제 27 강 : 예외처리

※ 학습목표
✓ 예외처리의 개념을 설명할 수 있다.
✓ throw를 이용하여 예외를 발생 시킬 수 있다.
✓ throws를 이용하여 예외전가를 수행할 수 있다.
✓ try~catch~finally를 이용하여 예외처리를 수행할 수 있다.

1. 프로그램 오류
✓ 프로그램이 실행 중 어떤 원인에 의해서 오작동을 하거나 비정상적으로 종료되는 경우 오류라고 함
✓ 컴파일 에러 : 컴파일 시 발생하는 에러
✓ 런타임 에러 : 실행도중에 발생하는 에러
✓ 컴파일러는 문법적인 오류만 인식할 수 있음
✓ 자바에서는 런타임 에러를 에러(Error)와 예외(Exception)로 구분함

오류구분	설 명
예외(Exception)	프로그램 코드에 의해서 수습될 수 있는 다소 미약한 오류
오류(Error)	프로그램 코드에 의해서 수습될 수 없는 심각한 오류

2. 예외가 일어나는 상황
✓ 정수를 0으로 나누는 경우
✓ 배열의 index값이 음수 값을 가지거나, 크기를 벗어나는 경우
✓ 부적절한 형변환
✓ 입출력시 interrupt가 나타나는 경우
✓ 입출력하기 위한 파일이 존재하지 않는 경우
✓ 메서드 호출시

3. 예외처리 목적

✓ 예외의 발생으로 인한 실행 중인 프로그램의 갑작스런 비정상 종료를 막고, 정상적인 실행상태를 유지할 수 있도록 하는 것

4. 예외처리 구문

```
try{
        //예외가 발생할 가능성이 있는 코드(문장)
}catch(예외타입1 매개변수1 ){
        //예외발생시 처리할 코드 (예외가 발생할 때만 실행된다.)
}catch(예외타입2 매개변수2 ){
        //예외발생시 처리할 코드
} finally{
        //예외에 상관없이 실행 할 코드
}
```

5. 예외처리 실습

① 예외가 발생하는 경우

✓ 문법상의 오류가 없기 때문에 컴파일 시 Error를 발생하지 않음

✓ 실행도중 정수를 0으로 나눌 경우가 생길 수 있고 Exception 발생

```
1  package tommy.java.exam01;
2
3  public class Exception1 {
4          public static void main(String args[]) {
5                  int number = 50;
6                  int result = 0;
7                  for (int i = 0; i < 10; i++) {
8                          result = number / (int) (Math.random() * 5);
9                          System.out.println(result);
10                 }
11         }
12 }
```

② 예외를 처리한 경우

```
1  package tommy.java.exam02;
2
3  public class Exception2 {
4          public static void main(String args[]) {
5                  int number = 50;
6                  int result = 0;
7                  for (int i = 0; i < 10; i++) {
8                          try {
9                                  result = number / (int) (Math.random() * 5);
10                                 System.out.println(result);
11                         } catch (ArithmeticException e) {
12                                 // ArithmeticException이 발생하면 수행된다.
13                                 System.out.println("Exception 발생");
14                         }
15                 }
16         }
17 }
```

③ 단언과 예외(Exception)의 차이점은 무엇일까?

✓ 예외(Exception)는 특정한 코드에서 예외가 발생하므로 일어나는 비정상적인 프로그램 종료와 같은 1차적인 손실을 막고 예외에 대한 처리로 인해 프로그램의 신뢰성을 높이는 것이다.

✓ 하지만 단언은 어떤 결과를 위해 특정 코드나 변수의 값을 프로그래머가 예상하는 값이어야 하는 것을 검증하는 것에 차이가 있다

④ 예외처리 구문의 실행 순서 1

```
1  package tommy.java.exam03;
2
3  public class Exception3 {
4          public static void main(String args[]) {
5                  System.out.println(1);
6                  System.out.println(2);
7                  try {
8                          System.out.println(3);
9                          System.out.println(4);
10                 } catch (Exception e) {
11                         System.out.println(5);
12                 }
13                 System.out.println(6);
14         }
15 }
```

⑤ 예외처리 구문의 실행 순서 2

```
1  package tommy.java.exam04;
2
3  public class Exception4 {
4          public static void main(String args[]) {
5                  System.out.println(1);
6                  System.out.println(2);
7                  try {
8                          System.out.println(3);
9                          // ArithmeticException을 발생시킨다.
10                         System.out.println(0 / 0);
11                         System.out.println(4); // 실행되지 않는다.
12                 } catch (ArithmeticException ae) {
13                         System.out.println(5);
14                 } // try-catch의 끝
15                 System.out.println(6);
16         }
17 }
```

⑥ 참고 : Runtime 익셉션과 그 외의 익셉션

✓ runtime Exception은 컴파일 시 Error를 발생하지 않음

✓ 그 외 Exception은 컴파일 시 Error를 발생함

6. 예외발생

✓ throw 예약어

✓ throw new 발생시킬 예외객체 생성자

[실습]

```
1  package tommy.java.exam05;
2
3  public class Exception5 {
4          public static void main(String[] ar) {
5                  // Exception을 강제로 발생시킨다.
6                  throw new ArrayIndexOutOfBoundsException();
7          }
8  }
```

[실습] 예외처리 순서 : 다중 catch문

```
1  package tommy.java.exam06;
2
3  public class Exception6 {
4          public static void main(String args[]) {
5                  System.out.println(1);
6                  System.out.println(2);
7                  try {
8                          System.out.println(3);
9                          // ArithmeticException을 발생시킨다.
10                         System.out.println(0 / 0);
11                         System.out.println(4);// 실행되지 않는다.
12                 } catch (ArithmeticException ae) {
13                         if (ae instanceof ArithmeticException)
14                                 System.out.println("true");
15                         System.out.println("ArithmeticException");
16                 } catch (Exception e) {
17                         System.out.println("Exception");
18                 }
19                 System.out.println(6);
20         }
21 }
```

[실습] : finally를 포함한 완전한 예외처리 예제

```
1  package tommy.java.exam07;
2
3  public class Exception7 {
4          int[] array;
5
6          public Exception7() {
7                  array = new int[3];// 속성(멤버필드) 초기화
8          }
9
10         public void program() {
11                 for (int i = 0; i <= array.length; i++) {
12                         System.out.println("for문의 시작 " + i + "번째");
13                         try {
14                                 System.out.println(array[i]);
15                         } catch (Exception e) {
16                                 System.out.println("Exception 발생" + e);
17                                 return;
18                         } finally {
19                                 System.out.println("finally 영역");
20                         }
```

```
21                                  System.out.println("for문의 끝" + i + "번째");
22                          }
23                  }
24
25                  public static void main(String[] args) {
26                          Exception7 ref = new Exception7();
27                          ref.program();
28                          System.out.println("프로그램 끝!");
29                  }
30  }
```

7. Exception 클래스의 주요 메서드

메서드명	설 명
printStackTrace()	예외 발생 당시의 호출스택(Call Stack)에 있었던 메서드의 정보와 예외 메시지를 화면에 출력
getMessage()	발생한 예외클래스의 인스턴스에 저장된 예외 메시지를 얻을 수 있음

8. 예외전가

✓ throws 예약어

[접근제한] 반환형 메서드명([인자1, 인자2...]) throws 예외클래스1, 예외클래스2....

[실습]

```
1   package tommy.java.exam08;
2
3   public class Exception8 {
4           private static void test() throws Exception {
5                   System.out.println(6 / 0);
6           }
7           public static void main(String[] args) {
8                   try {
9                           test();
10                  } catch (Exception e) {
11                          System.out.println("예외 발생");
12                  }
13          }
14  }
```

9. 사용자 정의 예외

✓ 사용자 정의 Exception이 필요한 이유는 표준예외가 발생할 때 예외에 대한 정보를 변경하거나 정보를 수정하고자 한다면 사용자가 직접 작성하여 보안된 예외를 발생시켜 원하는 결과를 얻는데 있다.

✓ 사용자 정의 Exception을 작성하기 위해서는 Throwable을 받지 않고 그 하위에 있으면서 보다 많은 기능들로 확장되어 있는 Exception으로부터 상속을 받는 것이 유용하다. 물론 입/출력에 관련된 예외를 작성하기 위해 IOException으로부터 상속을 받는 것이 보편적인 것이다