스트림(Stream)이란

Stream은 자바 8에서 소개된 기능으로, 데이터 컬렉션(예: 리스트, 배열, 맵 등)을 다루는데 편리하고 간결한 방법을 제공하는 API이다.

Stream은 컬렉션의 요소를 처리하거나 변환하는데 사용되며, 함수형 프로그래밍과 람다식을 활용하여 데이터를 처리하는 강력한 도구이다.

스트림은 컬렉션, 배열, 파일 등을 스트림으로 변환하여 사용한다.

컬렉션의 경우 stream() 메서드를 호출하여 스트림을 얻을 수 있다

스트림(Stream)

```
List<String> list = new ArrayList<>();
Stream<String> stram = list.stream();
```

Stream API는 중간 연산과 최종 연산으로 나뉜다. 중간 연산은 스트림을 변환하거나 필터링하는 작업을 수행하며, 여러 개의 중간 연산을 연결하여 복잡한 작업을 수행할 수 있다. 최종 연산은 스트림의 처리 결과를 얻는 작업으로, 스트림을 닫는 작업을 수행한다.

중간 연산

filter(Predicate<T> predicate) 주어진 조건에 맞는 요소 선택한다.

map(Function<T, R> mapper) 요소를 다른 형태로 변환한다.

sorted(Comparator<T> comparator) 요소를 정렬한다.

distinct() 중복된 요소를 제거한다.

```
Stream<String> listStream = fruitList.stream();

listStream.sorted();
```

최종연산

forEach(Consumer<T> action) 각 요소에 대한 작업을 수행한다.

collect(Collector<T, A, R> collector) 요소를 컬렉션으로 수집한다.

count() 요소의 개수를 반환한다.

reduce(BinaryOperator<T> accumulator) 요소를 결합하여 결과를 얻는다.

```
listStream.sorted().forEach(element -> {
    System.out.println(element);
});
```

Stream API의 주요 특징

특징	설명
데이터 소스	스트림은 컬렉션, 배열, 파일 등의 데이터 소스에서 요소들을 처리한다.
람다식	스트림을 사용하면 람다식을 활용하여 데이터 처리 로직을 정의할 수 있다. 이를 통해 코드의 가독성과 유지보수성이 높아지며, 간결한 함수형 프로그래밍 스타일로 작성할 수 있다.
중간 연산 / 최종 연산	스트림은 중간 연산과 최종 연산으로 나뉜다. 중간 연산은 스트림을 반환하여 연속적으로 연산을 수행하며, 최종 연산은 스트림의 요소를 처리하고 결과를 반환한다.
지연 실행	스트림은 지연 실행을 지원한다. 즉, 최종 연산이 호출될 때까지 중간 연산은 실제로 수행되지 않고, 최종 연산이 호출될 때 한 번에 모든 중간 연산이 수행된다. 이는 성능을 향상시킬 수 있는 장점 중 하나이다.
병렬 처리	스트림은 멀티코어 프로세서를 활용하여 병렬 처리를 지원한다. 이를 통해 대량의 데이터를 더 빠르게 처리할 수 있다.