

# Java Study Skill Up Team Review

---


## 연산자

### 1. overflow, underflow

- overflow
  - 타입이 허용하는 최대값을 벗어나는 것
- underflow
  - 타입이 허용하는 최소값을 벗어나는 것
- byte
  - $127 + 1 = 128$  ↪ overflow 발생: -128
  - $-128 - 1 = -129$  ↪ underflow 발생: 127

### 2. NaN, Infinity

- NaN
  - Not a Number
- Infinity
  - 무한대

 좌측 피연산자가 정수이고 우측 피연산자가 0일 경우

```
package example;

public class Chapter03 {
    public static void main(String[] args) {
        int x = 5;
        int y = 0;
        int result = x % y;
        System.out.println(result);
    }
}
```

#### [ 오류 발생 ]

- ArithmeticException

#### [ 오류 해석 ]

- 수치 연산(수학적 연산을 컴퓨터로 수행)에서 예외가 발생한 경우 발생하는 오류이다.
- 정수를 0으로 나누려고 할 때 또는 정수를 나눌 때 결과가 정수가 아닌 경우 발생하는 오류이다.

#### [ 오류 해결 ]

- 연산 전에 0으로 나누는지 확인하고 예외 처리를 사용하여 오류를 처리해야 한다.

좌측 피연산자가 실수이거나 우측 피연산자가 0.0 또는 0.0f 라면 예외 발생은 없지만 결과는 Infinity 또는 NaN

```
package example;

public class Chapter03 {
    public static void main(String[] args) {
        System.out.println(5 / 0.0);    // Infinity
        System.out.println(5 % 0.0);    // NaN
    }
}
```

- 결론적으로 위와 같이 Infinity 또는 NaN 상태에서 계속해서 연산을 수행하면 안된다.
- 어떤 연산을 하더라도 결과는 계속 Infinity 와 NaN이 되기 때문이다.(영망)

그렇기 때문에 Infinity 또는 NaN 인지 먼저 확인을 하고 연산을 수행하는 것이 좋다.

```
package example;

public class Chapter03 {
    public static void main(String[] args) {
        int x = 5;
        double y = 0.0;
        double result = x / y;

        if (Double.isInfinite(result) || Double.isNaN(result)) {
            System.out.println("값 산출 불가 이유: " + (Double.isInfinite(result) ?
                "Infinity" : "NaN"));
        } else {
            System.out.println("이후 산출 수행");
        }
    }
}
```

- 출력: 값 산출 불가 이유: Infinity

### 3. 삼항 연산자

```
package example;

public class Chapter03 {
    public static void main(String[] args) {
        String ternaryOperatorField = "나는 삼항";
        boolean ternaryOperatorBooleanField = true;

        System.out.println(ternaryOperatorField == "나는 삼항" ? "YES 삼항" : "NO
삼항");
        System.out.println(ternaryOperatorField != "나는 이항" ? "NO 이항" : "YES
```

```

이항");
    System.out.println(ternaryOperatorBooleanField ? "true" : "false");
    System.out.println(!ternaryOperatorBooleanField ? "true" : "false");
}
}

```

- 삼항 연산자는 말 그대로 총 3개의 피연산자를 가진다.
- `boolean ? true : false`

## 조건문과 반복문

### 4. if 문

```

package example;

public class Chapter04 {
    public static void main(String[] args) {
        int x = 10;

        if (x == 0) {
            System.out.println("x가 0");
        } else if (x == 5) {
            System.out.println("x가 5");
        } else if (x == 7) {
            System.out.println("x가 7");
        } else if (x == 10) {
            System.out.println("x가 10");
        }
    }
}

```

- 코드가 길어진다면 아래 코드가 조금 더 가독성이 좋다고 볼 수 있다.

```

package example;

public class Chapter04 {
    public static void main(String[] args) {
        int x = 10;

        if (x == 0) {
            System.out.println("x가 0이면");
        }

        if (x == 5) {
            System.out.println("x가 5면");
        }

        if (x == 7) {
            System.out.println("x가 7이면");
        }
    }
}

```

```
    }

    if (x == 10) {
        System.out.println("x가 10이면");
    }
}
}
```

## 5. switch 문

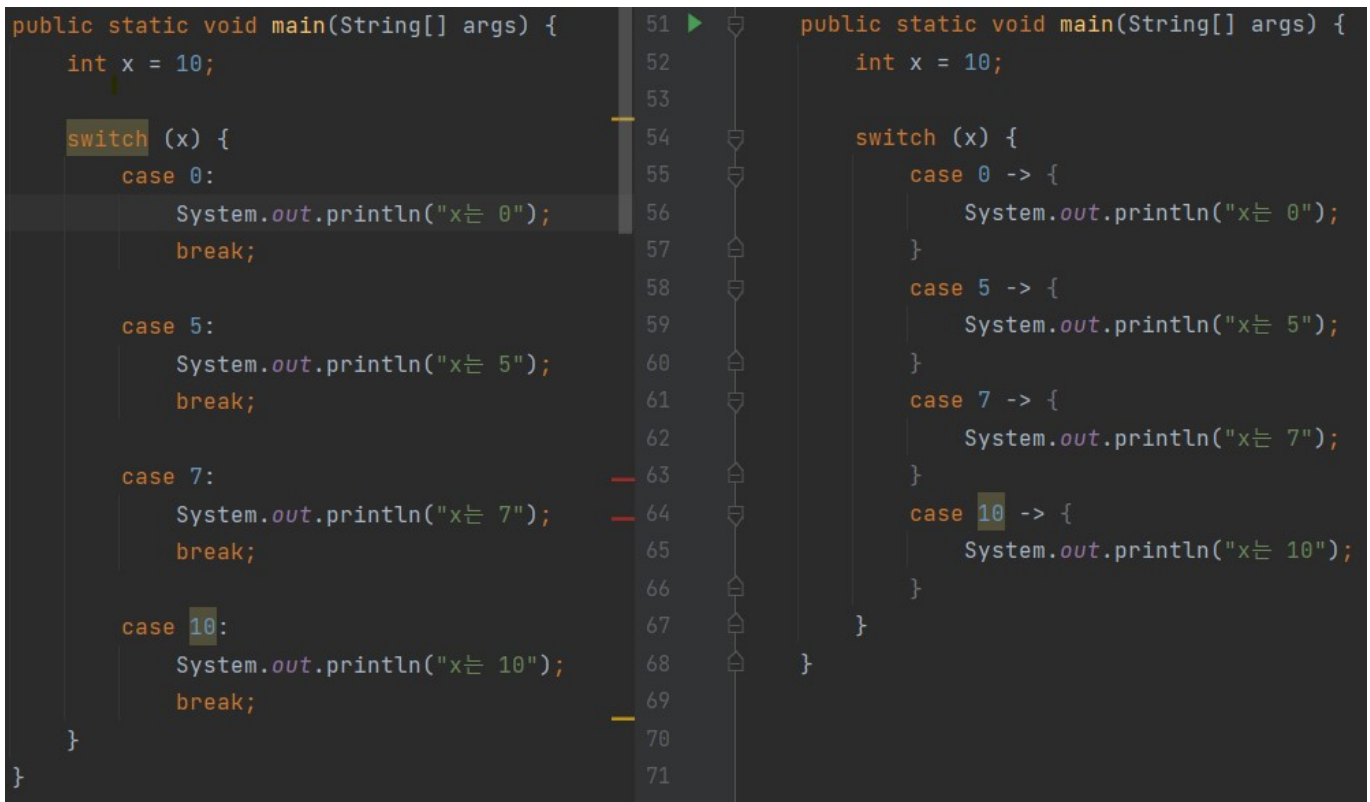
- 조건문 처리에 조금 더 코드가 간결해진다.

```
package example;

public class Chapter04 {
    public static void main(String[] args) {
        int x = 10;

        switch (x) {
            case 0 -> {
                System.out.println("x는 0");
            }
            case 5 -> {
                System.out.println("x는 5");
            }
            case 7 -> {
                System.out.println("x는 7");
            }
            case 10 -> {
                System.out.println("x는 10");
            }
        }
    }
}
```

- 자바 12 이후부터 Expressions(표현식) 사용이 가능하다.
- break 문을 없애는 대신 화살표와 중괄호를 사용해 가독성이 좋아졌다.
- 실행문이 하나만 있을 경우 중괄호를 생략할 수 있지만 넣는 것이 이슈 방지 차원으로 더 좋다.



- Java 13 이후부터는 yield 키워드로 값을 지정해서 대입할 수도 있다.
- 단 이 경우에는 default 반드시 존재해야 한다.

```

package example;

public class Chapter04 {
    public static void main(String[] args) {

        int x = 10;

        int yieldTestField = switch (x) {
            case 10 -> {
                int won = 1;
                yield won;
            }
            default -> 200;
        };

        System.out.println(yieldTestField);
    }
}

```

## 6. for 문

```


for(int i = 0; i <= 50; i++) {
    // 초기화, 조건식, 증감식
}

```

## 7. while 문

```
package example;

public class whileClass {
    public static void main(String[] args) {
        while(true) {
            System.out.println("while 입니다.");
            break;
        }
    }
}
```

- break; 문이 빠지면?
  -  무한 루프