1. ArrayList 와 LinkedList 있는 이유, 차이점

* ArrayList -> 내부적으로 데이터를 배열에서 관리하는 자료 구조
  + 데이터 검색에는 유리하지만, 데이터 추가, 삭제에는 성능이 떨어짐
  + 정적인 데이터 활용, 조회가 빈번한 데이터 일 때 사용
* LinkedList -> 데이터를 저장하는 각 노드가 이전 노드와 다음 노드의 상태만 알고 있는 자료 구조
  + 데이터 추가, 삭제에는 유리하지만, 데이터 검색에서 성능이 떨어짐
  + 동적으로 추가/삭제 요구사항이 빈번한 데이터 일 때 사용

왜 SuperList 같이 하나로 제공해주지 안고, 여러 개로 나눠서 컬렉션을 제공해줄까요?

일단 정의부터

Collection?

* 데이터를 저장하고, 검색하고 처리하는 데이터 묶음

Collection Framework?

* 데이터를 처리하기 위한 통일된 구조

배열을 쓰는 ArrayList 와 LinkedList 는 사용하는 용도 자체가 다르기 때문에 따로 나눠서 컬렉션을 제공한다!

1. 예외처리문 없어도 예외처리가 가능하지 않을까요? 왜 필요할까요?

예외는 어떤 상황에서 발생할까요?

메서드에서 예외를 던졌는데, 처리하지 않는다면 어떻게 될까요?

CompileException(Checked) 과 RuntimeException(unchecked) 의 차이

* 예외로 인한 비정상적인 종료를 막고 예외 상황일 때 정상적인 프로그램 실행을 계속 진행 할 수 있도록 하는 것이 예외 처리가 필요한 이유
* 잘못된 코드, 부정확한 데이터, 예외적인 상황에 의하여 오류가 났을 때
* 에러가 뜨면서 프로그램에 종료된다

CompileException(Checked) > 컴파일 시점에서 예외를 catch 하는지 정적으로 확인

컴파일 시점에서 예외에 대한 처리를 하지 않으면 컴파일 에러 발생

RuntimeException(unchecked) > 컴파일은 성공했지만 코드 로직에 의해 발생하는 예외 를 catch 하는 지 여부는 확인하지 않는다.

1. 자바 코드는 어떻게 컴퓨터에서 실행될까요?

컴파일과 인터프리터에 대해서 간략하게 설명해주세요!

바이트 코드에 대해서 설명해주세요. 왜 가상머신이나 바이트 코드를 사용할까요?

컴파일 과정 : 자바 코드 -> 자바 클래스 (바이트 코드) 파일 생성

실행 과정 : JVM(바이트 코드 JVM으로 가져옴 -> Byte Verifirer 로 바이트 코드 내부 오류 / 버그 검증과정을 거친다 -> 인터프리터로 OS 에 맞게 한 줄 씩 실행 ) -> OS -> 하드웨어

컴파일러 -> 프로그램 전체를 스캔하여 이를 모두 기계어로 번역하다.

전체를 스캔하기 때문에 대개 컴파일러는 초시 스캔 시간이 오래 걸림.

컴파일러는 초기 스캔을 마치면 실행 파일을 만들어 놓고 다음에 실행할 때 이전에 만들어 놓았던 실행파일을 실행하기 때문이다.

컴파일러는 고급언어로 작성된 소스를 기계어로 번역, Object Code 파일 생성, 이 코드를 묶어서 하나의 실행 파일로 다시 만드는 링킹 작업을 한다. 오류 메시지를 생성할 때 전체 코드를 검사한 후에 오류 메시지를 새성한다. 그래서 실행 전에 오류를 발견 못한다.

인터프리터 -> 프로그램 실행 시 한 번에 한 문장씩 번역한다

컴파일러와 같은 오브젝트 코드 생성 과정이 없기 때문에 메모리 효율이 좋다

프로그램을 실행 시키고 나서 오류를 발견하면 바로 실행을 중지 시킨다 실행 후에 오류를 알 수 있다.

오브젝트 코드?

1. 백엔드 개발자가 하는 일은 무엇인가