50 - 150;
16                  turtle.setPosition(x, y);
17
18                  int num = (int)(Math.random()*colors.length);
19                  turtle.fillColor(colors[num]);
20                  turtle.stamp();
21              }
22          }
23
24      }
25  }
```

Thank you!