KH정보교육원

예외처리(Exception)

예외처리

KH정보교육원

자바 오류 정의

오류(Error)	1. 프로그램 수행시 치명적 상황이 발생한 것 2. 소스상 해결이 불가능한 것을 에러라고 하며, 프로그램이 비정상적으로 종료됨
예외(Exception)	프로그램 오류 중 코드 수정을 통해 수습될 수 있는 오류

예외처리 - 오류의 종류

KH정보교육원

컴파일 에러

소스 상의 문법 Error

논리 에러

문법상 문제가 없고, 런타임에러도 발생하지 않지만 개발자 의도대로 작동하지 않음

런타임 에러

입력 값이 틀리거나 계산식의 오류로 발생

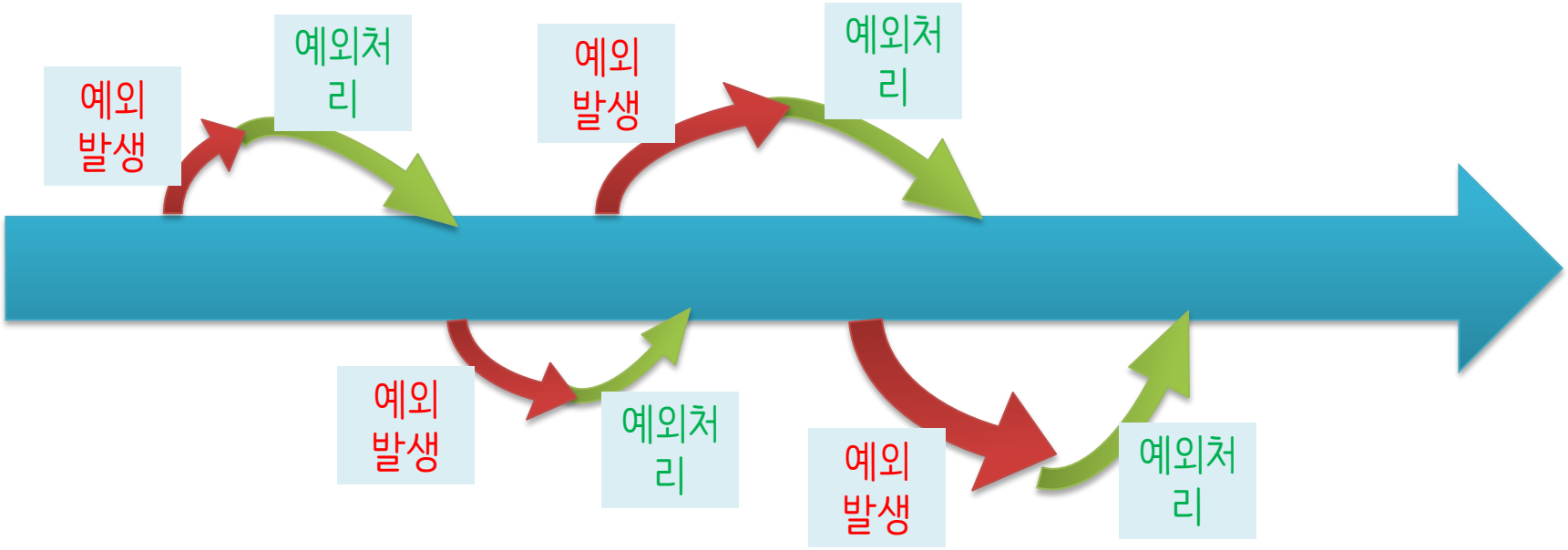
시스템 에러

컴퓨터 오작동으로 인한 에러

예외처리(try~catch)의 목적

KH정보교육원

이걸 대체 왜 하느냐?



목적 : 프로그램의 비정상 종료를 막고, 정상적인 실행상태를 유지함.

어떻게: 예외상황이 발생된 경우에 처리로직을 만듬.

예외처리

KH정보교육원

try~catch 표현식

```
try{
    //반드시 예외 처리를 해야 하는 구문 작성함
}catch(처리해야할예외클래스명 참조형 변수명){
    //잡은 예외 클래스에 대한 처리 구문 작성함
}finally{
    //실행 도중 해당 Exception이 발생을 하던,
    //안하던 반드시 실행해야 하는 구문 작성함
}
```

예외처리

KH정보교육원

예외처리 예제

```
public static void main ( String [] agrs) {
	Scanner sc = new Scanner(System.in);

	System.out.println("-------- 나눗셈 프로그램 -------");
	System.out.print("첫번째 수 입력 : ");
	int data1 = sc.nextInt();
	System.out.print("두번째 수 입력 : ");
	int data2 = sc.nextInt();

	int result = data1 / data2;
	System.out.println("결과 : " + result);

	System.out.println("감사합니다.");
}
```

```
public static void main ( String [] agrs) {
	Scanner sc = new Scanner(System.in);

	System.out.println("-------- 나눗셈 프로그램 -------");
	System.out.print("첫번째 수 입력 : ");
	int data1 = sc.nextInt();
	System.out.print("두번째 수 입력 : ");
	int data2 = sc.nextInt();
	try {
		int result = data1 / data2;
		System.out.println("결과 : " + result);
	}catch(Exception e) {
		System.out.println("0으로는 ..." + e.getMessage());
	}
}
```

예외처리

KH정보교육원

finally 예외처리 실습

```
public static void main(String [] agrs) {
    int i = 10;
    int j = 0;
    try{                // j가 0일 때 ArithmeticException이 발생
        int k = i / j;
        System.out.println(k);
    }catch(ArithmeticException e){    // catch에서 예외처리
        System.out.println("0으로 나눌 수 없습니다. : " + e.toString());
    }finally {          // finally블록은 생략 가능합니다.
        System.out.println("오류가 발생하든 안하든 무조건 실행되는 블록입니다.");
    }
}
```

예외처리

KH정보교육원

finally 사용 이유

우리가 흔히 App 과 DB를 연결해서 사용하게 되는데
정상적으로 DB 서버에 연결하여 사용하고 있었다면 마지막에는 DB서버와의
연결을 끊어주어야 한다.
(지속적인 연결은 리소스를 계속 잡고 있기 때문에
연결 제한이 발생할 수 있음)
그런데 여기서 정상적인 처리후에 연결 해제만 코딩 해놓게 된다면
비정상적인 처리(예외) 상황에서는 DB 연결이 해제
되지 않는 상황이 발생하게 되어
큰 문제가 발생할 수 있게 된다.
그렇기 때문에 finally 를 통해서 try 경우든 catch 경우에든
DB 연결 해제 처리를 하게 되어 DB에 영향을 주지 않도록 설정하는 것이다.

예외의 종류

KH정보교육원

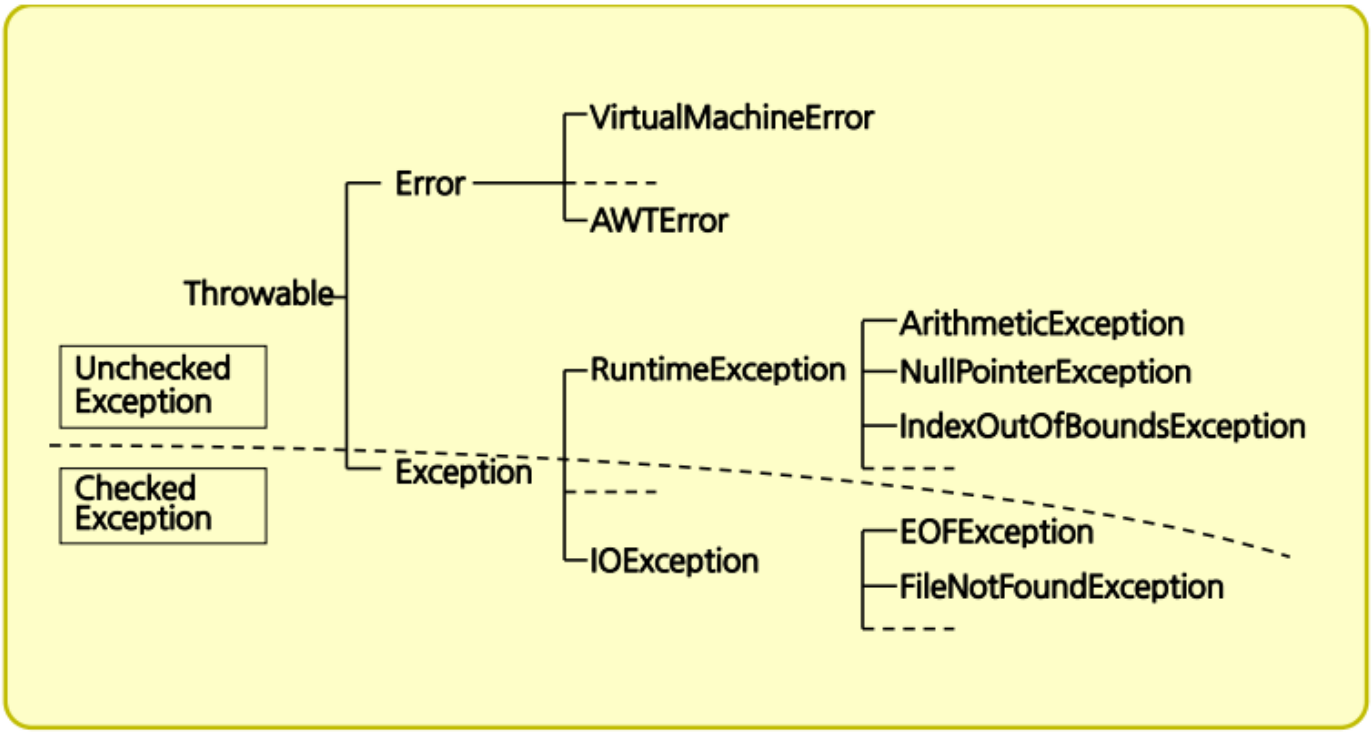
Exception 클래스 상속도

소스코드 상에서 반드시 예외처리해야 되는 Checked Exception

소스코드 상에서 명시적인 처리를 강제하지 않는 Unchecked Exception

예외의 종류

KH정보교육원



RuntimeException 후손클래스

KH정보교육원

예외 타입(예외 클래스)	예외 발생 경우	패키지
ArithmeticException	정수를 0으로 나눌 때 발생	java.lang
NullPointerException	null 레퍼런스를 참조할 때 발생	java.lang
ClassCastException	변환할 수 없는 타입으로 객체를 변환할 때 발생	java.lang
OutOfMemoryError	메모리가 부족한 경우 발생	java.lang
ArrayIndexOutOfBoundsException	배열의 범위를 벗어난 접근 시 발생	java.lang
IllegalArgumentException	잘못된 인자 전달 시 발생	java.lang
IOException	입출력 동작 실패 또는 인터럽트 시 발생	java.io
NumberFormatException	문자열이 나타내는 숫자와 일치하지 않는 타입의 숫자로 변환 시 발생	java.lang
InputMismatchException	Scanner 클래스의 nextInt()를 호출하여 정수로 입력받고자 하였지만, 사용자가 'a' 등과 같이 문자를 입력한 경우	java.util

예외처리

KH정보교육원

멀티 catch 표현식

```
try{
    //반드시 예외 처리를 해야 하는 구문 작성함
}catch(예외클래스명3 e){

} catch(예외클래스명2 e){

} catch(예외클래스명1 e){

}
```

이때,catch절의 순서는 상속관계를 따라 작성해야 한다. 후손클라스가 부모클라스보다 먼저 기술되어야 한다.

FileNotFoundException → IOException → Exception

예외처리

KH정보교육원

try~with~resource 표현식

```
try(반드시 close 처리 해야 하는 객체에 대한 생성 구문){
    //예외 처리를 해야 하는 구문 작성함
}catch(처리해야할예외클래스명 레퍼런스){
    //잡은 예외 클래스에 대한 처리 구문 작성함
}
```

자바 7에서 추가된 기능으로, finally에서 작성되었던 close()처리를 생략하고 자동으로 close처리 되게 하는 문장이다.

Exception 처리 방법

KH정보교육원

1. Exception 처리를 호출한 메소드에게 위임

-메소드 선언시 throws ExceptionName 문을 추가하여 호출한 상위 메소드에게 처리를 위임하여 해결한다.
-계속적으로 위임하면 main()까지 위임하게 되고, main()까지 가서도 예외처리가 되지 않는 경우 JVM이 비정상 종료된다.

Exception 처리 방법

KH정보교육원

2. Exception을 발생한 곳에서 직접 처리

- try~catch문을 이용하여 예외를 처리한다.
- try : exception 발생할 가능성이 있는 코드를 try구문 안에 기술한다.
- catch : try 구문에서 exception 발생시 해당하는 exception에 대한 처리를 기술한다.
  여러 개의 exception처리가 가능하나, exeption간의 상속관계를 고려해야 한다.
- finally : exception 발생 여부에 관계 없이, 꼭 처리해야 하는 logic은 finally 안에서 구현한다.
  중간에 return문을 만나도 finally구문은 실행한다.
  단, System.exit(0)를 만나면 무조건 프로그램을 종료한다.
  주로 java.io, java.sql 패키지의 메소드 처리시 이용한다.

예외처리

KH정보교육원

throws로 예외 던지기

```
public static void main(String[] args){
        ThrowTest t = new ThrowTest();
        try{
            t.methodA();
            System.out.println("정상수행");
        }catch(IOException e){
            System.out.println("IOException이 발생");
        }finally{
            System.out.prinln("프로그램 종료");
        }
}
```

```
public void methodA() throws IOException{
        methodB();
}
```

```
public void methodB() throws IOException{
        methodC();
}
```

```
public void methodC() throws IOException{
        throw new IOException();
        //Exception 발생시키는 구문
}
```

원칙적으로, 메소드를 호출한 곳에서 예외처리를 해야하므로, 메소드를 최초 호출한 main메소드에서 처리하고 있다.

사용자 정의 예외생성

KH정보교육원

Exception 클래스를 상속받아 예외 클래스 작성함.

Exception 발생하는 곳에서 throw new MyException()으로 발생

```
public class MyException extends Exception{
    public MyException(){}
    public MyException(String msg){
        super(msg);
    }
}
```

```
public class MyExceptionTest{
    public void checkAge(int age) throws MyException
{
        if(age < 19)
            throw new MyException("입장불가");
        else
            System.out.println("즐감"); }
}
```

```
public class Run{
    public static void main(String[] args){
        MyExceptionTest m
                        = new MyExceptionTest();
        try {
            m.checkAge(15);
        }catch(MyException e){
            System.out.println(e.getMessage());
        }
    }
}
```