

{ JAVA }

선린인터넷
웹 운영 과

예외 처리

Exception Handling

❖ 예외 Exception ...?

■ 프로그램 실행 중에 발생하는 오류

≠ 문법적인 에러 Syntax error

■ 사례

- 정수를 0으로 나누는 경우
- 배열의 첨자(index)가 범위를 벗어났을 경우
- 부적절한 형변환이 발생한 경우
- 입출력 시 인터럽트가 발생한 경우
- 지정한 파일이 존재하지 않은 경우.... 등 다양하다.

} 잘못된 사용자 조작 ,
개발자 코딩 실수

Exception Handling

자바의 예외 처리

호출할 **메소드**가 위험하다는 것...에 기반하여
(즉, 그 메소드가 예외를 발생시킬 수 있다는 것)

미리 “예외적인 상황”을 처리할 수 있도록 하여 대비한다.

호출하려는 메소드에서 예외를 발생시킬 수 있다는 것은 어떻게 알 수 있을까?

FileReader

```
public FileReader(String fileName)  
    throws FileNotFoundException
```

Creates a new `FileReader`, given the name of the file to read from.

Parameters:

`fileName` - the name of the file to read from

Throws:

`FileNotFoundException` - if the named file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading.

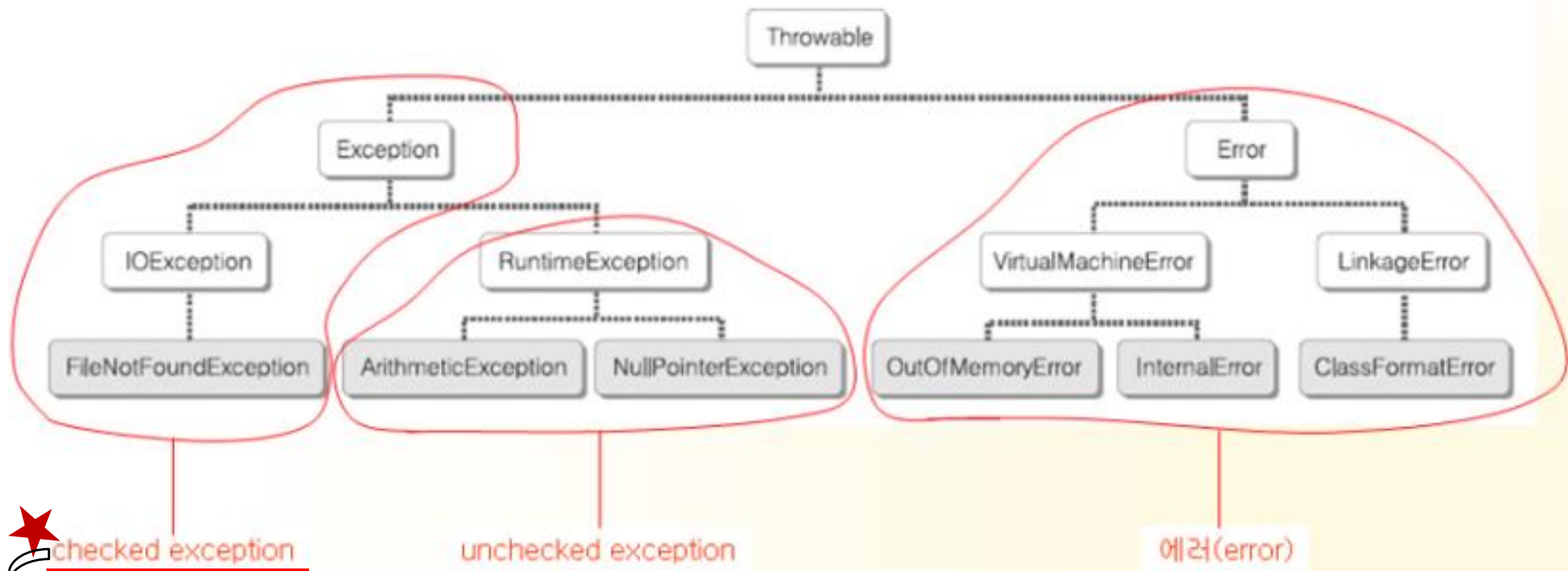
예외를 던질 가능성이 있는 예외를 반드시 선언해야 하므로, 메소드 선언에서 **throws 구문**을 찾는다.

언제 그런 예외가 발생할 수 있는가에 대한 설명이 있다.

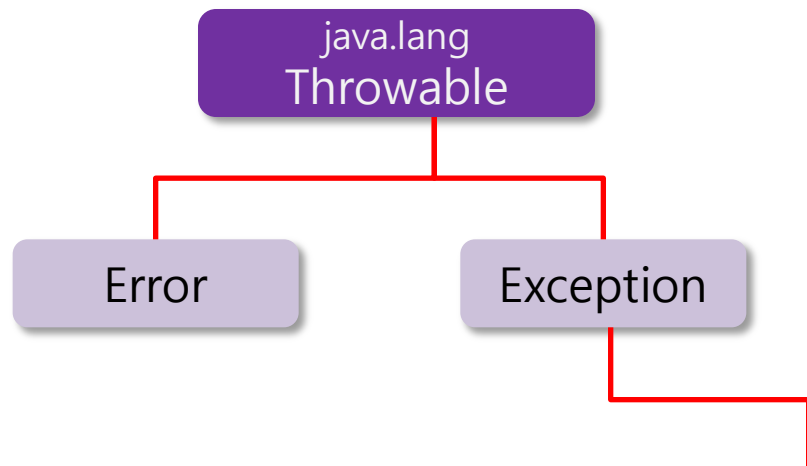
Exception Handling

❖ java.lang. Exception 클래스 : Throwable클래스를 상속받음.

- 자바에서 예외 처리를 위해 제공되는 클래스



파일 입출력, 메모리 입출력, 데이터베이스 입출력, 네트워크 입출력 등



Direct Known Subclasses:

AclNotFoundException, ActivationException, AlreadyBoundException, ApplicationException, AWTException, BackingStoreException, BadAttributeValueExpException, BadBinaryOpValueExpException, BadLocationException, BadStringOperationException, BrokenBarrierException, CertificateException, CloneNotSupportedException, DataFormatException, DatatypeConfigurationException, DestroyFailedException, ExecutionException, ExpandVetoException, FontFormatException, GeneralSecurityException, GSSEException, IllegalClassFormatException, InterruptedException, IntrospectionException, InvalidApplicationException, InvalidMidiDataException, InvalidPreferencesFormatException, InvalidTargetObjectTypeException, IOException, JAXBException, JMEException, KeySelectorException, LambdaConversionException, LastOwnerException, LineUnavailableException, MarshalException, MidiUnavailableException, MimeTypeParseException, MimeTypeParseException, NamingException, NoninvertibleTransformException, NotBoundException, NotOwnerException, ParseException, ParserConfigurationException, PrinterException, PrintException, PrivilegedActionException, PropertyVetoException, ReflectiveOperationException, RefreshFailedException, RemarshalException, RuntimeException, SAXException, ScriptException, ServerNotActiveException, SOAPException, SQLException, TimeoutException, TooManyListenersException, TransformerException, TransformException, UnmodifiableClassException, UnsupportedAudioFileException, UnsupportedCallbackException, UnsupportedFlavorException, UnsupportedLookAndFeelException, URISyntaxException, URIReferenceException, UserException, XAException, XMLParseException, XMLSignatureException, XMLStreamException, XPathException

❖ 예외 Exception 처리방법

- `try ~ catch (~ finally)` 구문

1. 예외 **처리** 코드

- 예외를 잡아주기

- `throws` 구문

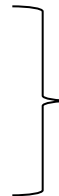

2. 예외 처리 **떠넘기기**

- 예외를 던지기 (책임 미루기)

1. 예외 처리 코드

❖ **try ~ catch (~ finally)** : 예외가 발생한 지점에서 직접 처리하는 방법

```
■ try {  
    예외를 발생시킬 가능성이 있는 문장들 ;  
}  
catch (예외타입 매개변수) {    // 여러 개의 catch문을 제공할 수 있음  
    예외타입(클래스)이 발생할 경우 처리할 문장들 ;  
}  
finally {    // 이하 생략이 가능함.  
    항상 실행할 필요가 있는 문장들 ;  
}
```



생략 가능



예제 : try~catch 적용해서 해결해봅시다

```
1 import java.io.*;
2 public class Exam {
3     public void method() {
4         // FileNotFoundException
5         FileReader f = new FileReader("a.txt");
6         int inp;
7         while((inp=f.read())!=-1) //IOException
8             System.out.print((char)inp);
9         f.close(); // IOException
10    }
11    public static void main(String[] args) {
12        new Exam().method();
13        // 아래 두줄과 같음.
14        // Exam a = new Exam();
15        // a.method();
16    }
17 }
```

Exception in thread "main" java.lang.Error: Unresolved compilation problems:
Unhandled exception type [FileNotFoundException](#)
Unhandled exception type [IOException](#)
Unhandled exception type [IOException](#)

예시 : try~catch 적용 예시

```
1 import java.io.*;
2 public class Exam {
3     public void method() {
4         try {
5             // FileNotFoundException
6             FileReader f = new FileReader("a.txt");
7             int inp;
8             while((inp=f.read())!=-1) //IOException
9                 System.out.print((char)inp);
10            f.close(); // IOException
11        } catch(Exception e) { // 무슨 예외인지 알 수 없게 처리
12            System.out.println("예외발생하였음");
13        }
14    }
15    public static void main(String[] args) {
16        new Exam().method();
17        // 아래 두줄과 같음.
18        // Exam a = new Exam();
19        // a.method();
20    }
21 }
```

Console   P

<terminated> Exam (1

예외발생하였음

2. 예외 처리 떠넘기기

❖ throws

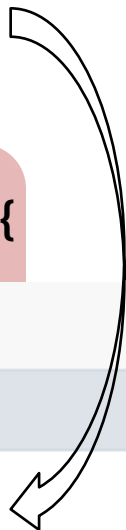
- 예외가 발생한 메소드를 호출한 곳으로 **예외 처리를 넘겨준다. (떠넘기기)**
- 따라서 throws 키워드가 있는 메소드를 사용할 때
try~catch문으로 예외사항을 반드시 처리해주어야 한다.
- throws 키워드는 메소드 선언부에 사용된다.

- 접근제한자 리턴타입 메소드이름(타입 변수) **throws 예외클래스** {
 메소드가 수행할 작업들(명령) ;
}

Method Detail

read

```
public abstract int read()  
                    throws IOException
```



예제 : throws 적용해서 해결해봅시다

```
1 import java.io.*;
2 public class Exam {
3     public void method() {
4         // FileNotFoundException
5         FileReader f = new FileReader("a.txt");
6         int inp;
7         while((inp=f.read())!=-1) //IOException
8             System.out.print((char)inp);
9         f.close(); // IOException
10    }
11    public static void main(String[] args) {
12        new Exam().method();
13        // 아래 두줄과 같음.
14        // Exam a = new Exam();
15        // a.method();
16    }
17 }
```

Exception in thread "main" java.lang.Error: Unresolved compilation problems:
Unhandléd exception type [FileNotFoundException](#)
Unhandléd exception type [IOException](#)
Unhandléd exception type [IOException](#)

예시 : throws 적용 예시

```
1 import java.io.*;
2 public class Exam {
3     public void method() throws Exception {
4         // FileNotFoundException
5         FileReader f = new FileReader("a.txt");
6         int inp;
7         while((inp=f.read())!=-1) //IOException
8             System.out.print((char)inp);
9         f.close(); // IOException
10    }
11    public static void main(String[] args) {
12        try {
13            new Exam().method(); // 미처리시 예러가 뜹니다!!!
14        } catch(Exception e) {
15            System.out.println("예외가 발생합니다.");
16        }
17    }
18 }
```

Console × Prob

<terminated> Exam (15) |

예외가 발생합니다.

try ~ catch 문

try~catch 구문의 실행

정상적인 실행의 경우

```
try{
```

예외발생 가능 코드;

```
catch(예외클래스 매개변수) {  
}
```

예외 처리 코드;

```
}
```

예외상황 발생의 경우

```
try{
```

예외발생



더 이상 실행 안됨

```
} catch(예외클래스 매개변수) {
```

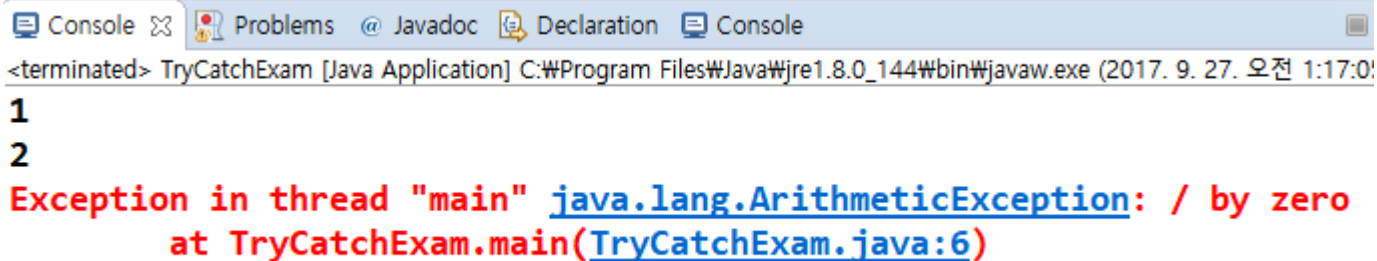
예외 처리 코드;

```
}
```

예제 : try~catch를 적용해봅시다

```
1 public class TryCatchExam {  
2     public static void main(String[] args) {  
3         System.out.println("1");  
4  
5         System.out.println("2");  
6         System.out.println( 1 / 0 );  
7         System.out.println("3");  
8  
9         System.out.println("4");  
10        System.out.println("5");  
11    }  
12 }
```

try~catch를
적용해봅시다



The screenshot shows an IDE's console window. The title bar includes tabs for 'Console', 'Problems', 'Javadoc', 'Declaration', and another 'Console'. The main content of the console shows the execution of the 'TryCatchExam' application. It prints '1' and '2' successfully, but then an exception occurs at line 6 of 'TryCatchExam.java'. The exception is a 'java.lang.ArithmeticException: / by zero'. The console text is as follows:

```
<terminated> TryCatchExam [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (2017. 9. 27. 오전 1:17:01)  
1  
2  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
    at TryCatchExam.main(TryCatchExam.java:6)
```

Exception Handling

예시 : try~catch 적용 결과 – 무엇이 출력되고, 무엇이 출력 안 되는지! 살펴보세요!

```
1 public class TryCatchExam {  
2     public static void main(String[] args) {  
3         System.out.println("1");  
4  
5         try {  
6             System.out.println("2");  
7             System.out.println( 1 / 0 );  
8             System.out.println("3");  
9         } catch (Exception e) {  
10            System.out.println("4");  
11        }  
12  
13        System.out.println("5");  
14    }  
15 }
```

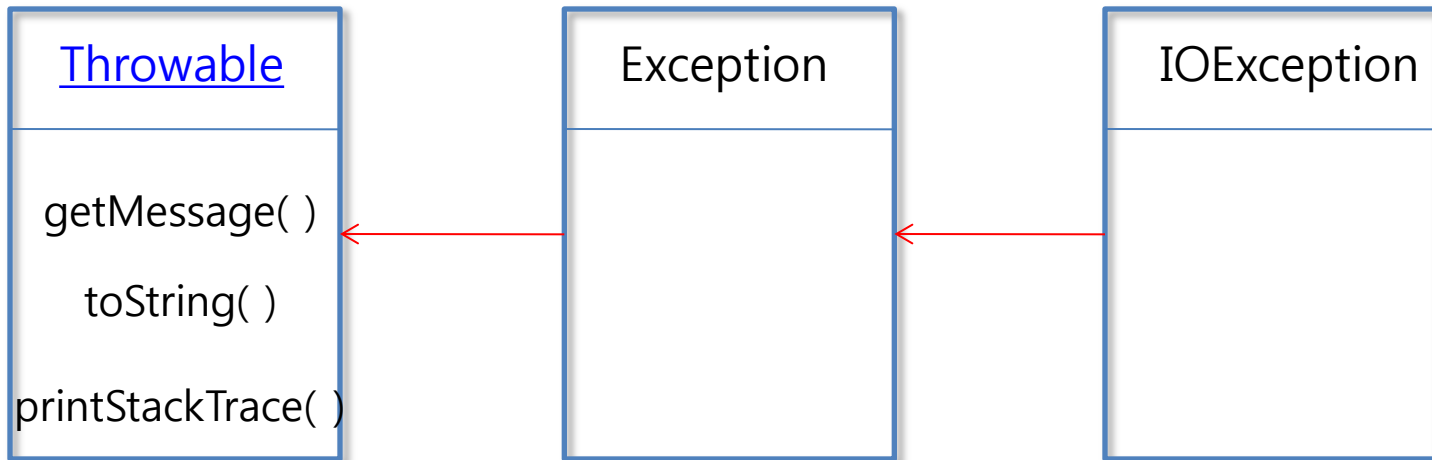


Console ✕
<terminated> TryC
1
2
4
5

Exception Handling

예외 객체 관련 메소드들...

```
try {  
    // 예외를 발생시킬 가능성이 있는 문장들 ;  
}  
catch (Exception e) {    // 여러 개의 catch문을 제공할 수 있음  
    // 예외타입(클래스)이 발생할 경우 처리할 문장들 ;  
}
```



❖ 예외 클래스의 메시지를 출력하기 위한 메소드들

- `getMessage()` : 예외 객체에 포함된 **메시지**를 얻음 → 간단한 힌트
문자열 값을 리턴
- `toString()` : 예외 객체에 대한 **좀더 자세한 정보**를 가져올 수 있음
문자열 값을 리턴
- `printStackTrace()` : 예외 발생하기까지 **메소드 호출 순서**를 **역순 출력**
→ 발생 원인, 어떤 로직을 통해 발생했는지 정보 등
직접 출력 (리턴값: void)

Exception Handling

예제1 : 메소드를 활용하여 예외 메시지를 출력해보자

```
1 public class ExceptionEx1 {  
2     public static void divide(int a, int b) {  
3         try {  
4             System.out.println(a+"/"+b+"="+a/b);  
5         } catch(Exception e) {  
6             System.out.println("e.getMessage() : "+e.getMessage());  
7             System.out.println("e.toString() : "+e.toString());  
8             System.out.println("e.printStackTrace() : ");  
9             e.printStackTrace();  
10        }  
11    }  
12    public static void main(String[] args) {  
13        divide(10,0);  
14    }  
15 }
```

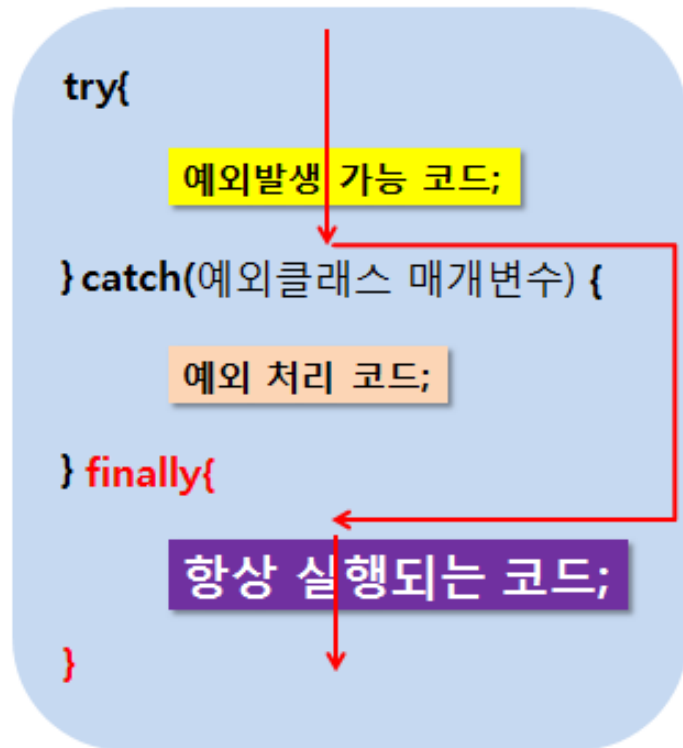
```
e.getMessage() : / by zero  
e.toString() : java.lang.ArithmeticException: / by zero  
e.printStackTrace() :  
java.lang.ArithmeticException: / by zero  
    at ExceptionEx1.divide(ExceptionEx1.java:4)  
    at ExceptionEx1.main(ExceptionEx1.java:13)
```

try ~ catch
~ finally 문

Exception Handling

try~catch~finally 구문의 실행

정상적인 실행의 경우



예외상황 발생의 경우



Exception Handling

예제2 : 명령행 매개변수 값을 주지 않았다고 가정해봅시다.

```
1 public class ExceptionEx2 {  
2     public static void main(String[] args) {  
3         try{  
4             String data1 = args[0];  
5             String data2 = args[1];  
6             int num1 = Integer.parseInt(data1);  
7             int num2 = Integer.parseInt(data2);  
8             System.out.println(num1 + num2);  
9         }catch(ArrayIndexOutOfBoundsException ae){  
10             System.out.println("배열의 범위를 벗어났습니다.");  
11         }finally{  
12             System.out.println("여기는 무조건 실행됩니다.");  
13         }  
14     }  
15 }
```

Console Problems @ JavaC

<terminated> ExceptionEx2 [Java Applica

배열의 범위를 벗어났습니다.
여기는 무조건 실행됩니다.

다중 try ~ catch 문

Exception Handling

다중 try~catch 구문의 사용

다중 **try~catch** 사용 : try절에서 발생할 예외사항이 여러 개일 경우

```
try{
    예외를 발생시킬 가능성이 있는 문장들;
}catch(예외타입_1 매개변수){
    예외 타입_1 의 예외가 발생할 경우 처리할 문장들;
} catch(예외타입_2 매개변수){
    예외 타입_2 의 예외가 발생할 경우 처리할 문장들;
}
```

다중 **try~catch~finally**

```
try{
    예외를 발생시킬 가능성이 있는 문장들;
}catch(예외타입_1 매개변수){
    예외 타입_1 의 예외가 발생할 경우 처리할 문장들;
}catch(예외타입_2 매개변수){
    예외 타입_2 의 예외가 발생할 경우 처리할 문장들;
}finally{
    항상 실행할 필요가 있는 문장들;
}
```


예제2 (수정)

```
1 public class ExceptionEx2 {
2     public static void main(String[] args) {
3         try{
4             String data1 = args[0];
5             String data2 = args[1];
6             int num1 = Integer.parseInt(data1);
7             int num2 = Integer.parseInt(data2);
8             System.out.println(num1 + num2);
9         }catch(NumberFormatException ne){
10             System.out.println("숫자로 변환할 수 없습니다.");
11         }catch(ArrayIndexOutOfBoundsException ae){
12             System.out.println("배열의 범위를 벗어났습니다.");
13         }finally{
14             System.out.println("여기는 무조건 실행됩니다.");
15         }
16     }
17 }
```

Console Problems @ Javac

<terminated> ExceptionEx2 [Java Applica

배열의 범위를 벗어났습니다.

여기는 무조건 실행됩니다.

❖ 자바의 주요 예외 클래스 (자세한 건 java-api참고)

예외 클래스)	예외 발생 경우
ArithmeticException	정수를 0으로 나눌 때
NullPointerException	Null 레퍼런스를 참조할 때
ClassCastException	변환할 수 없는 타입으로 객체를 변환할 때
ArrayIndexOutOfBoundsException	배열의 범위를 벗어나서 접근할 때
IOException	입출력 동작 실패 또는 인터럽트 시
NumberFormatException	문자열이 나타내는 숫자와 일치하지 않는 타입의 숫자로 변환하려는 경우

예제3 (실습) : try~catch~finally 구문을 실습해봅시다.

[조건]

파일명 : **ExceptionEx3.java**

- ❖ int형 데이터 3개를 저장할 수 있는 배열 **array**를 선언 및 객체 생성
- ❖ int형 변수 **a, b, c**에 사용자로부터 값을 입력받는다. (예: Scanner 클래스 등 활용)
- ❖ a/b (a를 b로 나눈) 결과를 **array[c]**에 저장해본다.

❖ **[예외처리]**

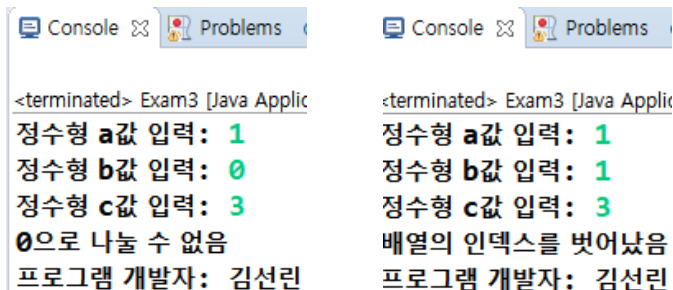
■ **0으로 나누었을 때**

- 0으로 나눌 수 없음을 화면에 **출력**

■ **배열의 인덱스를 벗어났을 때**

- 배열의 인덱스를 벗어났음을 화면에 **출력**

■ **예외 발생 유무와 관계없이** "프로그램 개발자 : 0 0 0"를 화면에 **출력**



```
<terminated> Exam3 [Java Applic
정수형 a값 입력: 1
정수형 b값 입력: 0
정수형 c값 입력: 3
0으로 나눌 수 없음
프로그램 개발자: 김선린

<terminated> Exam3 [Java Applic
정수형 a값 입력: 1
정수형 b값 입력: 1
정수형 c값 입력: 3
배열의 인덱스를 벗어났음
프로그램 개발자: 김선린
```



0 0 0는 본인 **성명**을 쓰자.

예제3 (실습) : try~catch~finally 구문을 실습해봅시다.

파일명 : ExceptionEx3.java

[Hint]

왼쪽 코드에 적절한 코드를
작성해봅시다! : try~catch~finally

```
1 import java.util.Scanner; // Scanner는 java.util 패키지
2 public class ExceptionEx3 {
3     public static void main(String[] args) {
4         // int형 배열(길이3) array 선언 및 생성
5         int[] array = new int[3];
6
7         // Scanner 객체 생성 - 표준입력 스트림(키보드)
8         Scanner sc = new Scanner(System.in);
9
10        // 키보드 입력값을 정수형 변수 a, b, c에 저장
11        System.out.print("정수형 a값 입력: ");
12        int a = sc.nextInt();
13
14        System.out.print("정수형 b값 입력: ");
15        int b = sc.nextInt();
16
17        System.out.print("정수형 c값 입력: ");
18        int c = sc.nextInt();
19
20        // a/b값을 array[c]에 저장하기
21        array[c] = a / b;
22
23        // 스트림 닫기
24        sc.close();
25    }
26 }
```

Console Problems

<terminated> Exam3 [Java Applic

정수형 a값 입력: 1

정수형 b값 입력: 0

정수형 c값 입력: 3

0으로 나눌 수 없음

프로그램 개발자: 김선린

Console Problems

<terminated> Exam3 [Java Applic

정수형 a값 입력: 1

정수형 b값 입력: 1

정수형 c값 입력: 3

배열의 인덱스를 벗어났음

프로그램 개발자: 김선린

Exception Handling

예제4 (실습) : try~catch~finally 구문을 실습해봅시다.

[조건]

파일명 : **ExceptionEx4.java**

```
1 public class ExceptionEx4 {  
2     public static void main(String[] args) {  
3         String str = null;  
4         System.out.println("문자열 : " + str.length());  
5     }  
6 }
```

Console Problems @ Javadoc Declaration Console
<terminated> ExceptionEx4 [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (2017.
Exception in thread "main" java.lang.NullPointerException
at ExceptionEx4.main(ExceptionEx4.java:4)

- ❖ 에러 메시지 대신 아래와 같이 출력되도록 위 프로그램을 개선해보자.
- ❖ 예외 발생 유무와 관계없이 "프로그램 개발자 : O O O"를 화면에 출력

Console Problems @ Javadoc Declaration Console
<terminated> ExceptionEx4 [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (2017.
객체없음 : **java.lang.NullPointerException**
프로그램 개발자 : 홍길동