에러와 예외

작성일시	@2022년 7월 27일 오후 1:20
강의 번호	JAVA
유형	
자료	
복습	☐
학습 소스 출처 1	
학습 소스 출처 2	
날짜	

에러 (error) : 프로그램 코드로는 처리할 수 없는 상황

—메모리 부족, 스택오버플로 등

—> 디버깅으로 해결

예외 (exception) : 프로그램 코드로 처리할 수 있는상황

—읽어야 할 파일이 없는 경우, 네트워크연결이 안되는 경우

—>예외처리(exception handling)로 해결

예외처리 클래스

예외처리를 위한 클래스들이 있다

가장 상위인 object 밑에 Throwable 이 있고 이밑에 두가지로나뉜다.

Error 의 하위 클래스 : error에 해당하고 디버깅이 필요하다. 하위클래스로는 OutOfMemoryError 가 있다.

Exception 의 하위클래스 : exception 에 해당하고 예외처리가 필요하다. 이하위클래스는 또 두가지로 나뉘는데.

checked exception : 꼭 예외처리가 필요한 경우에 쓰는 클래스들
ex)FileNotFoundException …

unchecked exception : 예외처리가 필수는 아님.

예외처리 기본

예외처리 할시 , try문 안의 예외발생 지점 아래의 모든 try구문은 처리하지않고 catch 문을 실행한다.

```
try{}
catch(){}
catch(){}
catch(){}
```

catch는 여러개 존재 할 수 있다.

또한 예외처리할시 상위 예외처리 클래스일 수록 아래로 가야한다. 왜냐하면 하위 클래스들은 전부 상위클래스를 상속받았기 때문에 더상위클래스일수록 하위클래스보다 커버하는 범위가 크다.

```
try(FileReader br= new FileReader("aaa.txt")) {
      br.read();
    }
    catch(FileNotFoundException e) {

    }
    catch(IOException e) {

    }
```

위와 같이 try()안에서 close할 변수를 작성하면, 따로 close를 하지 않아도 알아서 종료할때 close를 실행한다.

throws를 오버라이딩시 주의점

throw 사용를 사용하는 상위 클래스의 메소드를 하위 메소드가 오버라이딩할때,

throw 하는 예외클래스는 항상 부모가 받는 클래스보다 작거나 같아야한다.

```
class Parent{
  void mehodA() throws IOException{}
}

public class Test extends Parent{
  @Override
  void methodA() throws Exception{
  }
  //불가능!
}
```

Throw , Throws

```
public void method01() throws FileNotFoundException { //예외 발생시 호출한 메서드로 예외 넘기기
    Random r = new Random();
    int num=r.nextInt(2);

    if(num==0) {
      //고의로 예외 발생!!
      throw new FileNotFoundException();
    }
    System.out.println("method01 ㅈㅇ료");
  }
```

위의 코드를 보면 throw를 이용해 직접 예외를 발생시키고,

이를 throws를 받을 수 있다. 예외처리 클래스도 **클래스** 이기 때문에 new가 필요하다

사용자 정의 예외 클래스

```
public class MyCheckedException extends Exception {
  public MyCheckedException() {
    super("MyCheckedException");

  }
  public void mySolution() {
    System.out.println("예외 상황이 해결됐어요.");
  }
}
//이런식으로 코드를 짜고
```

```
//예외를 throws 할 메소드
public void method01() throws  MyCheckedException {
  Random r = new Random();
  int num=r.nextInt(2);

  if(num==0) {
    throw new MyCheckedException();
  }
  System.out.println("method01 ㅈㅇ료");
}
//메인
public static void main(String[] args) {
  ExceptionMain m= new ExceptionMain();
  try {

    m.method01();
  }
  catch(MyCheckedException e) {
    e.printStackTrace();
    e.mySolution();
  }


}
```

위의 코드처럼 Exception을 상속받아 사용자 정의 함수를 작성할 수 있다.