

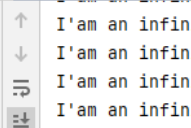
# 2021.05.12 Java

## <for를 이용한 반복문>

```
public class _1st_ForTest {  
    public static void main(String[] args) {  
  
        // for문의 구성;  
        // >> for(초기화 코드; 조건식 코드; 증감식 코드)로 구성.  
        // 초기화 코드는 for문에 최초 진입식 한 번만 실행.  
        // 조건식 코드는 for문 내부{}로 진입하기 위한 조건  
        // 증감식 코드는 for문 내부의 동작이 완료된 이후 동작할 코드  
        // 이후 다시 조건식 코드로 가서 조건을 비교하고 충족한다면 루틴을 반복  
        // 만약 조건식 코드에서 조건이 위배된다면 for문 종료.  
        for(int i =1; i <= 10; i++){  
            System.out.println("i = " + i);  
        }  
    }  
}
```

### <for를 이용한 무한루프>

```
_1st_ForTest.java ×  _2nd_InfinityLoopWithForTest.java ×
1 public class _2nd_InfinityLoopWithForTest {
2     public static void main(String[] args) {
3
4         // while(true)와 다르게 for의 경우엔 무조건 식을 만들 수 있음
5         // 조건부에 아무런 조건이 없다? >> 무조건
6         for(;;){
7             System.out.println("I'am an infinite loop too");
8         }
9
10
11        // for(int i = 1; i++){
12        //     System.out.println("i = " + i);
13        // }
14
15        //     for(int i = 1, j = 0; i++, j--){
16        //         System.out.println("i = " + i + ", j = " + j);
17        //     }
18
19    }
20 }
21
```

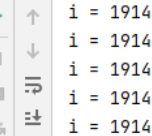


The screenshot shows a terminal window with a title bar that includes the text "Run:" and a tab labeled "\_2nd\_InfinityLoopWithForTest". The terminal output consists of ten lines, each containing the text "I'am an infinite loop too". On the left side of the terminal, there is a vertical toolbar with various icons, including a green play button at the top, a trash can icon, and a magnifying glass at the bottom. At the bottom of the terminal window, the text "Process finished with exit code -1" is displayed in blue.

```

1  ▶ public class _2nd_InfinityLoopWithForTest {
2  ▶     public static void main(String[] args) {
3
4      // while(true)와 다르게 for의 경우엔 무조건 식을 만들 수 있음
5      // 조건부에 아무런 조건이 없다? >> 무조건
6      for(;;){
7          System.out.println("I'am an infinite loop too");
8      }
9
10
11     for(int i = 1; ; i++){
12         System.out.println("i = " + i);
13     }
14
15     // for(int i = 1, j = 0; ; i++, j--){
16     //     System.out.println("i = " + i + ", j = " + j);
17     // }

```



```
Run: _2nd_InfinityLoopWithForTest
i = 191431
i = 191432
i = 191433
i = 191434
i = 191435
i = 191436
i = 191437
i = 191438
i = 191439
i = 191440
i = 191441
i = 191442
i = 191443
i = 191444
i = 191445
```

```

1 public class _2nd_InfinityLoopWithForTest {
2     public static void main(String[] args) {
3
4         // while(true)와 다르게 for의 경우엔 무조건 식을 만들 수 있음
5         // 조건부에 아무런 조건이 없다? >> 무조건
6         for(;;){
7             System.out.println("I'am an infinite loop too");
8         }
9
10
11         for(int i = 1; ; i++){
12             System.out.println("i = " + i);
13         }
14
15         for(int i = 1, j = 0; ; i++, j--){
16             System.out.println("i = " + i + ", j = " + j);
17         }
18     }
19 }
20

```

Run: \_2nd\_InfinityLoopWithForTest

```

i = 549457, j = -549456
i = 549458, j = -549457
i = 549459, j = -549458
i = 549460, j = -549459
i = 549461, j = -549460
i = 549462, j = -549461
i = 549463, j = -549462
i = 549464, j = -549463
i = 549465, j = -549464
i = 549466, j = -549465
i = 549467, j = -549466

```

**<Quiz11>**  
**for 사용해서**  
**1~10까지 출력하는 프로그램 만들기**

```

1 public class _3rd_Quiz11 {
2     public static void main(String[] args) {
3         // Quiz 11. for 사용해서
4         // 1~10까지 출력하는 프로그램 만들기
5         for(int i = 1; i <= 10; i++){
6             System.out.printf("%3d", i);
7             // %3d는 먼저 %d가 정수형 숫자를 출력하는 역할을 수행함을 상기.
8             // %3d에서 '3'의 의미는 '3칸을 확보' 하라는 뜻이다.
9             // 숫자 '10'의 경우 '2칸'을 차지하니까
10            // '%2d'를 사용했다면 9와 10이 구별이 안됨. 이런식으로 될거임 >> 7_8_910
11            // 100까지도 해볼 것
12
13            if(i % 5 == 0){
14                System.out.println();
15                // i 값이 5로 나뉘서 떨어지면 엔터 적용
16                // 아무것도 출력 안하고 엔터만 적용함(println()) >> 5에서 줄바꿈 되는 식으로 주출.
17            }
18        }
19    }
20

```

Run: \_3rd\_Quiz11

```

"C:\Program Files\Java\jdk-16\bin\java.exe" -javaagent:C:\Users\Samuel\AppData\Local\JetBrains\
1 2 3 4 5
6 7 8 9 10

```

```
25 for(int i = 1; i <= 100; i++) {
26     System.out.printf("%3d", i);
27     if (i % 5 == 0) {
28         System.out.println();
29     }
```

Run: \_3rd\_Quiz11 x

```
31 32 33 34 35
36 37 38 39 40
41 42 43 44 45
46 47 48 49 50
51 52 53 54 55
56 57 58 59 60
61 62 63 64 65
66 67 68 69 70
71 72 73 74 75
76 77 78 79 80
81 82 83 84 85
86 87 88 89 90
91 92 93 94 95
96 97 98 99 100
```

**<Quiz12>**  
for 사용해서  
1~20에서 3의 배수를 출력하는 프로그램 만들기.(if 사용 없이)

```
_3rd_Quiz12.java x
1 public class _3rd_Quiz12 {
2     public static void main(String[] args) {
3
4         //Quiz 12. for 사용해서
5         // 1~20에서 3의 배수를 출력하는 프로그램 만들기.(if 사용 없이)
6
7         for(int i = 3; i<=20; i+=3){
8             System.out.println(i);
9         }
```

Run: \_3rd\_Quiz12 x

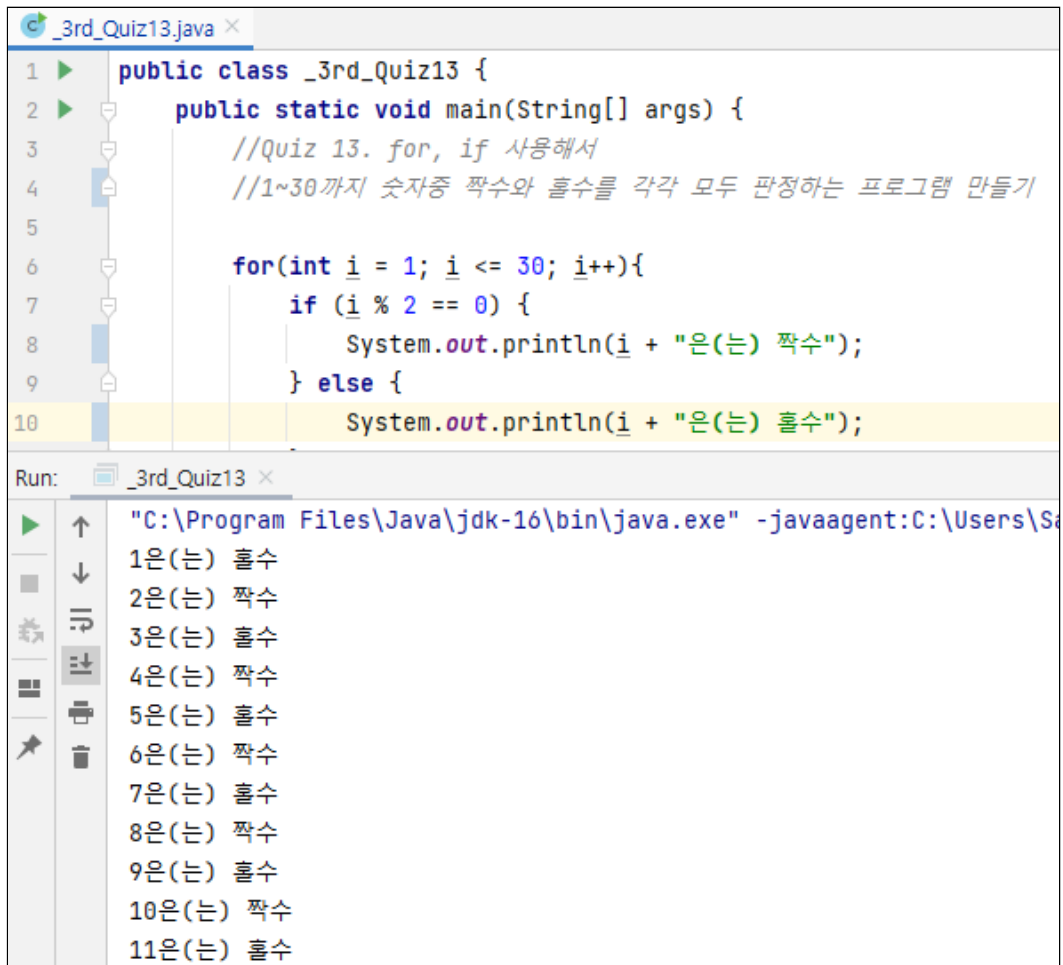
```
"C:\Program Files\Java\jdk-16\bin\java.exe" -javaagent:C:\User
3
6
9
12
15
18

Process finished with exit code 0
```

### <Quiz13>

for, if 사용해서

1~30까지 숫자중 짝수와 홀수를 각각 모두 판정하는 프로그램 만들기



The screenshot shows an IDE with a Java file named `_3rd_Quiz13.java`. The code defines a public class `_3rd_Quiz13` with a `main` method. The `main` method contains a loop from `i = 1` to `i = 30`. Inside the loop, it checks if `i` is even (`i % 2 == 0`). If even, it prints `i` followed by "(는) 짝수". If odd, it prints `i` followed by "(는) 홀수". The output window shows the results of the program execution, displaying the numbers 1 through 11 and their corresponding parity.

```
1 public class _3rd_Quiz13 {  
2     public static void main(String[] args) {  
3         //Quiz 13. for, if 사용해서  
4         //1~30까지 숫자중 짝수와 홀수를 각각 모두 판정하는 프로그램 만들기  
5  
6         for(int i = 1; i <= 30; i++){  
7             if (i % 2 == 0) {  
8                 System.out.println(i + "(는) 짝수");  
9             } else {  
10                System.out.println(i + "(는) 홀수");  
11            }  
12        }  
13    }  
14 }
```

Run: `_3rd_Quiz13`

```
"C:\Program Files\Java\jdk-16\bin\java.exe" -javaagent:C:\Users\S  
1은(는) 홀수  
2은(는) 짝수  
3은(는) 홀수  
4은(는) 짝수  
5은(는) 홀수  
6은(는) 짝수  
7은(는) 홀수  
8은(는) 짝수  
9은(는) 홀수  
10은(는) 짝수  
11은(는) 홀수
```

### <for문을 활용한 합산 + 정수형 / 정수형 & 정수형 / 소수점 차이>

```
1 public class _4th_AverageWithForTest {
2     public static void main(String[] args) {
3         int sum = 0;
4
5         for (int i = 1; i <= 10; i++){
6             sum += i;
7             System.out.println("sum = " + sum);
8         }
9         System.out.println("최종 합산 값 = " + sum);
10
11         float average = sum / 10.0f;
12         // 10.0f 말고 뭔가 다른 거 사용할 수 있을 것 같은데 생각해볼 것
13         // for문이 반복된 횟수를 나타내는 코드??
14         System.out.println("평균 = " + average);
15
16         // 위와 아래의 값이 다름.
17         // 아래의 케이스는 sum이 int형, 숫자 10도 default로 int형.
18         // 기본적으로 정수는 int, 소수점은 double을 채택함
19         //
20         // 위의 케이스는 sum은 int형이지만 나누는 숫자가 float이기 때문에
21         // 강제로 소수점 연산이 수행되어 5.5라는 결과를 얻게됨.
22         average = sum / 10;
23         System.out.println("평균 = " + average);
24     }
25 }
```

Run: \_4th\_AverageWithForTest

```
"C:\Program Files\Java\jdk-16\bin\java.exe" -javaagent:C:\Users\Samuel\
sum = 1
sum = 3
sum = 6
sum = 10
sum = 15
sum = 21
sum = 28
sum = 36
sum = 45
sum = 55
최종 합산 값 = 55
평균 = 5.5
평균 = 5.0
```

Q.

>11) float average = sum / 10.0f;에서

>10.0f 대신에 loop가 반복된 횟수를 넣는 코드는 없는지?

### <for문을 이용한 Math.random() 사용법 예제>

20210512\_Java > src > \_5th\_RandomTest > main

```
public static void main(String[] args) {  
    for ( ; ; ) {  
        // Math.random()은 [0~1)에 해당하는 소수점 데이터를 출력한다.  
        //// [0~1) 표현은 0 이상 1 미만 이라는 뜻 //  
        // 여기에 10을 곱하기 때문에 0.0 이상 10.0 미만 ( 9.9xxxxxxx 이하)  
        // 소수점이지만 값을 앞에서 int로 처리했기 때문에 소수점은 버려짐  
        // 따라서 0~9까지의 정수가 출력.  
        System.out.println((int) (Math.random() * 10)); // 헛갈릴까봐 소괄호를 조금 띄어둠.  
    }  
}
```

Run: \_5th\_RandomTest

0  
6  
5  
2  
3  
3  
8  
1  
5

## <주사위 만들기 + 랜덤 응용>

```

1 public class _6th_DiceTest {
2     public static void main(String[] args) throws InterruptedException {
3         System.out.println("주사위를 굴리자");
4
5         while (true) {
6             // [0.0 ~ 1.0)
7             // 0.0 이상 1.0미만
8             // 위에 6을 곱했으니 0.0이상 6.0미만(5.9xxxxxxx 이하)
9             // 거기에 +1을 하면 1.0이상 7.0미만(6.9xxxxxxx 이하)
10            // int형으로 정의했기 때문에 1~6이 출력됨.
11            System.out.println((int)(Math.random() * 6 + 1));
12            Thread.sleep(500);
13            // 위에 throws InterruptedException 자동추가하는 법 기억할 것
14            // (sleep에 error 났을 때 마우스 올려두면 나옴)
15            // 500도 기입하면 자동으로 millis 나오는 것도 개념확인
16            // 1초 = 1000 밀리세컨드 = 1000000 마이크로세컨드 = 1000000000 나노세컨드
17            // 1초 = 10^3 ms = 10^6 us = 10^9 ns
18            // 결국 0.5초 대기하라는 의미
19
20            // 랜덤에도 종류가 있음.
21            // 가우시안 랜덤(정규분포), 푸아송 분포, 이항 분포, 기하 분포 등등
22            // >> 인공지능 스터디에서는 중요한 요소긴 하지만 아직 배울 필요 없음
23            // Math.random() 이라는 랜덤은 Uniform Random >> 모든 항목이 동일한 확률을 가진다.
24        }
25    }
26}

```

Run: \_6th\_DiceTest

```

"C:\Program Files\Java\jdk-16\bin\java.exe" -javaagent:C:\Users\Samuel\AppData\Local\JetBra
주사위를 굴리자
6
6
3
4
5
2
4

```

```

System.out.println((int)(Math.random() * 6 + 1));
Thread.sleep(500);
// 위에 thr
// (sleep
// 500도
// 1초 = 1
// 1초 = 1
// 결국 0.5
// 랜덤에도
// 가우시안
// >>
// Math.ra

```

Unhandled exception: java.lang.InterruptedException

Add exception to method signature Alt+Shift+Enter More actions... Alt+Enter

java.lang.Thread  
public static void sleep(long millis)  
throws InterruptedException

Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers. The thread does not lose ownership of any monitors.

Params: millis – the length of time to sleep in milliseconds

Throws: **IllegalArgumentException** – if the value of millis is negative  
**InterruptedException** – if any thread has interrupted the current thread. The *interrupted status* of the current thread is cleared when this exception is thrown.

2) throws InterruptedException 추가방법  
> throws InterruptedException의 개념은 다음 시간에.