Java Collection Framework
- Stream 소개

학습내용

1. Stream 소개
2. Stream 처리 워밍업
3. Stream 처리 학습 방법

학습목표

- Java Collection Framework의 Stream의 개념을 설명할 수 있다.
- Stream 처리를 가볍게 수행하고 학습 방법을 설명할 수 있다.

Stream

Collection의 instance들에 대한
전반적인 연산을 지원하는 Abstraction

Stream의 Method들이 모여 데이터 처리의
파이프라인을 형성함

Stream


Stream
4
3
5
1
2
파이프라인
필터링 : 파란색이 아닌 것 거르기
Stream Method
변환 : 별 → 네모
정렬 : 숫자 오름차순
결과를 List로 모으기
List
2
4
5

Stream 생성 방법

Stream<T> Stream.of(T… values)

1. instance들을 통해 만들기

```
public static<T> Stream<T> of(T... values) {
    return Arrays.stream(values);
}
```

예 : Stream<Integer>

```
Stream<Integer> intStream = Stream.of(1,2,3,4);
```

Stream 생성 방법

Stream<T> Stream.of(T… values)

2. 기존의 Array를 통해 만드는 방법

```
public static <T> Stream<T> stream(T[] array) {
    return stream(array, 0, array.length);
}
```

예 : Stream<Integer>

```
Integer[] intArray = Arrays.stream(new Integer[]
{1,2,3,4});
```

Stream 생성 방법

Stream<T> Stream.of(T… values)

3. Collection의 stream Method로 생성

```
default Stream<E> stream() {
    return StreamSupport.stream(spliterator(), false);
}
```

예 : Stream<Integer>

```
List<Integer> list = List.of(1,2,3,4);
new ArrayList<Integer>(list).stream();
new LinkedList<Integer>(list).stream();
new HashSet<Integer>(list).stream();
new TreeSet<Integer>(list).stream();
```

Stream 생성 방법

Stream<T> Stream.of(T… values)

3. Collection의 stream Method로 생성

```
default Stream<E> stream() {
    return StreamSupport.stream(spliterator(), false);
}
```

예 : Stream<Integer>

```
Map<String, String> map = Map.of("Jack", "Data Engineering");
new HashMap<String,String>(map).entrySet().stream();
new HashMap<String,String>(map).keySet().stream();
new HashMap<String,String>(map).values().stream();
new TreeMap<String,String>(map).keySet().stream();
```

Stream 처리 워밍업 예제1

forEach : Stream의 각 instance를 소비 수행됨

```
Stream.of(1,2,3,4,5).forEach(new
Consumer<Integer>() {
    @Override
    public void accept(Integer t) {
        System.out.println(t);
    }
});
```

consumer라는 functional interface의 구현 받음

Stream 처리 워밍업 예제1

map: Stream의 각 instance를 변환 실제로 수행 안됨

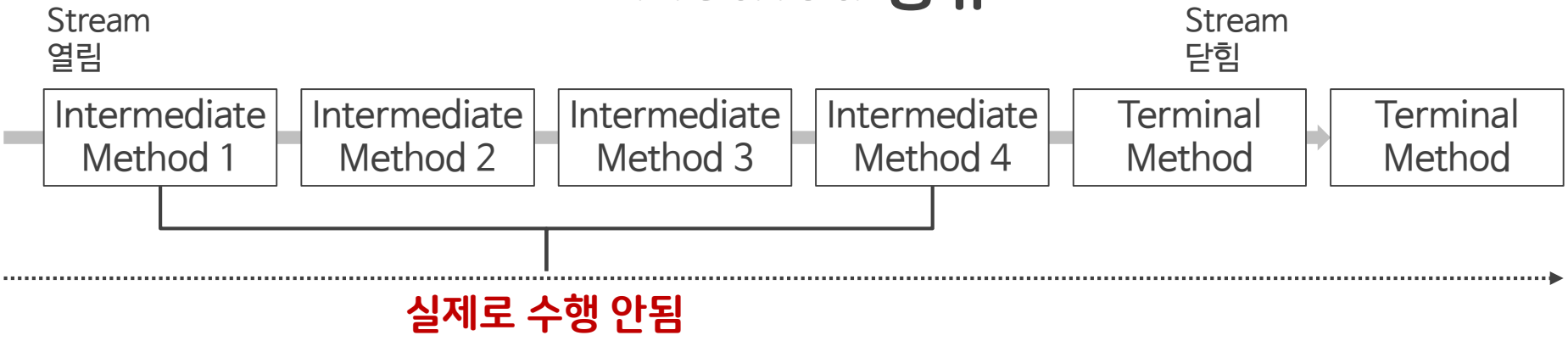
```
Stream.of(1,2,3,4,5).map(new Function<Integer,
Boolean>() {
  @Override
  public Boolean apply(Integer t) {
    if( t % 2 == 0)
      return true;
    else
      return false;
  }
});
```

정수값을 짝수 혹은 홀수의 boolean으로 변환 수행

[Stream 처리 워밍업]

예제1

Method 종류

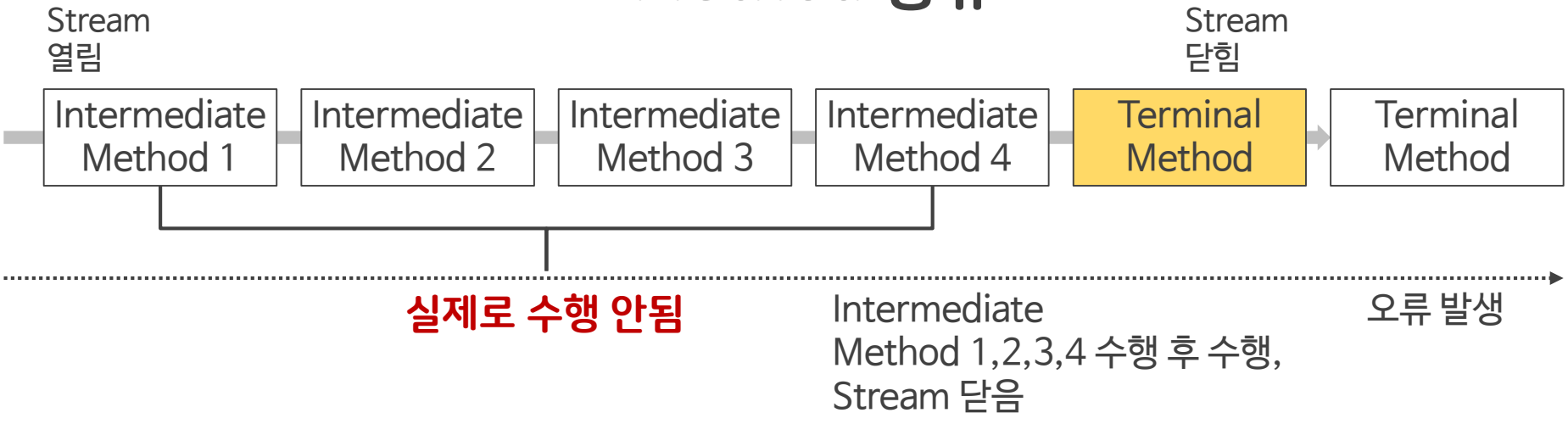


Intermediate Method

Terminal Method 호출 이후 수행시작 (Lazy Evaluation)

예 : map, filter, distinct, sorted, peek, limit, parallel

Method 종류



Terminal Methods

Stream 파이프라인의 처리를 시작하게 함(Eager Evaluation)

**Terminal Method 수행 이후에는 파이프라인이 종료되며
종료 이후로는 Stream 사용 불가**

[실습]

intermediate, terminal 디버깅

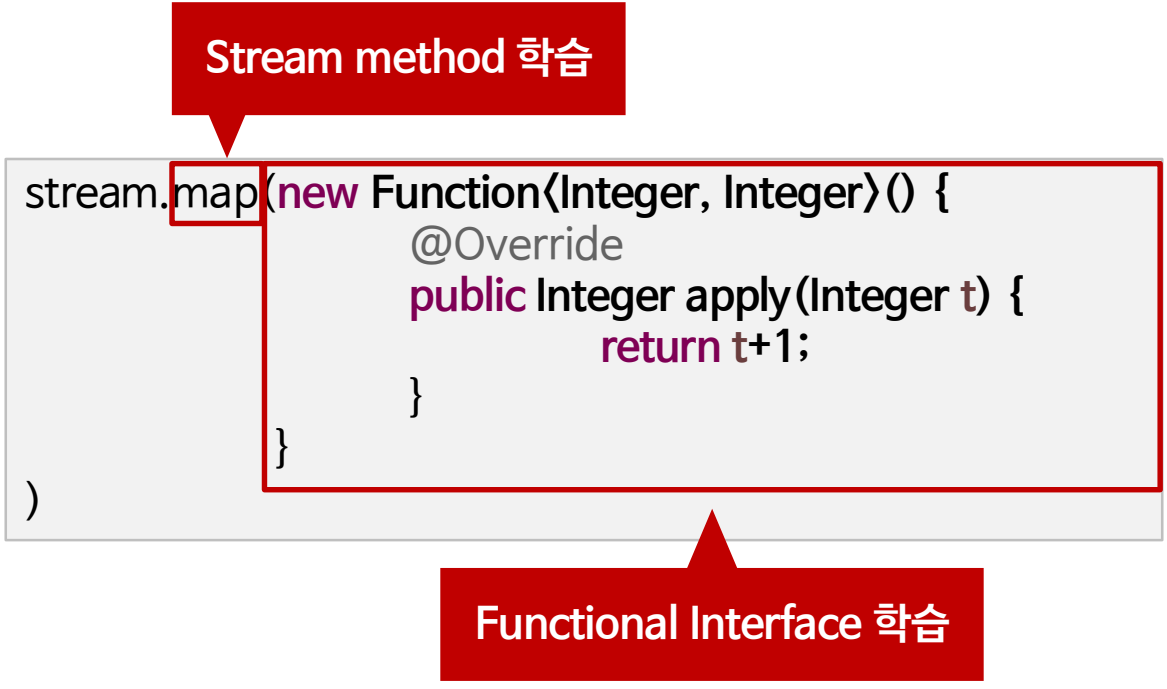
기본적인 Stream 처리를 통한
intermediate/terminal method 개념 학습

```
stream.map(new Function<Integer, Integer>() {
            @Override
            public Integer apply(Integer t) {
                        if ( e % 2 == 0 ) return true;
                        else return false;
            }
}).forEach( e -> System.out.println(e) );
```

Stream 학습 방법

```
stream.map(new Function<Integer, Integer>() {
                    @Override
                    public Integer apply(Integer t) {
                            return t+1;
                    }
            }
)
```

Stream 학습 방법


Stream method 학습
stream.map(new Function<Integer, Integer>() {
@Override
public Integer apply(Integer t) {
return t+1;
}
}
)
Functional Interface 학습

Remind

Java Collection Framework의 Stream 소개