34주차 발표_민서연

에러와 예외 처리

01 에러와 예외

에러

시스템이 종료되어야 할 수준의 상황과 같이
수습할 수 없는 심각한 문제

개발자가 미리 예측해서 방지 할 수 없다.

java.lang.Exception

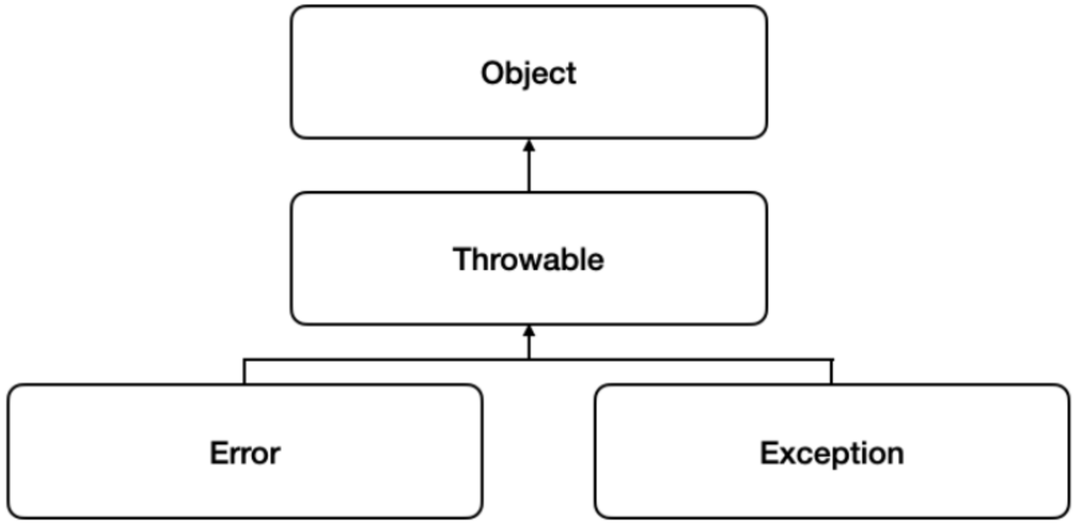
예외

개발자가 구현한 로직에서 발생한 실수나
사용자의 영향에 의해 발생

개발자가 미리 예측해서 방지 할 수 있다.

java.lang.Error

01 에러와 예외



unchecked

RunTimeException

unchecked

Throwable

Throwable 클래스는 예외 / 에러 처리를 할 수 있는 최상위 클래스

이 클래스를 상속받은 서브 타입만이 자바 가상 머신(JVM) 이나 throw 키워드에 의해 던져 질 수 있다.

01 에러와 예외

에러

컴파일 에러 (Compillation Error)

컴파일 단계에서 오류를 발견하면 에러 메세지를 출력해 준다.

대표적인 원인으로는 문법 구문 오류가 있다.

01 에러와 예외

에러 

논리에러 (Logic Error)

일종의 '버그'

프로그램이 실행하고 작동하는데는 아무런 문제가 없지만 결과가 예상과 달라 사용자가 작업을 수행하지 못하게 되어 서비스 이용에 지장이 생김

컴퓨터 입장에서는 에러 메세지를 알려주지 않기 때문에 전반적인 코드와 알고리즘을 개발자가 알고 있어야 한다.

01 에러와 예외

에러

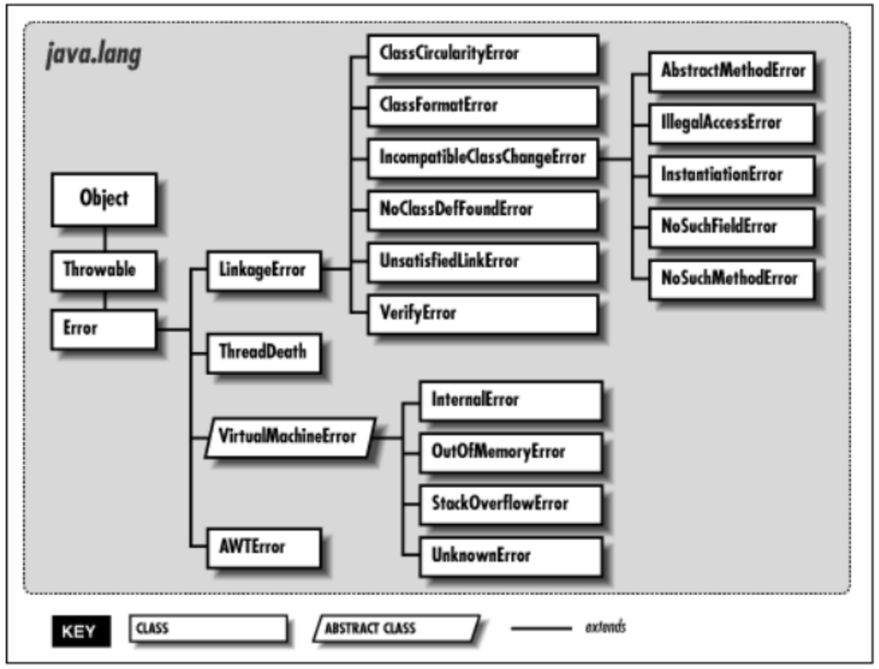
런타임 에러 (Runtime Error)

컴파일 에러를 잡아서 컴파일에는 문제가 없어도 프로그램 실행 중에 에러가 발생해서 잘못된 결과를 얻거나 외부적 요인으로 프로그램이 비정상적으로 종료

대체적으로 개발 시 설계 미숙으로 발생하는 에러가 대부분이고 개발자가 역추적해서 원인을 확인해야 한다.

01 에러와 예외

에러

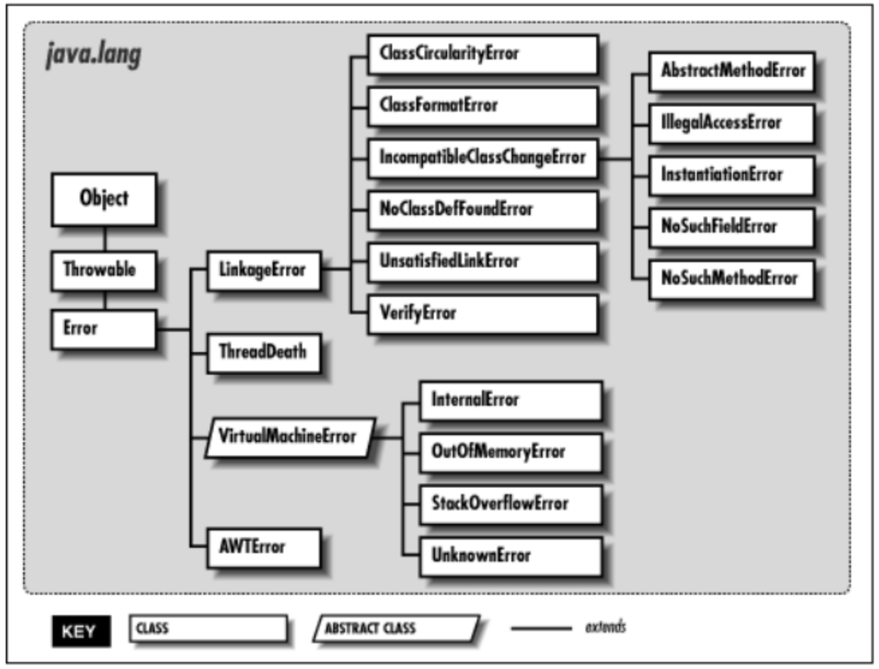


StackOverflowError
: 호출의 깊이가 깊어지거나 재귀가 지속되어 stack overflow가 발생 될 시에 던져지는 오류

OutOfMemoryError
: JVM이 할당된 메모리의 부족으로 더 이상 객체를 할당 받을 수 없을 때 던져지는 오류

01 에러와 예외

에러



StackOverflowError
: 호출의 깊이가 깊어지거나 재귀가 지속되어 stack overflow가 발생 될 시에 던져지는 오류

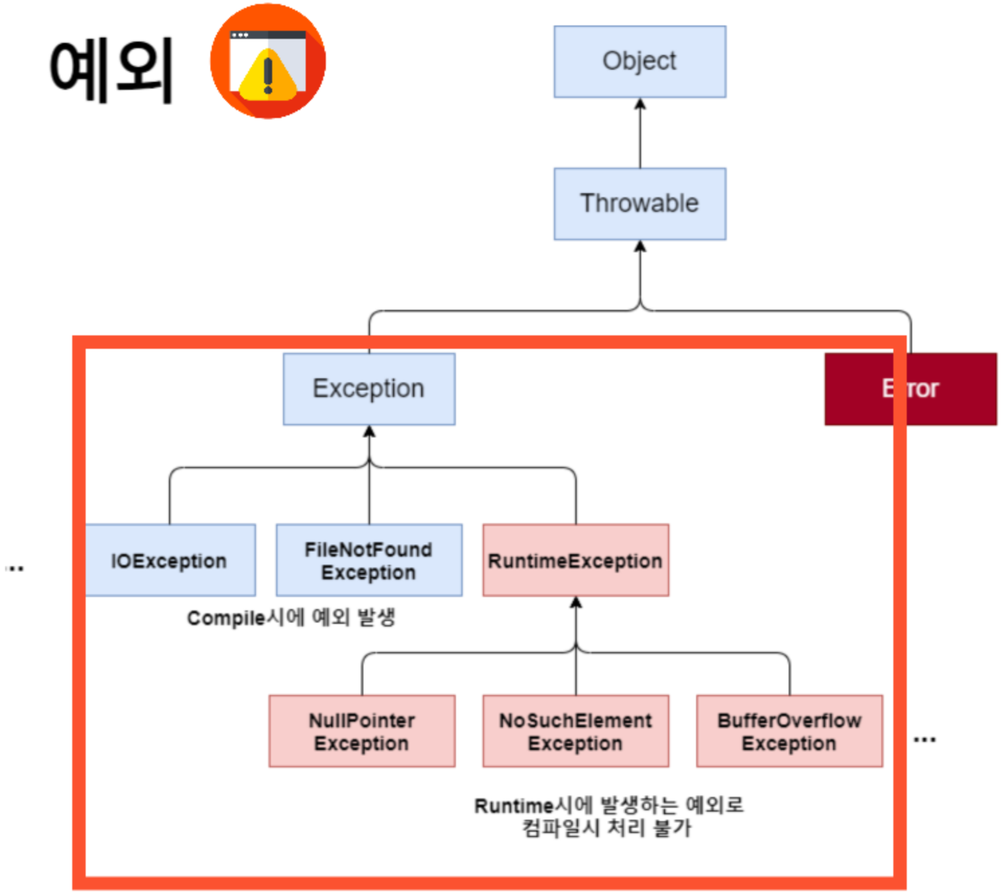
재귀를 사용 할 때에 더욱 조심하거나 가시적인 loop를 사용하면서 간접적인 예방을 할 수있음

OutOfMemoryError
: JVM이 할당된 메모리의 부족으로 더 이상 객체를 할당 받을 수 없을 때 던져지는 오류

heap을 늘려주거나 새는 메모리를 차단하는 방법

01 에러와 예외

예외



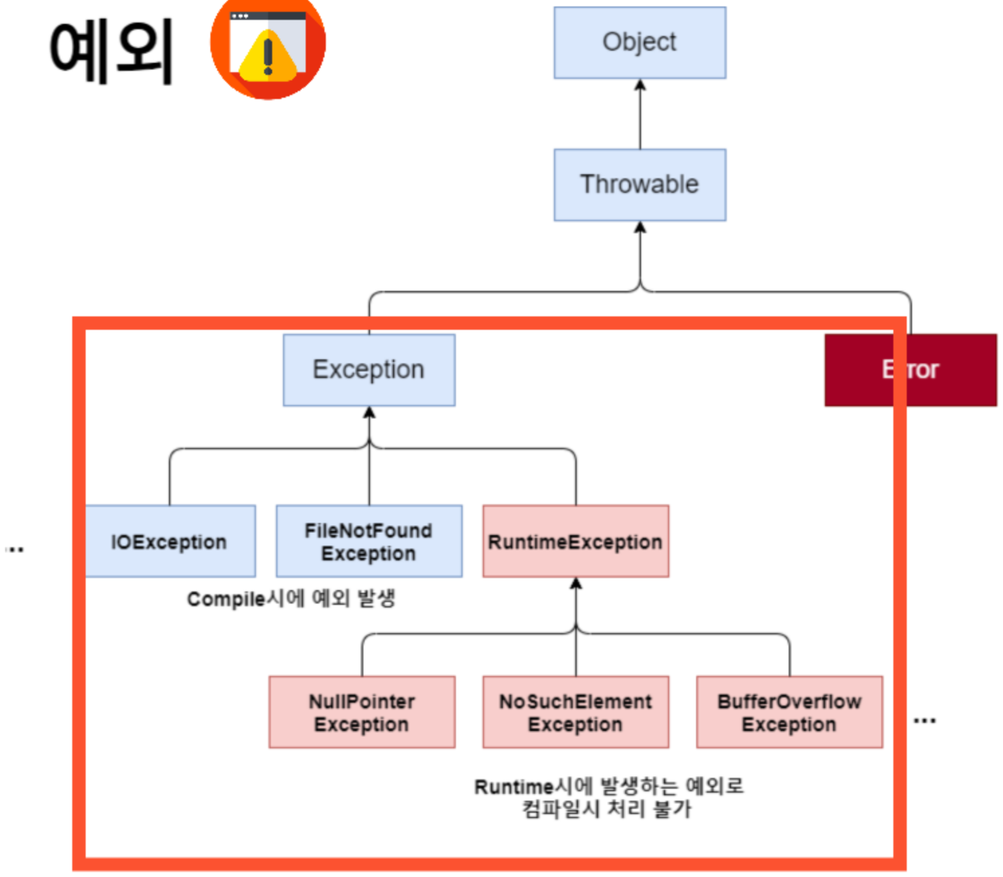
Exception은 RuntimeException과 그 외의 클래스 그룹으로 나뉜다.

RuntimeException: unchecked

그 외: checked

01 에러와 예외

예외

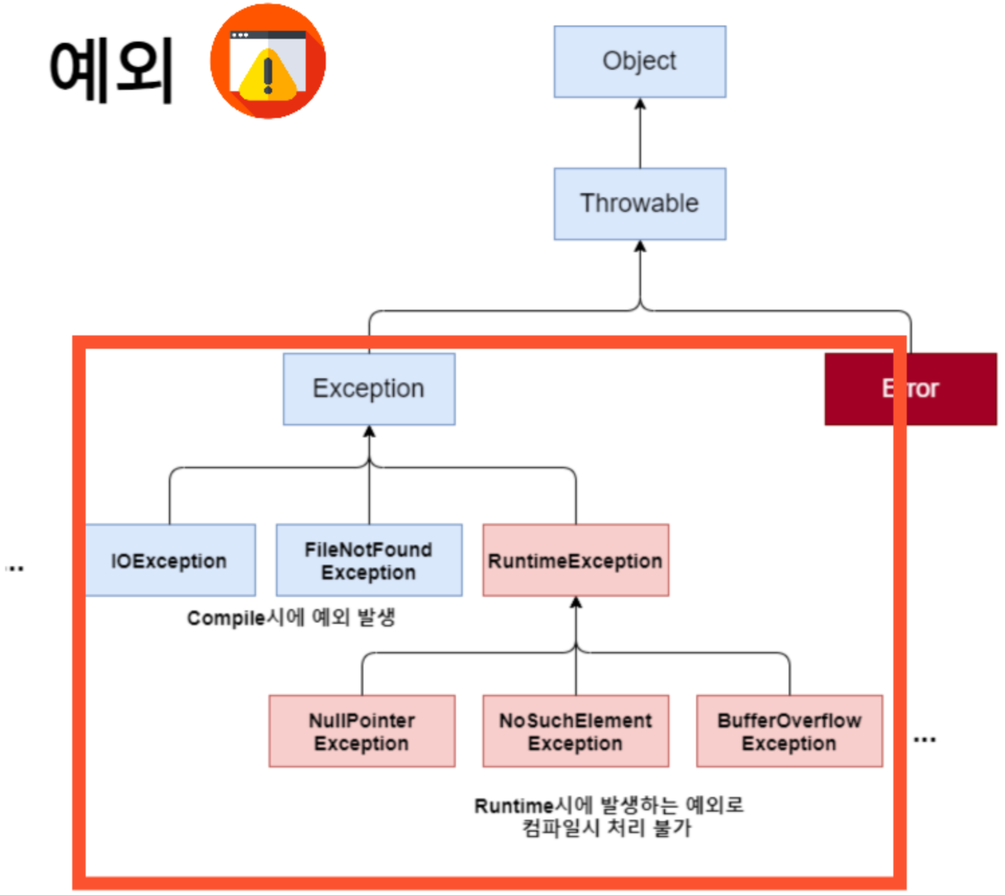


RuntimeException
: 프로그래머의 실수로 발생하는 예외

- *IndexOutOfBoundsException*
  : 배열의 범위를 벗어남
- *NullPointerException*
  : 값이 null인 참조 변수의 멤버를 호출
- *ClassCastException*
  : 클래스 간의 형 변환을 잘못함
- *ArithmeticException*
  : 정수를 0으로 나누는 산술 오류

01 에러와 예외

예외



그 외 Exception 하위 클래스
: 사용자의 실수와 같은 외적인 요인에 의해 발생하는
컴파일 시 발생하는 예

- FileNotFoundException
  : 존재하지 않는 파일의 이름을 입력
- ClassNotFoundException
  : 실수로 클래스의 이름을 잘못 기재
- DataFormatException
  : 입력한 데이터의 형식이 잘못된 경우

02 Spring에서의 에러 처리

Controller에서 @ExceptionHandler 사용하기

예외를 처리하는 메서드를 정의하고 @ExceptionHandler 어노테이션을 사용한다.

@ExceptionHandler 어노테이션이 있는 메서드는 전체 애플리케이션이 아닌
특정 컨트롤러에 대해서만 활성화가 된다.

```
public class exController{
    @ExceptionHandler({customException1.class,  customException2.class})
     public void handleException() {
          ...
     }
}
```

02 Spring에서의 에러 처리

HandlerExpceptionResolver 정의하기

응용프로그램에서 발생한 모든 예외를 해결할 수 있고 REST API 에서 균일한 예외 처리 메커니즘을 구현할 수 있다.

ExceptionHandlerExceptionResolver

: DispatcherServlet에서 활성화 된다. Spring 예외를 해당 HTTP 상태 코드로 해결하는 데 사용한다.

ResponseStatusExceptionResolver

: 주된 책임은 사용자 지정 예외에서 사용 가능한 @ResponseStatus 어노테이션을 사용해서 HTTP 상태코드에 매핑한다.

02 Spring에서의 에러 처리

@ControllerAdvice

Spring 3.2부터 @ControllerAdvice 어노테이션으로 @ExceptionHandler를 지원한다.

이를 통해 MVC 모델에서 탈피하고 @ExceptionHandler의 유형 안전성 및 유연성과 함께 ReponseEntity를 사용하는 메커니즘을 사용할 수 있다.

02 Spring에서의 에러 처리

ResponseStatusExcepion

HttpStatus와 reason, cause를 제공하는 인스턴스를 만들 수 있다.

하나의 예외 유형이 다중 다른 응답으로 이어질 수 있어서 @ExceptionHandler에 비해 결합도가 낮다.
사용자 정의 예외 클래스를 많이 만들 필요가 없다.
하나의 애플리케이션 내에서 다양한 접근 방식을 결합하는 것이 가능하다.

03 출처

https://toneyparky.tistory.com/40
https://sjh836.tistory.com/122
https://velog.io/@hameee/Java-Throwable
https://morningcoding.tistory.com/entry/Java31-%EC%98%88%EC%99%B8-%ED%81%B4%EB%9E%98%EC%8A%A4-Throwable-%ED%81%B4%EB%9E%98%EC%8A%A4
https://inpa.tistory.com/entry/JAVA-%E2%98%95-%EC%97%90%EB%9F%ACError-%EC%99%80-%EC%98%88%EC%99%B8-%ED%81%B4%EB%9E%98%EC%8A%A4Exception-%F0%9F%92%AF-%EC%B4%9D%EC%A0%95%EB%A6%AC
https://velog.io/@6v6/Spring-%EC%8A%A4%ED%94%84%EB%A7%81-%EC%97%90%EB%9F%AC-%EC%B2%98%EB%A6%AC-Error-Handling-for-REST-with-Spring
https://codedragon.tistory.com/4447