



10장. 예외처리

강사 김영석

A top-down view of a wooden desk. On the desk, there is a silver laptop with a black keyboard, a pair of black-rimmed glasses, a white coffee cup with a yellow handle, and a small green succulent in a pot. The word "CONTENT" is written in large, white, sans-serif capital letters over the left side of the image.

CONTENT

- 001 예외와 예외 클래스
- 002 실행 예외
- 003 예외 처리 코드
- 004 예외 종류에 따른 처리 코드
- 005 자동 리소스 닫기
- 006 예외 처리 떠넘기기
- 007 사용자 정의 예외와 예외 발생
- 008 예외 정보 얻기

001 예외와 예외 클래스

✓ 오류의 종류

■ 에러(Error)

- 하드웨어의 잘못된 동작 또는 고장으로 인한 오류
- 에러가 발생되면 프로그램 종료
- 정상 실행 상태로 돌아갈 수 없음

■ 예외(Exception)

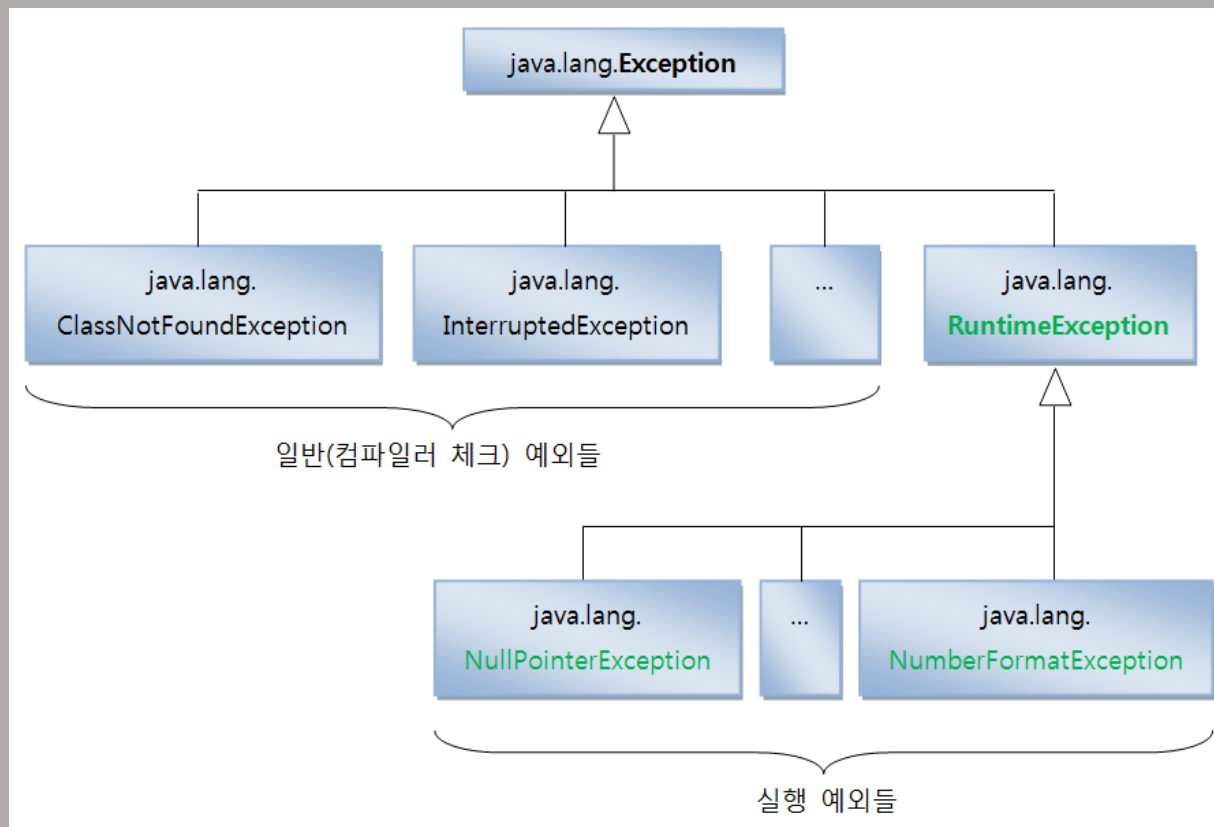
- 사용자의 잘못된 조작 또는 개발자의 잘못된 코딩으로 인한 오류
- 예외가 발생되면 프로그램 종료
- 예외 처리 추가하면 정상 실행 상태로 돌아갈 수 있음



✓ 예외의 종류

- 일반(컴파일 체크) 예외(Exception)
 - 예외 처리 코드 없으면 컴파일 오류 발생
- 실행 예외(RuntimeException)
 - 예외 처리 코드를 생략하더라도 컴파일은 되는 예외
 - 경험 따라 예외 처리 코드 작성 필요

✓ 예외 클래스



002 실행 예외(RuntimeException)

✓ NullPointerException

- 객체 참조가 없는 상태

➤ null 값 갖는 참조변수로 객체 접근 연산자인 도트(.) 사용했을 때 발생

```
String data = null;  
System.out.println(data.toString());
```

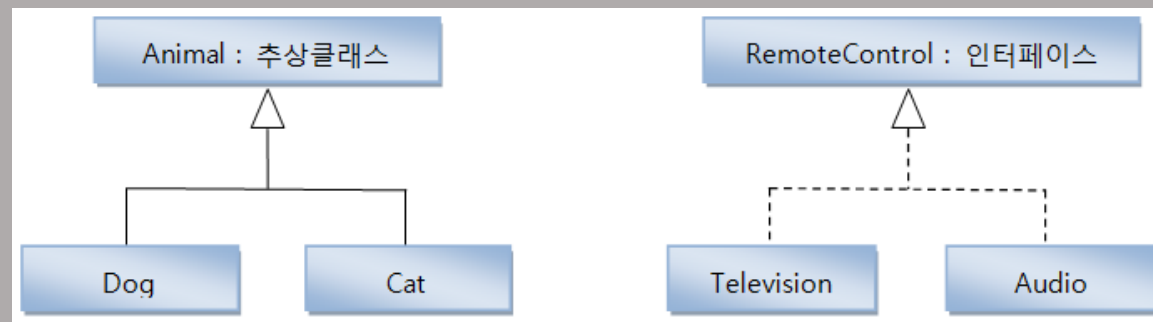
✓ ArrayIndexOutOfBoundsException

- 배열에서 인덱스 범위 초과하여 사용할 경우 발생

```
public static void main(String[] args) {  
    String data1 = args[0];  
    String data2 = args[1];  
  
    System.out.println("args[0]: " + data1);  
    System.out.println("args[1]: " + data2);  
}
```

✓ ClassCastException

- 타입 변환이 되지 않을 경우 발생



- 정상 코드

```
Animal animal = new Dog();
Dog dog = (Dog) animal;
```

```
RemoteControl rc = new Television();
Television tv = (Television) rc;
```

- 예외 발생 코드

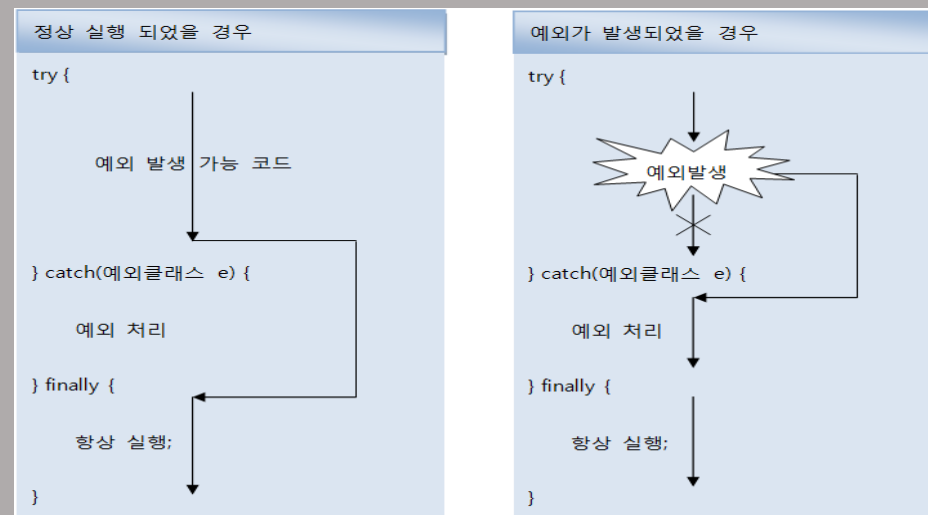
```
Animal animal = new Dog();
Cat cat = (Cat) animal;
```

```
RemoteControl rc = new Television();
Audio audio = (Audio) rc;
```

003 예외 처리 코드(try-catch-finally)

✓ 예외 처리 코드

- 예외 발생시 프로그램 종료 막고, 정상 실행 유지할 수 있도록 처리
 - 일반 예외: 반드시 작성해야 컴파일 가능
 - 실행 예외: 컴파일러가 체크해주지 않으며 개발자 경험 의해 작성
- try - catch - finally 블록 이용해 예외 처리 코드 작성

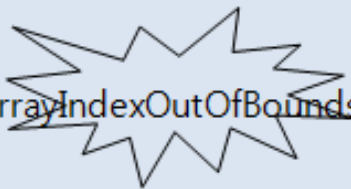


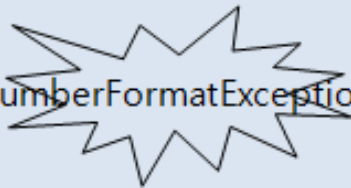
004 예외 종류에 따른 처리 코드

✓ 다중 catch

- 예외 별로 예외 처리 코드 다르게 구현

```
try {
```

 `ArrayIndexOutOfBoundsException` 발생

 `NumberFormatException` 발생

```
} catch(ArrayIndexOutOfBoundsException e) {
```

예외 처리 1

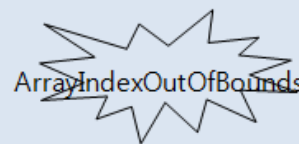
```
} catch(NumberFormatException e) {
```

예외 처리 2

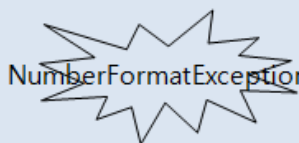
```
}
```

✓ catch 순서 – 상위 클래스가 위에 위치해야 한다

```
try {
```



ArrayIndexOutOfBoundsException 발생



NumberFormatException 발생

```
} catch(Exception e) {
```

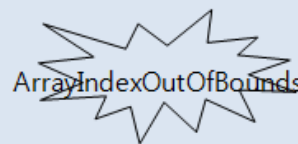
예외 처리 1

```
} catch(ArrayIndexOutOfBoundsException e) {
```

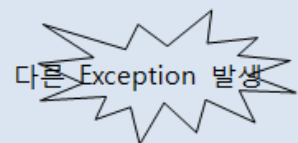
예외 처리 2

```
}
```

```
try {
```



ArrayIndexOutOfBoundsException 발생



다른 Exception 발생

```
} catch(ArrayIndexOutOfBoundsException e) {
```

예외 처리 1

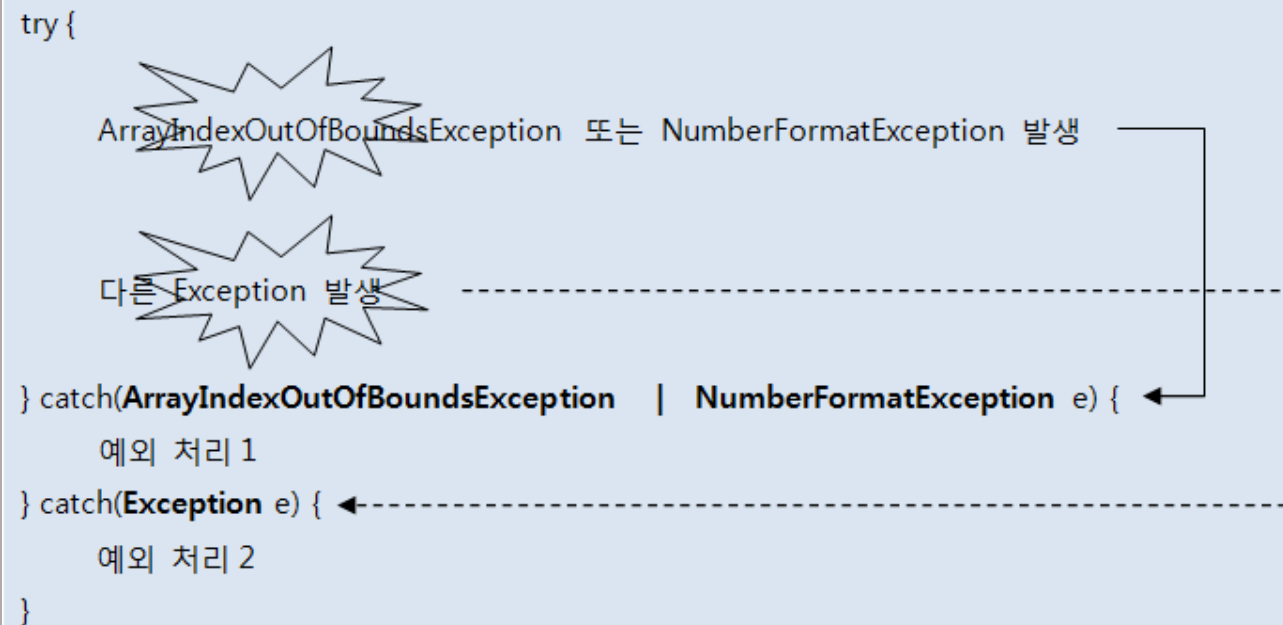
```
} catch(Exception e) {
```

예외 처리 2

```
}
```

✓ 멀티(multi) catch

- 자바 7부터는 하나의 catch 블록에서 여러 개의 예외 처리 가능
 - 동일하게 처리하고 싶은 예외를 |로 연결



005 자동 리소스 닫기

✓ try-with-resources

- 예외 발생 여부와 상관 없음
- 사용했던 리소스 객체의 `close()` 메소드 호출해 리소스 닫음
- 리소스 객체
 - 각종 입출력스트림, 서버소켓, 소켓, 각종 채널
 - `java.lang.AutoCloseable` 인터페이스 구현하고 있어야 함

006 예외 떠 넘기기

✓ throws

- 메소드 선언부 끝에 작성

```
리턴타입 메소드명(매개변수,...) throws 예외클래스 1, 예외클래스 2, ... {  
}
```

- 메소드에서 처리하지 않은 예외를 호출한 곳으로 떠 넘기는 역할

```
public void method1() {  
    try {  
        method2();  
    } catch(ClassNotFoundException e) {  
        //예외 처리 코드  
        System.out.println("클래스가 존재하지 않습니다.");  
    }  
}  
  
public void method2() throws ClassNotFoundException {  
    Class clazz = Class.forName("java.lang.String2");  
}
```

호출한 곳에서 예외 처리

007 사용자 정의 예외와 예외 발생

✓ 사용자 정의 예외 클래스 선언

- 자바 표준 API에서 제공하지 않는 예외
- 애플리케이션 서비스와 관련된 예외
 - Ex) 잔고 부족 예외, 계좌 이체 실패 예외, 회원 가입 실패 예외....
- 사용자 정의 예외 클래스 선언 방법

```
public class XXXException extends [ Exception | RuntimeException ] {  
    public XXXException() { }  
    public XXXException(String message) { super(message); }  
}
```

✓ 예외 발생 시키기

- 코드에서 예외 발생시키는 법

```
throw new XXXException()  
throw new XXXException("메시지");
```

- 호출된 곳에서 발생한 예외를 처리하도록

```
public void method() throws XXXException {  
    throw new XXXException("메시지");  
}
```

008 예외 정보 얻기

✓ getMessage()

- 예외 발생시킬 때 생성자 매개값으로 사용한 메시지 리턴

```
throw new XXXException("예외 메시지");
```

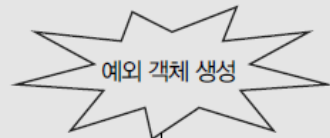
- 원인 세분화하기 위해 예외 코드 포함(예: 데이터베이스 예외 코드)
- catch() 절에서 활용

```
} catch(Exception e) {  
    String message = e.getMessage();  
}
```

✓ printStackTrace()

- 예외 발생 코드 추적한 내용을 모두 콘솔에 출력
- 프로그램 테스트하면서 오류 찾을 때 유용하게 활용

```
try {
```



```
} catch(예외클래스 e) {  
    //예외가 가지고 있는 Message 얻기  
    String message = e.getMessage();  
  
    //예외의 발생 경로를 추적  
    e.printStackTrace();  
}
```

A photograph of a server room with rows of server racks on both sides of a central aisle. The racks have glass doors and are filled with server units, some of which have glowing blue indicator lights. The ceiling has several long, rectangular light fixtures. The overall atmosphere is dimly lit, with the primary light sources being the overhead lights and the server unit indicators.

수고하셨습니다.