

# 컴퓨터 프로그래밍 (Computer Programming)

이 선 순



# 12. 파일 입출력



# 목차

01. 예외 처리

02. 표준입출력

03. 파일입출력

**01**

**예외 처리**

# 01 예외 처리

---

## ■ 오류가 발생하면...

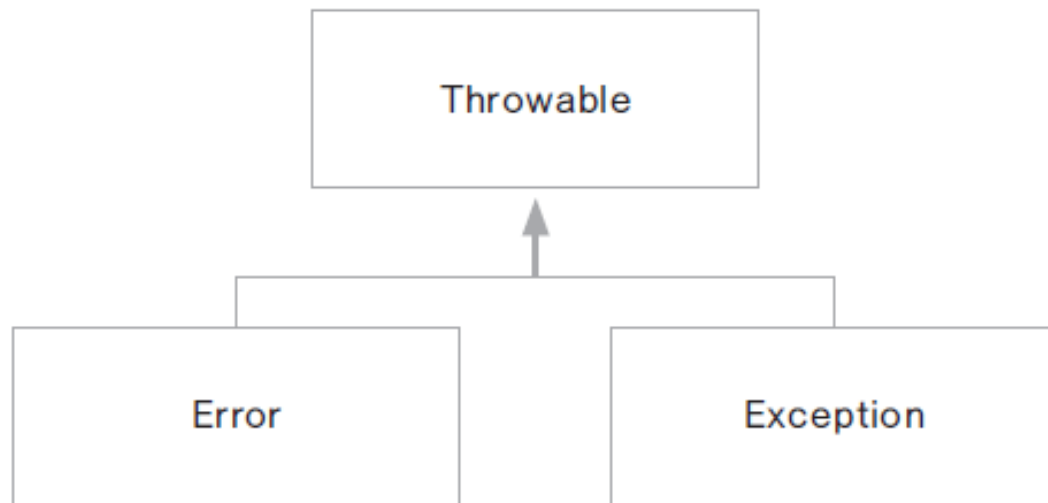
- 소프트웨어를 사용하다 보면 여러가지 상황의 오류가 발생함.
- 잘 접속되던 사이트가 접속이 안되거나, 스마트폰 앱이 갑자기 종류가 되는 경우
- 아무리 잘 만든 소프트웨어라도 이런 상황은 발생함
- 이런 일이 발생하더라도 갑자기 종료되는 상황이 발생하지 않도록 하는 것이 예외 처리임

## ■ 오류란 무엇인가요?

- 컴파일 오류(compile error) : 프로그램 코드 작성 중 발생하는 문법적 오류
- 실행 오류(runtime error) : 실행중인 프로그램이 의도하지 않은 동작(bug)을 하거나 프로그램이 중지되는 오류
- 실행 오류 시 비정상 종료는 서비스 운영에 치명적
- 오류가 발생할 수 있는 경우에 로그(log)를 남겨 추후 이를 분석하여 원인을 찾아야 함
- 자바는 예외 처리를 통하여 프로그램의 비정상 종료를 막고 log를 남길 수 있음

## ■ 오류와 예외

- 실행 오류는 크게 두가지가 있는데, 시스템 오류(error)와 예외(exception)
- 시스템 오류 : 자바가상머신에서 발생하는 오류로 프로그래머가 처리할 수 없음
  - 동적메모리가 없는 경우, 스택메모리의 오버플로가 발생한 경우 등
- 예외 : 프로그램에서 제어할 수 있는 오류
  - 프로그램에서 파일을 읽어 사용하려는데 읽어 들이려는 파일이 존재하지 않는 경우, 네트워크로 데이터를 전송하려는데 연결이 안된 경우, 배열 값을 출력하려는데 배열 요소가 없는 경우 등



# 01 예외 처리

## ■ 예외클래스의 종류

- 모든 예외 클래스의 최상위 클래스는 Exception
- Exception 클래스의 내용

### **Class Exception**

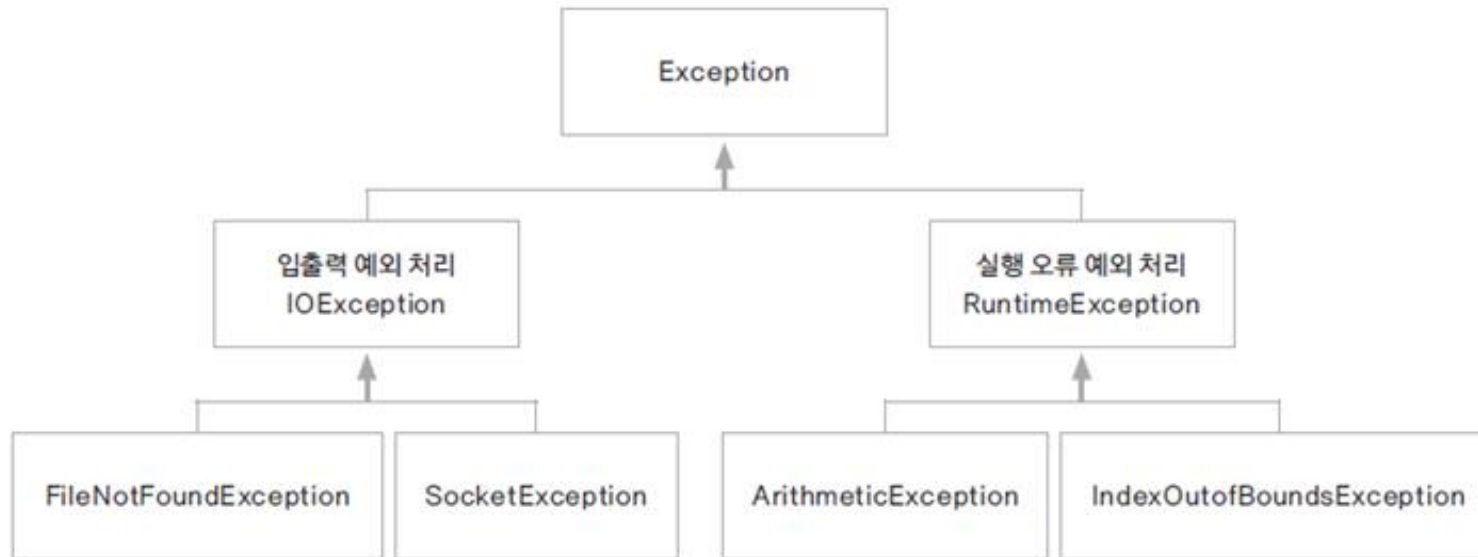
```
java.lang.Object  
    java.lang.Throwable  
        java.lang.Exception
```



# 01 예외 처리

## ■ 예외클래스의 종류

- 예외 타입은 클래스로서 서로 상속 관계이다.
- 다음은 Exception 하위 클래스 중 사용 빈도가 높은 클래스 위주로 계층도를 표현한 것
  - RuntimeException을 사용하면 ArithmeticException 등 그 아래의 예외가 모두 해당됨.
  - IOException 클래스 : 입출력에 대한 예외를 처리
  - RuntimeException : 프로그램 실행 중 발생할 수 있는 오류에 대한 예외를 처리



## ■ 예외 처리하기

- 이클립스 같은 개발 환경에서는 예외가 발생하면 대부분 처리하라는 컴파일 오류 메시지를 띄움 => try~catch 문을 사용하여 예외처리를 해야함.
- Exception 하위 클래스 중 RuntimeException은 try~catch 문을 사용하여 예외처리를 하지 않아도 컴파일 오류가 나지 않음
- RuntimeException 하위 클래스 중 ArithmeticException은 산술 연산 중 발생할 수 있는 예외, 즉 '0으로 숫자 나누기'와 같은 경우에 발생하는 예외임.
- 이렇게 컴파일러에 의해 체크되지 않는 예외는 프로그래머가 알아서 처리해야 하므로 주의해야함

# 01 예외 처리

## ■ 예외 처리하기

### ■ 구문오류 경우

- 오타 혹은 정의되지 않은 변수를 사용하는 등 구문상의 오류일때 이클립스 에디터에 빨간줄이 나타남.
- 행 번호 앞에 빨간색의 작은 x로 구문상의 오류 표시
- 'String'을 'Strong'으로 입력하여 오류 발생함
- 구문오류 발생 시 실행이 안되나 그림과 같이 강제로 실행 가능하나 더 이상 진행되지 않고 화면에 오류가 뜬
- 구문 오류 수정 후 실행해야 함
- 구문 오류는 예외 처리에 포함되지 않음

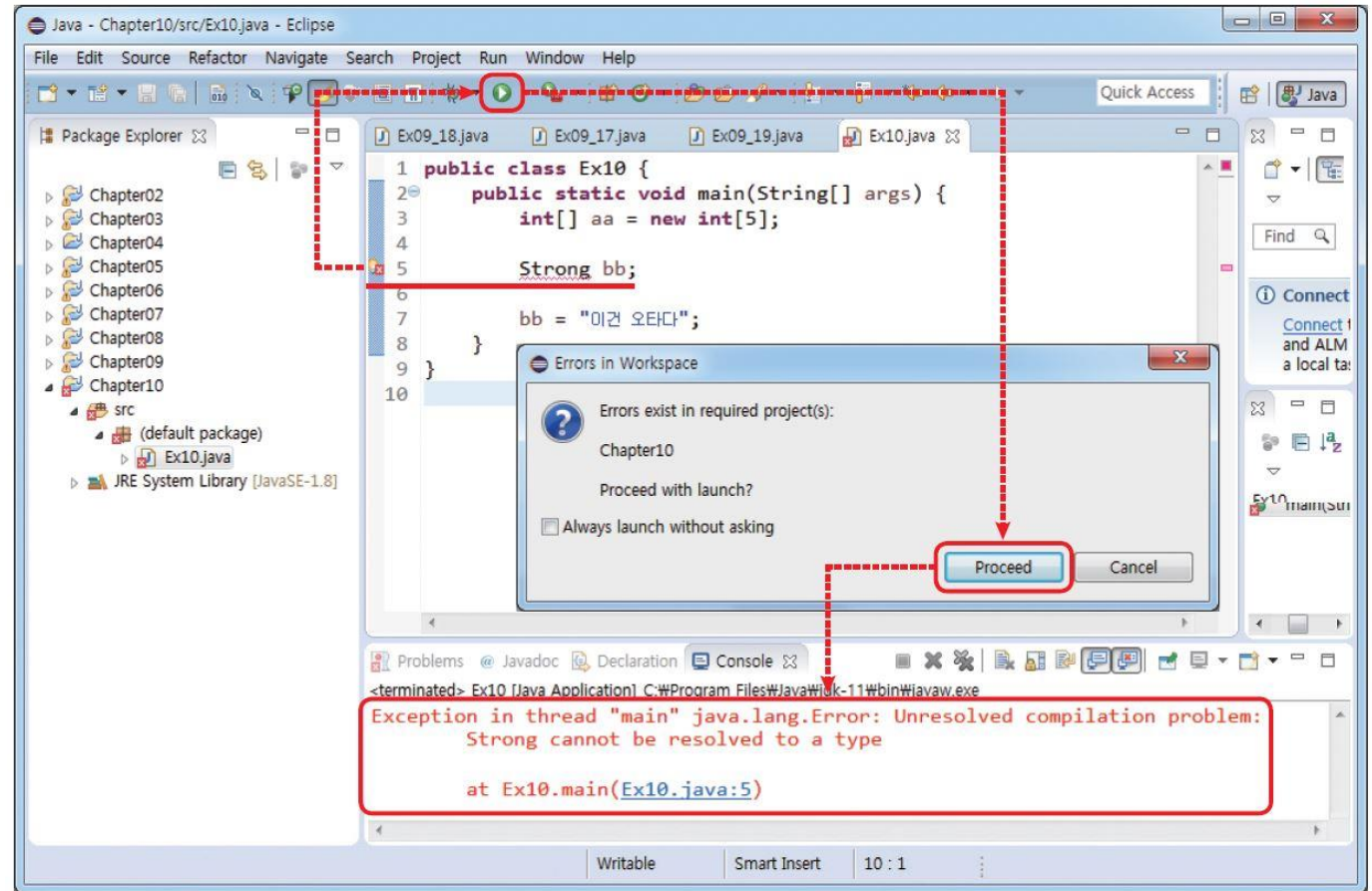


그림 10-1 구문 오류 발생의 예

# 01 예외 처리

## ■ 예외 처리하기

### ■ 구문오류가 없는데 실행 시 오류가 발생하는 경우

- 이클립스 에디터에는 오류 표시가 없는데 실행하면 오류가 발생하는 경우
- 그림과 같이 배열을 'int[]aa = new int[3]으로 3개를 설정하고 aa[3] 같은 곳을 접근하는 경우
- 오류가 발생하면 오류의 원인과 행 번호가 표시되는데, 그 구분을 클릭하면 오류가 발생한 행으로 커서가 이동함
- JAVA에서 예외처리는 그림과 같이 **구문에는 오류가 없으나 실행 시 오류가 발생하는 경우에 대신 처리해줌**

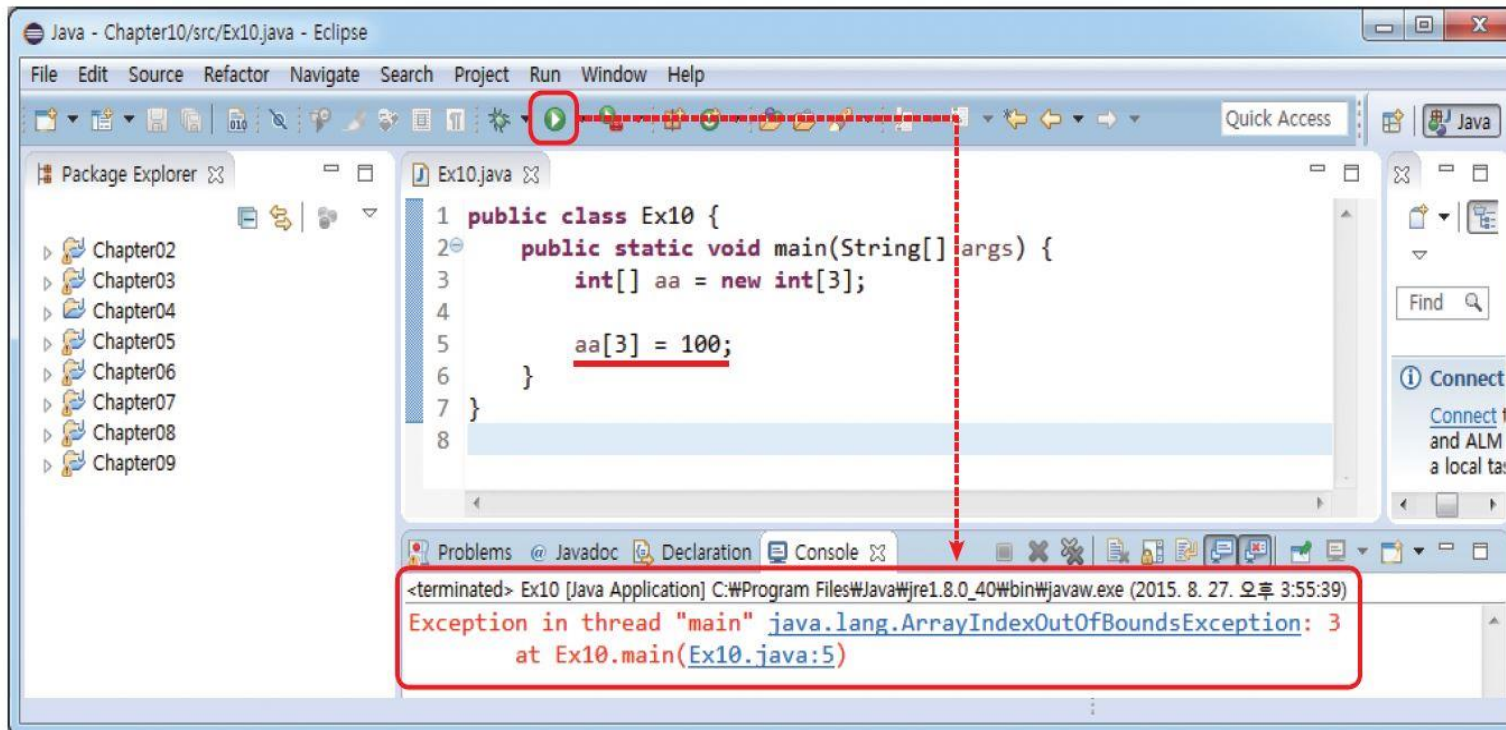


그림 10-2 실행 시 오류 발생의 예

# 01 예외 처리

## ■ 예외 처리의 기본 형식

- 예외 처리(exception handling)는 오류가 발생할 경우 프로그래머가 작성한 부분이 실행되도록 try~catch로 준비하는 것

- try ~ catch문

- 예외 타입 다음의 e는 변수로서 오류 내용이 여기에 포함: try 블록 안에서 예외가 발생했을 때 예외를 처리하는 부분

```
try {  
    JAVA 코드...  
} catch (예외 타입 e) {  
    예외 발생 시 이 부분이 실행됨  
}
```

- try ~ catch문 사용

```
int[] arr = new int[5];  
  
for (int i = 0 ; i < 5 ; i++){  
    arr[i] = i;  
    System.out.println(arr[i]);  
}
```

0  
1  
2  
3  
4



```
int[] arr = new int[5];  
  
for (int i = 0 ; i <= 5 ; i++){  
    arr[i] = i;  
    System.out.println(arr[i]);  
}
```

0  
1  
2  
3  
4

-----  
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5  
at .(#17:1)

# 01 예외 처리

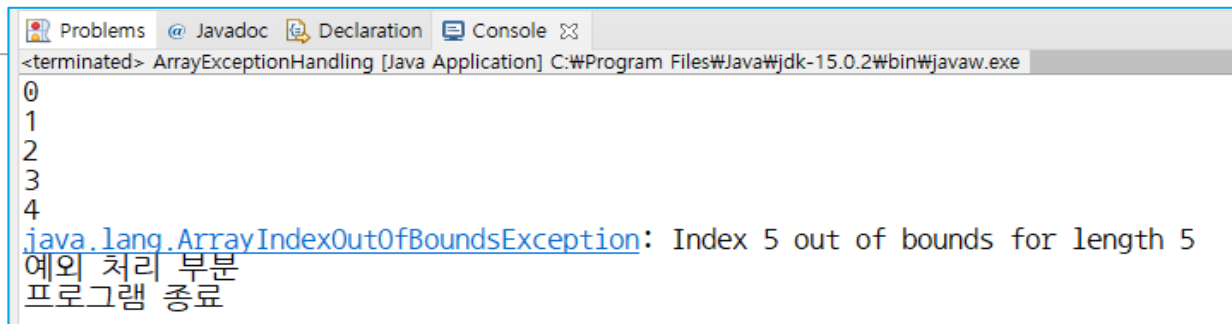
## ■ 예외 처리의 기본 형식

### ■ try ~ catch문 사용

```
1 package hello;
2
3 public class ArrayExceptionHandling {
4
5     public static void main(String[] args) {
6         int[] arr = new int[5];
7
8         try{
9             for(int i =0 ; i<=5 ; i++) {
10                 arr[i] =i;
11                 System.out.println(arr[i]);
12             }
13         } catch(ArrayIndexOutOfBoundsException e) {
14             System.out.println(e);
15             System.out.println("예외 처리 부분");
16         }
17         System.out.println("프로그램 종료");
18     }
19 }
```

예외가 발생할 수 있으므로 try블록에 작성

예외가 발생하면 catch 블록 수행



```
Problems @ Javadoc Declaration Console
<terminated> ArrayExceptionHandling [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe
0
1
2
3
4
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
예외 처리 부분
프로그램 종료
```

- 배열범위가 유효한 값 4까지는 배열에 저장되어 출력되고 그 다음 값을 배열에 넣으면 예외가 발생함
- 발생한 예외는 catch블록에서 처리하므로 System.out.println("프로그램 종료 ")문장까지 수행하고 프로그램이 정상종료됨
- 예외가 발생하여 프로그램이 바로 비정상 종료되었다면 System.out.println("프로그램 종료 ")문장을 수행할 수 없음.
- 예외 처리는 프로그램이 비정상 종료되는 것을 방지할 수 있으므로 매우 중요함.

# 01 예외 처리

## ■ 예외 처리의 기본 예

- [실습 10-1] [그림 10-2]의 오류를 예외 처리로 코딩 해봄

### 실습 10-1 예외 처리의 기본 예

```
01 public class Ex10_01 {
02     public static void main(String[] args) {
03         int[] aa = new int[3];
04         try {
05             aa[3] = 100;
06         } catch (ArrayIndexOutOfBoundsException e) {
07             System.out.println("배열 첨자가 배열 크기보다 커요 ~~");
08             // ArrayIndexOutOfBoundsException 예외가 발생하면 7행을 실행한다.
09         }
10     }
11 }
```

----- 크기 3의 배열을 선언한다.

----- 기존에 실행되는 코드이다.

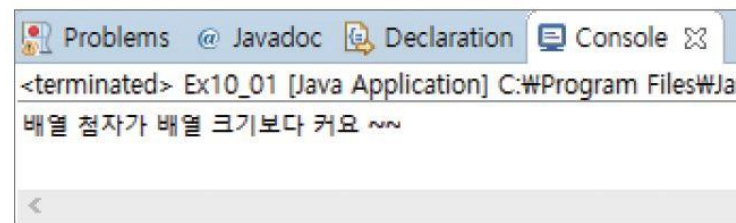


그림 10-3 실행 결과

- [실습 10-1]은 [그림 10-2]와 같이 5행이 실행되는데 try{ }로 묶여 있다는 것이 다름. 만약 try{ } 안에서 오류가 발생하고 그 오류가 ArrayIndexOutOfBoundsException 오류에 해당한다면 catch{ } 내부를 수행. ArrayIndexOutOfBoundsException은 배열의 인덱스가 실제 크기보다 큰 경우에 발생하는 오류.

# 01 예외 처리

## ■ 예외 처리의 전체 형식

- 만약 try 블록 안에서 발생할 수 있는 예외 상황이 여러 개라면 catch 블록을 예외 상황 수만큼 구현해야 함
- 한번 열어놓은 리소스를 해제하는 코드를 try~catch~catch ... 각 블록에 모두 작성해야 한다면 매우 번거로움.
- 이때 사용하는 블록이 finally임
- 일단 try 블록이 수행된다면 finally 블록은 어떤 경우에도 반드시 수행되어야 하며 생략은 가능함. try나 catch문에 return문이 있어도 수행됨
- try~catch~catch ... 각 블록마다 리소스를 해제하지 않고 finally 블록에서 한번만 해제해주면 됨

```
try {  
    JAVA 코드...  
} catch (예외 타입 1 e) {  
    예외 1 발생 시 이 부분이 실행됨  
} catch (예외 타입 2 e) {  
    예외 2 발생 시 이 부분이 실행됨  
} finally {  
    이 부분은 마지막에 무조건 실행됨  
}
```



# 01 예외 처리

## ■ 예외 처리의 전체 형식

### ■ [실습10-2] 예외 처리의 전체 예

```
1 public class Ex10_02 {  
2     public static void main(String[] args) {  
3         int[] aa = new int[3];  
4         try {  
5             aa[2] = 100 / 0; //0으로 나누는 오류가 발생  
6             aa[3] = 100;  
7         } catch (ArrayIndexOutOfBoundsException e) {  
8             System.out.println("배열 첨자가 배열 크기보다 커요~~");  
9         } catch (ArithmeticException e) {  
10            System.out.println("0으로 나누는 등의 오류예요 ~~"); //0으로 나누는 오류 등 수식오류 발생하면 10행 실행  
11        } finally {  
12            System.out.println("이 부분은 무조건 나와요 ~~"); //오류 발생 여부와 관계없이 실행  
13        }  
14    }  
15 }
```

Problems @ Javadoc Declaration Console

<terminated> Ex10\_02 [Java Application] C:\Program Files\Java\jdk-8.0.60\bin\java.exe  
0으로 나누는 등의 오류예요 ~~  
이 부분은 무조건 나와요 ~~

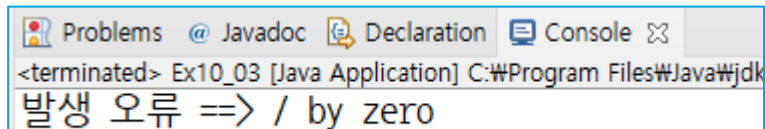
# 01 예외 처리

## ■ 오류메시지 출력

- JAVA에서는 오류가 발생하면 그 오류의 원인을 문자열로 가지고 있음.
- 예외 처리 형식인 `catch(예외 타입 e){}`에서 `e`에는 다양한 오류의 내용이 저장되어 있으며, 필요한 경우 오류를 확인할 수 있음 : `e.getMessage()` 메소드 이용

### ■ [실습10-3] 오류 내용의 출력 예

```
1 public class Ex10_03 {  
2     public static void main(String[] args) {  
3         int a = 100, b = 0;  
4         int result;  
5         try {  
6             result = a / b; //0으로 나누는 오류 발생  
7         } catch (ArithmeticException e) {  
8             System.out.print("발생 오류 ==> ");  
9             System.out.println(e.getMessage()); //e변수에서 메시지를 출력  
10        }  
11    }  
12 }
```



Problems @ Javadoc Declaration Console

<terminated> Ex10\_03 [Java Application] C:\Program Files\Java\jdk  
발생 오류 ==> / by zero

- 7행의 `ArithmeticException` 클래스 타입의 변수 `e`에 대해 9행의 `getMessage()` 메소드로 오류 내용을 추출해서 출력. '/ by zero' 오류 메시지.

# 01 예외 처리

## ■ 오류메시지 직접 만들기

- JAVA가 제공하는 오류 메시지를 사용하지 않고 직접 만들어 사용할 수 있음

```
throw new Exception("사용자가 만든 오류 메시지");
```

# 01 예외 처리

## ■ 오류 메시지 직접 만들기

### ■ [실습10-4] 오류 메시지 직접 만들기 예

```
1 public class Ex10_04 {  
2     public static void main(String[] args) {  
3         int a = 100, b = 0;  
4         int result;  
5         try {  
6             if (b == 0) // 나누는 값 b가 0이면 8행을 실행하기 전에 오류가 발생한다.  
7                 throw new Exception("0으로 나누려고요? 안됩니다.");  
8             result = a / b;  
9         } catch (Exception e) { // 예외 타입을 Exception으로 변경  
10            System.out.print("발생 오류 ==> ");  
11            System.out.println(e.getMessage());  
12        }  
13    }  
14 }
```

Problems @ Javadoc Declaration Console

<terminated> Ex10\_04 [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\java.exe  
발생 오류 ==> 0으로 나누려고요? 안됩니다.

## Self study10-1

---

1. [실습10-4] 를 수정하여 a가 0인 경우에도 '0은 나뉘도 0입니다.' 오류가 추가로 발생하도록 해보자.

Hint : catch는 여러 개를 추가해도 된다.

질문은 이메일을 이용해주세요.  
[ds.june2@gmail.com](mailto:ds.june2@gmail.com)

**감사합니다**