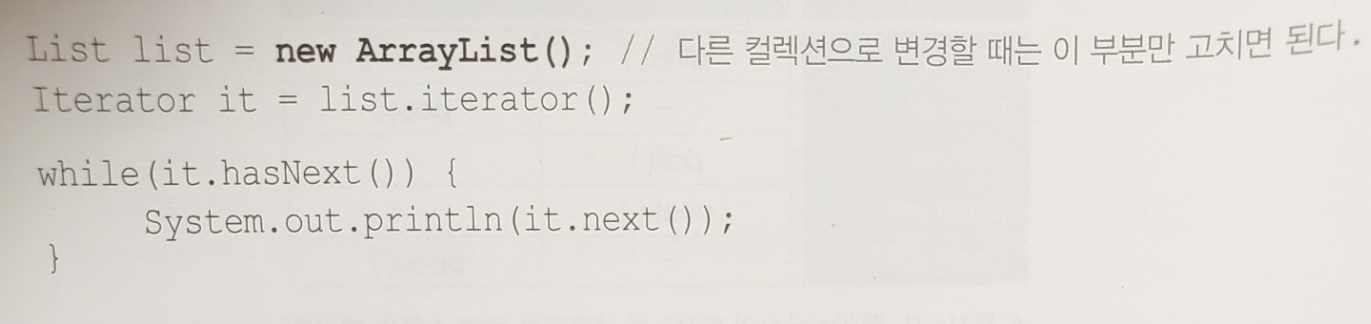
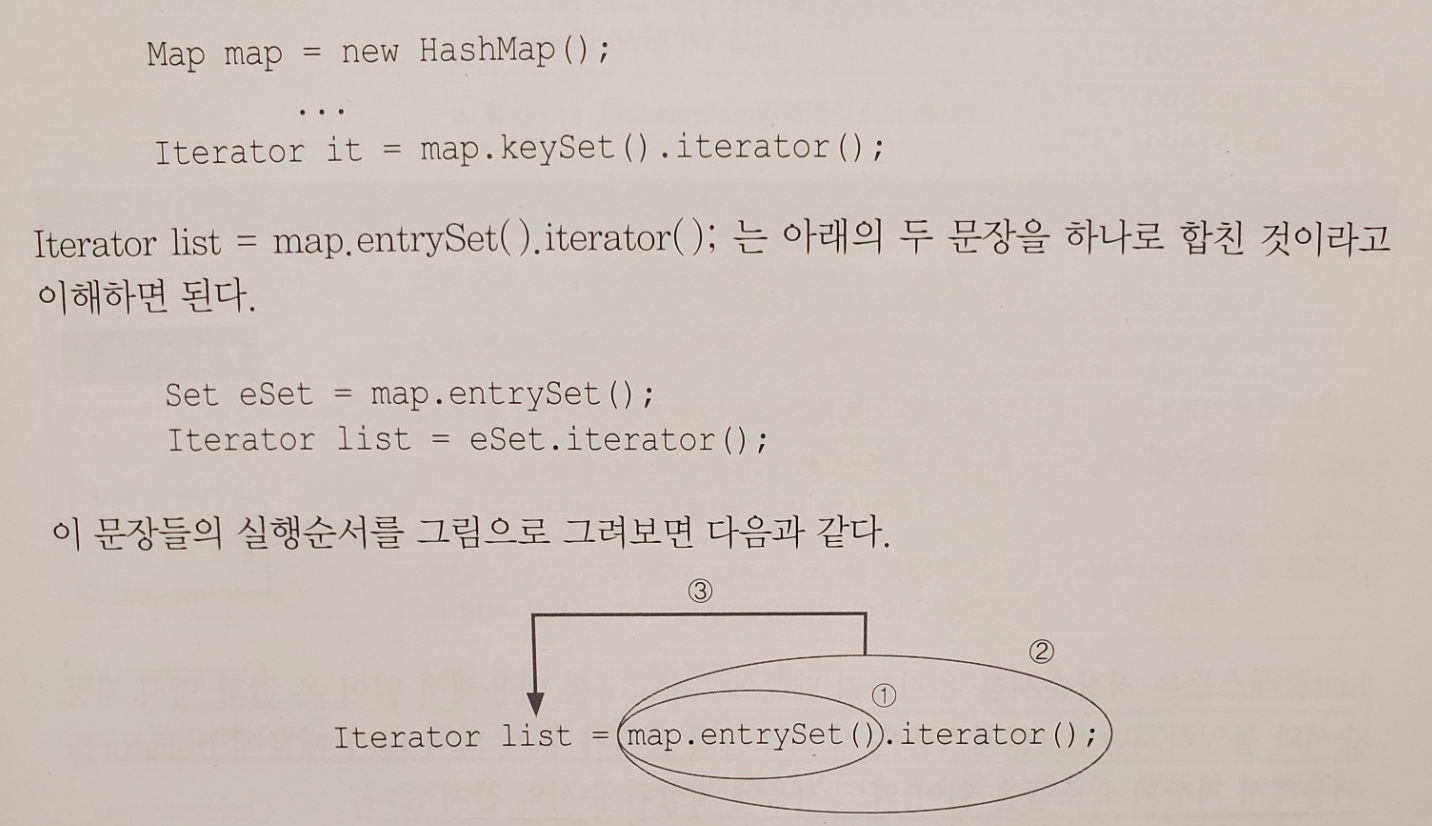
614p.

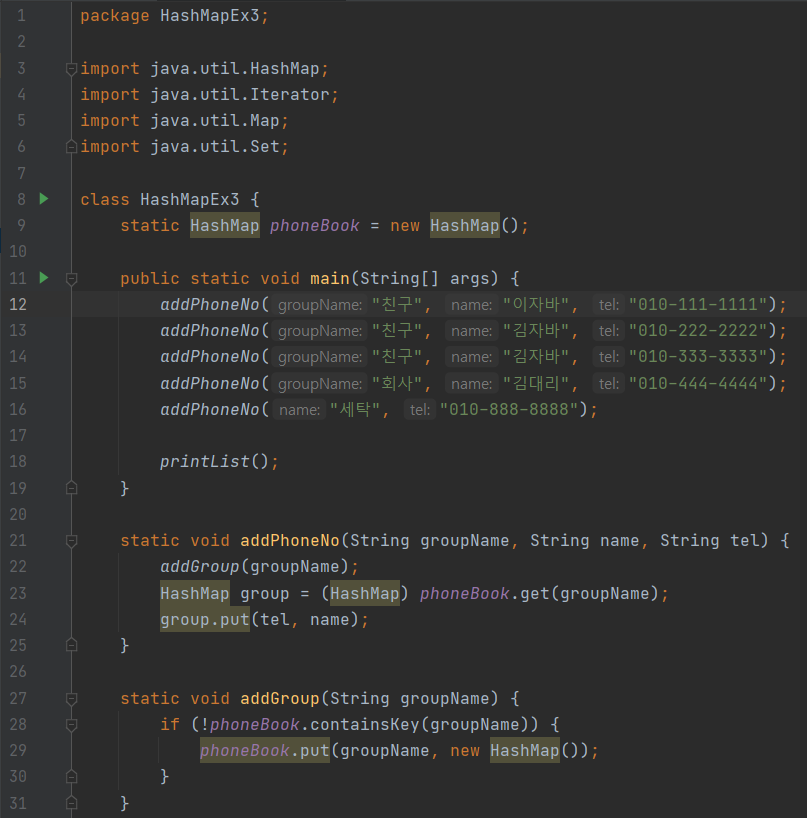
Iterator, ListIterator, Enumeration은 모두 컬렉션에 저장된 요소를 접근하는데 사용되는 인터페이스이다. Enumeration은 Iterator의 구버젼이며, ListIterator는 Iterator의 기능을 향상 시킨 것이다.

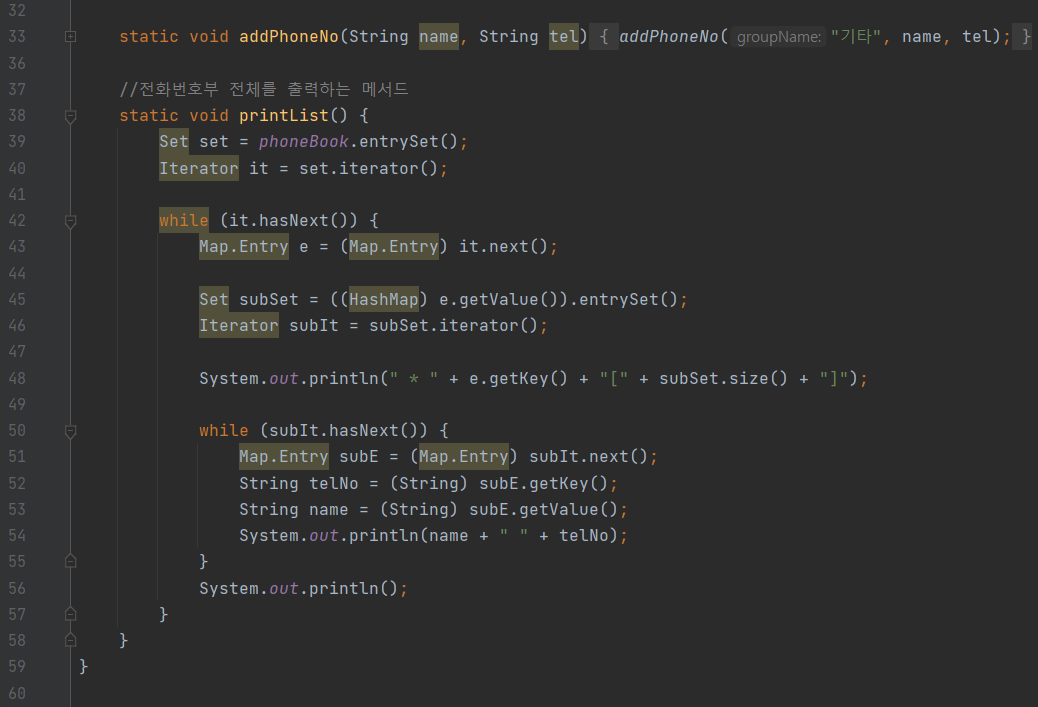
iterator()는 Collection 인터페이스에 정의된 메서드이므로 Collection 인터페이스의 자손인 List와 Set에도 포함되어 있다.



Map 인터페이스를 구현한 컬렉션 클래스는 키(key)와 값(value)을 쌍(pair)으로 저장하고 있기 때문에 iterator()를 직접 호출할 수 없고, 그 대신 keyset()이나 entrySet()과 같은 메서드를 통해서 키와 값을 각각 따로 Set의 형태로 얻어 온 후에 다시 iterator()를 호출해야 Iterator를 얻을 수 있다.







HashMap은 데이터와 키를 모두 Object 타입으로 저장하기 때문에 HashMap의 값(value)으로 HashMap을 다시 저장할 수 있다. 이렇게 함으로써 하나의 키에 다시 복수의 데이터를 저장할 수 있다. 먼저 전화번호를 저장할 그룹을 만들고 그룹 안에 다시 이름과 전화번호를 저장하도록 했다. 이때 이름대신 전화번호를 키로 사용했다는 것을 확인하자. 이름은 동명이인이 있을 수 있지만 전화번호는 유일하기 때문이다.

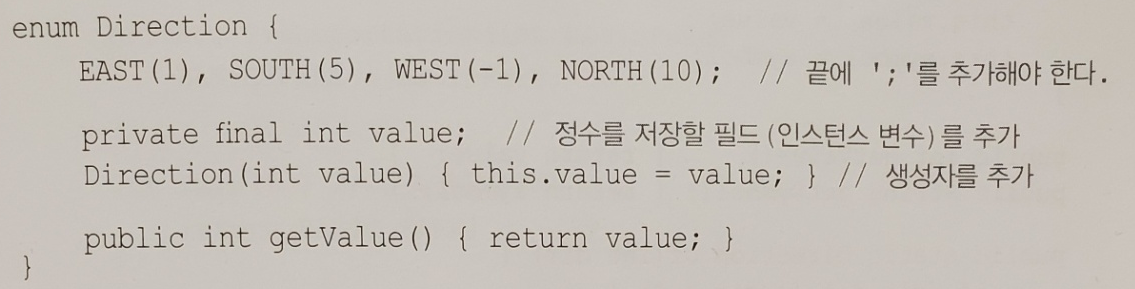
691p. 열거형 (enums)

자바의 ‘타입에 안전한 열거형(typesafe enum)’에서는 실제 값이 같아도 타입이 다르면 조건식의 결과가 false가 된다. 이처럼 값뿐만 아니라 타입까지 체크하기 때문에 타입에 안전하다고 하는 것이다.

Enum 클래스에 정의된 ordinal()이 열거형 상수가 정의된 상수를 반환하지만, 이 값을 열거형 상수의 값으로 사용하지 않는 것이 좋다. 이 값은 내부적인 용도로만 사용되기 위한 것이기 때문이다. 열거형 상수의 값이 불연속적인 경우에는 이때는 다음과 같이 열거형 상수의 이름 옆에 원하는 값을 괄호()와 함께 적어주면 된다.

Enum Direction { EAST(1), SOUTH(5), WEST(-1), NORTH(10) }

그리고 지정된 값을 저장할 수 있는 인스턴스 변수와 생성자를 새로 추가해주어야 한다. 이 때 주의할 점은, 먼저 열거형 상수를 모두 정의한 다음에 다른 멤버들을 추가해야한다는 것이다. 그리고 열거형 상수의 마지막에 ‘;’도 잊지 말아야 한다.

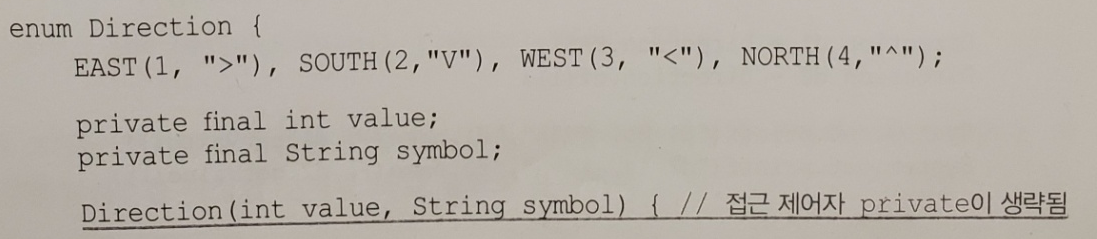


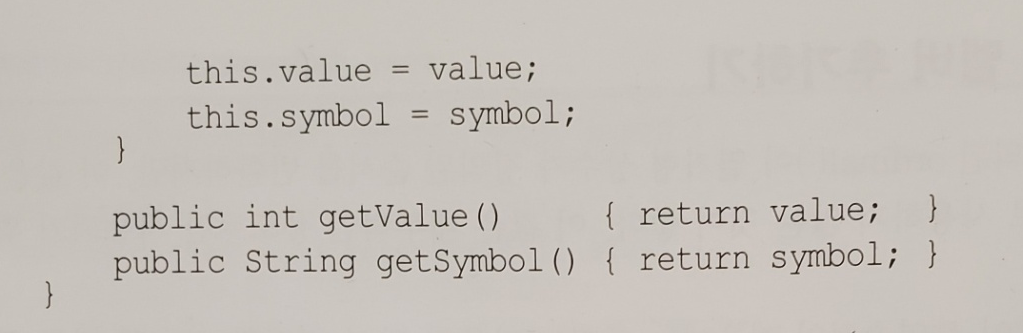
열거형의 인스턴스 변수는 반드시 final이어야 한다는 제약은 없지만, value는 열거형 상수의 값을 저장하기 위한 것이므로 final을 붙였다. 그리고 외부에서 이 값을 얻을 수 있게 getValue()도 추가하였다.



열거형 Direction에 새로운 생성자가 추가되었지만, 위와 같이 열거형의 객체를 생성할 수 없다. ,열거형의 생성자는 제어자가 묵시적으로 private 이기 때문이다.

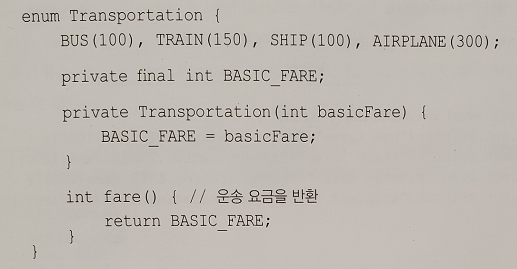
필요하다면, 다음과 같이 하나의 열거형 상수에 여러 값을 지정할 수도 있다. 다만 그에 맞게 인스턴스 변수와 생성자 등을 새로 추가해주어야 한다.



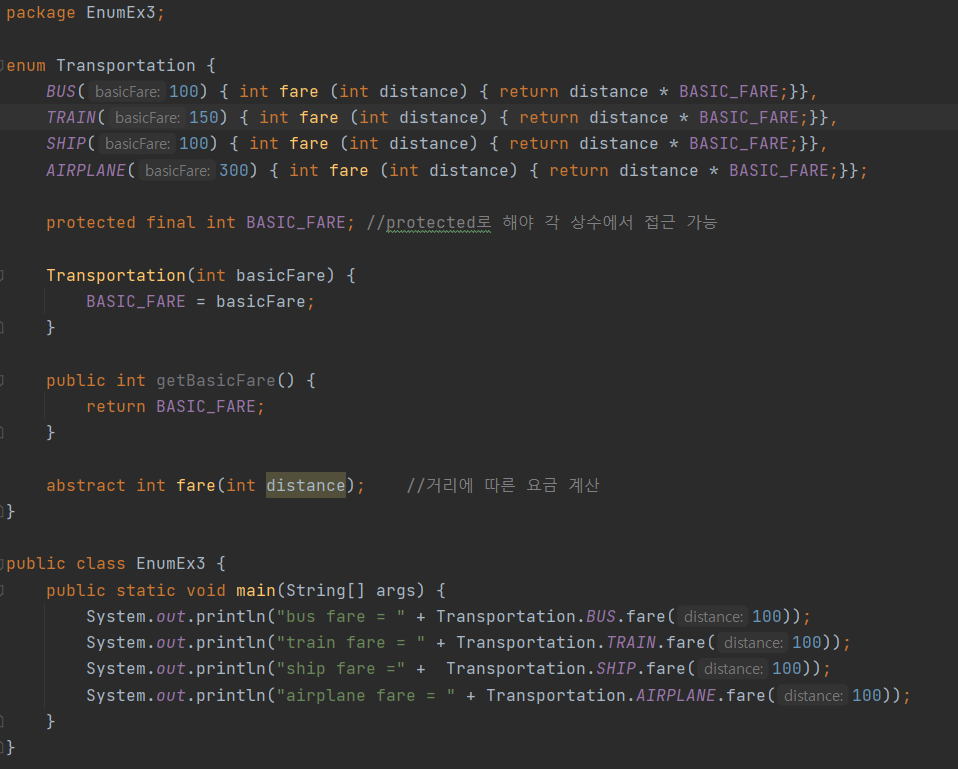


열거형에 추상 메서드 추가하기

열거형 Transportation은 운송 수단의 종류 별로 상수를 정의하고 있으며, 각 운송 수단에는 기본요금(BASIC\_FARE)이 책정되어 있다.



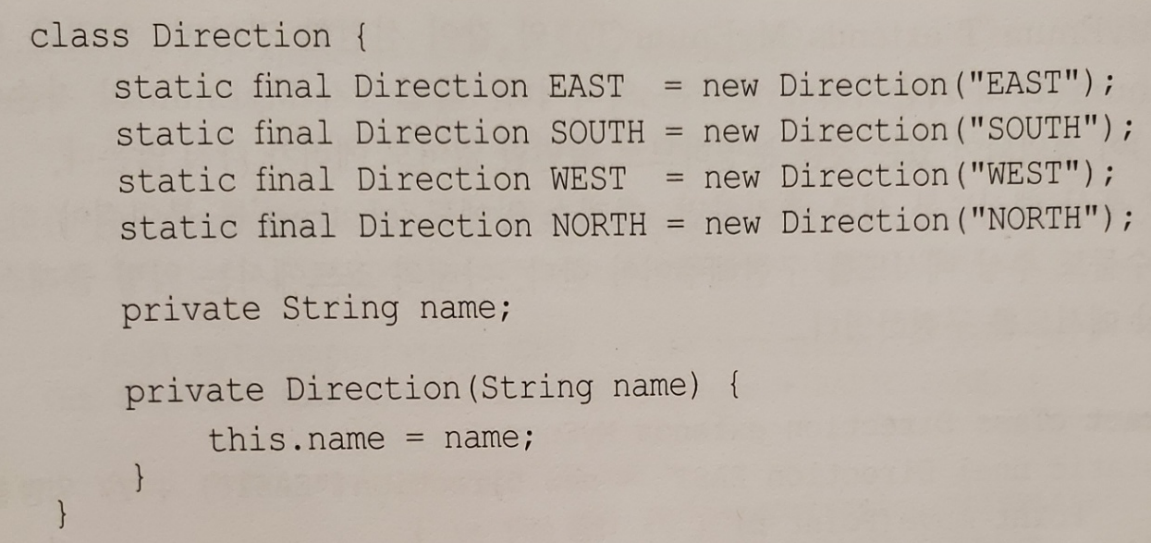
그러나 이것만으로는 부족하다. 거리에 따라 요금을 계산하는 방식이 각 운송 수단마다 다를 것이기 때문이다. 이럴 때, 열거형에 추상 메서드 ‘fare(int distance)’를 선언하면 각 열거형 상수가 이 추상 메서드를 반드시 구현해야 한다.



만일 열거형 Direction이 다음과 같이 정의되어 있을 때,

enum Direction {EAST, SOUTH, WEST, NORTH}

사실은 열거형 상수 하나하나가 Direction객체이다. 위의 문장을 클래스로 정의한다면 다음과 같을 것이다.



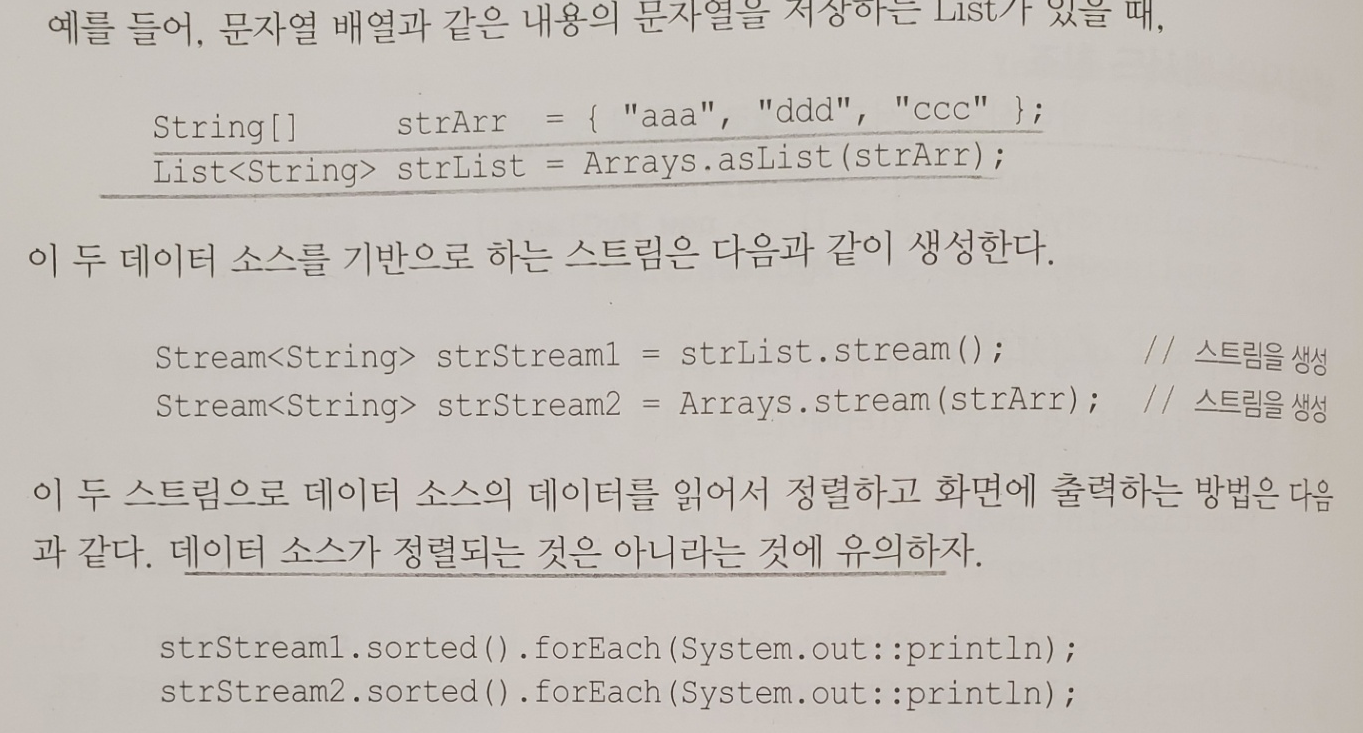
Direction 클래스의 static 상수 EAST, SOUTH, WEST, NORTH의 값은 객체의 주소이고, 이 값은 바뀌지 않는 값이므로 ‘==’로 비교가 가능한 것이다.

814p.

지금까지 우리는 많은 수의 데이터를 다룰 때, 컬렉션이나 배열에 데이터를 담고 원하는 결과를 얻기 위해 for문과 Iterator를 이용해서 코드를 작성해왔다. 그러나 이러한 방식으로 작성된 코드는 너무 길고 알아보기 어렵다. 그리고 재사용성도 떨어진다.

또 다른 문제는 데이터 소스마다 다른 방식으로 다뤄야한다는 것이다. Collection이나 Iterator와 같은 인터페이스를 이용해서 컬렉션을 다루는 방식을 표준화하기는 했지만, 각 컬렉션 클래스에는 같은 기능의 메서드들이 중복해서 정의되어 있다. 예를 들어 List를 정렬할 때는 Collections.sort()를 사용해야하고, 배열을 정렬할 때는 Arrays.sort()를 사용해야 한다.

이러한 문제점들을 해결하기 위해서 만든 것이 ‘스트림’이다. 스트림은 데이터 소스를 추상화하고, 데이터를 다루는데 자주 사용되는 메서드들을 정의해 놓았다. 데이터 소스를 추상화하였다는 것은, 데이터 소스가 무엇이던 간에 같은 방식으로 다룰 수 있게 되었다는 것과 코드의 재사용성이 높아진다는 것을 의미한다.



스트림은 데이터 소스를 변경하지 않는다

그리고 스트림은 데이터 소스로부터 데이터를 읽기만 할 뿐, 데이터 소스를 변경하지 않는다는 차이가 있다. 필요하다면, 정렬된 결과를 컬렉션이나 배열에 담아서 반환할 수도 있다.

