Ch9
예외처리

객체지향프로그래밍(기본) 2019

경상대학교 항공우주및소프트웨어공학전공

예외처리

프로그램을 작성 할 때 발생하는 예외적인 상황이 있을 수 있다. 사용자의 문제 일 수도 있고 시스템 상의 문제 일 수도 있다.

이러한 상황에 대처하기 위해 문제가 발생할 가능성이 높은 곳에 예외적인 상황이 발생했을 때의 처리 방안을 선언한다.

자바에서 try, catch, throws를 사용하여 예외를 처리한다.

9.1 프로그램의 위험요소 파악하기

자바에서 예외처리는 프로그램 실행 중에 생길 수 있는 문제를 예외적인 상황으로 인식하여 처리 할 수 있다.

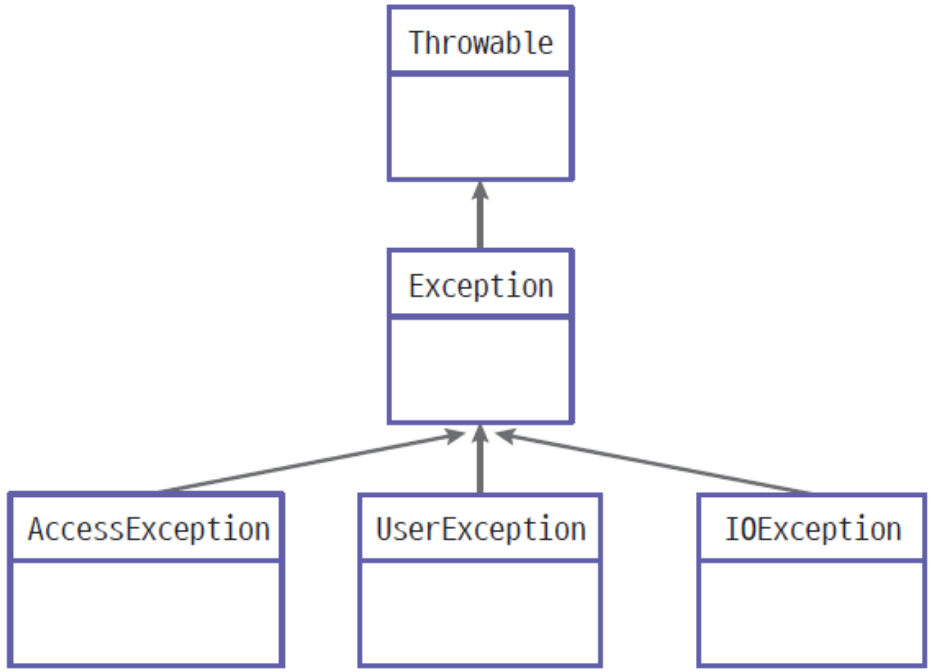
프로그램에서 오류를 발생할 수 있는 메소드를 사용할 때 메소드에 오류를 발생 시킬 수 있는 문제점에 대하여 선언해야 한다.

예외를 발생시킬 수 있는 문제가 발생하면 바로 던져줄 수 있도록 throw 명령어를 이용하여 메소드에 위험 요소를 선언한다.

위험 요소가 선언된 메소드는 반드시 try 구문 안에서 위험 요소가 있는지 파악되어야 하고 위험 요소 발생시 catch 구분에서 예외를 받아 처리해 주어야 한다.

Exception 클래스

자바에서 예외는 Exception 클래스를 기반으로 하는 다양한 하위 클래스로 만들어 지며 Exception 클래스는 Throwable 클래스를 부모 클래스로 가진다.



예외 상황

예외상황이 발생하면 프로그램은 뒤의 코드를 실행 시키지 않고 중단되며 콘솔 창에 오류의 형태와 상황에 대한 정보를 출력한다.

이같이 간단한 예외상황으로 인해 프로그램 전체가 멈추는 상황이 발생 할 수 있기 때문에 이런 상황을 방지하기 위해서 예외처리를 해야 한다.

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

```
public class ExceptionTest {
  public static void main(String[] args) {

    int x;
    x = 3/0;  <-- 오류 발생

    System.out.println("프로그램 끝");
  }
}
```

9.2 try, catch, throws

try구문 안에서 예외가 발생하지 않는다면 catch 구문 안의 코드는 실행되지 않지만 try구문 안에서 catch의 예외 종류에 맞는 예외가 발생하면 catch 구문 안의 코드는 실행된다.

try 문 뒤에 꼭 catch 문이 선언되어야 한다.

```
try {
  (예외 검사)
} catch( 예외 종류 ) {
  (예외 상황 처리)
}
```

예외 상황에서의 예외처리

```
try {
  x = 3/0;
  System.out.println("나눗셈 계산");
} catch(ArithmeticException e) {
  System.out.println("오류 발생");
}
```

오류가 발생하여 실행되지 않는 부분

결과

```
오류 발생
프로그램 끝
```

예외가 없을 때의 예외처리

예외를 발생하지 않는 상황에서는 catch 구문은 실행되지 않는다.

```
try {
  x = 3/10;
  System.out.println("나눗셈 계산");
} catch(ArithmeticException e) {
  System.out.println("오류 발생");
}
```

오류가 발생하지않았기 때문에 실행되지 않음

결과

```
나눗셈 계산
프로그램 끝
```

throws 가 선언 된 메소드

throws 가 선언 된 메소드는 throw 키워드를 이용하여 예외를 던져줘야 하며 메소드는 try 구문 안에서만 선언 할 수 있다.

```
try {
  bad.badCode(true);
  System.out.println("프로그램 실행");
}catch (Exception e){
  System.out.println("오류 발생");
}
```

```
class BadClass {
  public  void badCode(boolean bool ) throws Exception{
    if (bool){
     throw new Exception();
    }
```

예외를 던지도록 선언함

예외를 생성하여 던짐

9.3 다중 예외 처리하기

하나의 메소드에서 여러 개의 예외를 던질 수 있다.
하지만 catch 구문에서는 하나의 예외만 잡아야 한다.

```
try {
  bad.badCode(true);
  System.out.println("프로그램 실행");
}catch (Exception e){
  System.out.println("Exception 오류 발생");
}catch (IOException e){
  System.out.println("IOException 오류 발생");
}
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Unreachable catch block for IOException. It is already handled by the
catch block for Exception
```

다중 예외 처리하기 문제 해결

여러 개의 예외를 처리할 때 범위가 작은 것부터 큰 것으로 나열해야 한다.

```
  try {
    bad.badCode(true);
    System.out.println("프로그램 실행");
  }catch (IOException e){
    System.out.println("IOException 오류 발생");
  }catch (Exception e){
     System.out.println("Exception 오류 발생");
  }
}
```

결과

```
Exception 오류 발생
```

9.4 예외를 처리하는 방법

예외를 처리할 때 예외가 발생하거나 안하거나 꼭 실행을 해야 부분이 있을 수 있다.

이런 상황을 위해서 자바는 finally 키워드를 이용하여 무조건 실행할 내용을 지정 할 수 있다.

```
try {
  bad.badCode(true);
System.out.println("프로그램 실행");
}catch (Exception e){
  System.out.println("오류 발생");
}finally {
  System.out.println("프로그램 종료");
}
```

결과

```
오류 발생
프로그램 종료
```

결과

```
프로그램 실행
프로그램 종료
```

예외 던지고 받기

자바에서 예외는 꼭 처리하거나 다른 곳으로 던져야 한다.

예외를 처리하는 방법은 try~catch 구문을 이용하여 예외를 검사하고 처리하는 것을 말하며 예외를 다른 곳으로 던지는 것은 throws 키워드를 이용하여 예외를 다른 곳으로 보낸다.

```
try {
   System.out.println(divide());
}catch(NumberFormatException e){
  System.out.println("숫자를 입력해 주세요.");
}catch(ArithmeticException e){
  System.out.println("분모가 0 입니다.");
}
```

```
public int divide() throws ArithmeticException {
```