**Java 배열 정렬 기능**

2017-03-19

이승진

**학습목표**

Java 표준 라이브러리의 배열 정렬 기능을 학습한다.

**목차**

[1. Arrays 클래스의 sort 메소드 2](#_Toc477723659)

[1) 기본 자료형 배열 정렬 2](#_Toc477723660)

[2) 일부분만 정렬 3](#_Toc477723661)

[3) 객체 배열 정렬 3](#_Toc477723662)

[2. Comparable 인터페이스 4](#_Toc477723663)

[1) Comparable 인터페이스 4](#_Toc477723664)

[2) 기본 자료형 배열과 객체 배열 5](#_Toc477723665)

[3) Person 클래스 배열 정렬 6](#_Toc477723666)

[3. Comparator 인터페이스 8](#_Toc477723667)

[1) 순서 기준이 몇 개인가? 8](#_Toc477723668)

[2) Comparator 인터페이스 8](#_Toc477723669)

[3) Person 클래스 배열 정렬 9](#_Toc477723670)

[4. 연습문제 11](#_Toc477723671)

[1) Integer 클래스 존재 이유 11](#_Toc477723672)

[2) 내림차순 정렬 11](#_Toc477723673)

# Arrays 클래스의 sort 메소드

## 기본 자료형 배열 정렬

Java의 숫자형 기본 자료형(byte, char, int, long, float, double)의 배열은

Arrays 클래스의 sort 메소드 사용하여 정렬할 수 있다.

Arrays 클래스의 sort 메소드

public static void sort(byte[] a);

public static void sort(char[] a);

public static void sort(int[] a);

public static void sort(long[] a);

public static void sort(float[] a);

public static void sort(double[] a);

### int[] 정렬

|  |
| --- |
| package net.skhu.sort;  import java.util.Arrays;  public class Example1 {  public static void main(String[] args) {  int[] a = { 17, 14, 11, 19, 13, 15, 20, 12, 16, 18 };  Arrays.sort(a);  System.out.println(Arrays.toString(a));  }  } |

### double[] 정렬

|  |
| --- |
| package net.skhu.sort;  import java.util.Arrays;  public class Example3 {  public static void main(String[] args) {  double[] a = { 17.1, 14.3, 11.2, 19.5, 11.3, 14.1, 17.5, 19.2 };  Arrays.sort(a);  System.out.println(Arrays.toString(a));  }  } |

## 일부분만 정렬

Arrays 클래스의 sort 메소드

public static void sort(byte[] a, int fromIndex, int toIndex);

public static void sort(char[] a, int fromIndex, int toIndex);

public static void sort(int[] a, int fromIndex, int toIndex);

public static void sort(long[] a, int fromIndex, int toIndex);

public static void sort(float[] a, int fromIndex, int toIndex);

public static void sort(double[] a, int fromIndex, int toIndex);

배열에서 fromIndex와 toIndex 사이의 항목만 정렬한다.

toIndex 위치는 포함하지 않음에 주의하자. (fromIndex <= 인덱스 < toIndex)

|  |
| --- |
| package net.skhu.sort;  import java.util.Arrays;  public class Example1c {  public static void main(String[] args) {  int[] a = { 17, 14, 11, 19, 13, 15, 20, 12, 16, 18 };  Arrays.sort(a, 0, 5);  System.out.println(Arrays.toString(a));  }  } |

## 객체 배열 정렬

### String[] 정렬

|  |
| --- |
| package net.skhu.sort;  import java.util.Arrays;  public class Example2a {  public static void main(String[] args) {  String[] a = { "zero", "one", "two", "three", "four", "five", "six", "seven" };  Arrays.sort(a);  System.out.println(Arrays.toString(a));  }  } |

### Integer[] 정렬

|  |
| --- |
| package net.skhu.sort;  import java.util.Arrays;  public class Example2b {  public static void main(String[] args) {  Integer[] a = { 17, 14, 11, 19, 13, 15, 20, 12, 16, 18 };  Arrays.sort(a);  System.out.println(Arrays.toString(a));  }  } |

위 예제에서 배열은 Integer 객체의 배열임에 주목하자.

# Comparable 인터페이스

## Comparable 인터페이스

Comparable 인터페이스를 구현한 클래스 객체의 배열은

Arrays 클래스의 sort 메소드 사용하여 정렬할 수 있다.

public static void sort(Comparable[] a);

### Comparable 인터페이스

|  |
| --- |
| interface Comparable<T> {  int compareTo(T obj)  } |

Comparable 인터페이스는 compareTo 메소드를 포함한다.

compareTo 메소드는, this 객체와 파라미터 객체를 비교하여, this 객체가 크면 양의 정수를 리턴하고,

this 객체가 작으면 음의 정수를 리턴하고, 두 객체의 값이 동일하면 0을 리턴한다.

### Comparable 인터페이스를 구현한 클래스

String, Date, Byte, Character, Integer, Long, Float, Double 등

https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html 참고

String 클래스, Integer 클래스도 Comparable 인터페이스를 구현했기 때문에,

Arrays.sort 메소드로 정렬할 수 있다.

## 기본 자료형 배열과 객체 배열

기본 자료형 배열에는 기본 자료형 값이 저장된다.

예를 들어 int 배열에는 int 값이 저장된다.

참조형 배열에는 참조만 저장된다.

예를 들어 Integer 객체 배열에는 객체에 대한 참조만 저장된다.

Java 언어에서, 변수나 배열 안에 객체가 들어있을 수 없다.

변수나 배열에 들어있을 수 있는 것은, 객체가 아니고, 객체에 대한 참조이다.

Java 언어에서 배열도 일종의 객체이다.

객체는 heap segment 영역에 생성된다.

|  |
| --- |
| package net.skhu.sort;  import java.util.Arrays;  public class Example3 {  public static void main(String[] args) {  int[] a1 = new int[] { 10, 11, 12 };  int[] a2 = { 10, 11, 12 };  Integer[] b1 = new Integer[] { new Integer(10), new Integer(11), new Integer(12) };  Integer[] b2 = new Integer[] { 10, 11, 12 };  Integer[] b3 = { 10, 11, 12 };  System.out.println(Arrays.toString(a1));  System.out.println(Arrays.toString(a2));  System.out.println(Arrays.toString(b1));  System.out.println(Arrays.toString(b2));  System.out.println(Arrays.toString(b3));  }  } |

위 예제에서 a1 배열과 a2 배열은 완전히 동일한 배열이다.

단지 a2를 좀 더 간결하게 구현했을 뿐이다.

마찬가지로 b1, b2, b3 배열로 서로 완전히 동일한 배열이다.

단지 b2와 b3를 좀 더 간결하게 구현했을 뿐이다.



## Person 클래스 배열 정렬

Person 클래스가 Comparable 인터페이스를 구현하면,

Arrays.sort 메소드를 사용하여 Person 객체 배열을 정렬할 수 있다.

### Person.java

|  |
| --- |
| package net.skhu.sort.comparator;  public class Person implements Comparable<Person> {  String name;  int age;  public Person(String name, int age) {  this.name = name;  this.age = age;  }  public String getName() {  return name;  }  public int getAge() {  return age;  }  @Override  public boolean equals(Object obj) {  if ((obj instanceof Person) == false) return false;  Person p = (Person)obj;  return (this.name == null ? p.name == null : this.name.equals(p.name)) && this.age == p.age;  }  @Override  public String toString() {  return String.format("Person{name=\"%s\", age=%d}", name, age);  }  @Override  public int compareTo(Person p) {  int r = this.name.compareTo(p.name); // 먼저 이름(name)을 비교한다.  if (r != 0) return r; // 이름이 같지 않다면, 이름 비교 결과를 리턴한다.  return this.age - p.age; // 이름이 같다면, 나이(age) 비교 결과를 리턴한다.  }  } |

compareTo 메소드는, this 객체와 파라미터 객체를 비교하여,

this 객체가 크면 양수를, this 객체가 작으면 음수를, 동일하면 0을 리턴하도록 구현해야 한다.

위 compareTo 메소드로 비교하여 정렬하면,

먼저 이름 순서로 정렬되고, 이름이 동일한 경우에 나이 순서로 정렬될 것이다.

### Example4.java

|  |
| --- |
| package net.skhu.sort.comparator;  import java.util.Arrays;  public class Example1 {  public static void main(String[] args) {  Person[] a = new Person[] {  new Person("홍길동", 18),  new Person("임꺽정", 22),  new Person("전우치", 23)  };  Arrays.sort(a);  System.out.println(Arrays.toString(a));  }  } |

# Comparator 인터페이스

## 순서 기준이 몇 개인가?

정수(integer)의 경우에 어느 수가 앞이고 어느 수가 뒤인지 순서가 정해져 있다.

날짜(date)의 경우에도 어느 날자가 앞이고 어느 날짜가 뒤인지 순서가 정해져 있다.

요일, 월, 가나다, abs 순서도 정해져 있다.

그런데 사람의 순서는 여러가지 기준이 있을 수 있다.

이름 순서, 나이 순서, 키 순서, 재산 순서 등등

### 순서 기준이 한 개인 경우

순서 기준이 한 개인 경우에는, 그 순서를 비교하는 메소드를 그 객체에 구현하는 것이 좋다.

즉 Integer 클래스, Date 클래스는 Comparable 인터페이스를 구현하여

그 클래스에 CompareTo 메소드를 구현한다.

### 순서 기준이 여러 개인 경우

순서 기준이 여러 개인 경우에는, 그 각각의 비교 방법을 별도의 클래스로 구현하는 것이 좋다.

즉 Person 클래스에 비교 방법을 구현하는 것 보다,

PersonNameComparator, PersonAgeComparater 등 비교 클래스를 따로 구현하는 것이 좋다.

이 비교 클래스는 Comparator 인터페이스를 구현하여 compare 메소드를 구현해야 한다.

## Comparator 인터페이스

|  |
| --- |
| interface Comparator<T> {  int compare(T obj1, T obj2);  } |

Comparator 인터페이스는 compare 메소드를 포함한다.

compare 메소드는, obj1 객체와 obj2 객체를 비교하여, obj1 객체가 크면 양의 정수를 리턴하고,

obj1 객체가 작으면 음의 정수를 리턴하고, 두 객체의 값이 동일하면 0을 리턴한다.

## Person 클래스 배열 정렬

### Person.java

|  |
| --- |
| package net.skhu.sort.comparator;  public class Person {  String name;  int age;  public Person(String name, int age) {  this.name = name;  this.age = age;  }  public String getName() {  return name;  }  public int getAge() {  return age;  }  @Override  public boolean equals(Object obj) {  if ((obj instanceof Person) == false) return false;  Person p = (Person)obj;  return (this.name == null ? p.name == null : this.name.equals(p.name)) && this.age == p.age;  }  @Override  public String toString() {  return String.format("Person{name=\"%s\", age=%d}", name, age);  }  } |

### PersonNameComparator.java

|  |
| --- |
| package net.skhu.sort.comparator;  import java.util.Comparator;  public class PersonNameComparator implements Comparator<Person> {  @Override  public int compare(Person p1, Person p2) {  int r = p1.name.compareTo(p2.name); // 먼저 이름(name)을 비교한다.  if (r != 0) return r; // 이름이 같지 않다면, 이름 비교 결과를 리턴한다.  return p1.age - p2.age; // 이름이 같다면, 나이(age) 비교 결과를 리턴한다.  }  } |

두 Person 객체가 동일한 경우에만 compare 메소드가 0을 리턴해야 하는 것이 바람직하기 때문에,

클래스 이름이 PersonNameComparator 이지만 age 까지 비교하였다.

### PersonAgeComparator.java

|  |
| --- |
| package net.skhu.sort.comparator;  import java.util.Comparator;  public class PersonAgeComparator implements Comparator<Person> {  @Override  public int compare(Person p1, Person p2) {  int r = p1.age - p2.age; // 먼저 나이(age)를 비교한다.  if (r != 0) return r; // 나이가 같지 않다면, 나이 비교 결과를 리턴한다.  return p1.name.compareTo(p2.name); // 나이가 같다면, 이름(name) 비교 결과를 리턴한다.  }  } |

두 Person 객체가 동일한 경우에만 compare 메소드가 0을 리턴해야 하는 것이 바람직하기 때문에,

클래스 이름이 PersonAgeComparator 이지만 name 까지 비교하였다.

### Example1.java

|  |
| --- |
| package net.skhu.sort.comparator;  import java.util.Arrays;  public class Example1 {  public static void main(String[] args) {  Person[] a = new Person[] {  new Person("홍길동", 18),  new Person("임꺽정", 22),  new Person("전우치", 23)  };  Arrays.sort(a, new PersonNameComparator());  System.out.println(Arrays.toString(a));  Arrays.sort(a, new PersonAgeComparator());  System.out.println(Arrays.toString(a));  }  } |

먼저 이름 순서로 정렬하여 출력하고,

그 다음 나이 순서로 정렬하여 출력한다.

# 연습문제

## Integer 클래스 존재 이유

Java에 기본 자료형 int가 이미 있는데, Integer 클래스가 필요한 이유는 무엇일까

생각해 봅시다.

## 내림차순 정렬

이 강의 노트 예제들을 내림차순 정렬로 수정해 보라.

기본 자료형을 정렬하는 Arrays 클래스의 sort 메소드에는 내림차순 정렬 기능이 없다.

오름차순으로 정렬한 수 배열의 순서를 뒤집어야한다.

객체를 정렬하는 경우에는, Comparator 클래스의 compare 메소드를 수정하면

내림차순 정렬이 가능하다.