**객체지향 프로그래밍** | 최윤정 교수님 | 과제#6 Summary 이서현 2276219  
**예외처리 (Exception Handling)**주요내용:  예외와 오류  예외 클래스 Try-Throw-Catch-Finally  예외 처리 코드  예외 종류에 따른 처리 코드  
  잡기 Vs. 예외 처리 떠넘기기  사용자 정의 예외와 예외 발생시키기

**에러(Error) vs. 예외(Exception)**

* **에러 - 컴파일 에러**
* 대부분 문법오류(syntax error).
* 하드웨어의 잘못된 동작 또는 고장으로 인한 오류.
* 에러 발생 시 프로그램 비정상적 종료, 정상 실행 상태로 돌아갈 수 없음.
* **예외 - 런타임 에러 (Runtime Error)**
* 대부분 컴파일은 통과, 실행 중 도중 오류발생.
* 실행 중에 발생하게 되는 예상치 못한 상황에 의한 실행오류.
* 사용자의 잘못된 조작을 예상하지 못하거나 개발자의 잘못된 코딩으로 인해 발생.
* 예외가 발생되면 프로그램 종료, 예외를 잡아 처리하는 과정을 추가하면 정상 실행 상태로 돌아갈 수 있음.

예외적인 상황 처리: 자바의 Exception 클래스와 이 클래스의 서브 클래스들을 사용해 예외를 처리

**\*일반 예외(Checked Expection):** 반드시 예외처리 코드 작성해야.

**\*실행 예외(RuntimeException):** 예외처리 코드 생략해도 컴파일은 되나 시행착오에 따라 예외처리 코드 작성 필요.

실행 예외(RuntimeException)의 예)

* NullPointerException

 객체 참조가 없는 상태. null 값 갖는 참조변수로 객체 접근 연산자인 도트(.) 사용했을 때 발생

* ArrayIndexOutOfBoundsException

 배열에서 인덱스 범위 초과하여 사용할 경우 발생

* ClassCastException  
   타입 변환이 되지 않을 경우 발생 (올바르지 않은 캐스팅 – 실행 중 죽는다)

Exception 클래스는 두 개의 생성자를 갖는다:

 Exception(String s) : 메시지로 Exception 객체를 생성  Exception() : “널(null) 문자열”로 Exception 객체를 생성하는 기본 생성자

발생한 Exception 정보를 알아오는 메소드:

 String getMessage() : Exception 객체에 저장된 문자열 또는 널(null)을 반환

또한 Exception 클래스는 많은 서브클래스들을 갖고 있다.

예외가 발생했을 때 : 전달(throw) & 잡기(catch)

**예외 처리 코드(try-catch-finally)** : 예외 발생시 프로그램 종료를 막고, 정상 실행을 유지할 수 있도록 처리함.

[try🡪 (throw)🡪catch🡪finally] 순서로 실행

try: 예상되는 문제가 있을 것 같은, 예외가 발생할 가능성이 있는 코드. 주요 프로세스가 진행된다.

( throw: try블록에서 Exception클래스 타입으로 생성한 객체를 catch블록에 매개변수로 전달. )

catch: Exception 객체가 전달될(throw) 때 제어가 해당 catch블록으로 이동된다. 반드시 1개 이상 정의돼야 함.

* 예외처리 작업. 예외별로 여러 개 블록 존재 가능.
* catch 블록의 코드가 실행된 후 catch 블록 다음에 이어지는 문장을 계속 수행한다.

finally: 선택 작성. 정의되었다면 예외 발생과 전달 여부에 관계없이 항상 실행되어 정리용, 마무리용으로 사용된다.

|  |
| --- |
| **try** {  // 예외가 발생할 가능성이 있는 코드.  // Exception 객체 e 생성코드  // e 전달하기(throw) // e를 catch 블록으로 전달하는 코드, 자동전달된다.  } **catch** (Exception e) {  // e 객체가 전달된(throw) 때 제어가 해당 catch 블록으로 이동된다  // Exception이 발생했을 때, 이를 처리하기 위한 코드  } **finally** {  // 항상 실행  } |

**예외 종류에 따른 처리 코드**

다중 catch – 예외별로 예외 처리 코드를 다르게 구현!

\*catch 순서 : 가장 작고 구체적인 예외부터 잡기! 예외의 상위클래스가 하위클래스보다 아래에 위치해야 한다. 점진적으로 크고 세지도록. 최상위 예외는 가장 마지막에 위치시킨다. (ex. Exception이 젤 위면 안 됨!)

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**예외전달하기, 예외 떠넘기기 : 직접 잡아 주는대신 호출된 곳으로 넘김**

throws: 전달만 함. 메소드 선언부 끝에 작성. 메소드에서 처리하지 않은 예외를 호출한 곳으로 떠넘김

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

Exception 클래스와 서브클래스의 상위클래스 하위클래스 관계

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**사용자 정의 예외와 예외 발생**

 사용자 정의 예외 클래스 선언 - Exception 클래스나 RuntimeException 클래스를 상속하여 만든다.

**예외 정보 얻기**

 getMessage() : 예외를 발생시킬 때 생성자 매개값으로 메시지를 리턴. 전달한 메시지는 catch()에서 활용.

throw new XXXException("예외 메시지");

 printStackTrace()

예외 발생 코드의 추적 내용을 모두 콘솔에 출력. 프로그램 테스트하면서 오류 찾을 때 유용히 활용.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명