



이것이 자바다 개정판

17.1 스트림이란?

17.2 내부 반복자

17.3 중간 처리와 최종 처리

17.4 리소스로부터 스트림 얻기

17.5 요소 걸러내기(필터링)

17.6 요소 변환(매핑)

17.7 요소 정렬

17.8 요소를 하나씩 처리(루핑)

17.9 요소 조건 만족 여부(매칭)

17.10 요소 기본 집계

17.11 요소 커스텀 집계

17.12 요소 수집

17.13 요소 병렬 처리

스트림

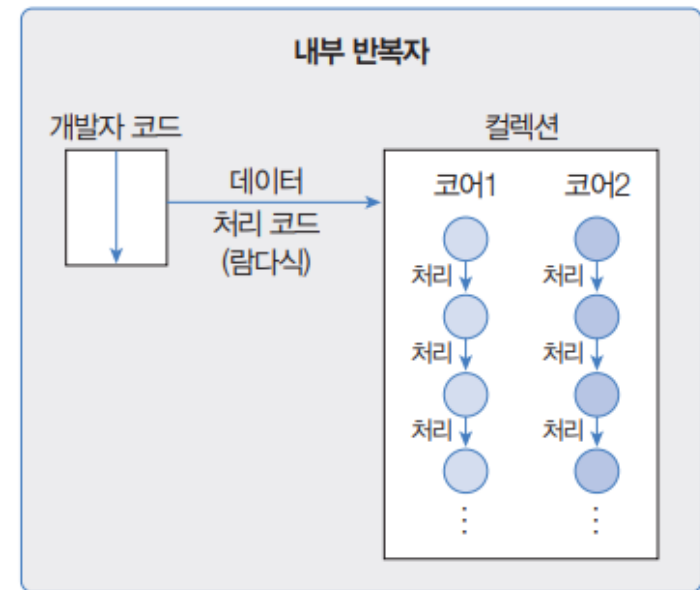
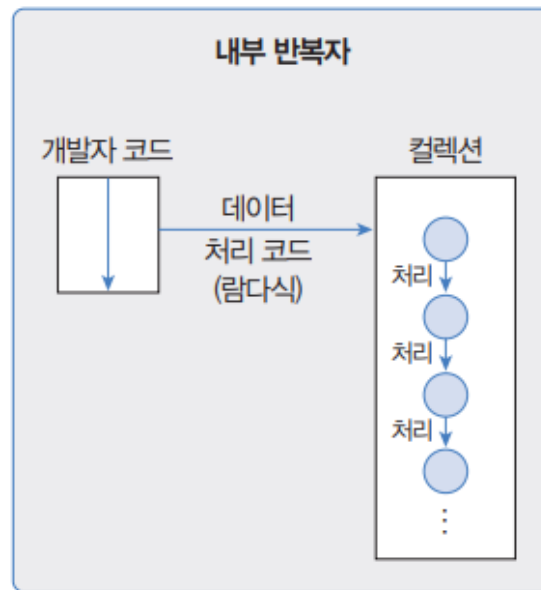
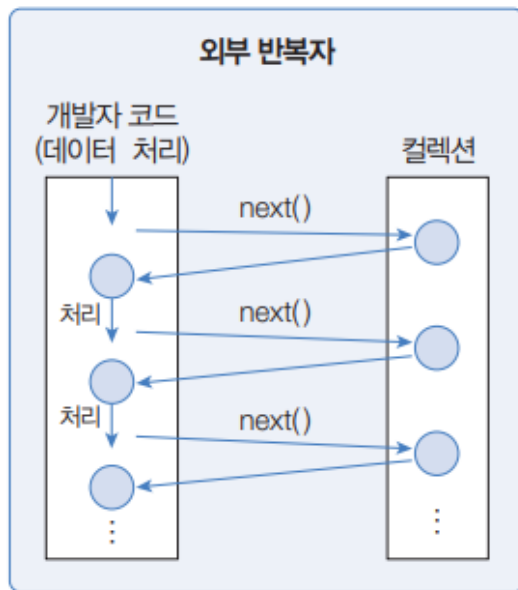
- Java 8부터 컬렉션 및 배열의 요소를 반복 처리하기 위해 스트림 사용
- 요소들이 하나씩 흘러가면서 처리된다는 의미

```
Stream<String> stream = list.stream();  
stream.forEach( item -> //item 처리 );
```

- List 컬렉션의 stream() 메소드로 Stream 객체를 얻고, forEach() 메소드로 요소를 어떻게 처리할지를 랴다식으로 제공
- 스트림과 Iterator 차이점
 - 1) 내부 반복자이므로 처리 속도가 빠르고 병렬 처리에 효율적
 - 2) 랴다식으로 다양한 요소 처리를 정의
 - 3) 중간 처리와 최종 처리를 수행하도록 파이프 라인을 형성

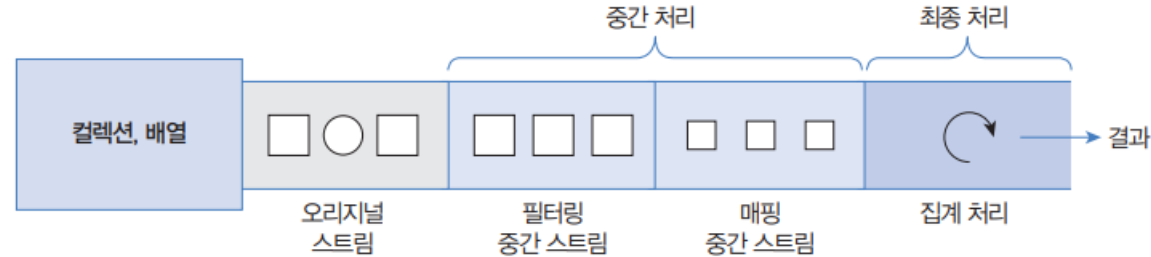
내부 반복자

- 요소 처리 방법을 컬렉션 내부로 주입시켜서 요소를 반복 처리
- 개발자 코드에서 제공한 데이터 처리 코드(람다식)를 가지고 컬렉션 내부에서 요소를 반복 처리
- 내부 반복자는 멀티 코어 CPU를 최대한 활용하기 위해 요소들을 분배시켜 병렬 작업 가능

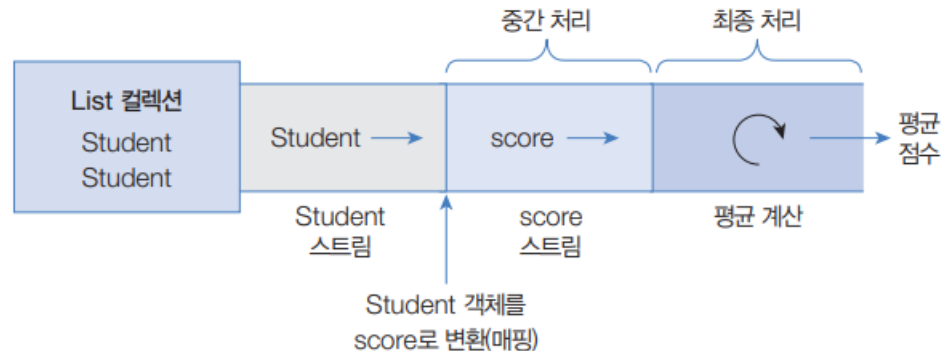


스트림 파이프라인

- 컬렉션의 오리지널 스트림 뒤에 필터링 중간 스트림이 연결될 수 있고, 그 뒤에 매핑 중간 스트림이 연결될 수 있음

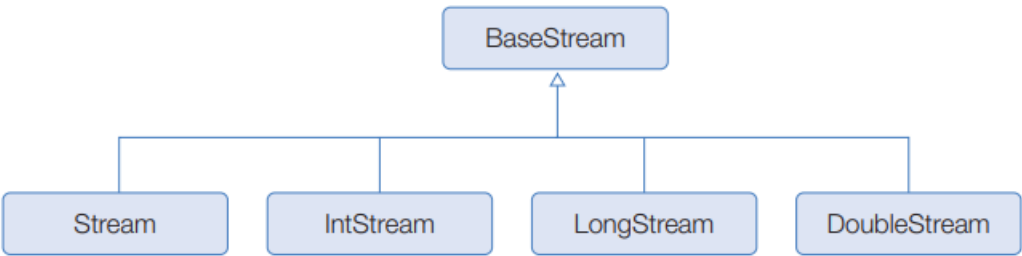


- 오리지널 스트림과 집계 처리 사이의 중간 스트림들은 최종 처리를 위해 요소를 걸러내거나(필터링), 요소를 변환시키거나(매핑), 정렬하는 작업을 수행
- 최종 처리는 중간 처리에서 정제된 요소들을 반복하거나, 집계(카운팅, 총합, 평균) 작업을 수행



스트림 인터페이스

- java.util.stream 패키지에는 BaseStream 인터페이스를 부모로 한 자식 인터페이스들은 상속 관계
- BaseStream에는 모든 스트림에서 사용할 수 있는 공통 메소드들이 정의



리턴 타입	메소드(매개변수)	소스
Stream<T>	java.util.Collection.stream() java.util.Collection.parallelStream()	List 컬렉션 Set 컬렉션
Stream<T> IntStream LongStream DoubleStream	Arrays.stream(T[]), Stream.of(T[]) Arrays.stream(int[]), IntStream.of(int[]) Arrays.stream(long[]), LongStream.of(long[]) Arrays.stream(double[]), DoubleStream.of(double[])	배열
IntStream	IntStream.range(int, int) IntStream.rangeClosed(int, int)	int 범위
LongStream	LongStream.range(long, long) LongStream.rangeClosed(long, long)	long 범위
Stream<Path>	Files.list(Path)	디렉토리
Stream<String>	Files.lines(Path, Charset)	텍스트 파일
DoubleStream IntStream LongStream	Random.doubles(...) Random.ints() Random.longs()	랜덤 수

컬렉션으로부터 스트림 얻기

- `java.util.Collection` 인터페이스는 스트림과 `parallelStream()` 메소드를 가지고 있어 자식 인터페이스인 `List`와 `Set` 인터페이스를 구현한 모든 컬렉션에서 객체 스트림을 얻을 수 있음

배열로부터 스트림 얻기

- `java.util.Arrays` 클래스로 다양한 종류의 배열로부터 스트림을 얻을 수 있음

숫자 범위로부터 스트림 얻기

- `IntStream` 또는 `LongStream`의 정적 메소드인 `range()`와 `rangeClosed()` 메소드로 특정 범위의 정수 스트림을 얻을 수 있음

파일로부터 스트림 얻기

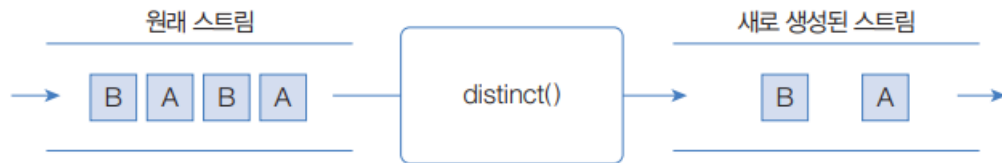
- `java.nio.file.Files`의 `lines()` 메소드로 텍스트 파일의 행 단위 스트림을 얻을 수 있음

필터링

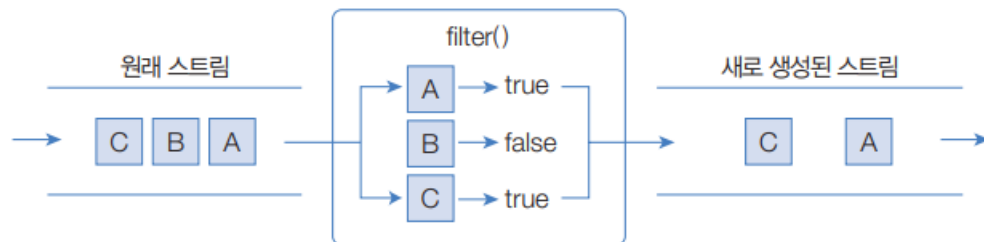
- 필터링은 요소를 걸러내는 중간 처리 기능

리턴 타입	메소드(매개변수)	설명
Stream IntStream LongStream DoubleStream	distinct()	- 중복 제거
	filter(Predicate<T>)	- 조건 필터링
	filter(IntPredicate)	- 매개 타입은 요소 타입에 따른 함수형 인터페이스이므로 람다식으로 작성 가능
	filter(LongPredicate)	
	filter(DoublePredicate)	

- distinct() 메소드: 요소의 중복을 제거



- filter() 메소드: 매개값으로 주어진 Predicate가 true를 리턴하는 요소만 필터링

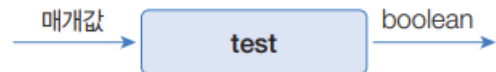


17.5 요소 걸러내기(필터링)

- Predicate: 함수형 인터페이스

인터페이스	추상 메소드	설명
Predicate<T>	boolean test(T t)	객체 T를 조사
IntPredicate	boolean test(int value)	int 값을 조사
LongPredicate	boolean test(long value)	long 값을 조사
DoublePredicate	boolean test(double value)	double 값을 조사

- 모든 Predicate는 매개값을 조사한 후 boolean을 리턴하는 test() 메소드를 가지고 있다.



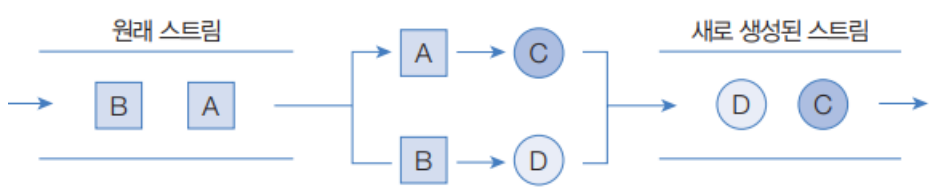
```
T -> { ... return true }  
또는  
T -> true; //return 문만 있을 경우 중괄호와 return 키워드 생략 가능
```

매핑

- 스트림의 요소를 다른 요소로 변환하는 중간 처리 기능
- 매핑 메소드: mapXxx(), asDoubleStream(), asLongStream(), boxed(), flatMapXxx() 등

요소를 다른 요소로 변환

- mapXxx() 메소드: 요소를 다른 요소로 변환한 새로운 스트림을 리턴



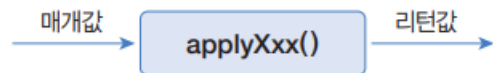
리턴 타입	메소드(매개변수)	요소 → 변환 요소
Stream<R>	map(Function<T, R>)	T → R
IntStream LongStream DoubleStream	mapToInt(ToIntFunction<T>)	T → int
	mapToLong(ToLongFunction<T>)	T → long
	mapToDouble(ToDoubleFunction<T>)	T → double
Stream<U>	mapToObj(IntFunction<U>)	int → U
	mapToObj(LongFunction<U>)	long → U
	mapToObj(DoubleFunction<U>)	double → U
DoubleStream DoubleStream IntStream LongStream	mapToDouble(IntToDoubleFunction)	int → double
	mapToDouble(LongToDoubleFunction)	long → double
	mapToInt(DoubleToIntFunction)	double → int
	mapToLong(DoubleToLongFunction)	double → long

17.6 요소 변환(매핑)

- 매개타입인 Function은 함수형 인터페이스

인터페이스	추상 메소드	매개값 → 리턴값
Function<T,R>	R apply(T t)	T → R
IntFunction<R>	R apply(int value)	int → R
LongFunction<R>	R apply(long value)	long → R
DoubleFunction<R>	R apply(double value)	double → R
ToIntFunction<T>	int applyAsInt(T value)	T → int
ToLongFunction<T>	long applyAsLong(T value)	T → long
ToDoubleFunction<T>	double applyAsDouble(T value)	T → double
IntToLongFunction	long applyAsLong(int value)	int → long
IntToDoubleFunction	double applyAsDouble(int value)	int → double
LongToIntFunction	int applyAsInt(long value)	long → int
LongToDoubleFunction	double applyAsDouble(long value)	long → double
DoubleToIntFunction	int applyAsInt(double value)	double → int
DoubleToLongFunction	long applyAsLong(double value)	double → long

- 모든 Function은 매개값을 리턴값으로 매핑(변환)하는 applyXxx() 메소드를 가짐



```
T -> { ... return R; }  
또는  
T -> R;  //return 문만 있을 경우 중괄호와 return 키워드 생략 가능
```

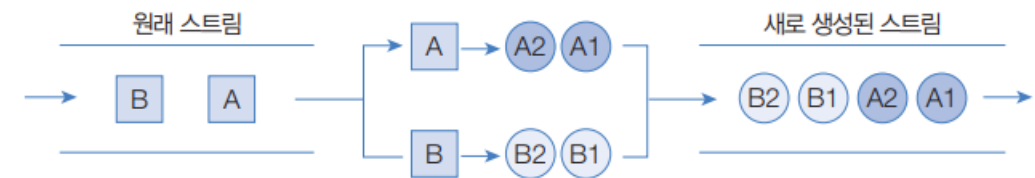
17.6 요소 변환(매핑)

- 기본 타입 간의 변환이거나 기본 타입 요소를 래퍼(Wrapper) 객체 요소로 변환하려면 간편화 메소드를 사용할 수 있음

리턴 타입	메소드(매개변수)	설명
LongStream	asLongStream()	int → long
DoubleStream	asDoubleStream()	int → double long → double
Stream<Integer> Stream<Long> Stream<Double>	boxed()	int → Integer long → Long double → Double

요소를 복수 개의 요소로 변환

- flatMapXxx() 메소드: 하나의 요소를 복수 개의 요소들로 변환한 새로운 스트림을 리턴



리턴 타입	메소드(매개변수)	요소 → 변환 요소
Stream<R>	flatMap(Function<T, Stream<R>>())	T → Stream<R>
DoubleStream	flatMap(DoubleFunction<DoubleStream>())	double → DoubleStream
IntStream	flatMap(IntFunction<IntStream>())	int → IntStream
LongStream	flatMap(LongFunction<LongStream>())	long → LongStream
DoubleStream	flatMapToDouble(Function<T, DoubleStream>())	T → DoubleStream
IntStream	flatMapToInt(Function<T, IntStream>())	T → IntStream
LongStream	flatMapToLong(Function<T, LongStream>())	T → LongStream

정렬

- 요소를 오름차순 또는 내림차순으로 정렬하는 중간 처리 기능

리턴 타입	메소드(매개변수)	설명
Stream<T>	sorted()	Comparable 요소를 정렬한 새로운 스트림 생성
Stream<T>	sorted(Comparator<T>)	요소를 Comparator에 따라 정렬한 새 스트림 생성
DoubleStream	sorted()	double 요소를 올림차순으로 정렬
IntStream	sorted()	int 요소를 올림차순으로 정렬
LongStream	sorted()	long 요소를 올림차순으로 정렬

Comparable 구현 객체의 정렬

- 스트림의 요소가 객체일 경우 객체가 Comparable을 구현하고 있어야만 sorted() 메소드를 사용하여 정렬 가능. 그렇지 않으면 ClassCastException 발생

```
public Xxx implements Comparable {
    ...
}
```

```
List<Xxx> list = new ArrayList<>();
Stream<Xxx> stream = list.stream();
Stream<Xxx> orderedStream = stream.sorted();
```

Comparator를 이용한 정렬

- 요소 객체가 Comparable을 구현하고 있지 않다면, 비교자를 제공하면 요소를 정렬시킬 수 있음

```
sorted((o1, o2) -> { ... })
```

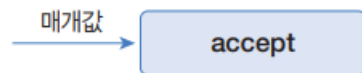
- 괄호 안에는 o1이 o2보다 작으면 음수, 같으면 0, 크면 양수를 리턴하도록 작성
- o1과 o2가 정수일 경우에는 Integer.compare(o1, o2)를, 실수일 경우에는 Double.compare(o1, o2)를 호출해서 리턴값을 리턴 가능

루핑

- 스트림에서 요소를 하나씩 반복해서 가져와 처리하는 것

리턴 타입	메소드(매개변수)	설명
Stream<T> IntStream DoubleStream	peek(Consumer<? super T>)	T 반복
	peek(IntConsumer action)	int 반복
	peek(DoubleConsumer action)	double 반복
void	forEach(Consumer<? super T> action)	T 반복
	forEach(IntConsumer action)	int 반복
	forEach(DoubleConsumer action)	double 반복

- 매개타입인 Consumer는 함수형 인터페이스. 모든 Consumer는 매개값을 처리(소비)하는 accept() 메소드를 가지고 있음



```
T -> { ... }  
또는  
T -> 실행문; //하나의 실행문만 있을 경우 중괄호 생략
```

매칭

- 요소들이 특정 조건에 만족하는지 여부를 조사하는 최종 처리 기능
- `allMatch()`, `anyMatch()`, `noneMatch()` 메소드는 매개값으로 주어진 Predicate가 리턴하는 값에 따라 `true` 또는 `false`를 리턴

리턴 타입	메소드(매개변수)	조사 내용
boolean	<code>allMatch(Predicate<T> predicate)</code> <code>allMatch(IntPredicate predicate)</code> <code>allMatch(LongPredicate predicate)</code> <code>allMatch(DoublePredicate predicate)</code>	모든 요소가 만족하는지 여부
boolean	<code>anyMatch(Predicate<T> predicate)</code> <code>anyMatch(IntPredicate predicate)</code> <code>anyMatch(LongPredicate predicate)</code> <code>anyMatch(DoublePredicate predicate)</code>	최소한 하나의 요소가 만족하는지 여부
boolean	<code>noneMatch(Predicate<T> predicate)</code> <code>noneMatch(IntPredicate predicate)</code> <code>noneMatch(LongPredicate predicate)</code> <code>noneMatch(DoublePredicate predicate)</code>	모든 요소가 만족하지 않는지 여부

집계

- 최종 처리 기능으로 요소들을 처리해서 카운팅, 합계, 평균값, 최대값, 최소값 등 하나의 값으로 산출하는 것

스트림이 제공하는 기본 집계

- 스트림은 카운팅, 최대, 최소, 평균, 합계 등을 처리하는 다음과 같은 최종 처리 메소드를 제공

리턴 타입	메소드(매개변수)	설명
long	count()	요소 개수
OptionalXXX	findFirst()	첫 번째 요소
Optional<T> OptionalXXX	max(Comparator<T>) max()	최대 요소
Optional<T> OptionalXXX	min(Comparator<T>) min()	최소 요소
OptionalDouble	average()	요소 평균
int, long, double	sum()	요소 총합

Optional 클래스

- Optional, OptionalDouble, OptionalInt, OptionalLong 클래스는 단순히 집계값만 저장하는 것이 아니라, 집계값이 없으면 디폴트 값을 설정하거나 집계값을 처리하는 Consumer를 등록

리턴 타입	메소드(매개변수)	설명
boolean	isPresent()	집계값이 있는지 여부
T double int long	orElse(T) orElse(double) orElse(int) orElse(long)	집계값이 없을 경우 디폴트 값 설정
void	ifPresent(Consumer) ifPresent(DoubleConsumer) ifPresent(IntConsumer) ifPresent(LongConsumer)	집계값이 있을 경우 Consumer에서 처리

최종 처리에서 average 사용 시 요소 없는 경우를 대비하는 방법

- 1) isPresent() 메소드가 true를 리턴할 때만 집계값을 얻는다.
- 2) orElse() 메소드로 집계값이 없을 경우를 대비해서 디폴트 값을 정해놓는다.
- 3) ifPresent() 메소드로 집계값이 있을 경우에만 동작하는 Consumer 랴다식을 제공한다.

스트림이 제공하는 메소드

- 스트림은 기본 집계 메소드인 `sum()`, `average()`, `count()`, `max()`, `min()`을 제공하지만, 다양한 집계 결과물을 만들 수 있도록 `reduce()` 메소드도 제공

인터페이스	리턴 타입	메소드(매개변수)
Stream	Optional<T>	<code>reduce(BinaryOperator<T> accumulator)</code>
	T	<code>reduce(T identity, BinaryOperator<T> accumulator)</code>
IntStream	OptionalInt	<code>reduce(IntBinaryOperator op)</code>
	int	<code>reduce(int identity, IntBinaryOperator op)</code>
LongStream	OptionalLong	<code>reduce(LongBinaryOperator op)</code>
	long	<code>reduce(long identity, LongBinaryOperator op)</code>
DoubleStream	OptionalDouble	<code>reduce(DoubleBinaryOperator op)</code>
	double	<code>reduce(double identity, DoubleBinaryOperator op)</code>

- `reduce()`는 스트림에 요소가 없을 경우 예외가 발생하지만, `identity` 매개값이 주어지면 이 값을 디폴트 값으로 리턴

필터링한 요소 수집

- Stream의 collect(Collector<T,A,R> collector) 메소드는 필터링 또는 매핑된 요소들을 새로운 컬렉션에 수집하고, 이 컬렉션을 리턴
- 매개값인 Collector는 어떤 요소를 어떤 컬렉션에 수집할 것인지를 결정
- 타입 파라미터의 T는 요소, A는 누적기accumulator, 그리고 R은 요소가 저장될 컬렉션

리턴 타입	메소드(매개변수)	인터페이스
R	collect(Collector<T,A,R> collector)	Stream

리턴 타입	메소드	설명
Collector<T, ?, List<T>>	toList()	T를 List에 저장
Collector<T, ?, Set<T>>	toSet()	T를 Set에 저장
Collector<T, ?, Map<K,U>>	toMap(Function<T,K> keyMapper, Function<T,U> valueMapper)	T를 K와 U로 매핑하여 K를 키로, U를 값으로 Map에 저장

요소 그룹핑

- Collectors.groupingBy () 메소드에서 얻은 Collector를 collect() 메소드를 호출할 때 제공
- groupingBy()는 Function을 이용해서 T를 K로 매핑하고, K를 키로 해 List<T>를 값으로 갖는 Map 컬렉션을 생성

리턴 타입	메소드
Collector<T,?,Map<K,List<T>>>	groupingBy(Function<T, K> classifier)

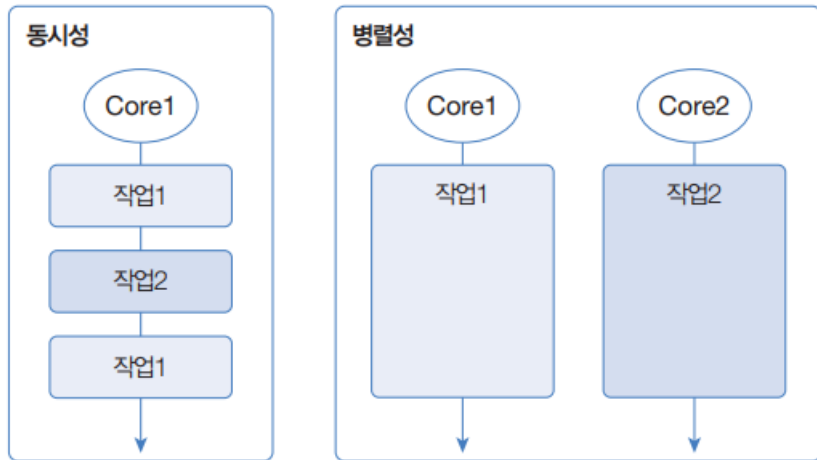
K		
0	...	n
T	...	T

- Collectors.groupingBy() 메소드는 그룹핑 후 매핑 및 집계(평균, 카운팅, 연결, 최대, 최소, 합계)를 수행할 수 있도록 두 번째 매개값인 Collector를 가질 수 있음

리턴 타입	메소드(매개변수)	설명
Collector	mapping(Function, Collector)	매핑
Collector	averagingDouble(ToDoubleFunction)	평균값
Collector	counting()	요소 수
Collector	maxBy(Comparator)	최대값
Collector	minBy(Comparator)	최소값
Collector	reducing(BinaryOperator<T>) reducing(T identity, BinaryOperator<T>)	커스텀 집계 값

동시성과 병렬성

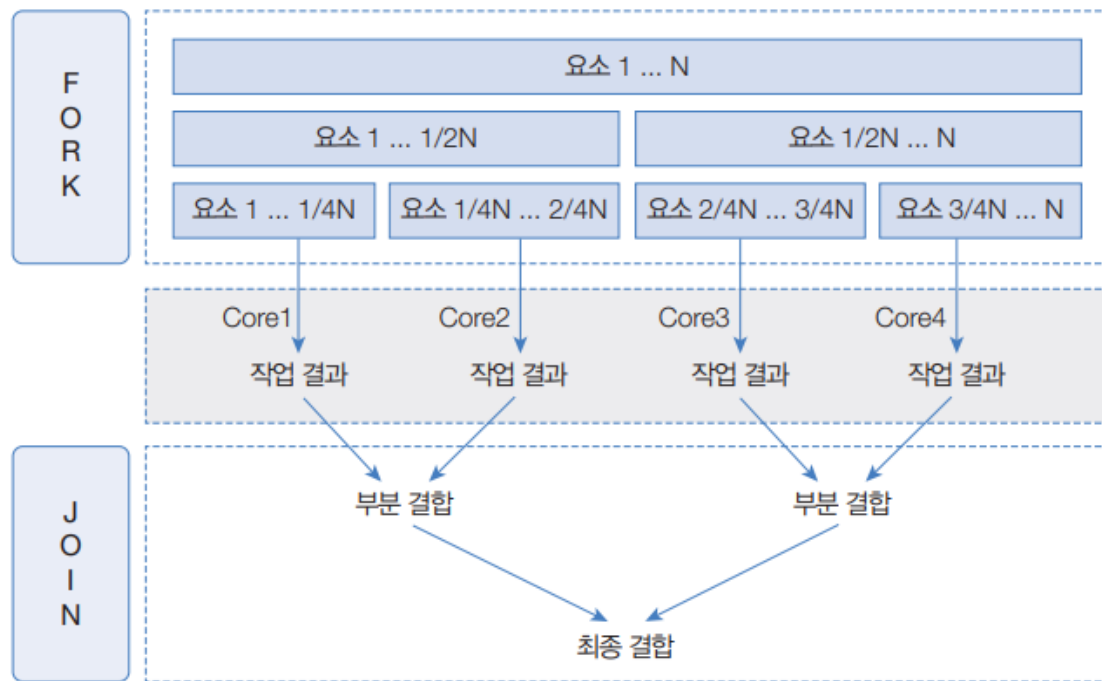
- 동시성: 멀티 작업을 위해 멀티 스레드가 하나의 코어에서 번갈아 가며 실행하는 것
- 병렬성: 멀티 작업을 위해 멀티 코어를 각각 이용해서 병렬로 실행하는 것



- 데이터 병렬성: 전체 데이터를 분할해서 서브 데이터셋으로 만들고 이 서브 데이터셋들을 병렬 처리해서 작업을 빨리 끝내는 것
- 작업 병렬성: 서로 다른 작업을 병렬 처리하는 것

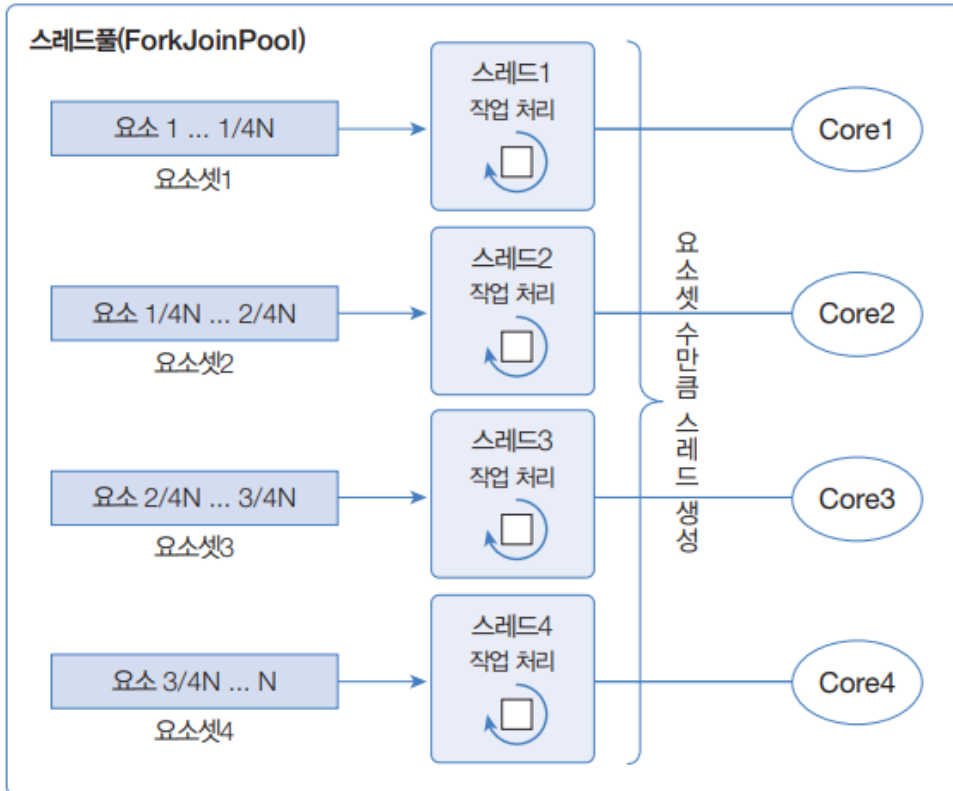
포크조인 프레임워크

- 포크 단계: 전체 요소들을 서브 요소셋으로 분할하고, 각각의 서브 요소셋을 멀티 코어에서 병렬로 처리
- 조인 단계: 서브 결과를 결합해서 최종 결과를 만들어냄



17.13 요소 병렬 처리

- 포크조인 프레임워크는 ExecutorService의 구현 객체인 ForkJoinPool을 사용해서 작업 스레드를 관리



병렬 스트림 사용

- 자바 병렬 스트림은 백그라운드에서 포크조인 프레임워크가 사용하므로 병렬 처리 용이
- `parallelStream()` 메소드는 컬렉션(List, Set)으로부터 병렬 스트림을 바로 리턴
- `parallel()` 메소드는 기존 스트림을 병렬 처리 스트림으로 변환

리턴 타입	메소드	제공 컬렉션 또는 스트림
Stream	<code>parallelStream()</code>	List 또는 Set 컬렉션
Stream	<code>parallel()</code>	<code>java.util.Stream</code>
IntStream		<code>java.util.IntStream</code>
LongStream		<code>java.util.LongStream</code>
DoubleStream		<code>java.util.DoubleStream</code>

병렬 처리 성능에 영향을 미치는 요인

- 요소의 수와 요소당 처리 시간
- 스트림 소스의 종류
- 코어(Core)의 수

Thank you!