

## 10-1. 예외 클래스

혼자 공부하는 자바(개정판) (신용권 저)

- 시작하기 전에
- 예외와 예외 클래스
- 실행 예외
- 키워드로 끝내는 핵심 포인트
- 확인문제



## 시작하기 전에

[핵심 키워드] : 예외, 예외 클래스, 일반 예외, 실행 예외

### [핵심 포인트]

자바에서 컴퓨터 하드웨어 관련 고장으로 인해 응용프로그램 실행 오류가 발생하는 것을 에러라 하고, 그 외 프로그램 자체에서 발생하는 오류를 예외라고 한다. 예외의 종류와 발생 경우를 알아본다.



# 시작하기 전에

## ❖ 예외 (Exception)

- 사용자의 잘못된 조작 또는 개발자의 잘못된 코딩으로 인해 발생하는 프로그램 오류
- 예외 처리 프로그램 통해 정상 실행상태 유지 가능
- 예외 발생 가능성이 높은 코드 컴파일할 때 예외 처리 유무 확인



# 예외와 예외 클래스

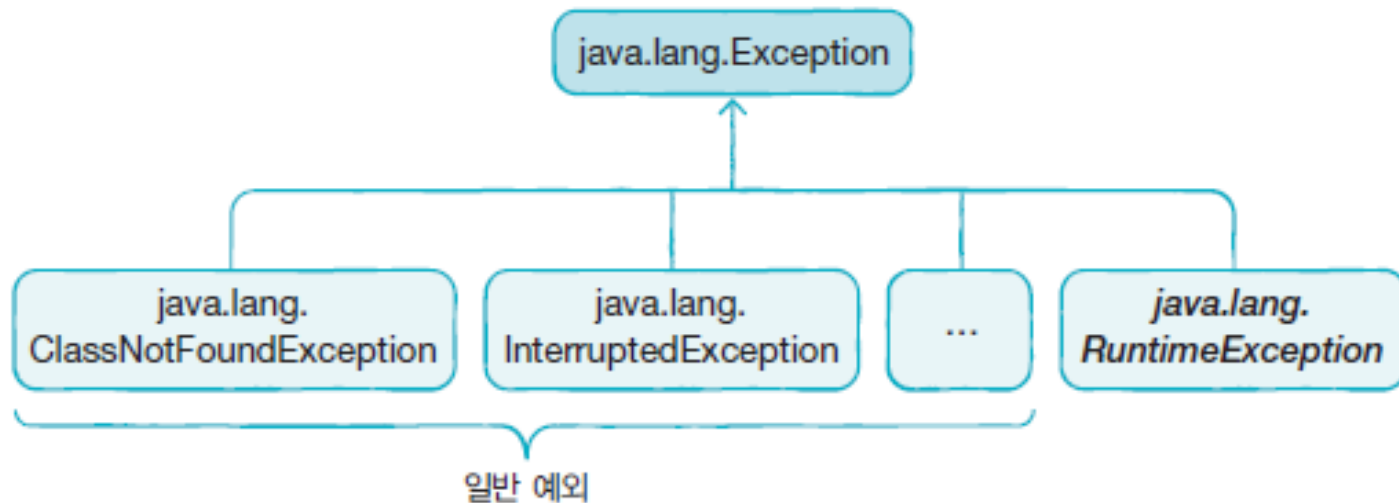
## ❖ 일반 예외 (exception)

- 컴파일러 체크 예외
- 자바 소스 컴파일 과정에서 해당 예외 처리 코드 있는지 검사하게 됨

## ❖ 실행 예외 (runtime exception)

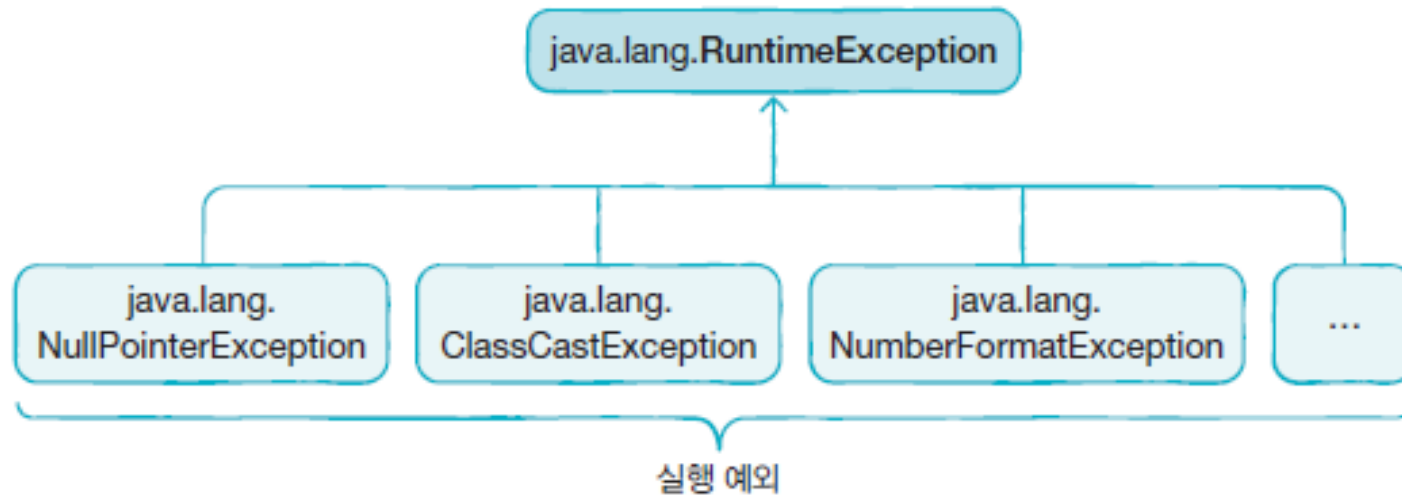
- 컴파일러 런 체크 예외
- 실행 시 예측할 수 없이 갑자기 발생하기에 컴파일 과정에서 예외처리코드 검사하지 않음

## ❖ 자바에서는 예외를 클래스로 관리



# 예외와 예외 클래스

- RuntimeException 클래스 기준으로 일반 및 실행 예외 클래스 구분



# 실행 예외

## ❖ 개발자의 경험에 의해서 예외 처리 코드 작성해야 함

- 예외처리코드 없을 경우 해당 예외 발생 시 프로그램 종료

## ❖ NullPointerException

- 가장 빈번하게 발생하는 실행 예외
- `java.lang.NullPointerException`
- 객체 참조가 없는 상태의 참조 변수로 객체 접근 연산자 도트를 사용할 경우 발생

```
01 package sec01.exam01;
02
03 public class NullPointerExceptionExample {
04     public static void main(String[] args) {
05         String data = null;
06         System.out.println(data.toString());
07     }
08 }
```

### 실행결과

```
Exception in thread "main" java.lang.NullPointerException
    at NullPointerExceptionExample.main(NullPointerExceptionExample.java:6)
```



## ❖ ArrayIndexOutOfBoundsException

- 배열에서 인덱스 범위를 초과할 경우
- `java.lang.ArrayIndexOutOfBoundsException`

```
01 package sec01.exam02;  
02  
03 public class ArrayIndexOutOfBoundsExceptionExample {  
04     public static void main(String[] args) {  
05         String data1 = args[0];  
06         String data2 = args[1];  
07  
08         System.out.println("args[0]: " + data1);  
09         System.out.println("args[1]: " + data2);  
10     }  
11 }
```

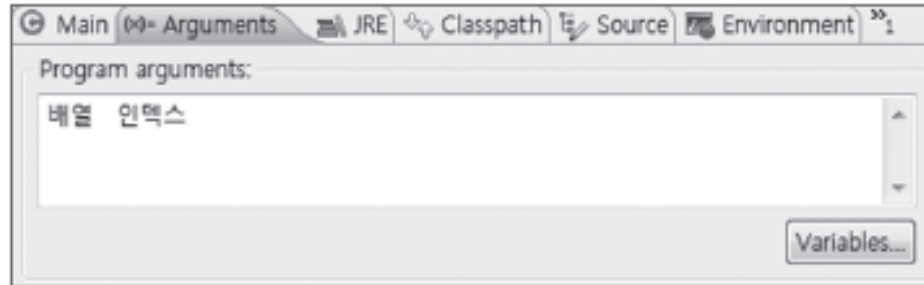
실행결과

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 0 out of bounds  
for length 0  
    at ArrayIndexOutOfBoundsExceptionExample.main(ArrayIndexOutOfBoundsExceptionExample.  
    java:5)
```



# 실행 예외

- 이클립스 - [Run] - [Run Configuration] - [Arguments]탭 - [Program arguments]  
아래와 같이 입력하여 해결



```
01 package sec01.exam03;
02
03 public class ArrayIndexOutOfBoundsExceptionExample {
04     public static void main(String[] args) {
05         if(args.length == 2) {
06             String data1 = args[0];
07             String data2 = args[1];
08             System.out.println("args[0]: " + data1);
09             System.out.println("args[1]: " + data2);
10         } else {
11             System.out.println("두 개의 실행 매개값이 필요합니다.");
12         }
13     }
14 }
15 }
16 }
```



## ❖ NumberFormatException

- 문자열을 숫자로 변환하는 경우

리턴 타입	메소드 이름(매개 변수)	설명
int	<code>Integer.parseInt(String s)</code>	주어진 문자열을 정수로 변환해서 리턴
double	<code>Double.parseDouble(String s)</code>	주어진 문자열을 실수로 변환해서 리턴

- 숫자가 변환될 수 없는 문자가 포함된 경우 `java.lang.NumberFormatException` 발생



## 실행 예외

```
01 package sec01.exam04;
02
03 public class NumberFormatExceptionExample {
04     public static void main(String[] args) {
05         String data1 = "100";
06         String data2 = "a100";
07
08         int value1 = Integer.parseInt(data1);
09         int value2 = Integer.parseInt(data2);    //NumberFormatException 발생
10
11         int result = value1 + value2;
12         System.out.println(data1 + "+" + data2 + "=" + result);
13     }
14 }
```

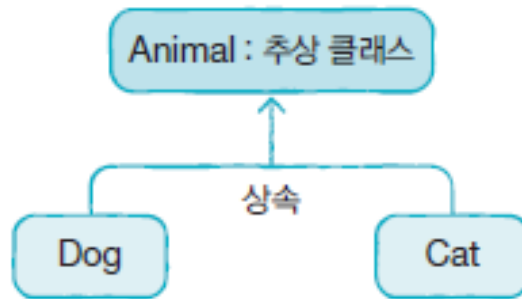
### 실행결과

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "a100"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    at java.base/java.lang.Integer.parseInt(Integer.java:652)
    at java.base/java.lang.Integer.parseInt(Integer.java:770)
    at NumberFormatExceptionExample.main(NumberFormatExceptionExample.java:9)
```



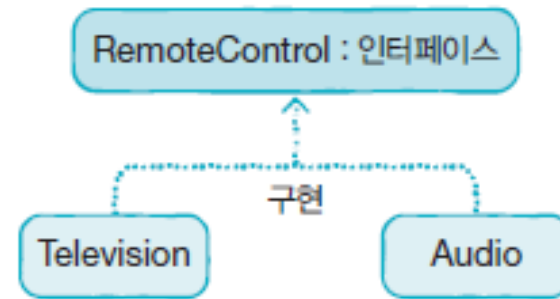
## ❖ ClassCastException

- 상위 및 하위 클래스 그리고 구현 클래스와 인터페이스 간 타입 변환 가능
- 위 관계가 아닌 경우 ClassCastException 발생



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

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



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

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

## 실행 예외

- instanceof 연산자로 타입 변환 가능 여부를 미리 확인

```
Animal animal = new Dog() ;  
if(animal instanceof Dog) {  
    Dog dog = (Dog) animal;  
} else if(animal instanceof Cat) {  
    Cat cat = (Cat) animal;  
}
```

```
Remocon rc = new Audio();  
if(rc instanceof Television) {  
    Television tv = (Television) rc;  
} else if(rc instanceof Audio) {  
    Audio audio = (Audio) rc;  
}
```



## ❖ 예시 – ClassCastException

```
01 package sec01.exam05;
02
03 public class ClassCastExceptionExample {
04     public static void main(String[] args) {
05         Dog dog = new Dog();
06         changeDog(dog);
07
08         Cat cat = new Cat();
09         changeDog(cat);
10     }
11
12     public static void changeDog(Animal animal) {
13         //if(animal instanceof Dog) {
14             Dog dog = (Dog) animal;    //ClassCastException 발생 가능
15         //}
16     }
17 }
18
19 class Animal {}
20 class Dog extends Animal {}
21 class Cat extends Animal {}
```



## 실행 예외

실행결과

```
Exception in thread "main" java.lang.ClassCastException: class Cat cannot be cast to class
Dog (Cat and Dog are in unnamed module of loader 'app')
    at ClassCastExceptionExample.changeDog(ClassCastExceptionExample.java:14)
    at ClassCastExceptionExample.main(ClassCastExceptionExample.java:9)
```



# 키워드로 끝내는 핵심 포인트

- **예외** : 사용자의 잘못된 조작 또는 개발자의 잘못된 코딩으로 인해 발생하는 프로그램 오류. 예외 발생 시 프로그램이 곧바로 종료되나, 예외 처리 통해 정상 실행상태를 유지할 수 있음
- **예외 클래스** : 자바에서는 예외를 클래스로 관리함. 프로그램 실행 중 예외가 발생하면 해당 예외 클래스로 객체를 생성하고 예외 처리 코드에서 예외 객체를 이용할 수 있도록 해줌.
- **일반 예외** : 컴파일러 체크 예외. 프로그램 실행 시 예외 발생 가능성 높기 때문에 자바 소스 컴파일 과정에서 해당 예외 처리 코드 있는지 검사함.
- **실행 예외** : 컴파일러 런 체크 예외. 실행 시 예측할 수 없이 갑자기 발생하기 때문에 컴파일 과정에서 예외 처리 코드 존재 여부를 검사하지 않음





## ❖ 예외에 대한 아래 설명 중 틀린 것을 고르세요

- 예외는 사용자의 잘못된 조작, 개발자의 잘못된 코딩으로 인한 프로그램 오류를 말한다.
- RuntimeException의 하위 클래스는 컴파일러가 예외 처리 코드를 체크하지 않는다.
- 예외는 클래스로 관리된다
- Exception의 하위 클래스는 모두 일반 예외에 해당한다.





Thank You!