

프로그래밍 언어 응용

chapter14

예외처리

제공된 자료는 훈련생의 수업을 돕기 위한 것으로, 타인과 공유하시면 안됩니다.

Contents

part.1

예외 클래스

part.2

예외 처리하기

part.3

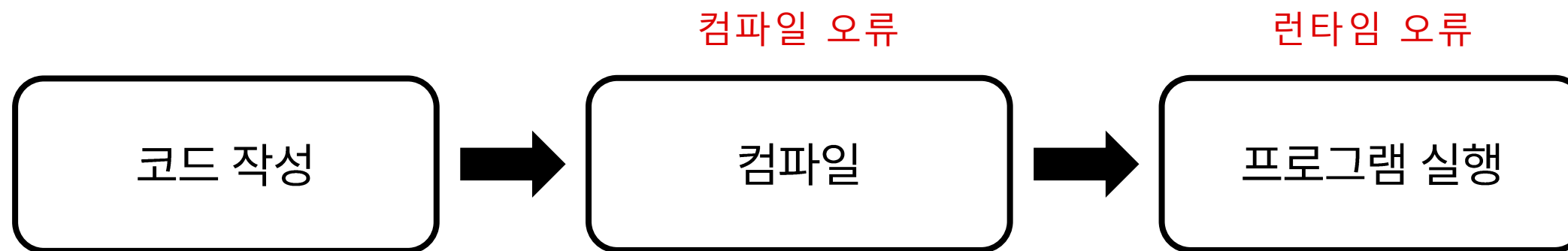
예외 처리 마무리

오류란?

- 프로그램 개발 중에는 다양한 이유로 인해 오류가 발생할 수 있다.

오류의 종류

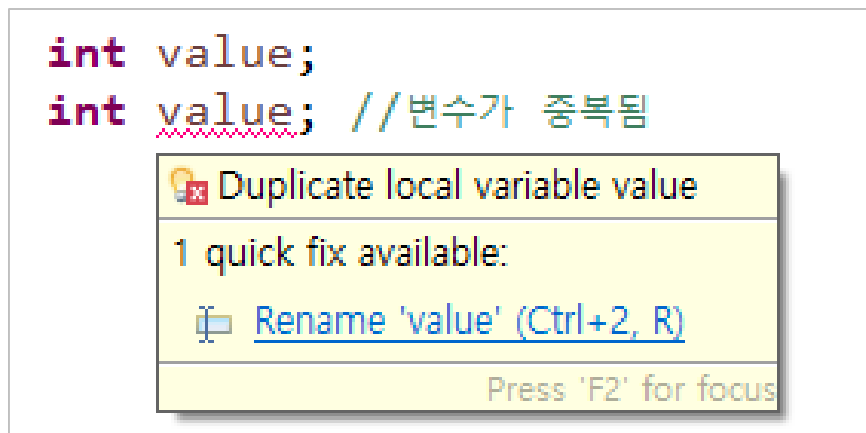
- 컴파일 오류 : 소스코드를 컴파일 할 때 발생하는 에러
- 런타임 오류 : 프로그램 실행 중에 발생하는 에러



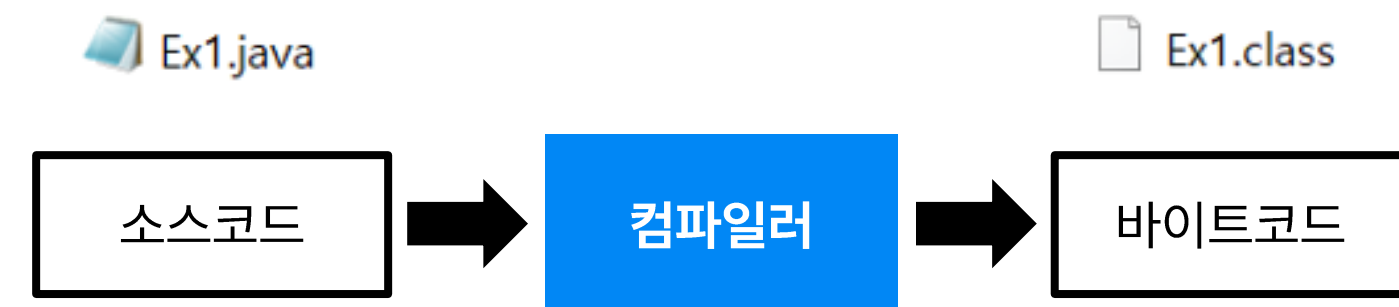
<프로그램 개발 과정>

컴파일 오류란?

- 컴파일 오류는 잘못된 문법을 사용했을 때 발생한다.
(예시: 변수 이름 중복, 데이터 타입 오류, 오타, 문장에 세미콜론이 없음, 블록이 완성이 안됨)
- 컴파일러가 소스 코드를 컴파일하는 동안 문법 오류를 발견하면 컴파일에 실패하고,
- 개발자에게 오류 메시지를 보내준다.



<변수 이름이 중복된 경우>

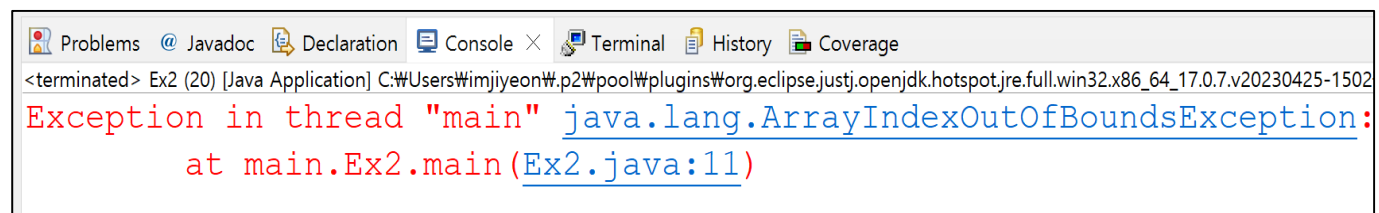


<컴파일 과정>

런타임 오류란?

- 런타임오류는 프로그램이 실행 중일 때 발생한다.
(예시: 0으로 나누기, 잘못된 형 변환, 배열 범위를 벗어남, Null 참조)
- 런타임오류가 발생하면 프로그램이 갑자기 종료된다.

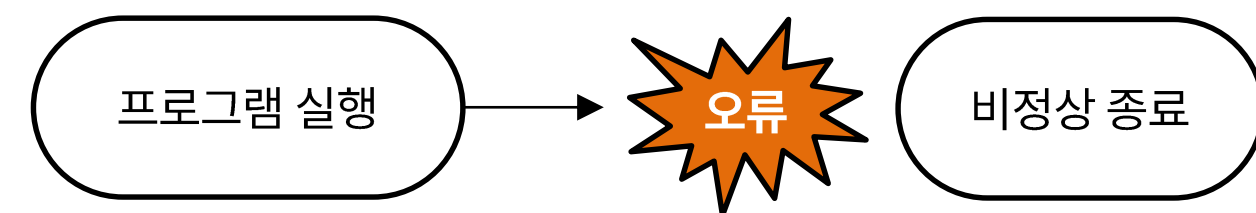
```
int[] arr = new int[5];  
arr[5] = 5;
```



The screenshot shows an IDE console window with the following text:

```
<terminated> Ex2 (20) [Java Application] C:\Users\Wimjiyeon\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.7.v20230425-1502  
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:  
    at main.Ex2.main(Ex2.java:11)
```

<런타임 오류가 발생한 경우>

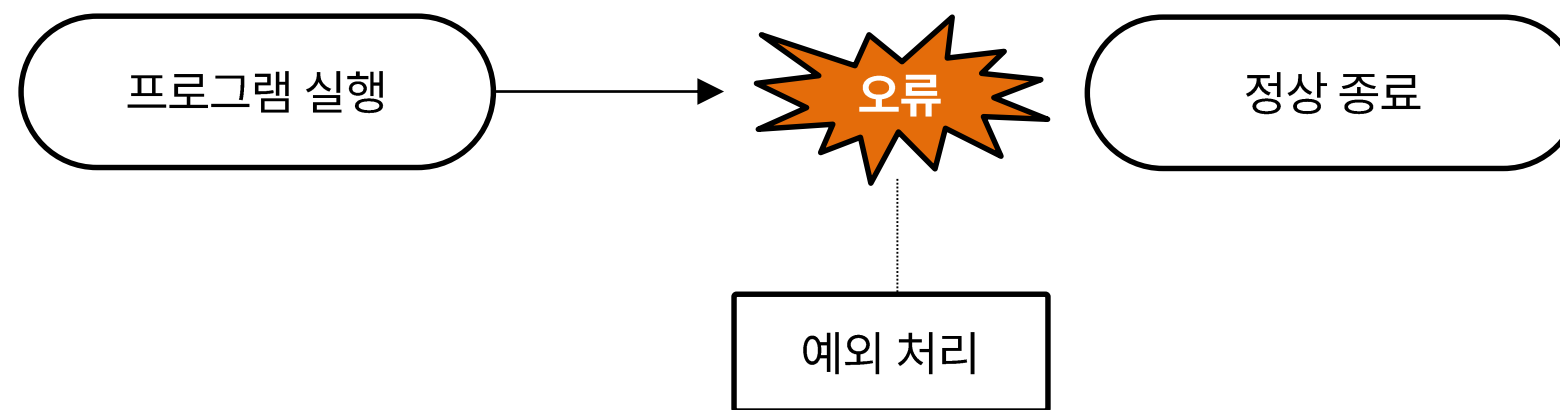


시스템 오류

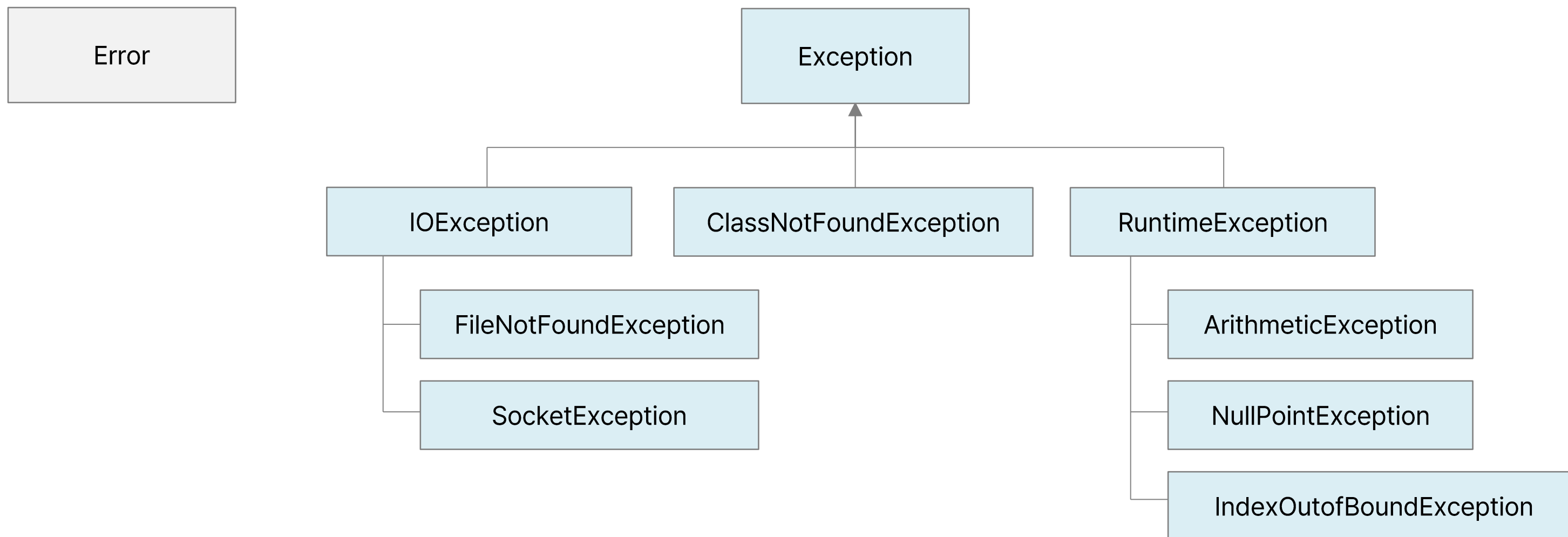
- 하드웨어나 운영체제와 관련된 문제로 발생함
- 파일 시스템 오류, 메모리 부족, 하드웨어 장애, 운영체제 장애
- 프로그램에서 처리할 수 없는 심각한 에러

예외

- 예상하지 못한 문제 또는 특정 조건을 만족하면 발생함.
- 프로그램에서 처리할 수 있는 간단한 에러
- 예외처리를 통해 프로그램이 비정상 종료되는 것을 막을 수 있음



예외를 처리하는 클래스는 다음과 같다.



예외 클래스	설명
FileNotFoundException	파일을 열거나 읽을 때, 파일이 없으면 발생
SocketException	네트워크 통신 중에 연결이 되지 않으면 발생
ClassNotFoundException	클래스를 찾지 못하면 발생
ArithmeticException	숫자를 0으로 나누면 발생
NullPointerException	빈 객체를 참조하면 발생
IndexOutOfBoundsException	배열의 인덱스 범위를 벗어나면 발생

예외 처리하기

예외처리하기

'try-catch문' 은 예외를 처리하는 기본 문법이다.

예외가 발생할 것을 미리 대비하여 프로그램이 비정상적으로 종료되는 것을 막는다.

try-catch문

```
try {  
    예외가 발생할 수 있는 코드  
} catch ( 처리할 예외 타입 e) {  
    try 블록 안에서 예외가 발생했을 때 처리할 코드  
}
```

try-catch문을 사용하여 예외 처리 하기

```
try {  
    arr[5] = 5;  
} catch (IndexOutOfBoundsException e) {  
    System.out.println(e);  
}
```

에러가 처리되는 과정

1. try 블록의 코드가 실행된다.
실행 중에 에러가 발생한다.
1. 해당 에러와 일치하는 catch 블록을 찾는다.
2. catch 블록의 코드를 실행한다.
3. 다음 문장을 계속해서 수행한다.

```
try {  
    arr[5] = 5;  
} catch (IndexOutOfBoundsException e) {  
    System.out.println(e);  
} catch (ArithmeticException e) {  
    System.out.println(e);  
}  
  
System.out.println("프로그램이 정상 종료됩니다.");
```

예외처리를 하지 않는다면
이 지점에서 비정상 종료됨

마지막 문장까지 실행하고
정상적으로 종료됨

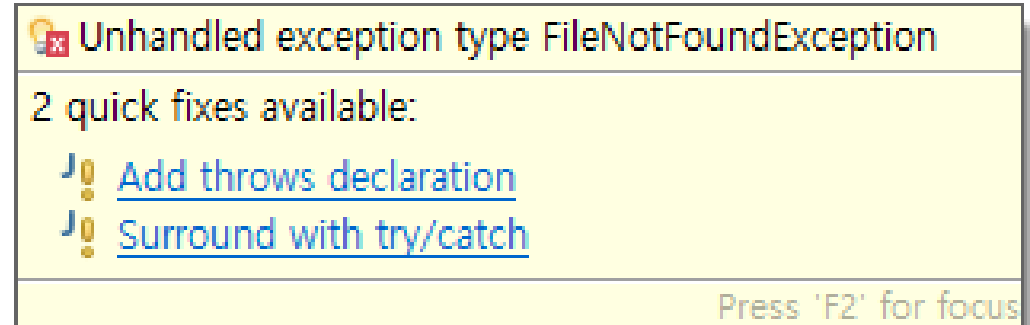
checked exception

- 컴파일러가 코드를 확인하고, 예외처리 코드가 없으면 컴파일 오류를 발생시킨다.
- 오류가 발생할 경우를 대비하여 미리 예외를 처리하도록 유도하는 것이다.
- 대표적인 예시로 IOException, SQLException 이 있다.

unchecked exception

- 예외처리를 하지 않아도 컴파일이 된다.
- 대표적인 예시로 NullPointerException, ArithmeticException 이 있다.

```
FileInputStream fis = new FileInputStream("a.txt");
```



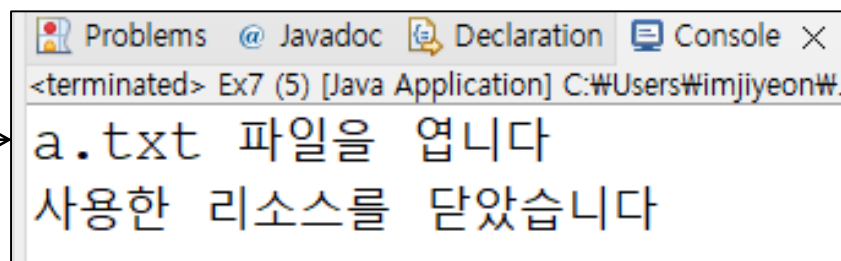
	Checked Exception	Unchecked Exception
처리 여부	반드시 예외처리를 해야함	예외처리를 안해도 됨
확인 시점	컴파일 단계	실행 단계
예외 종류	<ul style="list-style-type: none">IOExceptionSQLException	RuntimeException의 하위 예외 <ul style="list-style-type: none">ArithmeticExceptionNullPointerExceptionIndexOutOfBoundsException

finally 블록은 예외 발생 여부와 상관없이 항상 실행되는 블록이다.
이 블록은 주로 리소스 해제와 같은 마무리 작업을 수행하기 위해 사용한다.

텍스트 파일을 사용하고 닫기

```
try {  
    fis = new FileInputStream("src/main/a.txt");  
} catch (FileNotFoundException e) {  
    System.out.println(e.getMessage());  
} finally {  
    if(fis != null) {  
        try {  
            fis.close();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

결과



```
<terminated> Ex7 (5) [Java Application] C:\Users\Wimjiyeon#\n  
a.txt 파일을 엽니다  
사용한 리소스를 닫았습니다
```

예외를 처리하는 방법에는 2가지가 있다.

1. 예외가 발생하는 메소드에서 직접 예외를 처리하는 것이다.
2. 예외를 직접 처리를 하지 않고, 'throws' 키워드를 사용하여 예외처리를 넘기는 것이다.
그리고 메소드를 호출하는 쪽에서 예외 처리를 해야 한다.

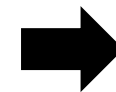
예외를 던짐

```
void openTextFile() throws FileNotFoundException {
    FileInputStream fis = new FileInputStream("src/main/a.txt");
}
```

예외를 처리함

```
public static void main(String[] args) {
    openTextFile();
}
```

Unhandled exception type FileNotFoundException
2 quick fixes available:
• Add throws declaration
• Surround with try/catch



```
void main(String[] args) {
    try {
        openTextFile();
    } catch (FileNotFoundException e) {
        System.out.println("파일이 존재하지 않습니다.");
    }
}
```