제 29 강

예외처리

교재:p162~167

목차

1. 예외/예외처리의 개념

1) 예외(Exception)란?

: 프로그램 실행 중 발생하는 오류 중에서 처리가 가능한 것을 의미

- 에러: 개발자가 조치를 취할 수 없는 수준 ex) 메모리 부족, JVM 동작 이상
- 컴파일 에러: 컴파일 시 발생하는 에러 ex)오타, 잘못된 자료형 등
- 런타임 에러: 프로그램 실행 도중에 발생하는 에러
- 로직 에러: 실행은 되지만 의도와는 다르게 동작하는 에러
- 예외: 다른 방식으로 처리 가능한 오류 ex) 입력값 오류, 네트워크 문제

1. 예외/예외처리의 개념

<실습> Exam-60.java

예외에 대한 간단한 예시 코드 실습

```
public class Exception1 {
    public static void main(String[] args) {
        int a=0;
        int b=2;
        int c= b/a;
    }
}
```

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at day29.Exception1.main(Exception1.java:7)
```

ArithmeticException: 산술연산 관련 오류

: 0으로 나눌 수 없기 때문에 발생하는 오류

1. 예외/예외처리의 개념

1) 예외처리(Exception Handling)란?

: 예외가 발생했을 때 이를 적절히 처리하여 프로그램이 비정상적으로 종료되는 것을 막는 방법

예시>

1. 입력 오류?

숫자 입력 해야 하는 곳에 문자 입력할 경우를 방지하기 위해

문자 입력 시 숫자를 입력해야 한다고 사용자에게 알려줌

2. try-catch-finally

try -catch-finally

: 예외발생 시 , 적절하게 처리하기 위해 자바에서 제공하는 예외처리 문법

[try -catch문]

```
try{
        예외가 발생할 수 있는 명령;
}
catch(발생할 수 있는 예외 클래스명){
        예외 발생 시 실행할 명령;
}
```

2. try-catch-finally

<실습> Exam-61.java

예외 발생 시 처리하는 방법 실습

```
public class Exception2 {
    public static void main(String[] args) {
        try {
            int []a= {2,0};
            int b=4;
            int c=b/a[2];
        }
        catch(ArithmeticException e) {
        System.out.println("산술 오류 발생");
        }
        catch(ArrayIndexOutOfBoundsException e) {
            System.out.println("배열 길이 오류 발생");
            }
        System.out.println("예외 처리 공부 중");
    }
}
```

<예외 클래스>
ArithmeticException: 산술연산 관련 오류
ArrayIndexOutofBoundsException: 배열 인덱스 접근 오류

<실행결과>

```
배열 길이 오류 발생
예외 처리 공부 중
```

int c= b/a[2]; 명령에서 오류가 발생한다.

a의 배열 크기는 2이므로, 인덱스는 [0],[1]만 가능한데
a[2]로 접근했으므로 배열의 인덱스 접근 오류

→ 여기서 배열에 접근 한 후 값을 가져와야 하는데
접근 시 오류가 났으므로 Index 예외가 발생한다.

2. try-catch-finally

finally 문

: try-catch문에 선택적으로 추가할 수 있는 문법으로, 오류가 발생하든 하지 않든 무조건 실행하는 구문

[finally 문]

```
try{
        예외가 발생할 수 있는 명령;
}
catch(발생할 수 있는 예외 클래스명){
                예외 발생 시 실행할 명령;
                                        }
finally{
        예외가 발생하든 안하든 무조건 실행하는 명령;
}
```

2. try-catch-finally

<실습> Exam-63.java

finally는 N/W 연결 시. 외부와의 연결을 종료하는 작업에 주로 사용

```
public class Finally {
    public static void main(String[] args) {
        int a=0;
        int b=2;
        try {
            System.out.println("외부로 접속");
        }
        catch(ArithmeticException e) {
        System.out.println("오류가 발생했습니다.");
        }
        finally {
            System.out.println("무조건 연결 해제");
        }
    }
}
```

<실행결과>

```
외부로 접속
무조건 연결 해제
```

2. try-catch-finally

<실습> Finally2.java

a,b를 입력받아 오류를 발생시키지 않아도 finally 구문 작동 확인하기

```java
import java.util.Scanner;
public class Finally2 {
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int a,b;
        System.out.print("a:");
        a=sc.nextInt();
        System.out.print("b:");
        b=sc.nextInt();
        try {
            int c=b/a;
            System.out.println("c:"+c);
        }
        catch(ArithmeticException e) {
        System.out.println("오류가 발생했습니다.");
        }
        finally {
            System.out.println("무조건 실행~");
        }
    }
}
```

<오류 발생 x 실행결과>

```
a:2
b:3
c:1
무조건 실행~
```

<오류 발생 실행결과>

```
a:0
b:2
오류가 발생했습니다.
무조건 실행~
```