Network

강사 이상훈
학생고동영

예외(Exception)

예외는 error 의 일종 , 프로그램 수행 시 또는 컴파일시에 불능상태를 만들어 버림

예외 처리

Exception 예외가 발생할 것을 대비하여 미리 예측해 이를 소스상에서 제어하고 처리하도록 만드는것

정리

예외란 error에 일종이며 , 발생시 시스템 및 프로그램을 불능상태로만듬

하지만 이를 막기 위해 예외 처리를 통해 , 시스템 및 프로그램 정상실행 상태로 유지하도록 함

1) Try - Catch - Finally

Try 블록 : 실제 코드가 들어가는 곳으로써 예외 Exeption이 발생할 가능성이 있는 코드

Catch 블록 : Try 블록에서 Exeption이 발생하면 코드 실행 순서가 Catch 쪽으로 오게됨. 즉 예외에 대한 후 처리 코드

Finally 블록 : Try 블록에서의 Exeption과 발생 유무와 상관 없이 무조건 수행되는 코드 (옵션이라 생략이 가능

예외 떠넘기기

```
public static void main(String[] args)
        throws IOException, InterruptedException {  // main 메서드에서는 IOException 이 발생할 수 있도 있는
                                                    // 작업을 할 것이며, 만약에 발생한다면, 이를 처리하지않고
                                                    //  호출한 곳으로 에러를 던진다는 의미

    ServerSocketManager ssm =
```

메소드 내부에서 예외가 발생할 수 있는 코드를 작성할 때

try - catch 블록으로

예외를 처리하는 것이 기본이지만, 경우에 따라서는 메소드를 호출한 곳으로

예외를 떠넘길 수도 있습니다. 이때 사용하는 키워드가 throws입니다.

throws를 왜사용하는걸까 ?

1) 예외를 메소드의 사용자에게 전가하여 메소드는 기능에 집중하고, 사용자는 발생할 수 있는 예외를 미리 확인하고 처리 루틴을 설정해두기 위해 사용된다.

Socket

```
private ServerSocket servSock; // -서버 프로그램에서 사용하는 소켓
                               // -포트를 통해 연결 요청이 오기를 대기
                               // -요청이 오면 클라이언트와 연결을 맺고 해당 클라이언트와 통신하는
                               //  새 소켓을 만드는 일을 한다.
                               // -새로 만들어진 소켓은 클라이언트 소켓과 데이터를 주고 받는다.

private Socket[] clntSockArr;  // - 서버 프로그램으로 연결 요청
                               // - 데이터 전송을 담당
```