Map은 java 스트림에서 요소를 변환하거나 매핑하는데 사용되는 중용한 연산자

‘map’ 연사자는 스트림 내의 각 요소를 새로운 값으로 변환하거나 특정 함수 또는 매핑 작업을 적용하는데 사용 됩니다. 다음은 ‘Map’을 사용하는 일반적인 상황과 용도입니다.

1. 요소 변환 : 스트림 내의 요소를 다른 형태로 변환할 때 map을 사용합니다. 예를 들어. 숫자 리스트에서 각 숫자를 제곱하는 경우

List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);

List<Integer> squaredNumbers = numbers.stream()

.map(n -> n \* n)

.collect(Collectors.toList());

1. 특정 속성 추출 : 객체 목록에서 특정속성을 추출하여 새로운 목록을 생성할 때 map을 사용합니다. 이는 객체에서 원하는 정보를 추출할 때 유용합니다.

List<Person> people = ... // 어떤 객체 목록

List<String> names = people.stream()

.map(Person::getName) // Person 객체에서 이름 추출

.collect(Collectors.toList());

3. 객체 변환 : 개게를 다른 형식으로 변환하거나 특정 필드를 업데이트할 때 map을 사용합니다. 예를 들어, 날짜를 포멧팅하거나 필드를 수정하는 경우.

List<Event> events = ... // 어떤 이벤트 목록

List<String> eventNames = events.stream()

.map(event -> {

event.setDescription(event.getDescription().toUpperCase()); // 설명을 대문자로 변경

return event;

})

.map(Event::getName) // Event 객체를 이름 문자열로 변환

.collect(Collectors.toList());

4. Optional 값을 추출: map은 Optional 객체 내에서 값을 추출할 때도 사용됩니다. 이것은 Optional을 통해 안전하게 값에 접근하는데 도움이 됩니다.

Optional<String> optionalValue = ... // 어떤 Optional

Optional<Integer> mappedValue = optionalValue.map(Integer::parseInt);

5. 스트림 평면화 : map은 스트림의 요소를 다른 스트림으로 매핑하는데 사용되며, 이를 통해 스트림을 평면화할 수 있습니다. 이것은 스트림의 중첩 구조를 해제하거나 평탄화할 때 유용합니다.

List<List<Integer>> nestedList = Arrays.asList(

Arrays.asList(1, 2, 3),

Arrays.asList(4, 5),

Arrays.asList(6, 7, 8)

);

List<Integer> flattenedList = nestedList.stream()

.flatMap(List::stream)

.collect(Collectors.toList());

System.out.println(flattenedList);

[1, 2, 3, 4, 5, 6, 7, 8]