# Exception Ⅱ - Make Exception Class

## Exception Class의 정의와 throw

- 나이를 입력하라고 했더니, -20살을 입력했다.
- 이름 정보를 입력하라고 했더니, 나이 정보를 입력했다.
  (Code Level의 ERROR가 아닌 논리흐름 상의 Exception)

- JVM이 감지할 수 없는 Exception은 개발자가 직접 JVM에게 Exception 메커니즘을 구동시키라고 알려줘야 한다.
  이 때 사용하는 Keyword가 throw이다.

### i. Exception Class의 정의

```
// 1. Exception Class를 상속만 받으면 Sub Class는 Exception Class가 될 수 있다.
class AgeInputException extends Exception {

     public AgeInputException(){
          // 부모 Class의 생성자 호출, 이 때 인자로 전달되는 문자열은 getMessage() Method 호출 시 반환되는 문자열
          super("유효하지 않은 나이가 입력되었습니다."); // Exception의 원인
     }

     /*
      *  2. catch(){ }문의 중괄호 안의 로직을 Exception Class 내에 구현하면 된다.
      */
}
```

### ii. 프로그래머 정의 Exception Class의 핸들링

```
class ExceptionHandling {

     public static void main(String[] args){
          System.out.print("나이를 입력하세요 : ");

          // 4. readAge에서 넘어온 Exception의 instance를 try~catch문에서 처리
          try{
               int age = readAge();
               System.out.println("당신의 나이는 "+age+"세입니다.");
          } catch(AgeInputException e){
               System.out.println(e.getMessage());
          }
```

```
        }

                                // throws : 예외발생 지점에서 내가 예외처리를 하지 않고, 나를 호출한 영역으로 Exception을 던지겠다.
        public static int readAge() throws AgeInputException {
            Scanner scan = new Scanner(System.in);
            int age = scan.nextInt;

            // 1. 예외발생!
            if(age < 0){
                // 2. 정의한 Exception Class의 instance 생성
                AgeInputException except = new AgeInputException();

                // 3.throw keyword로 JVM에게 Excetion의 instance를 던지면서 예외처리 메커니즘을 구동.
                throw except;
```

그런데 예외발생 영역(readAge Method 내부)에 try~catch문이 없다, 그러면 JVM이 instance를 어디에 전달할까??
메소드 안에 try~catch문이 있었다면 바로 처리 했지만, 없으므로 메소드를 호출한 영역으로 Exception의 Instance가 넘어간다.

```
            }

            return age;
        }
    }
```

---

**예외처리를 하지 않으면?**

```
class ExceptionHandling {
                                        // 내가 처리하지 않고 Main Method를 호출한 JVM으로 던진다.
    public static void main(String[] args) throws AgeInputException {
        System.out.print("나이를 입력하세요 : ");

        int age = readAge();
        System.out.println("당신의 나이는 "+age+"세입니다.");
    }

                        // 내가 처리하지 않고 main Method로 던진다.
    public static int readAge() throws AgeInputException {
        Scanner scan = new Scanner(System.in);
        int age = scan.nextInt;

        if(age < 0){
            AgeInputException except = new AgeInputException();
            throw except;
```

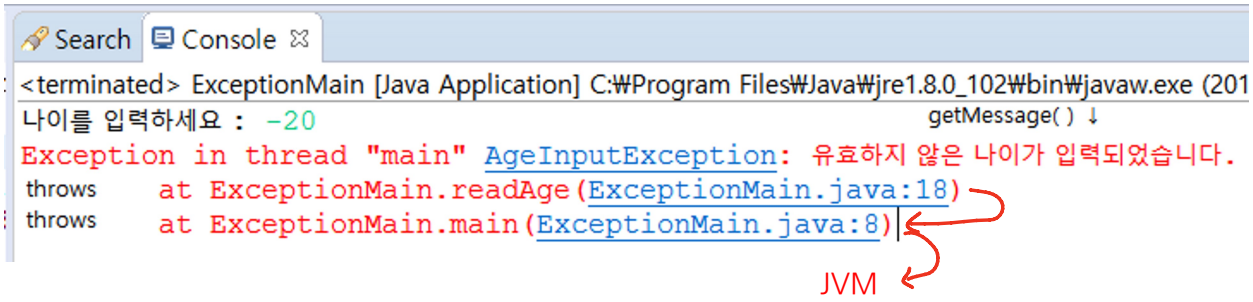
```
        }

        return age;
    }
}

class AgeInputException extends Exception {
    public AgeInputException(){
        super("유효하지 않은 나이가 입력되었습니다.");
    }
}
```

---

## JVM이 직접 Exception을 처리하는 과정

i. JVM이 Exception Instance를 최종적으로 전달 받으면 getMessage( ) Method를 호출해서 반환되는 문자열 출력

ii. JVM이 printStackTrace( ) Method를 호출하여 예외상황이 발생되서 Exception의 Instance가 전달(throws)되었던 모든 과정을 출력해준다.




( main Method를 실행한 주체는 JVM이므로 JVM에게 Exception이 최종전달 )

iii. 그 후 프로그램 종료

---