2020.12.16 Park Seol Ah

14. 예외 처리

(1) 예외(Exception)란?

프로그램에서 문제가 발생될 만한 곳을 예상하여 사전에 프로그래밍을 해놓는 것으로, 예외가 발생하더라도 계속해서 프로그램이 작동되도록 하기 위하여 예외 처리를 한다.

(2) 예외처리 문법

1) **try / catch :**

try {
try블럭 ; Exception이 발생할 가능성이 있는 명령문들(문제가 발생할 수 있는 로직을 기술)

} catch(Exception타입 Exception변수) {
그 Exception을 처리하는 명령문(try블록안에서 문제가 발생했을 때 대처방안 기술);

} finally {
Exception 발생 여부와 상관없이 맨 마지막에 실행할 명령문;

2) **throws** : 나를 호출한(실행시키는) 쪽으로 예외를 '던져버리는' 방식

<ex>

```
private void actionA() throws ArrayIndexOutOfBoundsException {
        System.out.println("actionA 전반부");
        int[] arr = {0,1,2,3};
        System.out.println(arr[4]); // Exception 발생시, 호출한 곳으로 던짐
        System.out.println("actionA 후반부"); // 위에서 Exception 발생하면 이곳은 실행되지 않는다.
```

(3) 일반적으로 많이 보게 되는 예외들

1) **ArrayIndexOutOfBoundsException** : 배열 사용시 존재하지 않는 index값을 호출하면 발생한다.
2) **NullPointerException** : 존재하지 않는 객체를 가리킬 때 발생
3) **NumberFormatException** : 숫자로 변경할 수 없는 문자열을 변경하려고 할 때

(4) 강제로 예외 발생시키기

```
throw new Exception("에러메시지");
```

<ex> - 계좌에서 출금시, 잔액부족으로 인한 강제 예외 발생시키기

```
public void withdraw (int amount) throws Exception {
        if(balance>=amount) { // 잔액부족으로 강제 예외발생
        throw new Exception(amount+"원 출금하기에는 잔액이 부족합니다.");
```