**자바프로그래밍2 실습과제**

**제5주 Collections**

이클립스에서 자신의 이름으로 프로젝트를 만드세요. 그 아래에 w5라는 이름의 패키지를 만들고 (5주차) 그 아래에 아래 문제의 클래스들을 작성하세요.

**<Collections에 속하는 여러 클래스 사용 예>**

아래는 Collections Framework에 속하는 여러 인터페이스와 클래스를 사용하는 예를 보여주는 프로그램입니다. 각 프로그램을 읽고 출력이 어떻게 될지 예측해 보세요. 객체들의 모양을 종이에 그림으로 그려 보면 결과를 예측하는 데 도움이 됩니다.

[RererencingObjectsInCollection2](src\w5.demo\RererencingObjectsInCollection2.java)

[RererencingObjectsInCollection3](src\w5.demo\RererencingObjectsInCollection3.java)

[RererencingObjectsInCollection4](src\w5.demo\RererencingObjectsInCollection4.java)

[RererencingObjectsInCollection5](src\w5.demo\RererencingObjectsInCollection5.java)

[SetDemo](src\w5.demo\SetDemo.java)

[SortedSetDemo](src\w5.demo\SortedSetDemo.java)

[MapDemo](src\w5.demo\MapDemo.java)

[SortedMapDemo](src\w5.demo\SortedMapDemo.java)

프로그램을 실행하여 결과를 확인하세요. 프로그램은 w5.demo 패키지에 넣어 실행하세요.

이 문제의 결과물은 제출하지 않습니다. 그렇지만 **이 문제를 잘 이해해야 아래 문제들을 잘 풀 수 있고 중간고사를 잘 볼 수 있습니다. 소스코드를 잘 읽어보고 실행결과를 예상한 후 프로그램을 실행시켜 확인하세요. 각 클래스와 인터페이스의 사용 방법도 기억해 두세요.**

<객체 그림 - Object diagram>

[RererencingObjectsInCollection2](image\RererencingObjectsInCollection2.jpg)

[RererencingObjectsInCollection3](image\RererencingObjectsInCollection3.jpg)

[RererencingObjectsInCollection4](image\RererencingObjectsInCollection4.jpg)

[RererencingObjectsInCollection5](image\RererencingObjectsInCollection5.jpg) --> [더 세밀 한 그림](image\RererencingObjectsInCollection5_detail.jpg)

1. Collections framework의 대표적인 인터페이스들(List, Set, Map)을 사용하는 연습입니다. 이 문제를 통해 generic interface와 generic class를 사용하는 연습도 함께 하게 됩니다.

w5 패키지에 CollectionsExercise 라는 이름으로 클래스를 만들고 그 아래에 main 메소드를 두어 아래 작업을 하도록 코드를 작성하세요. 강의자료를 잘 읽고 아래 문제를 푸세요.

(1)

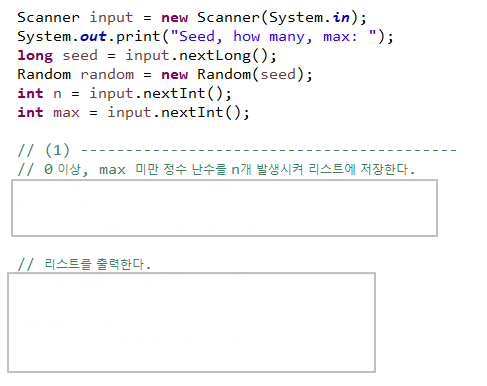
* 정수 난수를 발생시켜 list에 저장한다. (난수발생기의 seed, 난수 개수, 난수 최대 한도 등 세 가지를 사용자 입력으로 받습니다.)

List 중에서 ArrayList를 사용한다.

* 리스트를 출력한다.

출력할 때는 iterator를 사용한다.

코드 빼대 예를 들면 아래와 같습니다.



<힌트>

리스트를 생성할 때는 아래와 같이 합니다.

List<Integer> list = new ArrayList<>();

실행 예는 아래와 같습니다. 콘솔 화면의 일부만 갈무리한 것이어서 10 미만 정수 난수 100개 중 일부만 보입니다.



(2) 위 (1)에서 만든 main에 아래 코드를 추가하세요.

* 위 (1)에서 만든 list를 정렬한다.

list를 오름차순으로 정렬하려면 Collections 클래스의 static 메소드인 sort를 이용한다: Collections.sort(list). 그러면 list 내의 원소들이 오름차순으로 정렬된다.

표준 라이브러리에 Collections라는 이름의 클래스가 있다. 이 클래스는 Math 클래스처럼 많이 사용되는 기능들을 static 메소드 형태로 지원하는 유틸리티 클래스이다. java.util.Collections는 클래스이고 java.util.Collection은 인터페이스이다.

List에 정수를 저장하면 auto-boxing되어 Integer들이 저장된다. Integer은 Comparable(Comparable 인터페이스를 구현)로서 정수의 크기 순인 natural ordering이 정의되어 있기 때문에 정렬이 가능하다.

* 정렬된 리스트를 출력한다.

출력할 때는 iterator를 사용한다.

실행 예는 아래와 같습니다. 콘솔 화면의 일부만 갈무리한 것이어서 10 미만 정수 난수 100개 중 일부만 보입니다.



(3) 위 (2)에서 만든 main에 아래 코드를 추가하세요.

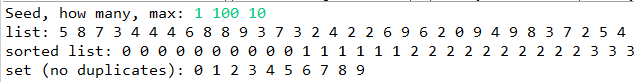
* (2)에서 만든 list에 들어 있는 원소들로 set을 생성한다. (중복된 원소 제거)

Set 중에서 HashSet을 이용한다. list에 들어 있는 원소들로 set을 생성하려면 아래와 같이 하면 된다.

Set<Integer> set = new HashSet<>(list)

* Set을 출력한다.

실행 예는 아래와 같습니다.



(4) 위 (3)에서 만든 main에 아래 코드를 추가하세요.

* list에 있는 각 원소가 얼마나 중복되었는지를 조사하여 map에 저장한다.

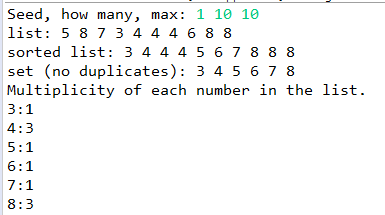
map에 <원소, 중복도>의 쌍으로 저장한다. 중복도는 영어로 multiplicity이다.

Map 중에서 HashMap을 사용한다.

* map의 내용을 출력한다.

map의 내용을 출력하는 방법은 강의자료에 설명되어 있다. (Map의 모든 원소를 읽는 법)

난수 개수가 10으로 주어진 경우의 실행 예는 아래와 같습니다.



위 그림에서 list의 각 숫자가 몇 번 중복되었는지를 확인할 수 있습니다. 0과 1, 2, 9는 리스트에 없으며, 3, 5, 6, 7은 중복되지 않았고, 4와 8은 각각 세 번씩 (중복도 = 3) 나타납니다.

<힌트>

리스트에 각 정수가 몇 개나 들어있는지를 맵에 저장하려면 아래와 같이 하면 됩니다.

A computer code with text

Description automatically generated with medium confidence

85번 줄에 있는 map.getOrDefault(i, 0)은

map에 키 i가 이미 있다면 그 키에 대응되는 value를, 그렇지 않다면 0을 반환합니다.

(5) 위 (4)에서 만든 main에 아래 코드를 추가하세요.

* 중복횟수의 도수분포(frequency distribution)를 조사하여 배열에 저장한다.

0번 방: 사용하지 않음

1번 방: 중복되지 않은 숫자가 몇 개인가?

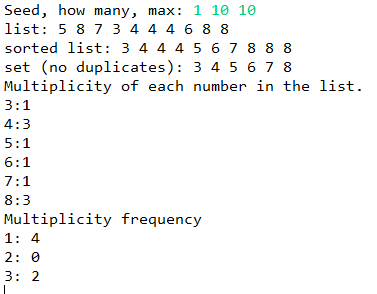
2번 방: 2중복된 숫자(2번 나타남)가 몇 개인가?

3번 방: 3중복된 숫자(3번 나타남)가 몇 개인가?

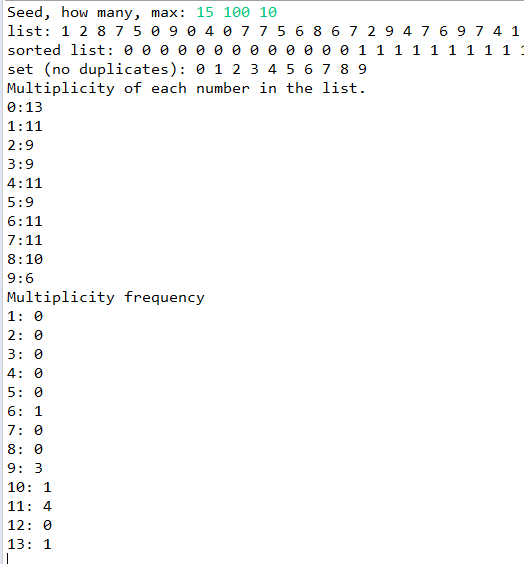
…

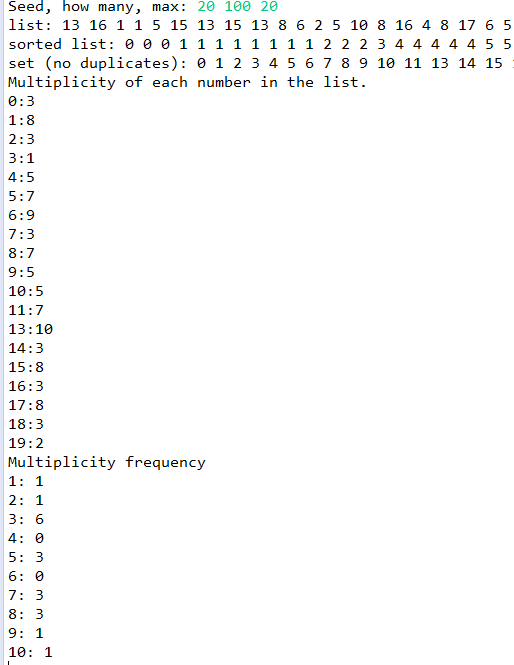
* 중복횟수의 도수분포를 출력한다.

실행 예는 아래와 같습니다. 중복이 되지 않고 단독으로만 존재하는 수자는 네 개(3, 5, 6, 7)이 있습니다. 2중복 수자는 없습니다. 3중복 수자는 두개(4, 8)이 있습니다. 4중복보다 중복도가 큰 수자는 없으므로 출력되지 않았습니다.



다른 예를 보이면 아래와 같습니다.





1. List와 ListIterator를 이용하는 방법을 연습하는 문제입니다. 이 문제를 통해 generic interface와 generic class를 사용하는 연습도 하게 됩니다.

우선 Iterator를 복습합니다. 모든 Collection에서 Iterator를 이용할 수 있습니다.

Iterator 인터페이스의 사용법은 아래 그림과 같습니다.



처음 세 개의 메소드에 주목하고 마지막 메소드는 다소 복잡하고 많이 쓰는 것이 아니므로 그냥 넘어갑니다.

아래에 사용법 예를 보입니다.

A screenshot of a computer program

Description automatically generated



리스트에서는 Iterator를 사용할 수 있을 뿐 아니라 ListIterator도 사용할 수 있습니다. 아래 예에서 it는 Iterator이고 lit는 ListIterator입니다.

List<String> list = new List<>();

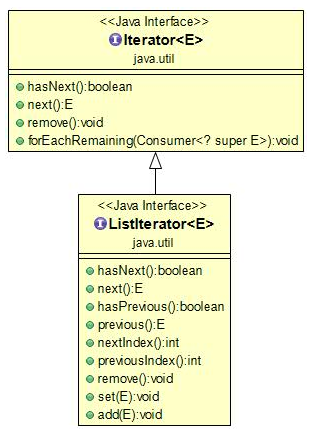
list.add(“일”);

list.add(“이”);

Iterator<String> it = list.iterator(); // Iterator를 얻는다.

ListIterator<String> lit = list.listIterator(); // ListIterator를 얻는다.

ListIterator는 Iterator의 서브인터페이스입니다. 따라서 Iterator의 모든 메소드를 갖고 몇 가지 자신의 고유한 메소드를 추가로 갖습니다.



아래는 ListIterator 사용 예입니다. ListIterator는 역방향으로도 이동하면서 list의 원소를 읽을 수도 있습니다.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

이상으로 Iterator와 ListIterator에 대한 복습을 마치고 문제를 풉니다.

이 문제에서는 주어진 개수의 **소수(素數, prime number)**를 구하는 프로그램을 작성합니다.

소수란?

1과 자기 자신 외의 약수를 갖지 않는 자연수. 1은 소수로 간주하지 않는다. 2는 소수이다.

n이 소수인지 판별하기 위해 2 이상, n 미만인 모든 정수로 n을 나누어 떨어지는지 살펴보는 방법은 지나치게 비효율적이다.

**<작은 소수부터 큰 소수 순으로 차례로 소수들을 찾아 내는 조금 더 효율적인 방법>**

2는 소수이다. list = [2]

3은 list에 있는 소수 2로 나누어 떨어지지 않으므로 소수이다. list = [2, 3]

4는 list에 있는 소수 2로 나누어 떨어지므로 소수가 아니다.

5는 list에 있는 소수 2로도 나누어 떨어지지 않고, 3으로도 나누어 떨어지지 않으므로 소수이다. list = [2, 3, 5]

6은 list에 있는 소수 2로 나누어 떨어지므로 소수가 아니다.

7은 list에 있는 소수 2로도, 3으로도, 5로도 나누어 떨어지지 않으므로 소수이다. list = [2, 3, 5, 7]

8은 list에 있는 소수 2로 나누어 떨어지므로 소수가 아니다. list = [2, 3, 5, 7]

9는 list에 있는 소수 3으로 나누어 떨어지므로 소수가 아니다. list = [2, 3, 5, 7]

...

즉, 어떤 수 n은 n보다 작은 소수들 중 하나로 나누어 떨어지면 소수가 아니다. 만약 n이 n보다 작은 소수들 어떤 것으로도 나누어 떨어지지 않으면 n은 소수이다.

주어진 개수의 소수(素數)를 구하는 프로그램을 아래 뼈대에 맞춰 작성하세요.

Graphical user interface, text, application

Description automatically generated

getPrimes 메소드를 구현할 때, 주어진 수 n개 만큼의 소수들을 찾기 위해서는, 2부터 시작하여 소수를 하나하나 차례로 찾아 나가야 합니다. 그래서 이미 찾은 소수들이 있을 때, 그 다음 소수를 찾는 getNextPrime 메소드를 작성하고, 이 메소드를 이용해 getPrimes 메소드를 구현합니다. getPrimes 메소드에서는 우선 리스트를 하나 (ArrayList이건 LinkedList이건 상관없습니다.) 생성하고 첫 소수인 2를 리스트에 넣습니다.

주의: List<Long> list = new ArrayList<>(); 문장 다음에 list.add(2)라고 적으면 컴파일에러입니다. 2는 int 타입 숫자이므로 Long 타입으로 자동포장(auto-boxing)되지 않습니다. lsit.add(2L)라고 적어야 long 타입 숫자 2L이 Long 타입 객체로 자동포장되어 list에 add됩니다.)

그리고는 getNextPrime 메소드를 호출하여 그 다음 소수를 구하고 그 소수를 리스트에 추가합니다. 그리고 다시 getNextPrime을 호출하여 다음 소수를 구하고 그 소수를 리스트에 넣는 작업을 반복합니다. 주어진 개수만큼 리스트가 찰 때까지 반복합니다. 주어진 개수의 소수들이 리스트에 차면 리스트를 반환합니다.

main에서는 SpecialNumberGenerator 인스턴스를 생성하고 이 객체에게 getPrimes 메소드를 호출하여 사용자가 지정하는 개수의 소수를 찾아냅니다. 그리고 찾아낸 소수들을 출력하고, 찾아낸 소수들 중 마지막 10개를 큰 수부터 작은 수 순으로 출력합니다. 만약 전체 소수가 10개 미만이라면 소수들을 모두 출력합니다.

리스트의 원소들을 차례로 출력하기 위해서는 ListIterator를 사용하세요. 아래는 List의 메소드인 listIterator 설명입니다. listIterator 메소드가 두개 있습니다.

Graphical user interface, text

Description automatically generated with medium confidence

두 번째 메소드는 주어진 index부터 차례로 방문하는 ListIterator를 반환합니다. 이 ListIterator에게 next를 호출하면 index번 방의 원소가, previous를 호출하면 index-1번 방의 원소가 반환됩니다. index는 0 이상, size() 이하이어야 합니다. list에 원소가 10개 있을 때, list.listIterator(10)을 호출하여 반환되는 ListIterator는 마지막 10번째 원소를 지나 리스트의 끝 부분을 가리키고 있습니다. 이 Listiterator에게 previous를 호출하면 마지막 원소인 9번 방의 원소가 반환됩니다.

출력 예는 아래와 같습니다.



1. 여러 가지 Collection을 이용하는 연습문제입니다.

주어진 개수의 못난이수(ugly numbers)를 구하는 프로그램을 작성합니다.

못난이수(ugly numbers)란?

2, 3, 5 외의 소수(素數) 약수를 갖지 않는 자연수.

(소인수가 2, 3, 5 뿐인 자연수)

(2를 0개 이상, 3을 0개 이상, 5를 0개 이상 서로 곱하여 얻을 수 있는 자연수)

1은 못난이수로 정한다.

아래 수는 못난이수이다.

1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, ...

7은 소수인 7을 약수로 가지므로 못난이수가 아니다.

11, 13도 소수인 11과 13을 각각 약수로 가지므로 못난이수가 아니다.

14도 소수인 7을 약수로 가지므로 못난이수가 아니다.

<작은 못난이수부터 큰 못난이수 순으로 못난이수들을 찾아 내는 방법>

1은 못난이수이다. 1을 못난이수 **집합**에 넣는다. {1}

1에 2, 3, 5를 곱한 수 2, 3, 5를 못난이수 집합에 넣는다. {1, **2, 3, 5**}

집합 중 1 다음에 있는 숫자가 2이다. 2에 2, 3, 5를 곱한 수 4, 6, 10을 집합에 추가한다. 새로 추가한 수를 집합의 끝에 추가하면 {1, 2, 3, 5, **4, 6, 10**}이 되어 오름차순으로 정렬되지 않으므로, 추가되는 수를 집합에 넣을 때는 오름차순으로 정렬이 되도록 신경 써서 올바른 위치에 넣는다. {1, 2, 3, **4**, 5, **6**, **10**}

집합 중 2 다음에 있는 숫자가 3이다. 3에 2, 3, 5를 곱한 수 6, 9, 15를 집합에 추가한다. 6은 이미 집합에 들어 있으므로 9와 15만 추가한다. {1, 2, 3, 4, 5, 6, **9**, 10, **15**}

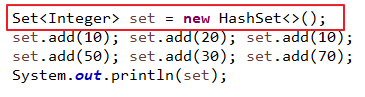
집합 중 3 다음에 있는 숫자가 4이다. 4에 2, 3, 5를 곱한 수 8, 12, 20을 집합에 추가한다. {1, 2, 3, 4, 5, 6, **8**, 9, 10, **12**, 15, **20**}

이런 과정을 반복한다.

위 방법에 따라 못난이수를 차례로 구할 때, 못난이수들을 저장할 Collection으로는 무엇이 적절할까요?

두 가지 성질이 필요합니다. 중복을 허용하지 않아야 하고 원소의 순서를 오름차순으로 관리해야 합니다.

중복을 허용하지 않는 Collection은 Set입니다. Set 인터페이스를 구현한 대표적인 클래스는 HashSet입니다. 아래 프로그램과 출력을 보세요.


10을 두 번 넣었지만 하나 밖에는 들어 있지 않습니다.

그런데 Set은 순서를 관리하지 않습니다. 위 예에서도 원소들의 순서가 제멋대로입니다.

Set중에도 순서를 관리하는 놈이 있습니다. SortedSet이 바로 그것입니다. SortedSet은 Set의 서브인터페이스로서, Set의 모든 성질을 물려받고, 순서를 관리하는 추가적인 성질을 갖습니다. SortedSet 인터페이스를 구현한 대표적인 클래스는 TreeSet입니다.

A screenshot of a computer code

Description automatically generated



원소의 중복을 허용하지 않으면서 natural ordering에 따라 원소의 순서를 관리합니다. 그러니까 우리는 위 못난이수를 찾아 저장하기 위해 SortedSet을 사용하면 될 것입니다.

위 p2 문제에서 작성한 SpecialNumberGenerator 클래스를 복사하여 SpecialNumberGenerator2 클래스를 만들고 SpecialNumberGenerator2에 못난이수를 찾아내는 아래 빨간네모 메소드를 추가하세요.

Graphical user interface, text, application

Description automatically generated

getUglies 메소드는 아래와 같이 구현할 수 있습니다.

SortedSet 타입 변수를 선언하고 SortedSet 클래스인 TreeSet<Long> 인스턴스를 생성한다.

API 문서에서 SortedSet 인터페이스에 어떤 메소드가 있는지 잘 읽어본다.

SortedSet 인터페이스의 메소드 중 적절한 것을 활용하여 getUglies 메소드를 완성한다.

[<작은 못난이수부터 큰 못난이수 순으로 차례로 못난이수들을 찾아 내는 방법>](#algorithm_uglies)은 "집합에 있는 **다음숫자**"에 2, 3, 5를 곱하여 집합에 추가합니다. 그런데 SortedSet은 List와 달리 k번 방의 원소를 읽는 get(k) 메소드를 지원하지 않습니다. SortedSet은 원소의 순서만 관리할 뿐 원소 번호까지 관리하지는 않습니다. 그래서 SortedSet을 사용할 때 "집합에 있는 **다음숫자**"를 얻기 위해서는 약간의 궁리가 필요합니다.

SortedSet 특유의 대표적인 메소드로는 first, headSet, last, tailSet, subSet 등이 있습니다.

만약 SortedSet x가 {1, 3, 6, 7, 9}라면, 이 메소드들은 아래와 같이 작동합니다.

x.first() --> 1

x.last() --> 9

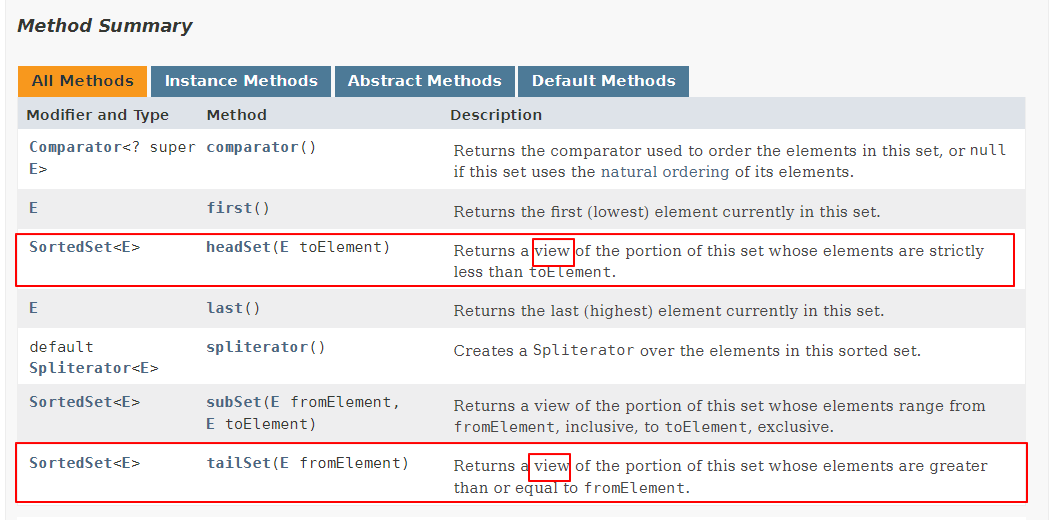
x.headSet(3) --> {1} // 3보다 작은 원소들의 부분집합

x.headSet(4) --> {1, 3} // 4보다 작은 원소들의 부분집합

x.tailSet(6) --> {6, 7, 9} // 6과 같거나 큰 원소들의 부분집합

x.tailSet(5) --> (6, 7, 9) // 5와 같거나 큰 원소들의 부분집합

특히 아래 설명에 주의하세요. [api](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/SortedSet.html#method-summary)

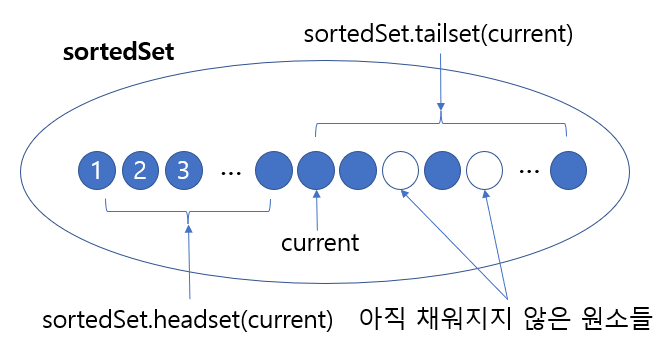


headSet과 tailSet은 SortedSet의 앞부분 원소들로 이루어지는 새 SortedSet을 만들어 반환하는 것이 아니고, 기존 SortedSet의 일부에 해당하는 **view**를 SortedSet으로 반환합니다. 그러니까 headSet이나 tailSet 호출에 의해 반환되는 SortedSet에 어떤 원소를 추가하거나 삭제하면 그 추가/삭제가 **원래의 SortedSet에도 반영**됩니다.

우리는 못난이수를 구하기 위해 "현재숫자"에 2, 3, 5를 곱합니다. 그리고는 "집합에 있는 **다음숫자**"를 찾아 그 수를 새로운 "**현재숫자**"로 삼고, 그 "**현재숫자**"에 2, 3, 5를 곱하여 못난이수를 구합니다. 집합 {1, 3, 6, 7, 9} 중 **현재숫자**가 3이라면 "집합에 있는 **다음숫자**"는 6입니다. **현재숫자**가 3인 상황에서 집합에 있는 **다음숫자**를 찾기 위해서는 위 메소드 중 어떤 메소드를 어떻게 이용하면 될지 생각해 보세요. (Set은 원소의 인덱스를 관리하지 않습니다.)

* SortedSet을 sset이라고 하고, “현재숫자”를 current라고 합시다. 처음에는 sset에 1L을 넣습니다. 그리고, current\*2L, current\*3L, current\*5L을 sset에 넣습니다. 이제 current를 sset 안의 다음 숫자로 업데이트해야 합니다, sset.tailSet(current+1L)을 호출하면 current+1L 이상인 원소들의 부분집합이 얻어집니다. 그 중에서 첫 번째 원소를 새로운 current로 삼으면 됩니다. 그리고, current\*2L, current\*3L, current\*5L을 sset에 넣고, 다시 current를 업데이트하는 작업을 반복합니다.

못난이수를 SortedSet에 추가해 나가면 아래 그림과 같은 자료구조가 만들어집니다.



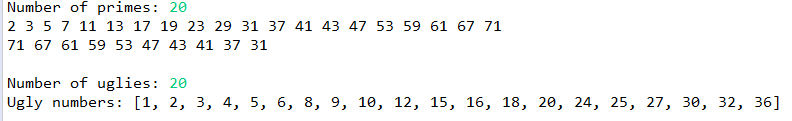
구하고자 하는 못난이수 개수가 n인 경우, sset.size() == n일 될 때 작업을 끝내면 안 됩니다. 위 그림에서 보듯, 아직 채워지지 않은 못난이수가 있을 수 있기 때문입니다. sortedSet.headSet(current).size() == n이 될 때까지 위 과정을 반복하고 sortedSet.headSet(current)를 반환해야 합니다.

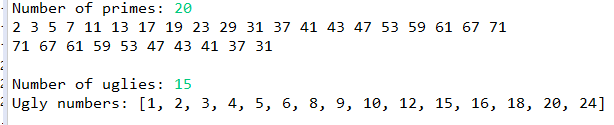
getUglies 메소드는 Collections<Long>을 반환해야 하는데 (문제에서 그렇게 요구했습니다.), 이렇게 만든 SortedSet<Long>도 Collection<Long>의 일종(서브타입)이므로 SortedSet<Long>을 반환하면 됩니다.

getUglies 메소드를 완성한 후 main에 getUglies 메소드를 이용하여 사용자가 지정하는 개수의 못난이수들을 구하여 출력하는 코드를 추가하세요. getUglies 메소드가 반환하는 SortedSet을 System.out.println을 이용하여 그냥 프린트하면 됩니다.

또, 소수들을 큰 수부터 작은 수 순으로 10개 출력하기 전에 소수들 전체를 오름차순으로 출력하게 하세요.

프로그램 실행 출력은 아래와 같습니다.





또, 아래 코드를 main에 추가하세요.

1. 이미 구한 소수들과 못난이수들의 교집합을 구하여 오름차순으로 화면에 출력한다.

p와 u가 각각 prime number와 ugly number를 갖고 있는 List와 Collection이라고 할 때, p와 u의 교집합은 어떻게 구할 수 있을까요? List의 메소드 중 어느 것을 사용하면 될지 궁리해 보세요. 물론 [API](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html) 문서를 잘 읽어 List에는 어떤 메소드가 있고 각 메소드가 무엇을 하는지 알아야 합니다. List 인터페이스에 intersection이라는 이름의 메소드는 없지만 그 기능을 하는 다른 이름의 메소드가 있습니다. 바로 retainAll입니다. p.retainAll(c)를 호출하면 p의 원소들 중 c에 있는 것들만 남고 나머지 원소들은 삭제됩니다.

1. 이미 구한 소수들과 못난이수들의 합집합을 구하여 합집합을 오름차순으로 화면에 출력한다.

p와 u가 각각 prime number와 ugly number를 갖고 있는 List와 Collection이라고 할 때, p와 u의 합집합은 어떻게 구할 수 있을까요? List 인터페이스에 union이라는 이름의 메소드는 없지만 그 기능을 하는 다른 이름의 메소드가 있습니다. 바로 addAll입니다. p.retainAll(c)를 호출하면 p의 끝에 c의 원소들이 추가됩니다. 여기에서 중복을 제거하고 오름차순으로 정렬하여 출력하면 됩니다.

1. 정렬된 합집합의 앞 10개를 차례로 화면에 출력한다. (합집합의 크기가 10 미만이면 전체를 출력합니다.)

합집합을 리스트로 만든 후 ListIterator를 이용하여 출력하세요.

1. 정렬된 합집합의 뒤 10개를 차례로 화면에 출력한다. (합집합의 크기가 10 미만이면 전체를 출력합니다.)

리스트의 끝 부분 원소 10개를 출력하기 위해서는 ListIterator를 사용합니다.

완성된 프로그램의 실행 예는 아래와 같습니다.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. 이 문제는 equals, hashcode, compareTo 메소드를 재정의하는 이유와 방법에 대한 이해를 돕기 위한 문제입니다. 우선 아래 설명을 잘 읽어 보세요.

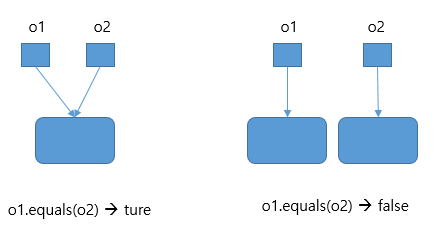
java.lang.Object 클래스는 인스턴스 메소드인 equals, hashcode 메소드를 가지고 있다.





**o1.equals(o2)** – o1이 o2와 같은 메모리에 있는지 검사하여 같은 메모리이면 true, 그렇지 않으면 false를 반환한다.

o1과 o2가 하나라면, o1과 o2는 같은 메모리 주소를 가지며, o1.equals(o2)는 true이다. 그림 1 (a). 그렇지 않으면 false이다. 그림 1 (b).



1. (b)

그림 1

**o.hashcode()** – 객체 o의 메모리 주소에 따라 정해지는 (메모리 주소를 일정 규칙에 따라 지지고 볶아 만들어지는) 정수를 반환한다. 그러니까 그림 1 (a)의 경우에는 o1과 o2의 주소가 같으므로, o1.hashcode() 반환값과 o2.hashcode() 반환값이 같다. 그림 1 (b)의 경우에는 o1.hashcode() 반환값과 o2.hashcode() 반환값이 대체로 다르다. (지지고 복아 정수를 만든다고 했는데, 그렇게 하는 작업을 hash라고 한다. 서로 다른 두 값을 각각 hash하면 대부분 서로 다른 값이 나오지만, 아주 낮은 확률로 같은 값이 나올 수도 있다.)

모든 클래스는 java.lang.Object의 서브클래스이므로 모든 클래스는 java.lang.Object로부터 equals, hashcode 메소드들을 상속한다. 상속한다는 것은 이 메소드를 슈퍼클래스로부터 물려받아 갖는다는 말이다. Cat 클래스도, Dog 클래스도 이 두 메소드를 갖는다. 예를 들면 아래와 같다. (Cat 클래스가 name과 age 등 두 개의 필드를 갖는다고 가정한다.)

Cat c1 = new Cat("Cookie", 2);

Cat c2 = new Cat(" Cookie", 2);

System.out.println(c1.equals(c2)); --> false // 위 그림 1의 (b)에 해당하므로....

그러나, 대개는 Object로부터 상속한 이 두 메소드를 재정의(override)해서 사용한다. 재정의란 상속한 메소드의 시그너처(메소드 이름과 파라미터)는 그대로 유지하되, 메소드 몸체를 다시 정의함으로써 메소드 동작을 바꿔준다는 말이다. 일반적으로 equals 메소드는 두 객체의 필드 값들이 서로 같을 때 true가 되도록 재정의한다. name과 age 등 두 개의 필드를 갖는 Cat 클래스의 equals 메소드를 재정의하는 코드 예를 보이면 아래와 같다.

A computer screen shot of a code

Description automatically generated

위와 같이 Cat 클래스에서 equals 메소드를 재정의하면 아래와 같이 된다.

Cat c1 = new Cat(" Cookie", 2);

Cat c2 = new Cat(" Cookie", 2);

System.out.println(c1.equals(c2)); --> true // equals 코드를 실행하는 c1과 파라미터로 주어진 c2의 name과 age가 서로 같으므로...

Cat 클래스의 equals 메소드 파라미터가 Cat 타입이 아니고 Object 타입인 이유는 이것이 수퍼클래스인 Object로부터 상속한 메소드이기 때문이다. (Object 클래스의 equals 메소드 파라미터가 Object 타입이다.) 그러나 실제로 프로그램을 실행할 때는 일반적으로 Cat 객체를 다른 Cat 객체와 비교할 것이므로, equals 메소드를 호출할 때 통상 Cat 인스턴스를 인자로 주게 된다. (Cat을 포함한 모든 객체는 Object의 서브클래스이므로 Cat도 Object이다. 따라서, Cat의 equals 메소드를 호출할 때, Object를 인자로 줄 수도 있고, Cat을 인자로 줄 수도 있다.)

**equals 메소드를 재정의할 때는 반드시 hashcode 메소드도 재정의해야 한다.**

o1과 o2가 어떤 한 클래스 객체일 때, o1.equals(o2)가 true이면, o1.hashcode()와 o2.hashcode()도 서로 같은 값을 갖도록, 그 클래스의 hashcode 메소드를 재정의해야 한다 (자바 언어 규정). 그렇게 하지 않으면 Collection의 일부 메소드를 실행할 때 에러가 날 수 있다.

Set에 새 객체를 add할 때 add 메소드는 set에 이미 들어 있는 각 객체가 새 객체와 equals한지 검사한다. 만약 equals한 게 있으면 새 객체를 add하지 않는다.

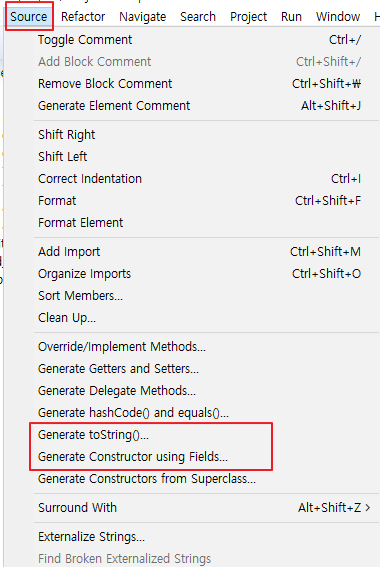
두 객체가 같은지 판별할 때, 많은 경우 곧바로 equals 메소드를 사용하지 않고 우선 두 객체의 hashcode를 계산하여 두 값이 같은지 먼저 살펴본다. hashcode가 같지 않으면 두 객체는 equals 하지 않은 것이다. 만약, 두 객체의 hashcode가 같으면, 추가적으로 두 객체가 같은지 equals 메소드를 통해 확인한다. 일반적으로 hashcode가 equals 보다 실행시간이 짧기 때문에 성능 향상을 위해 이런 2단계 계산을 한다.

아래 그림과 같은 Cat 클래스를 작성하세요.

A white background with black and red text

Description automatically generated

그리고 아래 그림의 빨간 네모 기능을 이용하여 생성자(Constructor)와 toString 메소드를 삽입하세요.



아래 그림과 같이 HashSet에 Cat 인스턴스들을 넣는 main 메소드를 실행하면 출력이 어떻게 될지 생각해 보세요. 이 Cat 클래스는 equals와 hashCode 메소드를 재정의하지 않았음에 유의하세요. HashSet 클래스는 Set 인터페이스를 구현한 대표적인 클래스입니다.

A computer code with text

Description automatically generated with medium confidence

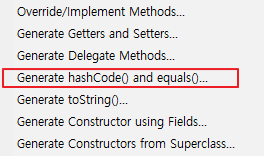
위 코드를 실행하여 예상이 맞는지 확인하세요.

다음으로 w5.overriding 패키지를 만드세요. 이클립스 메뉴, File-New-Package를 차례로 선택하면 나타나는 창에서 패키지 이름을 입력하고 “Create package-info.java” 앞의 네모는 체크 **해제**하세요.

w5 패키지와 w5.overriding 패키지는 서로 다른 패키지입니다. w5.overriding 패키지는 w5 패키지에 속하지 않습니다.

그리고, w5.Cat을 복사하여 새로 만든 w5.overriding 패키지에 붙여 넣고 아래 작업을 수행하세요.

Source 메뉴의 서브메뉴 중 아래 네모의 기능을 이용하여 equals, hashcode 메소드를 재정의하세요.



새로 만들어진 equals, hashcode 메소드를 읽어 보세요. 이 코드가 java.lang.Object로부터 상속한 equals, hashcode 메소드를 재정의(엎어쓰기)합니다. equals는 인자로 주어진 Object가 Cat이고, 그 Cat의 name과 age가 이 (equals 코드를 실행하는) Cat의 name과 age와 서로 같으면 true를 반환합니다. hashcode는 name과 age 값을 hash 함수로 처리하여 나온 결과를 int 값으로 반환합니다. (구체적인 계산식은 알 필요가 없습니다.)

main 메소드를 아래와 같이 수정하고 실행하면 결과가 어떻게 될지 예상해 보고 실제로 실행하여 결과를 확인하세요. 아래 코드의 Set s에는 몇 마리의 고양이들이 들어 있게 될까요?

A screenshot of a computer code

Description automatically generated

첫 번째 Set에는 고양이가 두 마리만 들어 있게 됩니다. 세 마리 중 두 마리가 서로 나이와 이름이 같기 때문입니다. 두 번째 Set에는 고양이가 다섯 마리 들어 있게 됩니다. 여섯 마리 중 두 마리가 서로 나이와 이름이 같기 때문입니다.

이번에는 고양이들을 SortedSet에 넣어 보겠습니다. w5.overriding 패키지를 복사하여 w5.overriding2라는 이름의 패키지로 붙여넣기 하세요. (w5.overriding 패키지를 오른쪽 버튼 클릭하여 Copy 한 후, 다시 w5.overriding 패키지를 오른쪽 버튼 클릭하여 Paste하여 나타나는 대화창에서 패키지 이름 편집). 그리고 새로 만들어진 w5.overriding2 패키지에서 작업을 진행하세요.

SortedSet은 원소의 natural ordering에 따라 원소들의 순서를 관리합니다. 그런데 우리의 Cat은 Comparable이 아니고 compareTo 메소드를 갖고 있지 않습니다. 따라서, natural ordering이 정의되어 있지 않습니다. SortedSet에는 Comparable 객체만 넣을 수 있습니다. Comparable이 아닌 것들은 어느 것이 더 큰 놈인지 알 수 없기 때문입니다.

w5.overriding2.Cat 클래스가 java.lang.Comparable 인터페이스를 구현하도록 코드를 수정하겠습니다.

A screenshot of a computer program

Description automatically generated

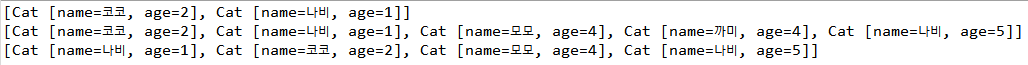
이렇게 함으로써 어린 고양이는 나이가 많은 고양이에게 공손해지는 "질서"가 만들어졌습니다. 이 경우 나이를 기준으로 natural ordering을 정의한 것입니다. (“@override” 부분은 어노테이션(annotation)이라고 합니다. 이 메소드가 재정의하는 코드라는 것을 컴파일러에게 알려줍니다. 이 어노테이션을 반드시 사용해야 하는 것은 아니지만 오류를 방지하는 데 도움을 줍니다.)

이제 고양이들을 SortedSet에 넣을 수 있습니다. SortedSet 인터페이스를 구현한 대표적인 클래스는 TreeSet입니다. 아래와 같은 main 메소드를 실행해 봅시다.

A screenshot of a computer program

Description automatically generated

실행 결과는 아래와 같습니다.



첫 두 줄은 Set을 출력한 것이고, 세 번째 출력은 SortedSet을 출력한 것입니다. 세 번째 출력을 보면 고양이들이 나이의 오름차순으로 정렬되어 있음을 볼 수 있습니다.

그런데 이상한 게 있지요? 두 번째 줄, Set 출력에는 원소가 다섯 개 있는데, 세 번째 줄 SortedSet 출력에는 원소가 네 개 밖에 없습니다. SortedSet은 Set의 서브인터페이스이므로 SortedSet도 Set의 일종이니까 이렇게 돼서는 곤란합니다. 일관성이 없지요.

이렇게 된 이유는 SortedSet이 두 원소가 같은지 판별할 때 compareTo 메소드를 사용하기 때문입니다. compareTo 메소드를 사용하면 두 원소가 같은지까지도 판별해 주기 때문에, 두 원소가 같은지 여부를 판별하기 위해 별도로 hashcode나 equals 메소드를 호출하지 않습니다. compareTo 메소드만으로 두 원소가 같은지 판별하는 경우, 두 고양이가 나이만 같으면 같다고 판별하게 됩니다. 그러니까, 위 예에서 1살짜리 Cookie 두 마리가 같다고 판별하고, 추가로 네 살 Butter와 네 살 Coffee도 서로 같다고 판별한 것입니다.

이렇게 되면 곤란합니다. Set과 SortedSet이 일관성이 유지되게 하려면 equals가 0일 때는 compareTo도 0이 되게 해 주어야 합니다. [API 문서도 이것을 명확히 권고하고 있습니다.](image/compareTo_equals.png) 아래에서 왼쪽이 참이면 오른쪽도 0이고, 오른쪽이 0이면 왼쪽도 참이 되도록 compareTo 메소드를 구현해야 합니다.

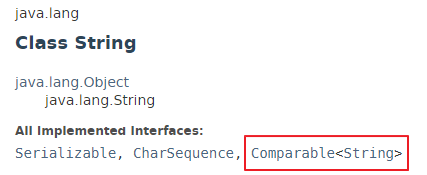
equals가 true <-- --> compareTo가 0

--------------------------------------------------------------------------------------------------------------

그러니까 compareTo를 구현할 때 나이에 따라 순서를 정하되 나이가 같은 경우에는 이름에 따라 (가령, 이름의 알파벳 오름차순으로) 순서가 정해지고, 나이와 이름 두 가지가 모두 같은 경우에만 0을 반환하게 해야 합니다.

이름은 String입니다. 두 개의 String을 알파벳 순으로 비교하려면 어떻게 하면 될까요?

아래 그림은 API 문서 중 [String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html) 클래스 머리 부분입니다.

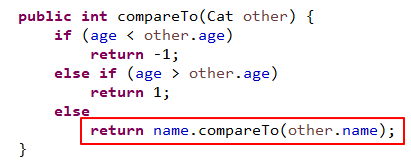


String 클래스는 Comparable<String> 인터페이스를 구현하고 있음을 알 수 있습니다. 그러니까 String은 compareTo 메소드를 지원합니다. String의 compareTo 메소드는 알파벳 순으로 크기를 판별합니다. 그러니까 String name1과 String name2가 있을 때, name1과 name2의 알파벳 선후관계를 알려면 아래와 같이 하면 됩니다.

name1.compareTo(name2)

위 값이 음수이면 name1이 name2보다 알파벳 순에서 앞서는 것입니다.

그러니까 Cat의 compareTo 메소드는 아래와 같이 적을 수 있습니다.



이 compareTo 메소드는 현재 고양이와 주어진 고양이(other)가 서로 나이와 이름이 모두 같을 때만 0을 반환합니다.

compareTo 메소드를 이렇게 수정하고 나서 main을 실행하면 아래와 같은 출력을 얻게 됩니다.

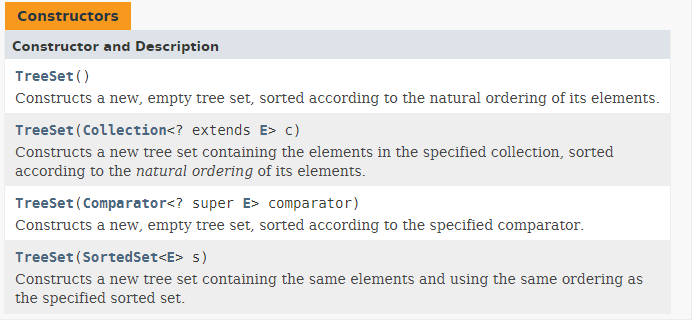
A screenshot of a computer

Description automatically generated

세 번째 줄에도 고양이가 다섯 마리만 보입니다. 이들은 나이 순으로, 나이가 같으면 이름 알파벳 순으로 정렬되어 있습니다. 여섯 마리 중 나이와 이름이 같은 두 마리는 같은 것으로 여겨져 한 마리만 보입니다.

지금까지 우리는 기본적으로 나이를 기준으로 고양이들의 질서(natural ordering)를 세웠지만, 때에 따라서는 다른 기준으로 정렬을 하고 싶을 수도 있습니다. 이럴 때 사용하는 게 바로 Comparator입니다. Comparator는 두 객체를 비교하여 어떤 게 더 큰지 판별해 주는 놈입니다.

TreeSet의 생성자는 아래와 같이 네 개가 있습니다.



우리가 지금까지 주로 사용해 온 것은 첫 번째, 두 번째 것이었습니다. 세 번째 것은 Comparator를 파라미터로 갖습니다. 세 번째 생성자를 이용해서 TreeSet을 만들면 이 TreeSet은 원소의 natural ordering에 따라 정렬하는 대신, 두 원소들 중 어떤 것이 더 큰지를 주어진 Comparator에게 일일이 물어보고 그 답에 따라 정렬해 줍니다.

이제 고양이 이름의 알파벳 순으로 고양이들을 먼저 정렬하고, 이름이 같은 경우에만 나이에 따라 정렬하는 프로그램을 작성해 봅시다. 단, Cat 클래스는 변경하지 않습니다. 그러니까 Cat의 natural ordering은 변하지 않습니다. 대신 Cat의 이름 알파벳 순으로 크기를 일차 판별하고, 이름이 같은 경우 나이에 따라 순서를 정해 주는 Comparator를 이용합니다.

Comparator는 클래스가 아니고 인터페이스이므로 Comparator 인터페이스를 구현한 클래스를 먼저 작성해야 합니다. 클래스가 있어야 객체를 생성할 수 있습니다.

아래는 [java.util.Comparator<T>](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Comparator.html) 인터페이스의 메소드입니다. (이 메소드 이름은 **compareTo**가 아니고 **compare**임에 유의하세요.) 우리는 이 메소드를 구현해야 합니다.



설명을 보면 o1이 작으면 음수를, o1이 크면 양수를, 같으면 0을 반환한다고 적혀 있습니다.

<참고> java.util.Comparator<T> 인터페이스에는 다른 메소드가 여러 개 더 있지만 이들은 모두 미리 구현되어 있는 static 메소드이거나 default 메소드이므로 프로그래머가 별도로 구현할 필요가 없습니다. 인터페이스를 구현하는 클래스들은 static 메소드나 default 메소드를 자동으로 갖게 됩니다.

아래 그림과 같은 클래스를 완성하세요.

Graphical user interface, text

Description automatically generated

compare 메소드는 c1의 이름이 c2의 이름보다 알파벳 순으로 앞서면 -1을, 그 반대이면 +1을 반환하면 됩니다. c1의 이름은 c1.name이고 c2의 이름은 c2.name입니다. 우리는 앞에서 Cat 클래스를 작성할 때 age와 name을 private으로 선언하지 않았으므로 같은 패키지 내에서는 이 필드를 읽을 수 있습니다. 만약 이 필드를 private으로 선언했다면 CatComparator 클래스에서 이렇게 바로 읽을 수 없습니다. Cat 클래스에 public getter 메소드를 두고 c1.getName()과 같은 방법으로 읽어야 합니다.

c1의 이름이 c2의 이름과 같으면 c1의 나이와 c2의 나이를 비교해야 합니다. c1의 나이가 c2의 나이보다 작으면 음수, 크면 양수, 같으면 0을 반환하게 합니다.

이렇게 해서 CatComparator 클래스를 작성했습니다. CatComparator 인스턴스를 생성하면 그 놈은 Comparator 인터페이스를 구현한 놈입니다. 즉, Comparator 객체입니다.

이제 SortedSet이 이 Comparator를 이용하여 고양이들을 정렬하게 해 봅시다.

A screenshot of a computer program

Description automatically generated

이 프로그램의 출력은 아래와 같습니다.

A screenshot of a computer

Description automatically generated

네 번째 줄에서 이름의 알파벳 순으로 우선 정렬하고 이름이 같은 경우에는 나이순으로 정렬된 것을 확인할 수 있습니다. 이름과 나이가 서로 같으면 같은 고양이라고 판별합니다. 물론, 이 경우에도 여섯 마리 고양이들 중 이름과 나이가 모두 같은 두 마리는 중복이 제거되어 그 중 한 마리만 포함됩니다.

w5.overriding2 패키지를 복사하여 w5.overridng3로 복사하고 w5.overriding3 패키지에서 다음 작업을 하세요.

고양이들의 이름과 나이를 입력하면 한 번은 natural ordering에 맞춰 정렬된 SortedSet을 출력하고 또 한 번은 이름 알파벳으로 우선 정렬, 이름이 같으면 나이순으로 정렬하는 SortedSet에 넣어 출력하도록 Cat의 main 프로그램을 수정하세요. 엔터 키를 연이어 두 번 누름으로써 빈 줄을 입력하거나, control-z(윈도), 혹은 control-d(ios)를 누름으로써 EOF(End-Of-File) 신호를 보내면 입력이 끝나도록 하세요.

실행 예는 아래와 같습니다.

A white background with black text

Description automatically generated

<힌트>

입력을 읽어 들일 때 Scanner의 nexLine 메소드를 사용하면 편리합니다. 빈 줄이 입력되면 nextLine은 빈 문자열(“”)을 반환합니다. EOF이 입력되면 Scanner의 hasNextLine 메소드가 false를 반환합니다.

A computer screen shot of a code

Description automatically generated

String 클래스의 strip 메소드는 문자열의 앞 뒤 white space를 제거합니다. split은 문자열을 단어들로 분해한 뒤, 단어들을 String 배열로 반환합니다. split을 호출할 때는, split이 문자열을 분해하여 단어들로 구분하여 나눌 때 사용할 구분자(delimiter)를 정규식 형태의 인자로 주어야 합니다. 정규식 “\\s+”는 whitespace가 하나 이상 연속된 것을 의미합니다.

w5.overriding3 패키지의 Cat과 CatComparator를 강온실 문제 w5p4\_CatComparator의 답으로 제출하세요. Cat은 public이고, CatComparator은 public이 아니어야 합니다. 패키지 선언문은 없어야 합니다.

끝.