예외(Exception) 처리

예외란?

문제 없을 것 같은 프로그램도 외부환경요인 등에 의해서 문제가 발생하곤 한다
프로그램에서 문제가 발생될 만한 곳을 예상하여 사전에
"문제가 발생하면 이렇게 해라" 라고
프로그래밍 하는 것을 예외처리 라고 한다.

예외처리의 필요성

프로그래머는 예외 처리를 왜 해야하나 ?
몇 달 동안 심혈을 기울여 만들어 놓은 프로젝트가 단순한 어떤 이유로
작동을 하지 않다면 프로그램의 프로세스가 완전 죽은 상태가 되버린다..
다시 재가동을 시키지 않으면 움직이지 않는 상태가 됨
예외의 필요성은 어느 한 부분에서 예외가 발생하더라도
계속해서 프로그램이 동작되도록 하는데 목적이 있다.
어떤 문제가 하나 발생되었다고 그 프로세스가 완전히 정지되어
시스템을 재가동하기 전까지는 구동을 아예 못한다고 한다면
너무 많은 손실이 클 것이다.
그래서 사소한 문제 같은 경우에는 그 문제를 우회해서 가는 방법으로
프로세스가 죽지 않고 계속 구동하도록 프로그래밍하는 방법이 필요하다.

예외처리 문법(try ~ catch)

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

}catch(익셉션타입 익셉션변수) {
    그 익셉션을 처리하는 명령문(try블록안에서 문제가 발생했을 때 대처방안 기술);

} finally {
    익셉션 발생 여부와 상관없이 맨 마지막에 실행할 명령문;
}
```

예외처리 문법(throws)

try ~ catch 예외 처리 방법은 예외가 발생했을 때
자체적으로 catch문을 이용해서 해결 했다.
이번에 살펴볼 throws의 경우에는 나를 호출한(실행시키는) 쪽으로 예외를 던져버리는 방식

※ DB 관련 Exception

ClassNotFoundException : 드라이브 이름을 찾지 못했을 때
SQLException : db url, id, pw가 올바르지 않을 때

예외처리는 보험이다.

예외가 발생 했을 때 프로세스가 죽지 않게 예외처리를 자체적으로 처리할 수도 있고(try~catch),
내가 처리 안하고 나를 부른데서 알아서 할테니 걍 던져 버릴 수도 있다(throws)