20. 예외처리(exception handling)

• **예외**(exception)란? – Roberts W. Sebesta 제6판에 **정의**된 것을 먼저 소개한다.

> **예외는 소프트웨어 또는 하드웨어에 의해 감지될 수 있고,**
> **특별한 처리가 요구될 수도 있고 아닐 수도 있는 독특한 사건(event)이다.**

// 먼저, 오류가 있는 원시코드를 예외처리한 JAVA 프로그램을 살펴본다.

```
public class Test
{
    public static void main(String args[])
    {
        int a, b, c;
        try
        {
            int k[] = new int[-100];          //배열크기를 음수로 잘못 설정한 경우(오류)
            k[-30] = 7;

            a = 5; b = 0;
            c = a / b;                        //수식을 0으로 나눔(오류)
            System.out.println(c);
        }
        catch(ArithmeticException exp){
            System.out.println("0으로 나누는 오류");
        }
        catch(NegativeArraySizeException exp){
            System.out.println("배열크기가 음수로 설정됨");
        }
        finally{
            System.out.println("예외 발생과 상관없이 실행되는 블록");
        }
        System.out.println("try 블록과는 무관함");
    }
}
```

[실행결과]
배열크기가 음수로 설정됨
예외 발생과 상관없이 실행되는 블록
try 블록과는 무관함

다음은 자바의 예외처리 구문이다.

```
try
{
    예외가 발생될 수 있는 문장을 기술
}
catch(예외타입1 exp1)
{
    try 블록에서 발생될 수 있는 예외를 처리함
}
catch(예외타입2 exp2)
{
    try 블록에서 발생될 수 있는 예외를 처리함
}
  :
finally
{
    try 블록에서 할당된 자원 등을 해제함
}
```

① try 블록

- try 블록은 예외가 발생될 수 있는 하나 이상의 문장을 묶은 블록이다.
- 통상적으로, **예외가 발생될 수 있는 하나 이상의 문장을 기술**한다.

② catch 블록

- 발생된 **예외를 말 그대로 잡아내는(catch) 블록**이다.
- catch 블록은 try 블록 다음에 0개 이상 기술할 수 있다.
- 즉, catch 블록 없이 finally 블록이 올 수 있다.
- 각각의 catch 블록은 서로 다른 예외처리를 담당한다.(예외 클래스 변수를 이용함)

③ finally 블록

- 예외 발생 여부와는 상관없이 컴파일러는 finally 블록을 **무조건 수행**한다.
- try 또는 catch 블록 다음에 하나의 finally 블록이 올 수 있다.(생략 가능)
- finally 블록은 try 블록에서 할당받은 자원을 해제해야 하는 경우 등에 사용한다.

[예제 1] 수식에서 0으로 나누는 경우

```
class Test
{
  static int y;
  public static void main(String args[])
  {
    int a = 1, b = 0;
    try{ y = a / b; }                                  //수식에서 0으로 나누는 오류 발생
    catch(ArithmeticException e){ y = 100; }           //수식에서 0으로 나누는 예외처리
    catch(NegativeArraySizeException e){ y = 200; }
    catch(IndexOutOfBoundsException e){ y = 300; }
    finally{ System.out.print(y); }                    //무조건 실행되는 곳(y = 100)
  }
}
//출력 : 100
```

- 실행 도중에 예외(오류)가 발생하면, 제어흐름은 catch 블록으로 이동된다.
- 제어흐름은 catch 블록 중에서 관련된 **예외처리 클래스**가 명시된 곳으로 이동된다.

◈ **자바의 예외처리 클래스**

- ArithmeticException : 수식에서 0으로 나누는 경우
- NegativeArraySizeException : 배열 크기가 음수로 설정된 경우
- IndexOutOfBoundsException : 배열, 문자열 등에서 인덱스 값이 벗어난 경우

◈ **catch 블록이 실행되는 2가지 경우(매우 중요!)**

- 실행 도중에 예외(오류)가 발생하였을 때
- throw문으로 예외 객체를 던질 때

◈ **자바의 예외처리에서 throw문**

- throw는 분기문의 한 종류이다.
- 해서, throw를 만나면 제어흐름이 이동된다.
- throw를 만나면 catch 블록의 어느 하나가 실행된다. 관련된 catch 블록이 실행된다.
- 즉, throw를 만나면 관련된 catch 블록의 예외가 처리된다.

[예제 2] 자바 예외처리에서 throw문

```
public class Test extends Exception      //Exception은 자바에서 제공하는 예외처리 클래스
{
  public static void main(String args[])
  {
    int x = 0;
    try
    {
      x = x + 1;                         //x = 1
      if(x<100) throw new Test();        //조건이 참이므로 예외 객체 Test를 던진다.
      x = x + 2;                         //실행되지 않음
    }
    catch(Test e){ x = x + 3; }          //예외 객체 Test를 받아 처리함(x = 1 + 3 = 4)
    catch(ArithmeticException e){ x = x + 4; } //수식에서 0으로 나누는 예외처리
    finally{ x = x + 5; }                //무조건 실행되는 곳(x = 4 + 5 = 9)
    System.out.println(x);               //출력 9 (x = 9)
  }
}
```

◈ 자바에서 사용자 정의 예외

- 자바에서 사용자 정의 예외를 만들려면,
- 먼저 자바에서 제공하는 예외 클래스 중에 어느 하나를 상속받아야 한다.
- 의미가 가까운 예외 클래스를 하나 선택한다.(관례)
- public class Test extends Exception에서 Test는 사용자 정의 예외 클래스가 된다.
- Exception은 자바에서 제공하는 예외처리 클래스이다.
- Exception을 상속받아서 사용자 정의 예외 클래스를 만들 수 있다.

◈ 자바 예외처리에서 throw문

- throw는 분기문의 한 종류이다.
- 해서, throw를 만나면 제어흐름이 이동된다.
- throw를 만나면 catch 블록의 어느 하나가 실행된다. 관련된 catch 블록이 실행된다.
- 즉, throw를 만나면 관련된 catch 블록의 예외가 처리된다.
- 오류가 발생되지 않았지만, throw문과 관련된 catch 블록의 예외가 처리된다.

기출문제 분석

1. 다음 중 자바의 예외처리에 대한 설명으로 옳은 것은? [2016년 국회 9급]

① throw는 처리가 정상적으로 이루어졌음을 알리는 명령이다.
② 예외가 발생할 가능성이 있는 문장은 try 블록에 넣는다.
③ catch 블록에는 예외 발생 여부에 상관없이 실행할 문장을 넣는다.
④ 발생한 예외에 대한 처리는 finally 블록에서 할 수 있다.
⑤ 예외가 발생하였으나 예외처리가 이루어지지 않은 경우 그 예외는 무시되고, 프로그램은 예외 발생 이후 지점부터 계속된다.

♤ 자바의 예외처리

```
int a = 1, b = 0, y;
try{ y = a / b; }                          //예외 발생 가능성이 있는 문장은 try 블록에 기술
catch(ArithmeticException e){ y = 1; }     //예외가 발생된 경우에 실행되는 곳
catch(NegativeArraySizeException e){ y = 2; }
catch(IndexOutOfBoundsException e){ y = 3; }
finally{ System.out.print(123); }          //무조건 실행되는 곳
```

정답 : ②

2. JAVA 언어의 예외처리에 대한 설명이다. 틀린 것은? [2004년 서울 9급]

① try-catch-finally 블록을 사용하여 처리된다.
② 한 개의 try 블록에 여러 개의 catch 블록이 존재할 수 있다.
③ finally 블록은 try 블록의 코드가 실행되면서 예외가 발생하더라도 정상적으로 수행한다.
④ 여러 종류의 예외처리를 위해서는 여러 개의 try-catch-finally 블록을 작성해야 한다.
⑤ 실행 도중에 throw를 만나면 catch 블록의 어느 하나가 실행된다.

♤ 자바의 예외처리

• 여러 종류의 예외처리를 위해서는 여러 개의 try-catch-finally 블록을 작성해야 한다.(×)
→ 여러 종류의 예외처리도 하나의 try-catch-finally 블록으로 가능하다.

정답 : ④

3. 다음 Java 프로그램의 실행 결과로 옳은 것은? [2016년 계리 9급]

```
class Division{
  public static void main(String[] args){
    int a, b, result;
    a = 3;
    b = 0;
    try{
      result = a / b;
      System.out.print("A");
    }
    catch(ArithmeticException e){ System.out.print("B"); }
    finally {  System.out.print("C");  }
    System.out.print("D");
  }
}
```

① ACD ② BCD
③ ABCD ④ BACD

♤ Java 프로그램에서 예외처리

```
class Division{
  public static void main(String[] args){
    int a, b, result;
    a = 3;
    b = 0;
    try{
      result = a / b;            //b = 0이므로, 0으로 나누는 오류 발생
      System.out.print("A");     //이 곳은 실행되지 않는다. 오류가 발생했으므로
    }
    catch(ArithmeticException e){ System.out.print("B"); }     //0으로 나누는 예외처리 B 출력
    finally {  System.out.print("C");  }     //예외처리에서 finally 부분은 무조건 실행된다. C 출력
    System.out.print("D");                   //예외처리 후에 실행, D 출력
  }
}
```

정답 : ②

4. 다음 Java 언어로 작성한 프로그램의 실행 결과는? [2017년 법무부 9급]

```
public class Test
{
  public static void main(String[] args)
  {
    int ar[] = {10, 20, 30, 40, 50};
    int sum = 0, a = 100, b = 0;
    try
    {
      for(int i = 0; i < ar.length; i++) { sum += ar[i]; }
      System.out.println(sum);
    }
    catch (ArrayIndexOutOfBoundsException e)
    {
      System.out.println("Array Index Out Of Bounds Exception");
    }
    try
    {
      float z = a / b;
      System.out.println(z);
    }
    catch (ArithmeticException e) { System.out.println("Arithmetic Exception"); }
  }
}
```

① 100
0.0
③ 150
Arithmetic Exception

② 100
Array Index Out Of Bounds Exception
④ 150
/ by zero at Test.main(Test.java:14)

♤ **예외처리**

• sum = 10 + 20 + 30 + 40 + 50 = **150** → 150 출력
• z = a / b = 100 / 0 → 0으로 나누는 오류 발생, **Arithmetic Exception** 출력

정답 : ③