

12장. 예외 처리

Exception



에러와 예외

오류의 종류

1. 에러(Error)

- 하드웨어의 오작동 고장으로 인한 오류
- 에러가 발생되면 프로그램이 종료되고 정상으로 돌아갈 수 없음

2. 예외(Exception)

- 사용자의 잘못된 조작 또는 개발자의 잘못된 코딩으로 인한 오류
- 예외가 발생되면 프로그램이 종료되고, 예외 처리를 하면 정상으로 돌아갈 수 있음

예외의 종류

1. 일반 예외(Exception)

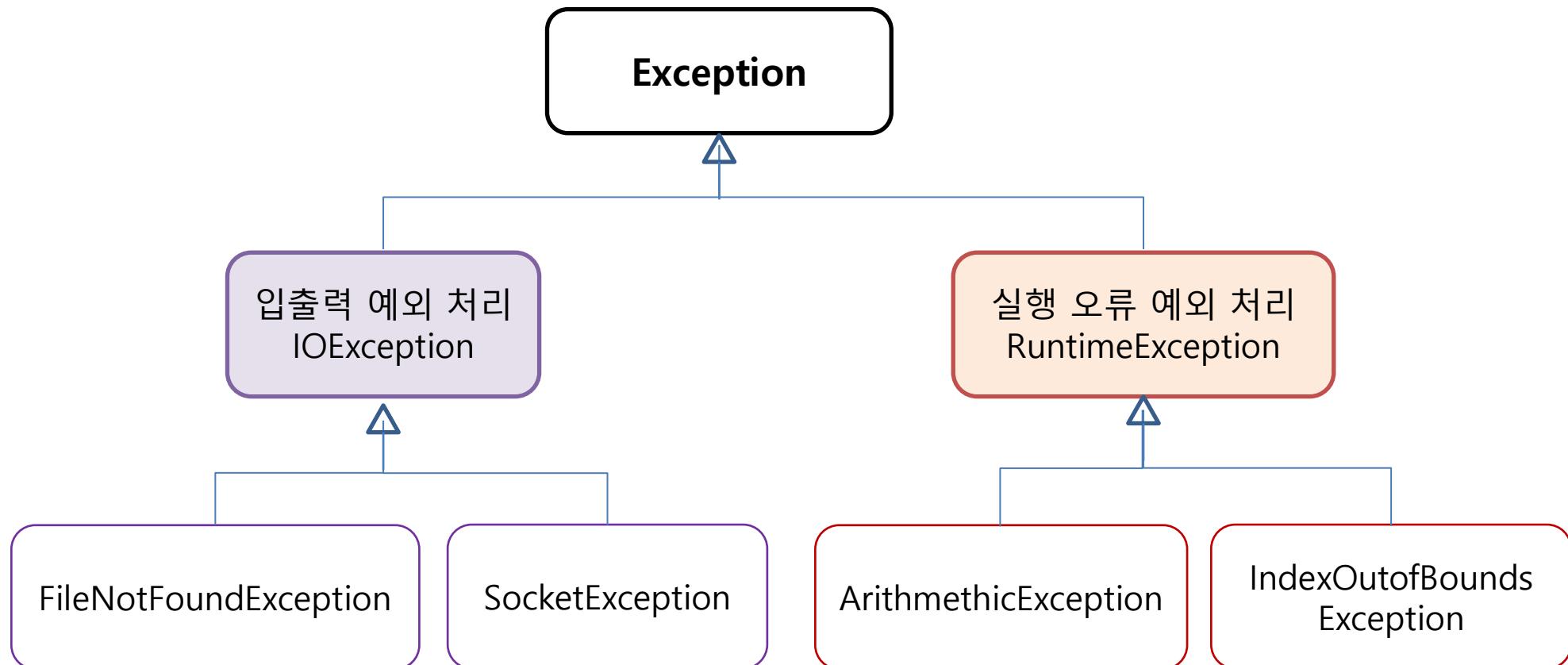
- 예외 처리가 없으면 컴파일 되지 않는 예외 – 컴파일 체크

2. 실행 예외(RuntimeException)

- 예외 처리를 생략해도 컴파일 되는 예외
- 개발자의 경험과 판단으로 예외 코드 작성 필요



예외 클래스의 종류



예외 클래스의 종류

Java.lang 패키지 -> Exception Summary

Exception Summary	
Exception	Description
ArithmeticException	Thrown when an exceptional arithmetic condition occurs.
ArrayIndexOutOfBoundsException	Thrown to indicate that an array has been accessed using an index that is outside the bounds of the array.
ArrayStoreException	Thrown to indicate that an attempt was made to store an object of the wrong type into an array.
ClassCastException	Thrown to indicate that an attempt was made to cast an object to a type to which it cannot be cast.
ClassNotFoundException	Thrown when an application tries to load a class using <code>Class.forName</code> and no definition of the class is found.
CloneNotSupportedException	Thrown to indicate that an attempt was made to clone an object that does not implement the <code>Cloneable</code> interface.
EnumConstantNotPresentException	Thrown when an application tries to access an enum constant that is not present in the class.
Exception	The class <code>Exception</code>

Module [java.base](#)
Package [java.lang](#)

Class ArithmeticException

`java.lang.Object`
 `java.lang.Throwable`
 `java.lang.Exception`
 `java.lang.RuntimeException`
 `java.lang.ArithmaticException`

All Implemented Interfaces:

`Serializable`



try ~ catch문

try ~ catch문

예외처리를 하면 예외 상황을 알려 주는 메시지를 볼 수 있고, 프로그램이 비정상적으로 종료되지 않고 계속 수행되도록 만들 수 있다.

try{

예외가 발생할 수 있는 코드

}catch(처리할 예외 타입 e){

예외를 처리하는 코드

}



try ~ catch문

try ~ catch문

```
public class ExceptionHandling extends Object{  
  
    public static void main(String[] args) {  
        //배열의 범위를 벗어난 경우 예외 처리  
        int[] arr = new int[3];  
  
        // 저장  
        try {  
            arr[0] = 10;  
            arr[1] = 20;  
            arr[2] = 30;  
            //arr[3] = 40;  
  
            for(int i = 0; i < arr.length; i++) {  
                System.out.println(arr[i]);  
            }  
        }catch(ArrayIndexOutOfBoundsException e) {  
            System.out.println(e);  
            System.out.println("예외 처리 부분");  
        }  
    }  
}
```



다중 try ~ catch

다중 try ~ catch문

```
public class ExceptionHandling2 {  
    public static void main(String[] args) {  
        //다중 try ~ catch문  
        try {  
            //예외 1  
            String name = null; //이름을 넣으면 문제 없음  
            System.out.println(name.toString()); //null 객체에 접근하여 오류 발생  
  
            //예외 2  
            int num1 = 10;  
            int num2 = 0;  
  
            int result = num1 / num2; //0으로 나눌 수 없음  
            System.out.println(result);  
        }catch(ArithmetricException e) {  
            System.out.println("0으로 나눌 수 없습니다.");  
        }catch(NullPointerException e) {  
            System.out.println(e);  
        }  
    }  
}
```



try~catch~finally문

try~catch~finally문 사용하기

프로그램에서 외부장치와의 연동시 초기화나 마무리 작업시 주로 사용한다.

이때 사용하는 블록이 finally인데 일단 try블록이 수행되면 어떤 경우에도 반드시 수행된다.



try~catch~finally문

```
Scanner scan = new Scanner(System.in);
System.out.println("=====");
System.out.println("1.포도 | 2.바나나 | 3.복숭아");
System.out.println("=====");
System.out.print("선택>");

try {
    int selNum = scan.nextInt();

    Fruit fruit = null;
    if(selNum == 1) {
        fruit = new Grape();
    }else if(selNum == 2) {
        fruit = new Banana();
    }else if(selNum == 3) {
        fruit = new Peach();
    }else {
        System.out.println("지원하지 않는 기능입니다.");
        //return;
    }
    fruit.showInfo();
}catch(NullPointerException e) {
    e.getMessage();
}catch(InputMismatchException e) {
    System.out.println("잘못된 입력입니다.");
}finally {
    System.out.println("프로그램을 종료합니다.");
    scan.close();
}
```



예외 처리 – throws

throws로 예외처리 미루기(떠넘기기)

예외 처리를 해당 메서드에서 하지 않고 미룬 후, 메서드를 호출하여 사용하는 곳에서 예외를 처리하는 방법이다.

메서드명 **throws** 예외클래스1, 예외클래스2,..{

```
public class ThrowsException {  
  
    public Class<?> loadClass(String className) throws ClassNotFoundException {  
        Class<?> c = Class.forName(className);  
        return c;  
    }  
  
    public static void main(String[] args) {  
        ThrowsException test = new ThrowsException();  
        try {  
            test.loadClass("java.lang.String");  
        } catch (ClassNotFoundException e) {  
            //e.printStackTrace();  
            System.out.println("클래스가 존재하지 않습니다.");  
        }  
    }  
}
```

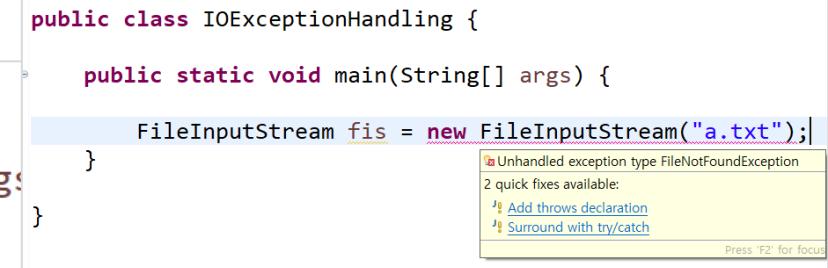


예외 처리

try ~ catch문 : 컴파일 오류의 예

✓ FileNotFoundException

```
public class IOExceptionHandling {  
    public static void main(String[] args) {  
        try {  
            FileInputStream fis = new FileInputStream("a.txt");  
        } catch (FileNotFoundException e) {  
            System.out.println(e);  
        }  
        System.out.println("수행 완료!!");  
    }  
}
```



[java.io.FileNotFoundException](#): a.txt (지정된 파일을 찾을 수 없습니다)
수행 완료!!



예외 처리

try ~ catch문

```
=====  
1.예금 | 2.출금 | 3.잔고 | 4.종료  
=====
```

선택>a

```
Exception in thread "main" java.util.InputMismatchException  
at java.base/java.util.Scanner.throwFor(Scanner.java:939)  
at java.base/java.util.Scanner.next(Scanner.java:1594)  
at java.base/java.util.Scanner.nextInt(Scanner.java:2258)  
at java.base/java.util.Scanner.nextInt(Scanner.java:2212)  
at banking.BankingSwitch.main(BankingSwitch.java:18)
```

```
public class BankingSwitch {  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        boolean run = true;  
        int balance = 0;  
  
        try {  
            while(run) {  
                System.out.println("=====");  
                System.out.println("1.예금 | 2.출금 | 3.잔고 | 4.종료");  
                System.out.println("=====");  
                System.out.print("선택>");  
  
                int selNum = sc.nextInt();  
                switch(selNum) {  
                    case 1:  
                        System.out.print("예금액>");  
                        balance += sc.nextInt();  
                        break;  
                    case 2:  
                        System.out.print("출금액>");  
                        balance -= sc.nextInt();  
                        break;  
                    case 3:  
                        System.out.println("잔고> " + balance);  
                        break;  
                    case 4:  
                        run = false;  
                        break;  
                    default:  
                        System.out.println("메뉴를 잘못 누르셨습니다. 다시 입력해 주세요");  
                        break;  
                }  
            }  
            System.out.println("프로그램 종료.");  
            sc.close();  
        }catch(Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```



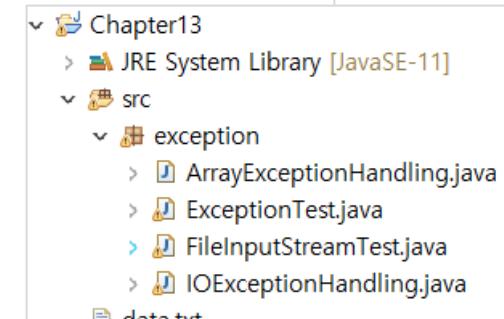
예외 처리

다중 try ~ catch문 사용하기

- ✓ 예외 상황이 여러 개라면 catch 블록을 예외 상황 수만큼 구현해야 한다

```
public class FileInputStreamTest {  
  
    public static void main(String[] args) {  
        try {  
            FileInputStream fis = new FileInputStream("data.txt");  
            //data.txt의 내용을 읽어서 출력하기  
            int i;  
            while((i=fis.read()) != -1) {  
                System.out.print((char)i);  
            }  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

상위 예외클래스를 아래쪽에 위치시켜야 함



예외 처리

try~catch~finally문 사용하기

```
public class FileInputStreamTest2 {  
    public static void main(String[] args) {  
        FileInputStream fis = null;  
        try {  
            fis = new FileInputStream("data.txt");  
            //data.txt의 내용을 읽어서 출력하기  
            int i;  
            while((i=fis.read()) != -1) {  
                System.out.print((char)i);  
            }  
        } catch (FileNotFoundException e) {  
            e.printStackTrace();  
        } catch (IOException e) {  
            e.printStackTrace();  
        } finally {  
            try {  
                fis.close(); //파일 입력 스트림 닫기  
            } catch (IOException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```



예외 처리

try~with~resources문 사용하기

close() 메서드를 명시적으로 호출하지 않아도 try 블록내에서 열린 리소스를 자동으로 닫도록 만들수 있다.

try~with~resource 문을 사용하려면 해당 리소스가 AutoCloseable 인터페이스를 구현해야 한다.

```
public class ExceptionHandling2 {
    public static void main(String[] args) {
        try(FileInputStream fis = new FileInputStream("a.txt")) {
            } catch (IOException e) {
                System.out.println(e);
            }
        System.out.println("여기도 수행됩니다.");
    }
}
```

java.io

Class FileInputStream

java.lang.Object

java.io.InputStream

java.io.FileInputStream

All Implemented Interfaces:

Closeable **AutoCloseable**



예외 처리

AutoCloseable 인터페이스

```
public class AutoCloseObj implements AutoCloseable{
    @Override
    public void close() throws Exception {
        System.out.println("리소스가 close() 되었습니다.");
    }
}

public class AutoCloseTest {
    public static void main(String[] args) {
        try(AutoCloseObj obj = new AutoCloseObj()){
            }catch(Exception e) {
                System.out.println("예외 부분입니다.");
            }
        }
}
```

리소스가 close() 되었습니다.



예외 처리

사용자 정의 예외 클래스 만들기

개발자가 직접 정의해서 만들어 내는 예외 클래스이다.

throw new 예외클래스;

```
package exception.thrownexam;

public class IDFormatException extends Exception{

    private static final long serialVersionUID = 1L;

    //생성자 - 매개변수로 예외 상황 메시지를 받음
    public IDFormatException(String message)
        super(message);
    }

}
```

Constructors

Modifier Constructor

Exception()

Exception(String message)

Exception 클래스
의 생성자 상속.



예외 처리

사용자 정의 예외 테스트하기

```
public static void main(String[] args) {
    IDFormatTest test = new IDFormatTest();

    //아이디가 null인 경우
    String userID = null;
    try {
        test.setUserID(userID);
    } catch (IDFormatException e) {
        System.out.println(e.getMessage());
    }

    //아이디가 7자인 경우
    userID = "abc1234";
    try {
        test.setUserID(userID);
    } catch (IDFormatException e) {
        System.out.println(e.getMessage());
    }
}
```

아이디는 null일 수 없습니다.
아이디는 8자 이상 20자 이하로 입력하세요.

