60200307 우연정

1. Exception handling
 1-1. Exception의 예시
2. Separation of Concerns
3. Refactoring

1. Exception handling
 Exception handling(예외 처리)이란, “프로그램 작성 언어의 능력 가운데 수행 도중 발생하는 예외 상황을 처리할 수 있도록 하는 기능”[IT용어사전]이다. 우리가 입력한 내용에 대해서 컴퓨터는 내용인지 제어인지 구별하지 못한다는 문제가 있다. 직접적 메세지와 그 메세지를 제어하는 메시지는 아예 다른 것이다. 예를 들어 숫자 1과 뒤로가기 키는 전혀 다르다. 숫자 1은 모니터에 찍어야 할 내용이고 뒤로가기 키는 모니터의 커서의 위치를 제어하는 제어문이다. 제어문과 내용은 다른 것이다.

 if는 계산하는 문장의 흐름을 바꿔주는 문장이다. 계산식과 if문은 전혀 다르다. 어떤 일을 수행하는 데에 있어서 문제가 발생하면 문제 해결 후 다시 돌아가는지 그만둬야 하는지 판단한다. 프로그램이 돌아가다 에러가 발생하는 것이 Exception이다. 이는 원래 하던 일과는 관계가 없다. 프로그램 내에서 같이 취급을 하면 제어문과 내용이 혼용된다. 되도록 정상적인 Computation과 Control이 분리되도록 프로그램을 짜는 것이 중요하다.

 혼용되지 않기 위해서는 미디어가 달라야 한다. 작업 프로그램에서도 계산과 예외를 다르게 만들어놓았다. read라는 함수에서, 에러가 나면, IOException이 전달되어 처리하도록 되어있다.
 Exception이 발생하면 받는 명령문이 try / catch 문이다. 에러가 나면 catch 부분을 처리하라는 뜻이다. e.printStackTrace() 는 에러의 위치를 찍어주는 것이다.

 함수 옆의 throws IOException 은 자기가 처리하던가 자기를 호출한 함수에게 밀어버린다는 의미이다. (위로 던진다.) 프로그램을 짤 때 예외 처리를 전부 위로 밀어버리기보다는 우리가 예외 처리를 직접 짜는 것이 좋다. 독립적으로 처리할 수 있어야 코드가 재사용하기 쉬워진다. 에러를 위로 밀어버리면 다른 사람이 내 코드를 보고 뭐가 에러인지 모르기 때문에, 에러는 안에서 처리해야 한다. 관심사를 잘 분리해야 한다.

 1-1. Exception의 예시
- IOException: “일종의 입력 또는 출력(I/O) 예외가 발생했음을 알리는 예외이다.”[https://docs.oracle.com/javase/7/docs/api/java/io/IOException.html]

- ArithmeticException: “예외적인 산술 조건이 발생하면 발생한다. 예를 들어 정수 "0으로 나누기"는 이 클래스의 인스턴스를 발생시킨다.”[https://docs.oracle.com/javase/7/docs/api/java/lang/ArithmeticException.html]

- IllegalArgumentException: “메소드에 부적절한 인수가 전달되었음을 나타내기 위해 발생한다.” [https://docs.oracle.com/javase/7/docs/api/java/lang/IllegalArgumentException.html]

- NumberFormatException: “문자열을 숫자 유형 중 하나로 변환하려고 시도했지만, 문자열에 적절한 형식이 없음을 나타내기 위해 발생한다.”[https://docs.oracle.com/javase/7/docs/api/java/lang/NumberFormatException.html]

2. Separation of Concerns

“컴퓨터 과학에서 separation of concerns는 컴퓨터 프로그램을 구별된 부분으로 분리하는 디자인 원칙으로, 각 부문은 개개의 관심사를 해결한다. 관심사의 분리는 정보를 잘 정의된 인터페이스가 있는 코드 부분 안에 캡슐화시킴으로써 달성한다. 관심사가 잘 분리될 때 독립적인 개발과 업그레이드 외에도 모듈 재사용을 위한 더 높은 정도의 자유가 있다. 모듈이 인터페이스 뒤에서 이러한 관심사의 세세한 부분을 숨기기 때문에 자유도가 높아짐으로써 다른 부분의 세세한 사항을 모르더라도, 또 해당 부분들에 상응하는 변경을 취하지 않더라도 하나의 관심사의 코드 부분을 개선하거나 수정할 수 있게 된다.”[위키백과]

우리가 만들 계산기 프로그램도 예외 처리를 직접 만들어야 독립적으로 처리할 수 있고 재사용이 쉬울 것이다.

3. Refactoring

코드는 주기적으로 정리정돈 해야 한다. “소프트웨어 공학에서 '결과의 변경 없이 코드의 구조를 재조정함'을 뜻한다. 주로 가독성을 높이고 유지보수를 편하게 한다. 사용자가 보는 외부 화면은 그대로 두면서 내부 논리나 구조를 바꾸고 개선하는 ‘유지보수’ 행위이다. Refactoring은 코드의 가독성을 향상시키고 복잡성을 감소시키는 효과를 가지며, 이는 소스 코드의 유지 보수성을 개선하고 확장성을 개선하기 위해 더 단순하고, 깔끔하거나, 표현력이 뛰어난 내부 아키텍처 또는 객체 모델을 만들 수 있게 한다”[위키백과]