

# JAVA Programming

## *Exception*

BeomSeok Kim

Department of Computer Engineering  
KyungHee University  
passion0822@khu.ac.kr

# Contents



- Examples of exception
- Handling technique for exception
- Example
- Exception handling
- Category of exception and classes

# Examples of exception



- 연산처리 과정
  - ✓ 정수를 0으로 나눌 때, 불능으로 인한 오류 발생
  - ✓ 할당된 범위 밖의 **배열요소**를 처리하고자 할 때
- 파일처리 과정
  - ✓ 사용하고자 하는 파일을 찾지 못하는 경우
  - ✓ 읽고 있는 파일의 끝(EOF)을 만났을 때
- 통신처리 과정
  - ✓ 통신을 통해 연결할 때 주소가 잘 못 지정된 경우
  - ✓ 통신 중 상대 측에서 일방적으로 연결을 끊는 경우
- 기타 특별한 상황의 발생에 대한 통보
  - ✓ 특정한 조건의 도래에 따른 인터럽트의 발생

# Handling technique for exception



- 예외의 정의
  - ✓ 프로그램에서 발생하는 예기치 않은 상황
  - ✓ 프로그램에서 일부러 도래하길 기다리는 상황
  
- 예외 발생시 조치방법
  - ✓ 예외(오류)의 원인이 된 프로그램이나 데이터의 보완
  - ✓ 예외의 발생에 대한 통보 및 조치 : 프로그래밍으로 처리

# Example

## ■ 프로그램 – 연산오류 발생

```
class Exception1 {  
    public static void main(String args[]) {  
        int i = 5; int j = 0;  
        int k = i / j;          // 불가능 : 예외상황 발생 -> 프로그램 수정 필요  
        System.out.println("k = " + k);  
    }  
}
```

## ■ 오류 메시지

Java.lang.ArithmeticException : / by zero  
At Exception1.main<Exception1.java.4>

- i의 값인 5를 j의 값인 0으로 나눌 때 오류 발생
- 예외 ArithmeticException이 발생

# Example

## ■ 프로그램 – 연산오류 발생

```
class Exception1 {  
    public static void main(String args[]) {  
        int i = 5; int j = 0;  
        int k = i / j;          // 불가능 : 예외상황 발생 -> 프로그램 수정 필요  
        System.out.println("k = " + k);  
    }  
}
```

## ■ 오류 메시지

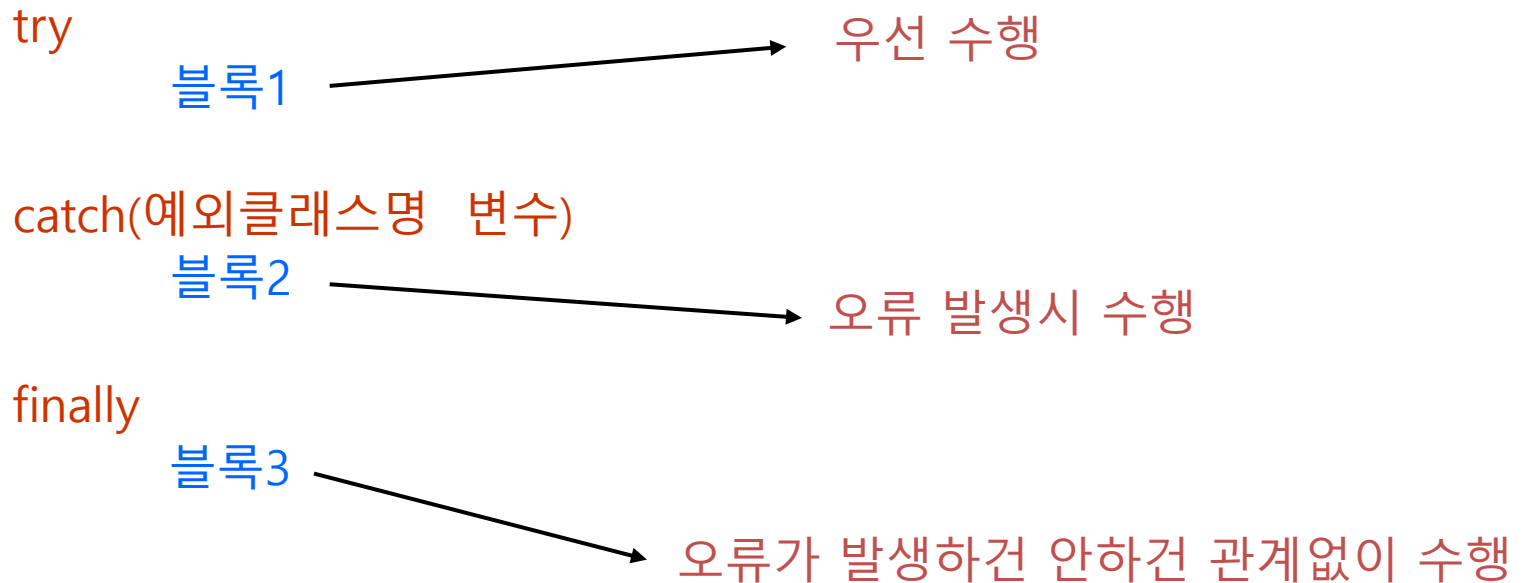
Java.lang.ArrayIndexOutOfBoundsException : 3  
At Exception2.main<ExceptionTest2.java.7>

- 생성된 배열의 크기를 초과한 배열요소의 인덱스 시도
- 예외 `ArrayIndexOutOfBoundsException`이 발생

# Exception handling

- 기본적인 예외 처리 구문
  - ✓ 'try-catch-finally' 구문 이용
  - ✓ 예외가 발생한 메소드 내에서 처리하는 방식

- 기본 문형



# Exception handling



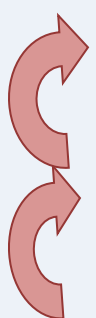
## ■ 프로그램 – 예외발생에 대한 처리

```
class Exception3 {  
    public void main(String args[]) {  
        int i = 0;  
        String array[] = {"one", "two", "three"}; // 3개 배열  
        while(i < 4) {    // 4개 배열(0 ~ 3)까지 인쇄  
            try {  
                System.out.println(array[i]); // 4번째 배열 : 오류  
            } catch(ArrayIndexOutOfBoundsException e) { // 예외 선언  
                System.out.println("배열 범위 초과"); // 배열오류시 수행  
                break;    // 정상 종료  
            } finally {  
                i++;    // try 문을 실행할 때마다 수행됨  
                System.out.println("배열요소 증가 : i = " + i);  
            }  
        }  
    } ...  
}
```



# Exception handling

- 예외의 처리가 정의되지 않으면
  - ✓ 예외가 발생한 메소드를 **호출한 메소드**로 전파됨
- 프로그램 – 예외의 전파



```
class Exception4
{
    public static void main(String args[]) {
        method1();
        System.out.println("End of program"); //비정상종료로 출력되지 않음
    }
    static void method1() {
        method2(); // (2) 오류 전파 : main()으로 전파
    }
    static void method2(){
        int i = 5; int j = 0;
        int k = i / j; // (1) 오류발생 : method1()으로 전파
        System.out.println("k = " + k);
    }
}
```

- 오류 메시지

```
Java.lang.ArithmeticException : / by zero
At Exception4.method2 Exception4.java.11>
At Exception4.method1 Exception4.java.7>
At Exception4.main Exception4.java.4>
```

# Exception handling



## ■ 예외의 전파순서(default)

- ✓ 예외 발생
  - 해당 메소드 내에서 처리
- ✓ 해당 메소드에 처리가 정의되어 있지 않으면
  - 호출 메소드로 이전하여 처리
  - 그러나 try-catch 문이 있으면 정의된 대로 처리
- ✓ 호출함수에 정의되어 있지 않으면, 그 앞의 호출 메소드로 전파
  - 최종 main() 메소드까지 전파
  - 비정상적인 종료


## ■ throws 문

- ✓ 예외 발생을 예측하고, 이 경우 호출 메소드로 이전함을 명시
- ✓ 적절한 이전 호출 메소드에서 try-catch를 이용한 처리방법 정의

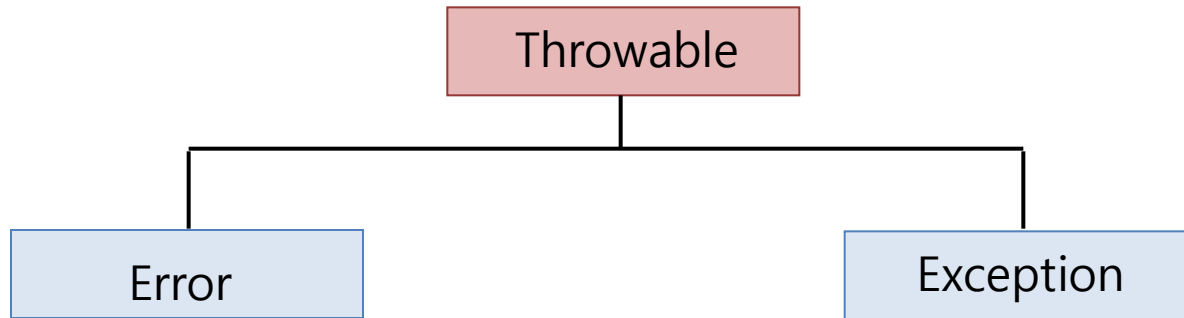
# Exception handling

## ■ 프로그램

```
class Exception5 {  
    public static void main(String args[]) {  
        method1();  
        System.out.println("End of program"); //정상종료로 정상출력  
    }  
    static void method1() {    // static인 main()에서 호출되므로 static 메소드가 됨  
        try {  
            method2();    // method2() 수행시의 연산 오류는 여기서 처리  
        } catch (ArithmeticException e) {    // 오류내역은 e 객체에서 관리됨  
            e.printStackTrace(); // e에 들어 있는 오류의 내용을 출력함  
        }  
    }  
    static void method2() throws ArithmeticException{  
        // throws : 호출 메소드로 예외를 전파  
        int i = 5; int j = 0;  
        int k = i / j; // 오류발생  
        System.out.println("k = " + k);  
    }  
}
```



# Category of exception and classes



- 모든 예외는 Throwable 클래스로부터 상속
  - ✓ Error : 복구가 불가능한 치명적인 오류
  - ✓ Exception : 프로그램의 수정이나 처리가 요구되는 오류나 상황

# Category of exception and classes



클래스	하위 클래스	발생 상황
RuntimeException	ArithmeticException	불능 등에 의한 산술적 오류
	NullPointerException	주소가 null 인 객체의 메소드나 변수에 접근
	IndexOutOfBoundsException	배열이나 문자열의 범위 초과
ClassNotFoundException		특정한 클래스를 찾지 못할 때 발생
InterruptedException		인터럽트의 발생(notify())의 호출 등의 이유로 발생)
IOException	FileNotFoundException	접근하려는 파일을 발견 못함
	EOFException	파일의 끝을 만남

# Category of exception and classes



- 응용 프로그램에 의한 예외의 발생
  - ✓ 시스템에서는 예외의 상황을 만나면 미리 정의된 예외를 발생시킴
  - ✓ 응용 프로그램에서도 필요한 경우에 예외를 발생시킬 수 있음
- 예외의 발생은 throw라는 키워드를 사용
  - ✓ 예외클래스는 Exception 클래스를 상속함
  - ✓ throw는 프로그램에서 정의하는 예외클래스로 예외를 발생시킴

```
class Grade {
    int grade;
    void setGrade(int score)
    {
        if (score > 100 || score < 0)           // 점수 범위 오류
            throw new OutOfBounds(score);      // 예외의 생성 및 발생
        else                                     // 정상 값
            grade = score / 10;
    }
}
class OutOfBounds extends Exception           // 예외 클래스의 정의
{
    OutOfBounds(int score) {
        super("OutOfBoundsException : " + score); // super = Exception
    }
}
```

**Thank You!**  
**Q&A**