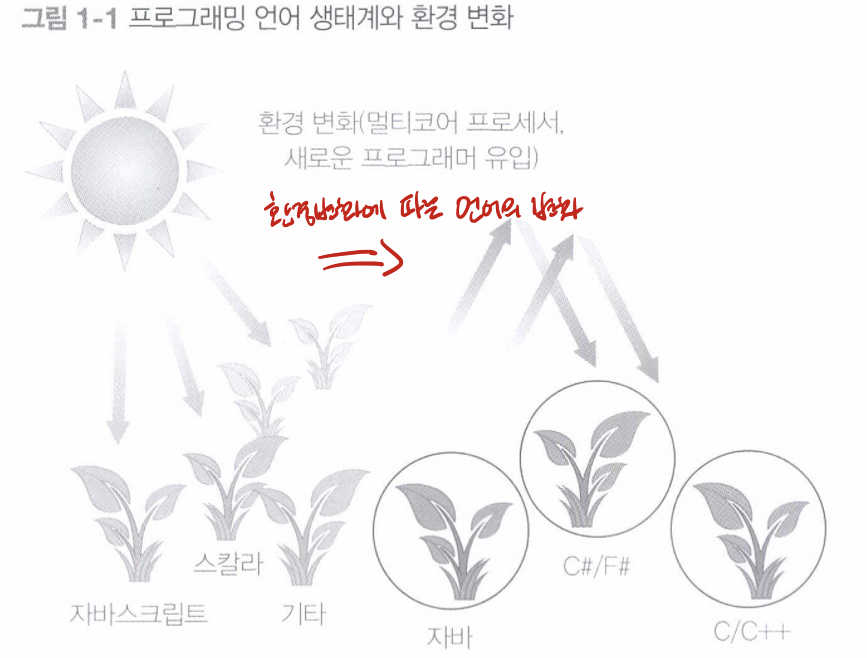
**[제 1 장]**

* 자바 8을 구성하는 핵심사항
  + 스트림 API
  + 메서드에 코드를 전달하는 기법
    - 함수형 프로그래밍에 최적화
  + 인터페이스의 디폴트 메서드

**1.1 왜 아직도 자바는 변화하는가?**

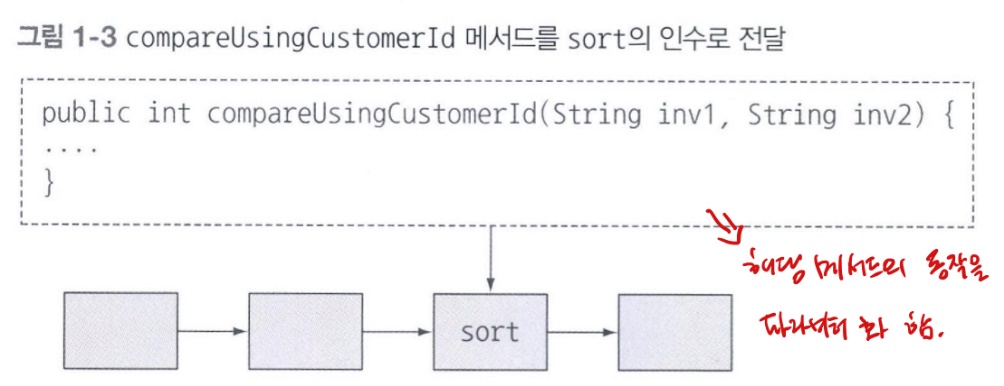
1.1.1 프로그래밍 언어 생태계에서 자바의 위치

* 자바의 위치
  + 스레드와 락을 지원
  + 다양한 임베디드 컴퓨팅 분야를 장악
* 자바와 빅데이터
  + 현 자바는 병렬 프로세싱(멀티코어 컴퓨터, 컴퓨팅 클러스터를 이용한 빅데이터 처리)을 활용하기엔 역부족
  + 이런 이슈로 인해 언어의 변화가 필요
* 프로그래밍 언어 생태계와 환경 변화
  + 

1.1.2 스트림 처리(자바 8 프로그램 개념1)

* 스트림이란
  + 한 번에 한 개씩 만들어지는 연속적인 데이터 항목들의 모임
* 스트림 API의 핵심
  + 한 번에 한 항목만 처리했던 스트림 방식을 자바8에서는 진행하려는 작업들을 추상화하여 일련의 스트림으로 처리 가능
  + 결론적으로, 스레드라는 복잡한 작업 대신 공짜로 병렬성을 얻을 수 있음

1.1.3 동작 파라미터화로 메서드에 코드 전달하기(자바 8 프로그램 개념2)

* 코드 일부를 API로 전달
  + 메서드를 다른 메서드의 인수로 넘기는 것이 가능
* 동적 파라미터화
  + 

1.1.4 병렬성과 공유 가변 데이터(자바 8 프로그램 개념3)

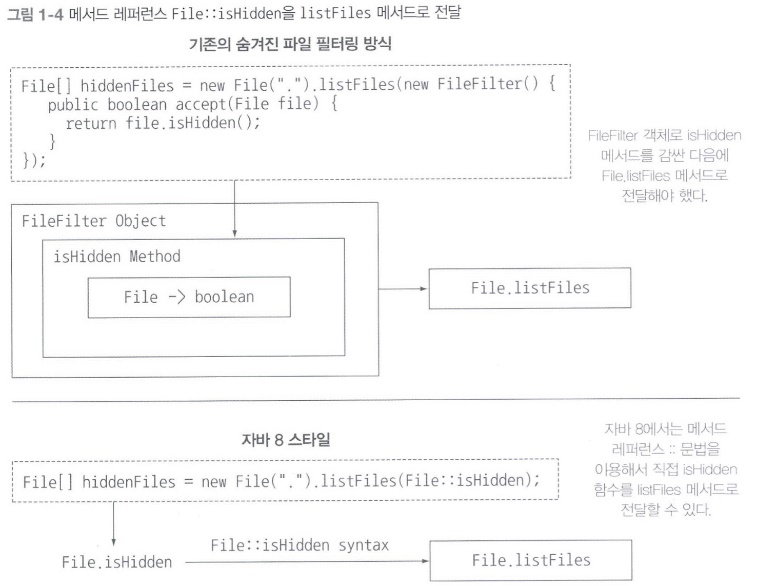
* 공짜로 얻게된 병렬성
  + 단, 스트림 메서드로 전달하는 코드는 다른 코드와 동시 실행 시 안전하게 실행되어야 함 -> 다른 코드가 공유된 가변 데이터에 접근을 막아야 함.
  + 기존에는 synchronized를 이용하여 공유된 가변 데이터의 보호를 구성함 -> 그러나 synchronized는 시스템에 악영향을 미침.
  + 자바 8 스트림을 이용할 시 기존의 자바 스레드 API보다 훨씬 쉽게 병렬성을 활용할 수 있음.

1.1.5 자바가 진화해야 하는 이유

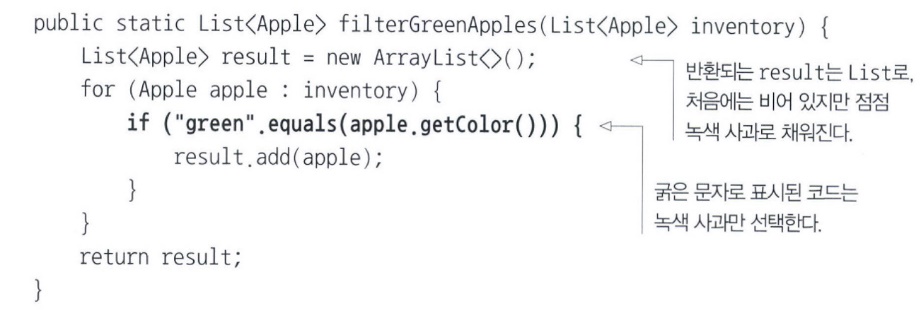
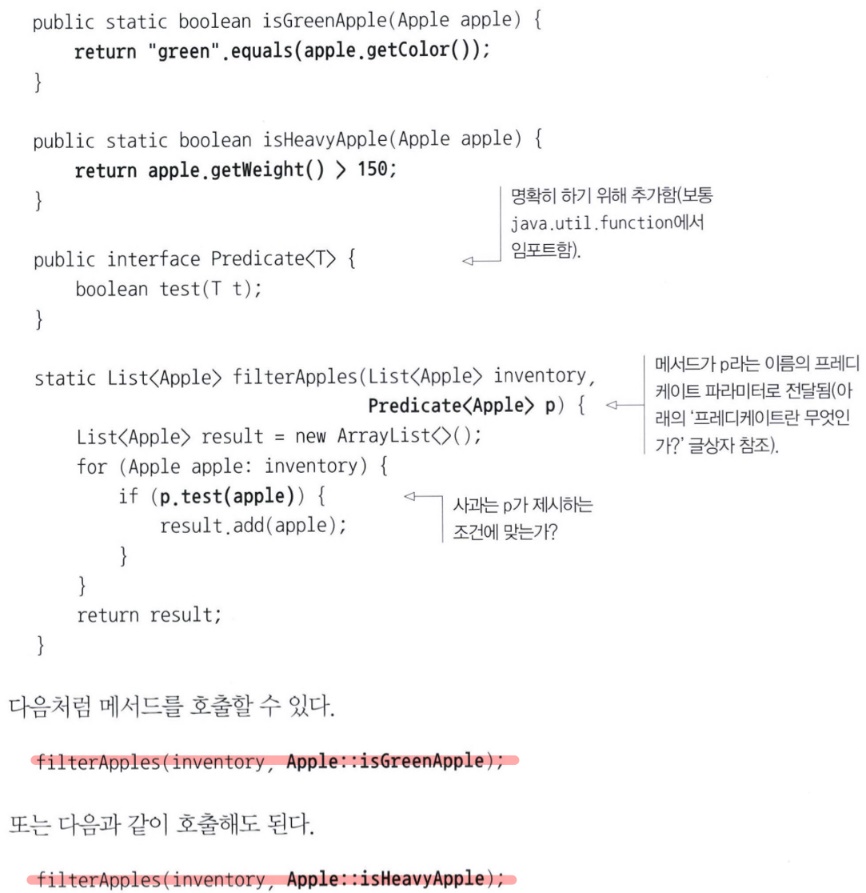
* 자바8의 가장 큰 변화
  + 고전적인 객체지향에서 벗어나 **함수형 프로그래밍**으로 다가서다

**1.2 자바 함수**

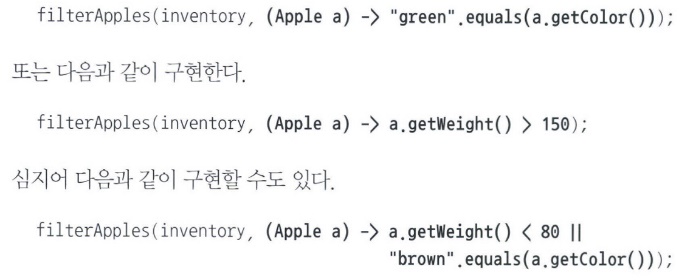
1.2.1 메서드와 람다를 일급 시민으로

* 메서드 레퍼런스
  + 
  + IsHidden이라는 함수는 File객체에 존재하므로 메서드 레퍼런스를 활용하여 listFiles에 직접적으로 전달 가능
  + 즉, 함수를 메서드의 파라미터로 전달 가능 -> 메서드를 일급값으로 취급
* 람다: 익명 함수
  + 람다를 포함한 함수도 값으로 취급 가능(일급)
  + (Int x) -> x+1 : x 인수 호출 시 x+1반환하는 동작으로 값으로 취급 가능

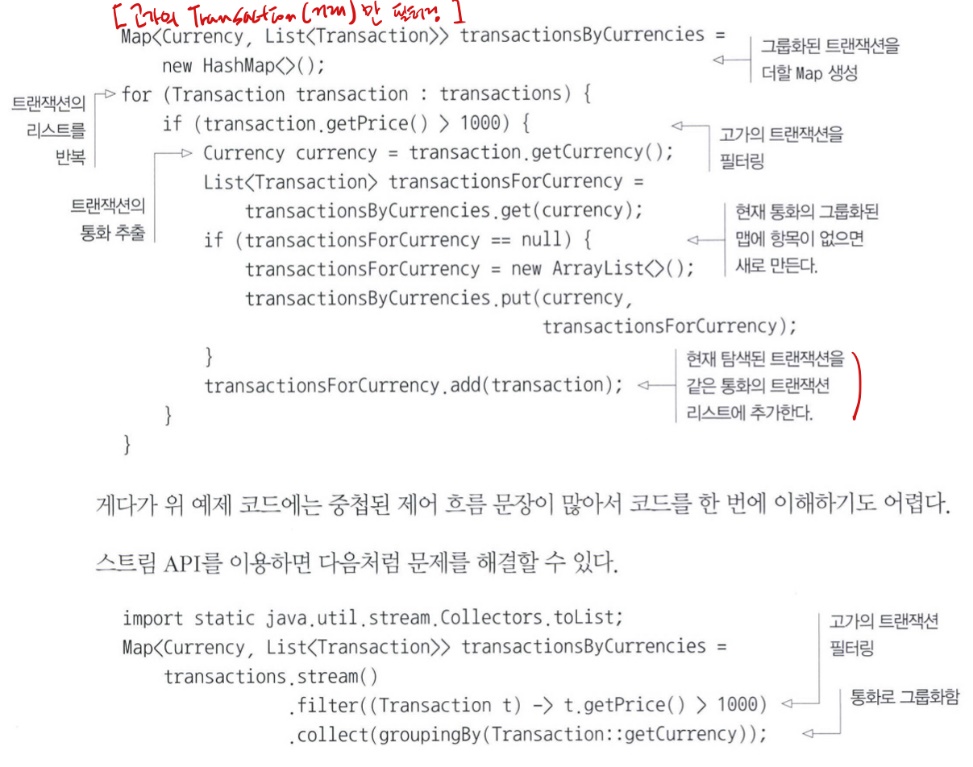
1.2.2 코드 넘겨주기: 예제

* Ex 1) 여러 사과가 인벤토리에서 녹색 사과만 꺼내겠다.(자바8 이전)
  + 
* Ex 2) 사과 무게 150그램 이상만 필터링
  + 
* Ex 1과 Ex 2는 굵게 표시한 문장만 다르고 모두 같음 -> 비효율적 || 코드에 버그 존재 시 고쳐야할 코드가 많아짐
* Ex 1, Ex 2) 자바 8 버전
  + 
  + 핵심은 메서드를 파라미터로 전달!
* 프레디케이트
  + 인수로 값을 받아 true나 false를 반환하는 함수
  + Function<Apple, Boolean> => Predicate<Apple> (더 표준적인 방식)

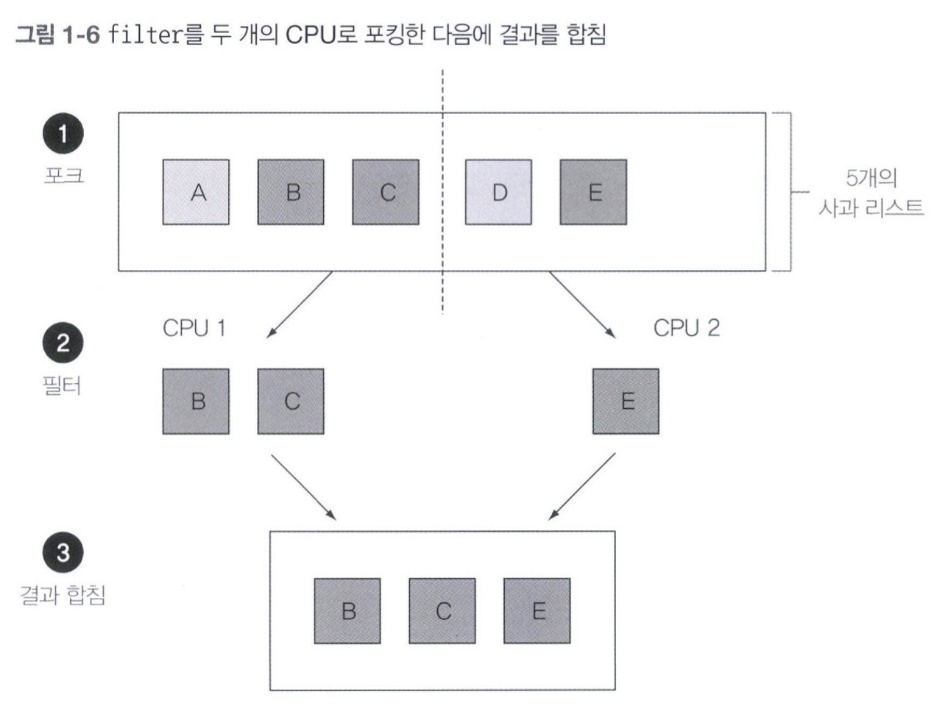
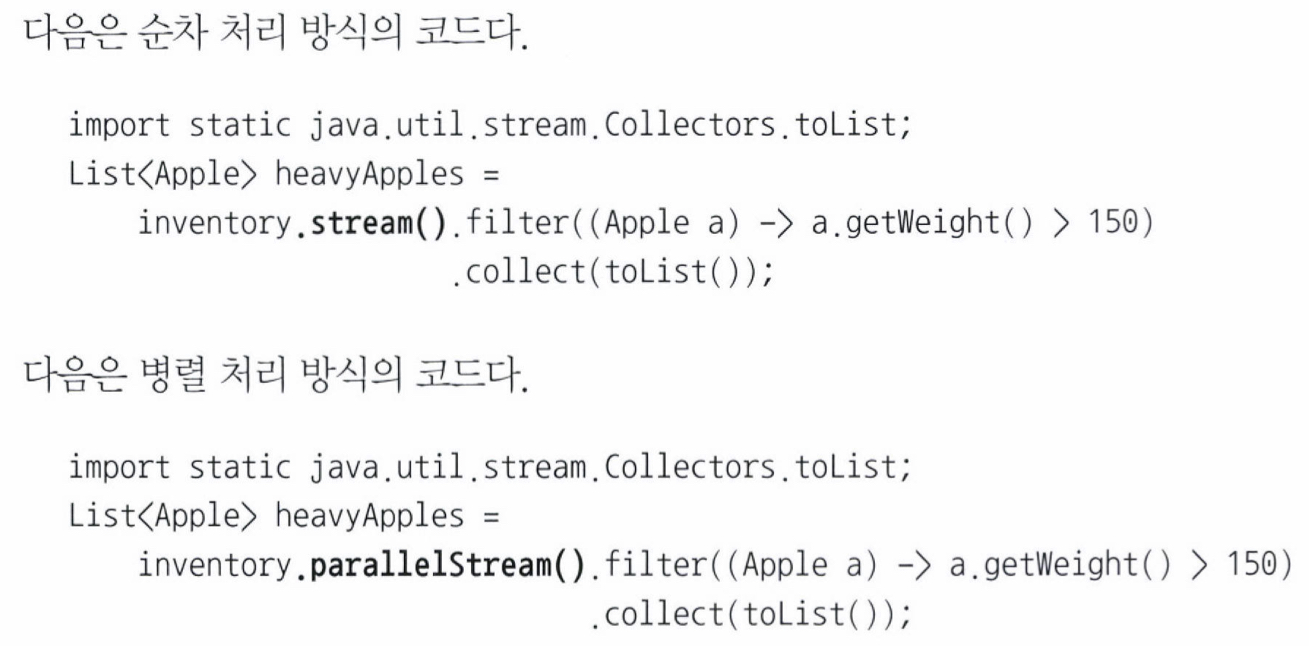
1.2.3 메서드 전달에서 람다로

* 소수로 사용하는 메서드는 람다로 대체
  + 

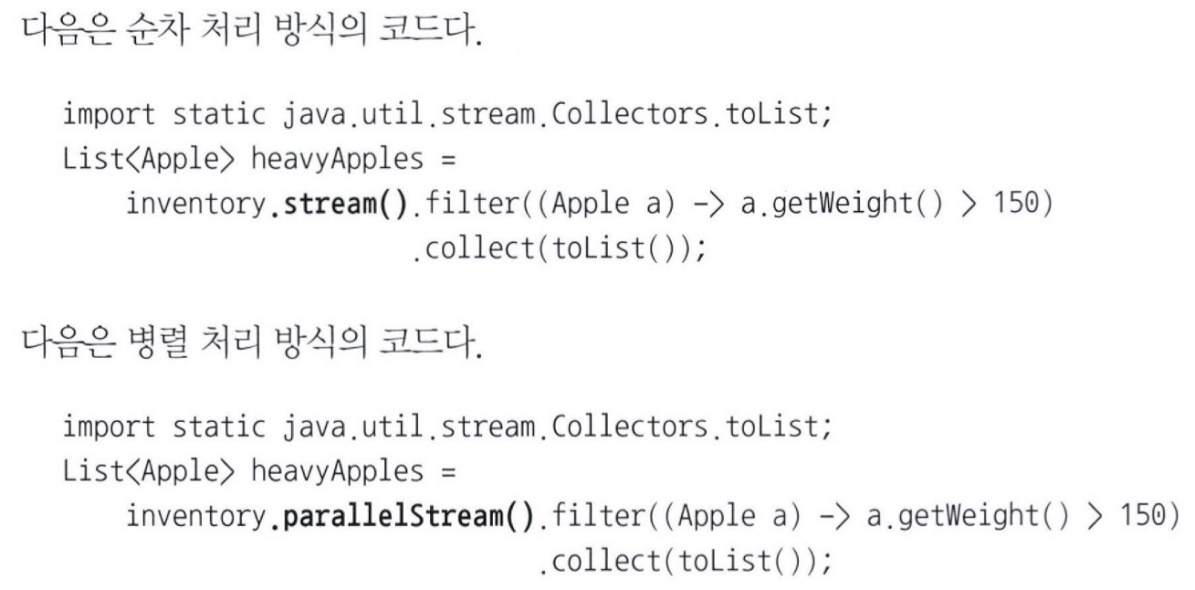
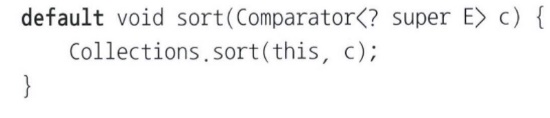
**1.3 스트림**

* 스트림 API 맛보기
  + 
  + 고가의 거래만 필터링하되, 해당 거래가 그룹화된 현재 통화에 없으면 항목을 추가까지 진행
* 스트림 API 라이브러리
  + 해당 라이브러리로 인해 외부 반복(for -each) 대신 내부 반복으로 데이터 처리 가능 -> 즉, 반복문을 명시할 필요가 없음.
* 멀티코어(여러개의 CPU)
  + 전통적인 자바 프로그램은 많은 CPU 중 단 하나만 사용
  + 자바 8에서는 멀티코어를 잘 활용할 수 있는 프로그래밍 스타일을 제공

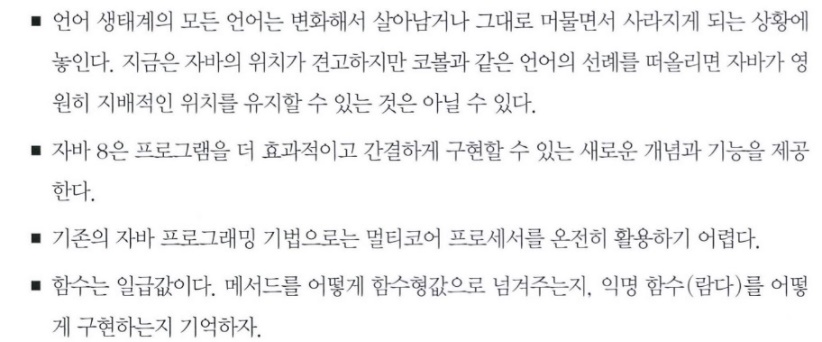
1.3.1 멀티스레딩은 어렵다

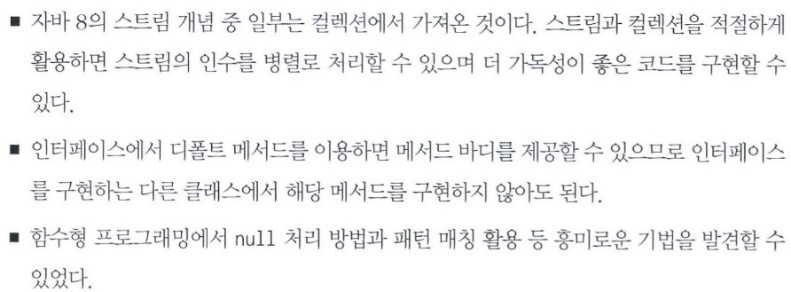
* 이전 자바 버전의 스레드
  + 멀티스레딩 환경에서 각각의 스레드는 동시에 공유된 데이터에 접근이 가능하여, 예상치 못한 결과값을 받을 수 있음.
* 스트림 API로 멀티 스레딩 문제 해결
  + 반복되는 패턴으로 주어진 조건에 따라 데이터를 **필터링, 추출, 그룹화**하는 기능을 지원
  + Ex) 리스트 필터링
    - 포킹 단계, 리스트 처리, 각 결과 머지
    - 
* 컬렉션과 스트림의 차이
  + 컬렉션은 어떻게 데이터를 저장하고 접근할 지에 중점을 둠
  + 스트림은 데이터에 어떤 계산을 할 것인지 묘사하는 것에 중점을 둠
  + But 베스트는, 컬렉션을 스트림으로 바꾸고, 병렬로 처리한 다음, 리스트로 다시 복원하는 것
* 병렬성을 공짜로 얻자(다시 언급)
  + 스트림과 람다식을 활용
  + 
* 자바 8의 요술방망이
  + 라이브러리에서 큰 스트림을 병렬로 처리할 수 있도록 작은 스트림으로 분할
  + Filter 같은 라이브러리 메서드로 전달된 메서드가 상호작용을 하지 않을 시 가변 공유 객체를 통해 공짜로 병렬성을 제공

**1.4 디폴트 메서드**

* 더 쉽게 변화할 수 있는 인터페이스를 만들기 위함
* 
  + 자바 8 이전에서는 해당 코드는 컴파일 불가 -> 해결을 위해서는 인터페이스를 생성하여 컬렉션 인터페이스에 stream 메서드를 추가하고 ArrayList에서 메서드를 구현해야 함.
  + 이 방법을 사용할 시 해당 인터페이스를 사용하는 모든 클래스는 새로 추가된 메서드를 구현해야 함.
* 구현 클래스에 구현하지 않아도 되는 인터페이스를 제공
  + 즉, 메서드 바디는 외부 클래스에 구현이 아니라 인터페이스의 일부로 포함됨 -> 디폴트 메서드
* 원래의 인터페이스 설계를 자유롭게 확장
  + 
  + List에 직접 sort 메서드를 호출 할 수 있음.
  + List를 구현하는 모든 클랫느는 sort를 구현하지 않아도 됨.

**1.6 요약**

* ****



질문) 함수형 프로그래밍에 자세한 설명 부탁드려욥