Collection<E> → Set<E> interface Ⅱ

Collection<E> → Set<E> 인터페이스를 구현하는 컬랙션즈 클래스

Set<E> Inteface

ii. TreeSet<E>

- 집합의 특성
- 트리 자료구조를 기반으로 데이터 관리
- 인스턴스를 정렬된 순서로 저장한다.

```
import java.util.Iterator;
import java.util.TreeSet;

public class TreeSetDemo {
     public static void main(String[] args) {
          TreeSet<Integer> sTree = new TreeSet<Integer>();
          // 순서가 앞서는 인스턴스부터 정렬하여 인스턴스 저장
          sTree.add(1);
          sTree.add(2);
          sTree.add(4);
          sTree.add(3);
          sTree.add(2);

          System.out.println("저장된 데이터 수 : "+sTree.size());

          TreeSet<E>의 Iterator<E>는 검색 시, 최소값부터 오름차순으로 검색 (저장의 오름,내림차순과 관계없음)
            • 4 3 2 1
              ←←←검색시작

            • 1 2 3 4
        검색시작→→→
          Iterator<Integer> iterator = sTree.iterator();

          while(iterator.hasNext()){
               System.out.println(iterator.next());
          }
     }
}
```

```
Command Prompt

저장된 데이터의 수 : 4
1
2
3
4
```

TreeSet<E>의 정렬기준 정의하기

정렬기준은 TreeSet<E>에 전달되는 Comparable<E> 인터페이스를 구현한 인스턴스의 compareTo() Method로 정렬되기 때문에, Overriding 해서 직접 정의해주어야 한다.

- compareTo() Method
  - 정렬기준 결정 역할 (TreeSet은 순서가 앞서는 인스턴스 부터 정렬)
  - 인스턴스 자신보다 인자로 전달된 인스턴스가 앞서면 양의 정수 반환
  - 인스턴스 자신보다 인자로 전달된 인스턴스가 뒤서면 음의 정수 반환
  - 인스턴스 자신과 인자로 전달된 인스턴스가 같으면 0 반환
  - 인스턴스의 앞섬, 뒤섬의 비교기준은 없기 때문에 기준은 개발자가 직접 정의한다.
    (조건문의 조건에 따라 오름차순이 될 수도, 내림차순이 될 수도 있다.)

```
public class Person implements Comparable<Person> {
     String name;
     int age;

     public Person(String name, int age){ this.name = name; this.age = age; }

     public void showDate(){ System.out.println(name+" "+age); }

     // 크기를 이용한 정렬기준 정의하기 (오름차순)
     @Override
     public int compareTo(Person p) {
          if(age > p.age)
               return 1;
          else if(age < p.age)
               return -1;
          else
               return 0;
     }
}

public class TreeSetDemo {
     public static void main(String[] args) {
          TreeSet<Person> sTree = new TreeSet<Person>();
          sTree.add(new Person("Lee", 24));
          sTree.add(new Person("Hong", 29));
          sTree.add(new Person("Choi", 21));

          Iterator<Person> iterator = sTree.iterator();
          while(iterator.hasNext()){
               iterator.next().showDate();;
          }
     }
}
```
