chapter 8. 예외처리

ch8-1 프로그램 오류

1.컴파일 에러 : 컴파일 할 때 발생하는 에러
2.런타임 에러 : 실행 시 발생하는 에러
3.논리적 에러 : 작성의도와 다르게 동작

<Java의 **런타임** 에러>

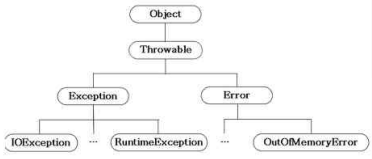
에러(error)	프로그램 코드에 의해서 수습될 수 없는 심각한 오류
예외(exception)	프로그램 코드에 의해서 수습될 수 있는 다소 미약한 오류

<예외처리 정의와 목적>

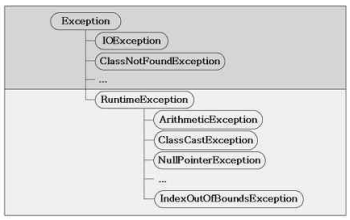
예외처리(exception handling)의

정의	프로그램 실행 시 발생할 수 있는 예외의 발생에 대비한 코드를 작성하는 것
목적	프로그램의 비정상 종료를 막고, 정상적인 실행상태를 유지하는 것

ch8-2 예외 클래스의 계층구조



[그림8-1] 예외클래스 계층도



[그림8-2] Exception클래스와 RuntimeException클래스 중심의 상속계층도

ch8-3 Exception과 RuntimeException

Exception클래스들	사용자의 실수와 같은 외적인 요인에 의해 발생하는 예외
RuntimeException클래스들	프로그래머의 실수로 발생하는 예외

ch8-4 예외처리하기. try-catch문

```
try {
    // 예외가 발생할 가능성이 있는 문장들을 넣는다.
} catch (Exception1 e1) {
    // Exception1이 발생했을 경우, 이를 처리하기 위한 문장을 적는다.
} catch (Exception2 e2) {
    // Exception2가 발생했을 경우, 이를 처리하기 위한 문장을 적는다.
} catch (ExceptionN eN) {
    // ExceptionN이 발생했을 경우, 이를 처리하기 위한 문장을 적는다.
}
```

ch8-5 try-catch문에서의 흐름

① try블럭 내에서 예외가 발생한 경우,

1. 발생한 예외와 일치하는 catch블럭이 있는지 확인한다.
2. 일치하는 catch블럭을 찾게 되면, 그 catch블럭 내의 문장들을 수행하고 전체 try-catch문을 빠져나가서 그 다음 문장을 계속해서 수행한다. 만일 일치하는 catch블럭을 찾지 못하면, 예외는 처리되지 못한다.

② try블럭 내에서 예외가 발생하지 않은 경우,

1. catch블럭을 거치지 않고 전체 try-catch문을 빠져나가서 수행을 계속한다.

ch8-6 예외의 발생과 catch블럭

-예외발생시 catch블럭을 찾아내려감
-일치하는 catch블럭이 없으면 예외처리 안됨
(비정상 종료되어 에러이후 모든 코드 실행불가)
-Exception이 선언된 catch블럭은 모든 예외처리 가능하므로 마지막에 넣는다.

ch8-7 PrintStackTrace() 와 GetNessage()

예외발생시 예외정보가 담긴 예외객체가 생성되고 catch문 참조변수에 객체주소가 담기게 된다.
예외객체 안에 자주 쓰이는 메서드 2가지!

printStackTrace( )	예외발생 당시의 호출스택(Call Stack)에 있었던 메서드의 정보와 예외 메시지를 화면에 출력한다.
getMessage( )	발생한 예외클래스의 인스턴스에 저장된 메시지를 얻을 수 있다.

ch8-8 멀티 catch블럭

-내용이 같은 catch블럭을 하나로 합친 것(JDK1.7부터)
합칠시 두 예외객체가 가지고 있는 공통된 메서드만 사용 가능함. (둘중 한쪽 객체만 가지고 있는 특정 메서드 사용불가)
-부모 자식관계인 catch블럭은 부모만 쓴다.

ch8-9 예외 발생시키기

1. 연산자 new를 이용해서 발생시키려는 예외 클래스의 객체를 만든 다음

```
Exception e = new Exception("고의로 발생시켰음");
```

2. 키워드 throw를 이용해서 예외를 발생시킨다.

```
throw e;
```

ch8-10 checked예외, unchecked예외

-checked예외 : 컴파일러가 예외처리 여부를 체크 (필수)
try-catch 안쓰면 컴파일 에러
----->Exception과 그 자손

-unchecked예외 : 컴파일러가 체크 안함 (선택)
try-catch 안써도 에러 노노
거의 모든 코드에 만들어줘야하기 때문에 체크안함
----->Runtime Exception과 그 자손

ch8-11~13 메서드에 예외 선언하기

-예외를 처리하는 방법 : 1.try-catch문 (직접처리) 2.예외 선언하기 (예외 떠넘기기, 알리기, 처리 X)

예외선언 : 메서드가 호출시 발생 가능한 예외를 자신을 호출하는 쪽에 알리는 것

참고 예외를 발생시키는 키워드 throw와 예외를 메서드에 선언할 때 쓰이는 throws를 잘 구별하자.

```
void method() throws Exception1, Exception2, ... ExceptionN {
    // 메서드의 내용
}

// method()에서 Exception과 그 자손 예외 발생 가능
void method() throws Exception {
    // 메서드의 내용
}
```

//예외가 발생한곳에서 catch 할 수도 있고, 해당 메서드를 호출한 곳에서 catch할 수 있는데 그것은 상황에 따라 판단 하여야 한다.

ch8-14 finally 블럭

-예외 발생여부와 관계없이 수행되어야 하는 코드를 try-catch문의 맨 마지막에 넣는다.

ch8-15,16 사용자 정의 예외 만들기

-우리가 직접 예외 클래스를 정의할 수 있다.
상속을 통해서 만든다)

1.조상은 Exception과 RuntimeException중에서 선택
2.예외 메시지를(String) 매개변수로 받는 생성자 필수 명시
3.그 안에 조상 생성자를 호출하는 생성자를 명시

ch8-17 예외 되던지기

예외를 처리한 후 다시 예외를 발생 시키는 것 (처리2번)
->양쪽에서 처리할 일이 있음..(분담처리)

ch8-18 연결된 예외 (예외 안의 예외)

-한 예외가 다른 예외를 발생시킬 수 있다.
-예외 A가 예외 B를 발생시키면, A는 B의 원인 예외

연결시킬 때 쓰는 메서드

Throwable initCause(Throwable cause)	지정한 예외를 원인 예외로 등록
Throwable getCause( )	원인 예외를 반환

<예외B를 발생시켜 A를 포함시키는 이유>

1.여러 예외를 하나로 묶어 다루기 위해

```
try {
   install();
} catch(SpaceException e) {
   e.printStackTrace();
} catch(MemoryException e) {
   e.printStackTrace();
} catch(Exception e) {
   e.printStackTrace();
}
```

```
결과 SpaceException: 설치할 공간이 부족합니다.
        at Ex8_13.install(Ex8_13.java:22)
        at Ex8_13.main(Ex8_13.java:4)
```

```
try {
   install();
} catch(InstallException e) {
   e.printStackTrace();
} catch(Exception e) {
   e.printStackTrace();
}
```

```
결과 InstallException: 설치 중 예외발생
        at Ex8_13.install(Ex8_13.java:17)
        at Ex8_13.main(Ex8_13.java:4)
Caused by: SpaceException: 설치할 공간이 부족합니다.
        at Ex8_13.startInstall(Ex8_13.java:31)
        at Ex8_13.install(Ex8_13.java:14)
        ... 1 more
```

```
void install() throws InstallException {
   try {
      startInstall();             // SpaceException 발생
      copyFiles();
   } catch (SpaceException e)     {
      InstallException ie = new InstallException("설치중 예외발생"); // 예외 생성
      ie.initCause(e); // InstallException의 원인 예외를 SpaceException으로 지정
      throw ie;        // InstallException을 발생시킨다.
   } catch (MemoryException me)   {
      ...
```

2.checked예외를 unchecked예외로 변경하려 할 때

```
static void startInstall() throws SpaceException, MemoryException {
  if(!enoughSpace())           // 충분한 설치 공간이 없으면...
    throw new SpaceException("설치할 공간이 부족합니다.");

  if (!enoughMemory())         // 충분한 메모리가 없으면...
    throw new MemoryException("메모리가 부족합니다.");
}
```

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

```
static void startInstall() throws SpaceException {
  if(!enoughSpace())           // 충분한 설치 공간이 없으면...
    throw new SpaceException("설치할 공간이 부족합니다.");

  if (!enoughMemory())         // 충분한 메모리가 없으면...
    throw new RuntimeException(new MemoryException("메모리가 부족합니다."));
} // startInstall메서드의 끝
```