제 30강

예외처리2

교재:p162~167

목차

예외/예외처리 복습

1) 예외(Exception)란?

: 프로그램 실행 중 발생하는 오류 중에서 처리가 가능한 것을 의미

2) 예외처리(Exception Handling)란?

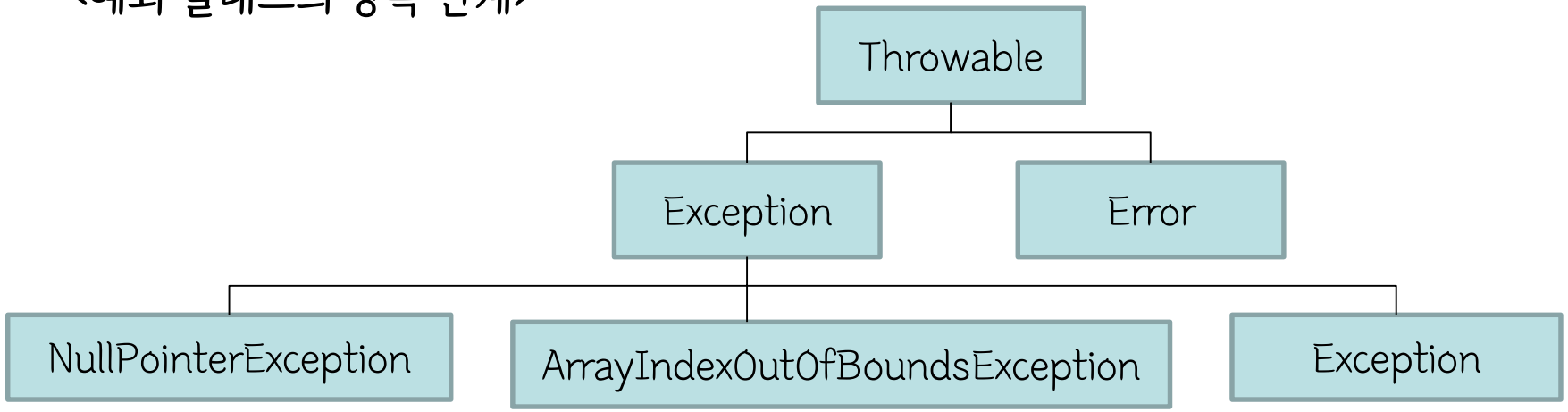
: 예외가 발생했을 때 이를 적절히 처리하여 프로그램이 비정상적으로 종료되는 것을 막는 방법

1. 객체로서의 예외

"자바에서 예외는 Exception 클래스의 객체!"

자바는 객체지향 언어이기에 예외도 객체로 처리한다.

<예외 클래스의 상속 관계>



1. 객체로서의 예외

"자바에서 예외는 Exception 클래스의 객체!"

- 모든 예외 클래스들은 Exception 클래스를 상속받으므로 Exception으로 처리 가능
- 예외 클래스들에서 Exception 클래스의 메서드 사용 가능

  ex) getMessage( ), toString( ),printStackTrace( ) …

<한번에 모든 예외 처리하는 방법>

```
try{
        예외가 발생할 수 있는 명령;
        }
catch(Exception e){
        예외 발생시 처리할 명령;
}
```

2. throw 와 throws

throw( )? 예외발생

: 프로그래머가 고의로 예외를 발생 시킬 때 사용 하는 방법

<구조>

```
Exception e = new Exception("Exception");
throw e;
```

2. throw 와 throws

<실습> Exam-64.java

```
public class Exception2 {
    public static void main(String[] args) {
        try {
            Exception e=new Exception("고의 예외");
            throw e;
        }
        catch(Exception e) {
            System.out.println("예외 발생");
            System.out.println(e.getMessage());
        }
    }
}
```

<실행결과>

```
예외 발생
고의 예외
```

Exception 생성자 호출 시 전달했던 문자열이
내부적으로 저장되어,
객체.getMessage( )를 호출하면 출력된다!

2. throw 와 throws

throws( )? 예외던지기

: 예외가 발생했을 경우 현재 메서드가 예외를 처리하지 않고 자신을 호출 한 쪽으로 예외 처리에 대한 책임을 넘기는 것

<구조>

```
void method( ) throws Exception{ … }
```

예외 던지기 시, 메서드 선언 부에 throws 키워드를 붙여, 메서드 호출 하는 부분에서 처리하도록 하는 기법

2. throw 와 throws

<실습> Exam-65.java

```
public class Exception3 {
    public static void main(String[] args) {
        try {
            methodA();
        }
        catch(Exception e) {
            System.out.println("메인에서 처리");
        }
    }
    public static void methodA() throws Exception{
        methodB();
    }
    public static void methodB() throws Exception{
        methodC();
    }
    public static void methodC() throws Exception{
        Exception e= new Exception();
        throw e;
    }
}
```

<실행결과>

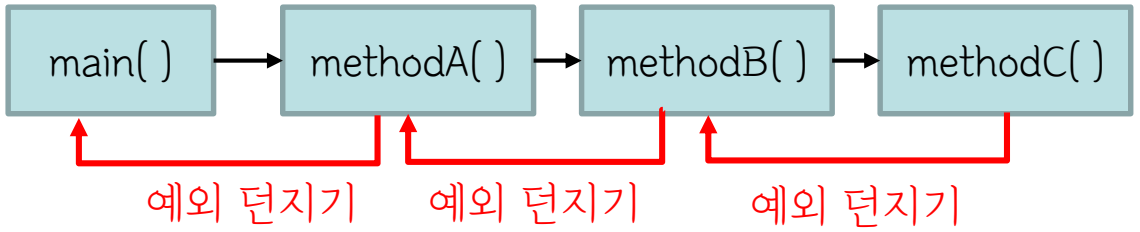
```
메인에서 처리
```

2. throw 와 throws

<실습> Exam-65.java

```
public class Exception3 {
    public static void main(String[] args) {
        try {
            methodA();
        }
        catch(Exception e) {
            System.out.println("메인에서 처리");
        }
    }
    public static void methodA() throws Exception{
        methodB();
    }
    public static void methodB() throws Exception{
        methodC();
    }
    public static void methodC() throws Exception{
        Exception e= new Exception();
        throw e;
    }
}
```

[코드 실행 과정]



2. throw 와 throws

<실습> Exam-66.java

try문 내부에 다른 try-catch문이 있을 경우 예외 처리 확인하기

```
public class RethrowExample {
    public static void main(String[] args) {
        try {
            System.out.println("외부 try...");
            try {
                System.out.println("내부 try...");
                Exception e=new Exception();
                throw e;
            }
            catch(Exception e) {
            System.out.println("(내부 try-catch) exception:"+e);
            System.out.println("예외 던지기 한번 더:");
            throw e;
            }
            finally {
                System.out.println("finally 구문 출력");
            }
            }
        catch(Exception e) {
            System.out.println("(외부 try-catch) exception:"+e);
        }
        System.out.println("종료");
    }
}
```

<실행결과>

```
외부 try...
내부 try...
(내부 try-catch) exception:java.lang.Exception
예외 던지기 한번 더:
finally 구문 출력
(외부 try-catch) exception:java.lang.Exception
종료
```

1. 외부 try 문 실행
2. 내부 try 문 실행
3. 내부 catch문 실행
4. 내부 finally문 실행
5. 외부 catch문 실행

3. 사용자 정의 예외 "자바에서는 사용자가 예외 클래스를 직접 정의 가능"

<실습> Exam-67.java

기존 예외클래스로 예의 표현 불가능 할 경우, 나만의 예외 클래스를 만들어 사용해보기

```
public class Exception5 {
    public static void main(String[] args) {
        int age=-19;
        try {
            ticketing(age);

        }catch(AgeException e) {
            e.printStackTrace();
        }
    }
    public static void ticketing(int age)throws AgeException{
        if(age<0) {
            throw new AgeException("나이 입력이 잘못되었습니다.");
        }
    }
}
```

예외 클래스 상속

```
class AgeException extends Exception{
    public AgeException() {}
    public AgeException(String message) {
        super(message);
    }
}
```

<실행결과>

```
day30.AgeException: 나이 입력이 잘못되었습니다.
        at day30.Exception5.ticketing(Exception5.java:15)
        at day30.Exception5.main(Exception5.java:7)
```