예외처리

목표:

- 예외가 프로그램에서 발생하는 상황에 대해서 처리할 프로세스 구조를 안다.
- 자바에서 예외를 처리하는 프로그램의 기본 구조를 안다.
- 여러가지 예외를 처리하는 **api**에 대해서 기능에 대해서 알고 실행할 줄 안다.
- 사용자 정의 예외 클래스를 선언하고 활용할 줄 안다.

생각해봅시다:

- 프로그래밍을 수행할 때, 발생할 수 있는 예외는 어떤 것이 있을까?
  - 컴파일 에러
  - 실행 에러
  - 외부 환경에 의해

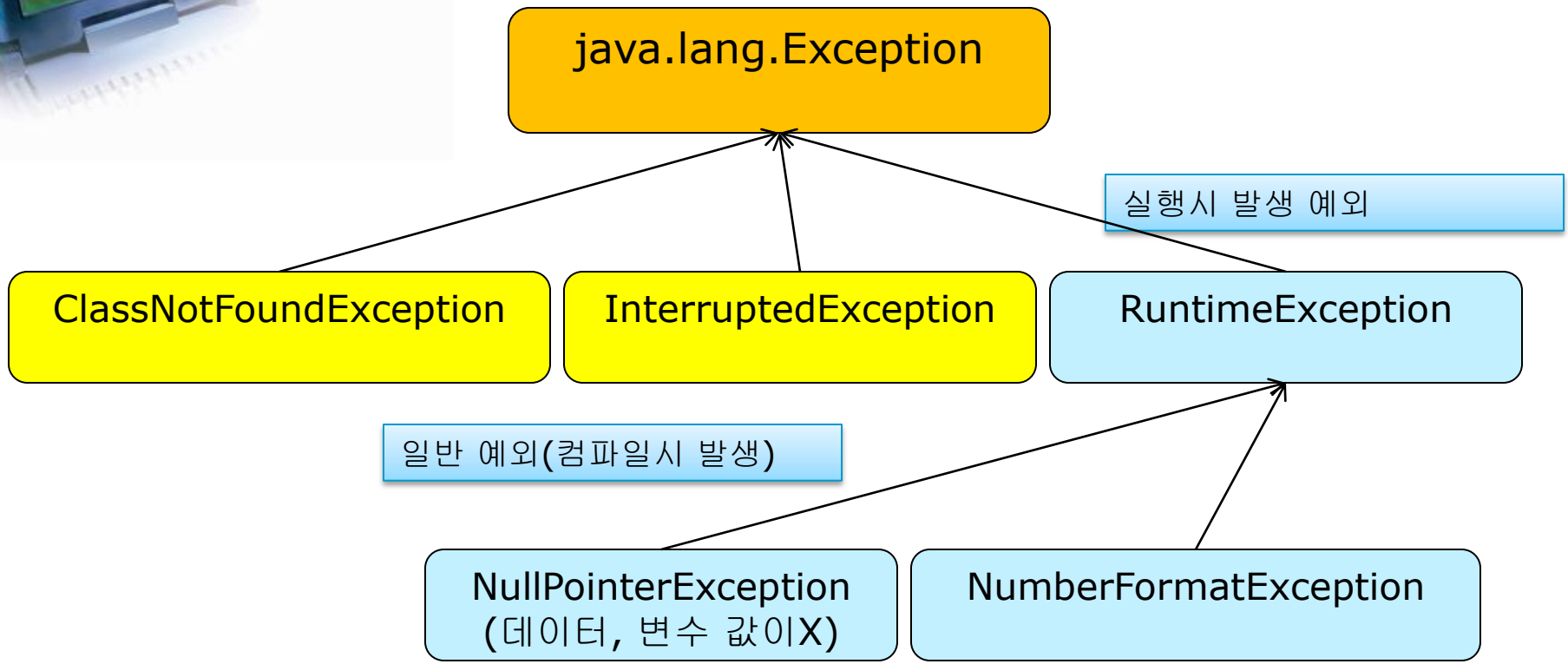
예외란?

- 컴퓨터 하드웨어적으로 오동작, 고장으로 응용프로그램 실행이 발생하는 것을 자바에서는 error라고 한다.
- JVM 실행 문제가 생기는 것으로 프로그램이 견고하게 만들었어도 실행이 불능하게 된다.
- 이런 프로그램에서 개발자는 대처할 방법이 없다.
- 자바에서 error 이외에 예외(Exception)라고 부르는 오류가 있다. 예외란 사용자의 잘못된 조작 또는 개발자 코딩 문제 발생하는 **프로그램 오류**를 말한다. 원래 이런 오류가 발생하면 프로그램 종료된다. 자바에서는 예외처리(Exception) 프로그램을 종료하지 않고 실행 상태 유지되도록 한다.
- 예외(Exception)

  일반 예외: 컴파일 체크 예외

  실행 예외: 컴파일 후, 실행 처리

예외 클래스 구조


java.lang.Exception
실행시 발생 예외
ClassNotFoundException
InterruptedException
RuntimeException
일반 예외(컴파일시 발생)
NullPointerException
(데이터, 변수 값이X)
NumberFormatException

실행 예외들.

- 컴파일 시 발생하는 것이 아니기에 개발자 경험이나, 간단한 테스트 코드로 예외처리를 하여야 하는 것을 말한다.
- NullPointerException
  - 객체 참조가 없는 상태, null값은 갖는 참조 변수로 접근 연산자(.) 사용해서 멤버를 호출했을 때 발생하는 예외
- ArrayIndexOutOfBoundsException
  - 배열에서 인덱스의 범위를 초과해서 사용했을 때, 발생하는 예외를 말한다.
- NumberFormatException
  - 문자열 → 숫자를 변경하는 경우, 문자열을 숫자로 변환하는데 숫자형 문자이어야 하는데, 그렇지 않을 때, 처리하는 객체 메소드에서 예외발생

예외 처리 코드:

- try{
  - 1.예외가 발생할만한 코드..
  - 2. 예외발생시, catch쪽에 정의된 예외클래스로 던짐
    - String name=null; System...(name.length());
    - NullpointerException 객체를 던짐.
  - 3. 상위 라인에 예외가 발생하면, 실행되지 않음
  - 4. 상위 라인에 예외가 발생하면, 실행되지 않음
- }catch(예외클래스 참조){
  - NullpointerException 이상의 상위클래스가 정의되어 있다
- }

예외 확인 예제

- 1단계
  - id를 입력하세요
  - id=null; 입력이 안된 경우, id="himan";
  - 입력된 id: @@@
  - catch
    - 아이디가 입력이 되지 않았습니다.
- 2단계
  - id, pass
  - id와 password가 유효할 때, @@님 환영합니다.
  - catch
    - id또는 password가 입력되지 않았습니다.

다중 catch

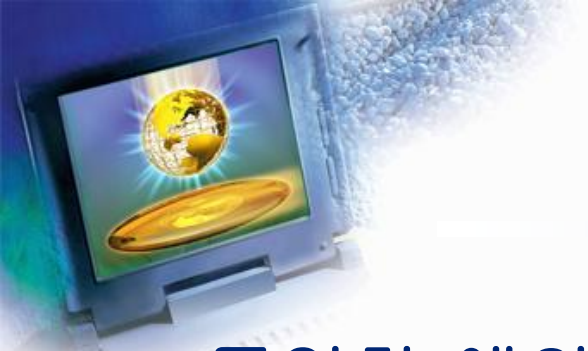
- 예외는 동시 다발적으로 여러 예외를 발생할 수 있지만, 여러 예외를 하나의 try{}catch(예외1){}catch(예외2){} 구문으로 처리할 수 있다. 그 외 예외 내용은 상위 클래스인 Exception을 통해 처리할 수 있다.
- 형식
  - try{
    - 여러 예외를 발생할 만한 코드..
    - 예외1
    - 예외2
  - }catch(예외1){
  - }catch(예외2){
  - }catch(**Exception** e){}

finally

- 예외를 발생하는 경우나 발생하지 않고 정상적으로 처리하는 경우 모두 다, 처리할 프로세스나 프로그램이 있을 때는 try catch문 마지막 블럭으로 finally{} 구문을 포함 시킨다.
- ex)
  - 파일입출력 클래스 선언
  - try{
  - 참조 = 파일입출력 객체생성..
  - ... 예외/ 발생하지..
  - }catch(예외){
  - }**finally**{
  - 파일입출력 객체 자원해제 .close();
  - }

예외 클래스에서 활용되는 메서드

- catch(예외 클래스 참조){
  - 참조.메서드();
- }
- e.getMessage() : 예외 가지고 있는 메시지를 문자열로 return
- e.printStackTrace() : 예외의 발생 경로를 추적내용 출력



예외 떠넘기기

- **동일한 예외 처리**가 여러 클래스에 분산되어 있어서 try~catch문을 여러 번 반복해야 할 때, 분산된 예외 처리를 최종적으로 호출되는 곳에서만 처리할 때, 각 **분산된 메서드에서 예외 떠넘기기를** 해서 한번에 try~catch으로 처리 할 수 있다.
- 리턴타입 메소드() **throws** 예외클래스1, 예외클래스2{
  - 예외가 발생할만한 코드(try catch를 한꺼번에 처리하기 위해 메서드명 옆에 throws로 떠넘기기 처리)
- }

사용자 정의 예외와 예외 발생

- 프로그램을 개발 시, 자바 표준 API에서 제공하는 예외 클래스에서 더 추가적인 예외 클래스를 만들어야 할 때가 있다.
  - 정의 ➔ 예외 발생시 예외객체 던지기 ➔ 던져진 예외를 try catch문에 의해 처리
- 이를 사용자 정의 예외 클래스라고 한다.
- class XXXException extends Exception{
  - public XXXException(){}
  - public XXXException(message){
    - super(message);
  - }
- }
- 예외 발생 : 내장된 예외는 throw new Exception(); 식으로 명시를 하지 않더라도 내부적으로 발생하지만, 사용자 정의 예외는 명시해야 한다.
  - throw new XXXException();

사용자 정의 예외와 예외 발생

- ex)
  - public void method() throws XXXException{
  - throws new XXXException();
    - // 명시적으로 호출 처리.
  - }

- 호출하는 곳에서 예외 처리.(try catch 처리)
  - try{
    - method();
  - }catch(XXXException e){ // 사용자 정의 예외 객체로 받아야 함.
    - 예외가 발생할 시, 처리할 내용
  - }

예외 클래스 메시지 전송

- 예외가 발생했을 때..
  - if이나 강제적으로
  - throw new XXXException("넘겨지는 메시지");
    - 정의된 곳에서 super(**msg**);
  - Exception
    - message = **msg** ; // 할당
    - getMessage() 에서 넘겨지는 메시지를 호출

확인예제

- 사용예외 클래스 정의
  - XXXException ex) ShoppingException
    - super를 통해 예외내용 입력하여 getMessage() 가져올 수 있도록 처리
- 프로그램 throw XXXException
  - Market
    - goShopping(쇼핑할 때, 보유할 현금)
    - buyProduct(쇼핑시, 물건을 구매 후, 지불금액)
      - 보유할 현금과 지불금액을 비교해서 금액이 적을 때 사용자 정의 예외 객체 호출.. (드디어, 안타까운 현실이군용!! message 전달)
- MAIN에서 호출 처리..
  - Martket....

사용정의 예외 클래스 숙제

- 1단계
  - PointException : 점수가 특정 점수 이하로 되었을 때, 예외 처리로 "불합격입니다. 재 수강을"
  - Exam
    - getPoint(임의의 점수) : 예외 처리 메서드
      - 입력 받은 점수가 60점이하일 때, 예외 객체 호출
  - Main()
- 2단계
  - PointException
    - 추가 메서드 : showResult()
      - @@과목, @@점 획득 하였는데, 결과는 재수강 입니다.
  - Exam
    - 과목명, 시험점수
    - getPoint() : 과목과 Random next()활용

정리 및 확인하기(숙제- 정답):

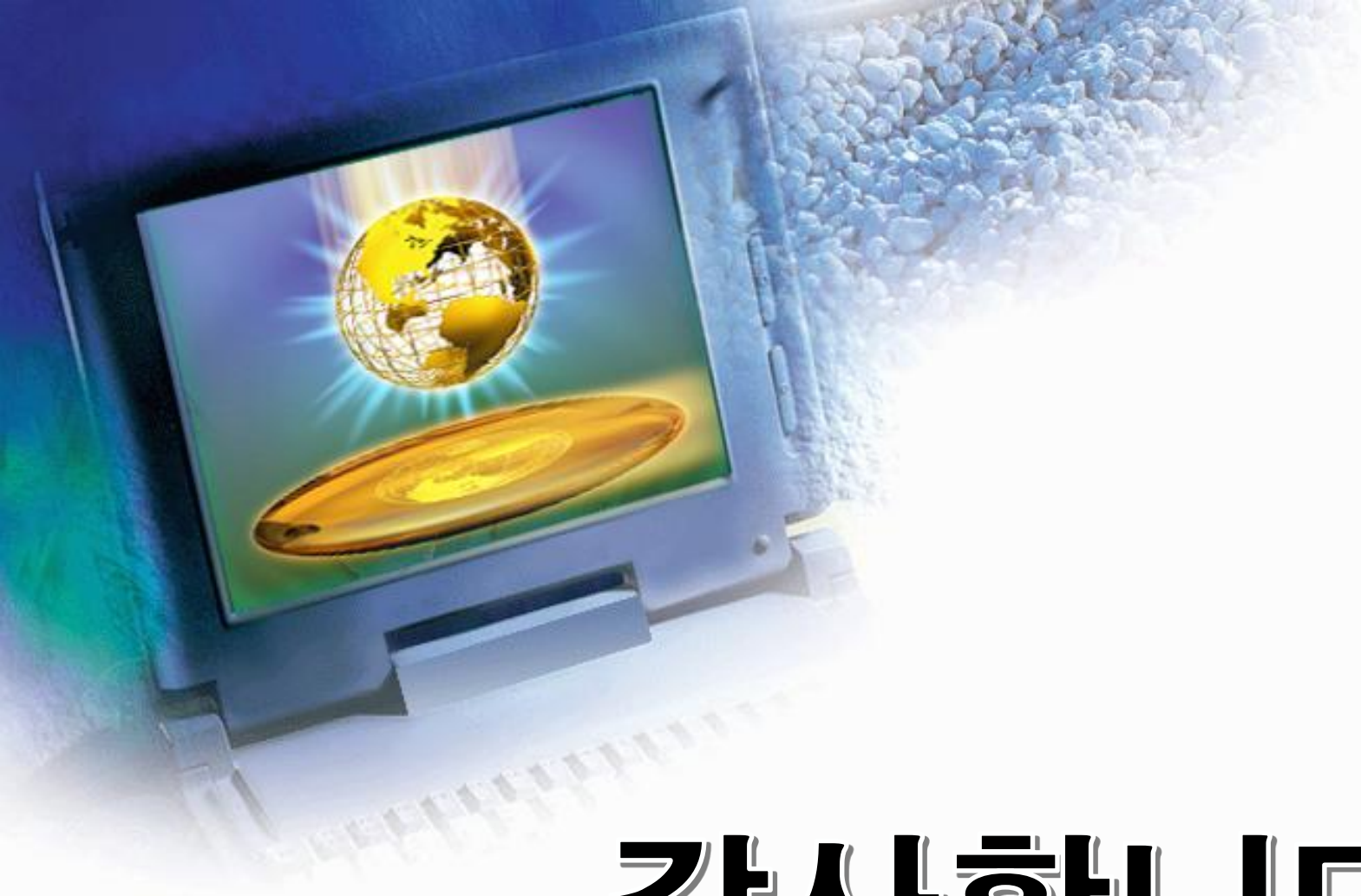
- 예외에 대한 설명 중 틀린 것은 무엇입니까?
  1. 예외는 사용자의 잘못된 조작, 개발자의 잘못된 코딩으로 인한 프로그램 오류를 말한다.
  2. RuntimeException의 하위 예외는 컴파일러가 예외 처리 코드를 체크하지 않는다.
  3. 예외는 try~catch블록을 사용해서 처리된다.
  4. 자바 표준 예외란 프로그램에서 처리할 수 있다.
- try~catch-finally 블록에 대한 설명 중 틀린 것은 무엇입니까?
  1. try{}블록에는 예외가 발생할 수 있는 코드를 작성한다.
  2. catch{}블록에서 return 블록에서 발생한 예외를 처리하는 블록이다.
  3. try{} 블록에서 return 문을 사용하면 finally{} 블록은 실행되지 않는 다.
  4. catch{} 블록은 예외의 종류별로 여러 개를 작성할 수 있다.

정리 및 확인하기(숙제- 정답):

- 예외에 대한 설명 중 틀린 것은 무엇입니까?
  1. 예외는 사용자의 잘못된 조작, 개발자의 잘못된 코딩으로 인한 프로그램 오류를 말한다.
  2. RuntimeException의 하위 예외는 컴파일러가 예외 처리 코드를 체크하지 않는다.
  3. 예외는 try~catch블록을 사용해서 처리된다.
  4. 자바 표준 예외란 프로그램에서 처리할 수 있다.
- try~catch-finally 블록에 대한 설명 중 틀린 것은 무엇입니까?
  1. try{}블록에는 예외가 발생할 수 있는 코드를 작성한다.
  2. catch{}블록에서 return 블록에서 발생한 예외를 처리하는 블록이다.
  3. try{} 블록에서 return 문을 사용하면 finally{} 블록은 실행되지 않는 다.
  4. catch{} 블록은 예외의 종류별로 여러 개를 작성할 수 있다.

정리 및 확인하기(숙제- 정답):

- throw에 대한 설명으로 틀린 것은 무엇입니까?
  1. 예외를 최초로 발생시키는 코드이다.
  2. 예외를 호출한 곳으로 떠넘기기 위해 메소드 선언부에 작성된다.
  3. throw로 발생된 예외는 일반적으로 생성자나 메소드 선언부에 throws로 떠넘겨진다
  4. throw 키워드 뒤에는 예외 객체 생성 코드가 온다.
- 다음과 같은 메소드가 있을 때, 예외를 잘못 처리한 것은 무엇입니까?
  1. try{method();}catch(Exception e){}
  2. void method2() throws Exception{method1{};}
  3. try{method1();}catch(Exception e){}catch(ClassNotFoundException e){}
  4. try{method1();}catch(ClassNotFoundException e){}catch(NumberFormatException e){}



감사합니다 !