Chapter 18

예외 처리

1.예외처리방법1
2.예외처리방법2
3.예외 떠넘기기
4.사용자 정의 예외 만들기

예외란?

예외에는 컴파일러 체크 예외와 실행 예외(Runtime Exception)가 있습니다.

- 컴파일러 체크 예외는 자바 소스를 컴파일하는 과정에서 예외 처리 코드를 검사하여 예외 처리 코드가 없다면 컴파일 오류가 발생합니다.
- 실행 예외는 컴파일하는 과정에서 예외처리 코드를 검사하지 않는 예외를 말합니다.

예외처리
- 에러에 대한 처리를 의미한다.
- 자바는 예외처리 메커니즘을 제공한다.
- 프로그램에서 문제가 될만한 부분을 예상하여 사전에 "문제가 발생하면 이렇게 처리하라" 라고 프로그래밍 하는 것을 예외 처리라고 합니다

```
public class ArrayIndex {

    public static void main(String[] args) {

        int[] arr = {3, 6, 9};
        System.out.println(arr[3]);

    }

}
```

예외가 발생하면
문제가 발생한 곳에 대한 정보 출력과
프로그램이 종료된다.

```
<terminated> ArrayIndex [Java Application] C:₩Program Files₩Java₩jre1.8.0_201₩bin₩javaw.exe (2019. 8. 4. 오후 10:36:55)
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
        at exception.runtime.ArrayIndex.main(ArrayIndex.java:8)
```

대표적인 실행 예외의 종류

주요 실행 예외

- 1. NullPointerException
- 객체 참조가 없는 상태, 즉 null 값을 갖는 참조 변수로 객체 접근 연산자인 dot(.)를 사용했을 때 발생합니다.

- 2. ArrayIndexOutOfBoundsException
- 배열에서 인덱스 범위를 초과하여 사용할 경우 발생합니다.

- 3. NumberFormatException
- 문자열로 되어 있는 데이터를 숫자로 변경하는 경우에 발생합니다.

- 4. ClassCastException
- 형 변환은 부모 클래스와 자식 클래스간에 발생하고 구현 클래스와 인터페이스 간에도 발생합니다. 이러한 관계가 아니라면 다른 클래스로 타입을 변환할 수 없습니다.

예외의 종류 까진 외울 필요는 없다.
하지만 이런 실행예외들은 개발자의 경험에 의해서 예외 처리 코드를 삽입해야 한다

예외처리 방법1 (try~catch~finally)

* try~ catch~ finally

- - try 블록에는 예외 발생 가능성이 있는 코드를 작성합니다. try 블록의 코드가 예외 발생 없이 정상 실행되면 catch 블록은 실행되지 않습니다.
- - try 내부에서 예외가 발생하면 즉시 실행을 멈추고 catch 블록으로 이동하여 예외 처리 코드를 실행합니다.
- 예외 발생 여부와 상관없이 항상 실행할 내용이 있다면 finally 블록 내부에 실행 내용을 작성합니다.

```
try {
   코드 작성 영역...
} catch(Exception e) {
   처리 영역...
}
```

```
try {
   코드 작성 영역...
} catch(Exception e) {
   처리 영역...
} finally {
   무조건 실행
}
```

다중 catch(둘 이상의 예외 처리)

* 다중 catch

- try 블록 내부는 다양한 종류의 예외가 발생할 수 있습니다. 예외가 여러 가지 발생한다면 다중 catch 블록을 작성하여 예외들을 처리합니다.

- catch 블록은 위에서부터 차례대로 검색되므로 상위 예외 클래스의 catch 블록이 위에 있다면 하위 예외 클래스의 catch블록은 실행되지 않습니다.

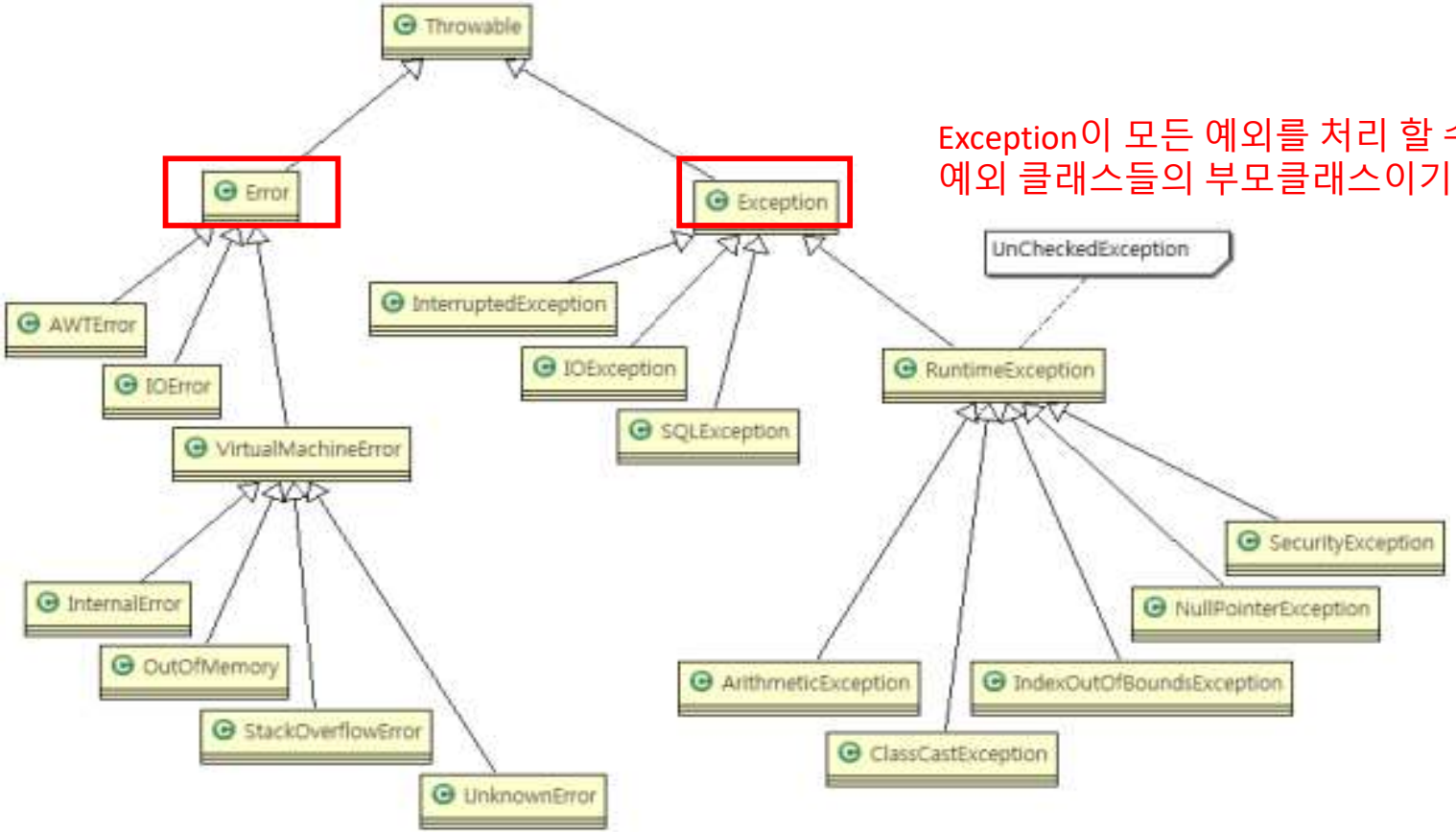
- catch() 괄호 안에 동일하게 처리하고 싶은 예외를 | 로 연결하면 됩니다. 이 방식을 사용할 때는 두 예외가 상속 관계가 있으면 안됩니다.

```
try {
    코드 작성 영역...
} catch(NumberFormatException e) {
    처리 영역...
} catch(ClassCastException e) {
    처리 영역...
} catch(Exception e) {
    처리 영역...
}
```

주의할 점
상위 예외 클래스가 하위 예외 클래스보다 아래쪽에 위치해야 합니다.

예외 처리 클래스도 상속 관계를 가진다

Error 클래스를 상속하는 예외 클래스의 예외 상황은 시스템 오류 수준의 예외 상황으로 프로그램 내에서 처리 할 수 있는 수준의 예외가 아니다. (심각한 에러)



Exception이 모든 예외를 처리 할 수 있는 이유는 예외 클래스들의 부모클래스이기 때문이다

예외처리 방법1 (예외 떠넘기기)

* throws

- try ~catch 구문이 예외가 발생했을 때 직접 해결을 하고자 하는 코드라면 throws는 메서드나 생성자를 호출한 곳으로 예외를 떠넘기는 코드입니다.

- 즉 예외처리를 직접 수행하지 않고 메서드 호출자에게 예외를 던지는 방법입니다.

- throws 키워드가 붙어있는 메서드는 반드시 try 블록 내부에서 호출되어야 합니다. 그리고 catch블록에서 떠넘겨 받은 예외를 처리해야 합니다.

- main도 thorws를 사용할 수 있습니다

호출부에서는 catch블록으로 예외를 반드시 처리해야 한다

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

    try {
        greet(3);
    } catch (Exception e) {
        System.out.println("배열의 참조범위를 벗어났습니다.");
    }
}
```

```
public class ThrowsExample1 {

    public static String[] greetings = {"안녕", "헬로", "니하오"};

    public static void greet(int index) throws Exception {
        System.out.println(greetings[index]);
    }
}
```

메소드의 throws절 선언을 통해 예외의 처리를 넘길 수 있다.

예외 강제 발생 시키기

- 사용자가 직접 선언한 예외 클래스나 자바가 제공하는 예외 API에서 예외를 강제 발생시키려면 throw라는 키워드를 이용합니다.

- 예외를 강제 발생시키며 메서드를 강제 종료 합니다

잘못된 값이 전달되면 반드시 강제 종료해야 하는 예

```
public static int calcSum(int n) throws Exception {

	if(n <= 0) {
		throw new Exception("매개값을 반드시 양수로 전달하세요.");
	}

	int sum = 0;
	for(int i=1; i<=n; i++) {
		sum += i;
	}

	return sum;
}
```

throw 구문은 강제 예외를 발생시키며 메서드를 종료한다
예외 던지기 구문으로 함께 처리한다

사용자 정의 예외

* 사용자 정의 예외

- 프로그램을 개발하다보면 자바 표준 API에서 제공하는 예외 클래스만으로 다양한 종류의 예외를 표현할 수 없습니다.

- 개발자가 만든 어플리케이션에서 자체적으로 생길 수 있는 예외는 개발자가 직접 예외 클래스를 정의해서 만들어야 합니다.

- 사용자 정의 예외 클래스는 Exception 클래스를 상속하여 사용하면 됩니다.
- 사용자 정의 예외 클래스의 이름은 Exception으로 끝나는 것이 좋습니다.

```
public class MyException extends Exception {

      public MyException() {}

      public MyException(String message) {
            super(message);
      }
      ……
}
```

일반적으로 기본생성자와
예외 메시지를 받는 생성자를 만듭니다

Chapter 18

수고하셨습니다