정렬과 검색
(sorting & searching)

egyou@induk.ac.kr

ADOPTIUM

기본 준비

- **입력 방법**
  - 배열 또는 collection 객체로 선언 및 초기화
    - System.in, Scanner 등을 활용한 입력
    - 테스트를 위한 랜덤 요소들 생성
- **활용 기술**
  - 기본 문법을 활용
  - Arrays, JCF, Comparable, Comparator 등 활용

기본 정렬

- **정렬 알고리즘**
  - 기본 : selection, insertion, bubble, merge, quick
- **최대 / 최소값**
  - 오름 차순 정렬 후 맨 마지막 값 / 맨 처음 값
  - 내림 차순 정렬 후 맨 처음 값 / 맨 마지막 값

StringOrderTest.java - String 배열 선택 정렬

```
package iducs.java.library;

public class StringOrderTest {
	public static void main(String[] args) {
		StringOrderTest sot = new StringOrderTest();

		String[] names = new String[10];

		for(int i=0; i < names.length; i++) {
			names[i] = "";
			for(int j=0; j < 5; j++) {
				 double dValue = Math.random();
				char cValue = (char)((dValue * 26) + 65);
				//char cValue = (char)((dValue * 26) + 97);
				names[i] = names[i] + "" + cValue;
			}
		}
```

```
        sot.printList(names, "random names list");
        long startTimeMillis = System.currentTimeMillis();
        String[] ascending = sot.ascendingOrder(names);
        sot.printList(ascending, "ascending ordered list");

        String[] descending = sot.descendingOrder(names);
        sot.printList(descending, "descending ordered list");
        long endTimeMillis = System.currentTimeMillis();
        System.out.println("₩n소요시간 : " + (endTimeMillis - startTimeMillis) + " ms");
    }

    private void printList(String[] strArr, String msg) {
        System.out.println("₩n---------- " + msg + " ----------");
        for(int k=0; k < strArr.length; k++) {
            System.out.printf("%-7s", strArr[k]);
        }
    }
```

계속

```
public String[] ascendingOrder(String[] strArr) {
    for(int i=0; i < strArr.length; i++) {
        for(int j = i+1; j < strArr.length; j++) {
            if(strArr[i].compareTo(strArr[j]) > 0) { // 기준값이 비교값보다 큰 경우 교체
                String imsi = strArr[i];
                strArr[i] = strArr[j];
                strArr[j] = imsi;
            }
        }
        //printList(names, "step" + (i + 1));
    }
    return strArr;
}
```

계속

```
    public String[] descendingOrder(String[] strArr) {
        for(int i=0; i < strArr.length; i++) {
            for(int j = i+1; j < strArr.length; j++) {
                if(strArr[i].compareTo(strArr[j]) < 0) { // 기준값이 비교값보다 적은 경우 교체
                    String imsi = strArr[i];
                    strArr[i] = strArr[j];
                    strArr[j] = imsi;
                }
            }
            //printList(names, "step" + (i + 1));
        }
        return strArr;
    }
}
```

StringOrderTest2.java - Arrays와 Comparator 활용

```
package iducs.java.library;

import java.util.Arrays;
import java.util.Comparator;

public class StringOrderTest2 {
	public static void main(String[] args) {
		StringOrderTest2 sot = new StringOrderTest2();

		String[] names = new String[10];

		for(int i=0; i < names.length; i++) {
			names[i] = "";
			for(int j=0; j < 5; j++) {
				 double dValue = Math.random();
				char cValue = (char)((dValue * 26) + 65); // 대문자
				//char cValue = (char)((dValue * 26) + 97); // 소문자
				names[i] = names[i] + "" + cValue;
			}
		}
```

계속

```
        sot.printList(names, "random names list");
        long startTimeMillis = System.currentTimeMillis();
        sot.ascList(names);
        sot.printList(names, "ascending ordered list");

        sot.desList(names);
        sot.printList(names, "descending ordered list");
        long endTimeMillis = System.currentTimeMillis();
        System.out.println("₩n소요시간 : " + (endTimeMillis - startTimeMillis) + " ms");
    }


    public void printList(String[] strArr, String msg) {
        System.out.println("₩n---------- " + msg + " ----------");
        for(int k=0; k < strArr.length; k++) {
            System.out.printf("%-7s", strArr[k]);
        }
    }
```

계속

```
    public void ascList(String[] strArr) {
        Comparator<String> asc = new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                // TODO Auto-generated method stub
                return o1.compareTo(o2);
            }
        };
        Arrays.sort(strArr, asc);
    }

    public void desList(String[] strArr) {
        Comparator<String> desc = new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                // TODO Auto-generated method stub
                return o2.compareTo(o1);
            }
        };
        Arrays.sort(strArr, desc);
    }
}
```

기본 검색

- **검색 알고리즘**
  - 순서 리스트(ordered list) 또는 비순서화된 리스트 등에서 어떤 원소/대상, 즉 키의 존재 및 그 위치를 찾는 방법
  - Linear, Binary,

- **자바 라이브러리에서 구현**
  - contains(), indexOf(), lastIndexOf(), binarySearch() ...

JavaSearchTest.java

```
package iducs.java.library;
import java.util.Arrays;
public class JavaUtilArraysBinarySearchTest {
	public static void main(String[] args) {
		int[] intArray = {1, 2, 9, 8, 7, 6, 5, 4, 3};
		System.out.println(seqSearch(intArray, 7));
		System.out.println(seqSearch(intArray, 11));

		Arrays.sort(intArray);
		System.out.println(Arrays.binarySearch(intArray, 7));
		System.out.println(Arrays.binarySearch(intArray, 11));
	}
	static int seqSearch(int[] a, int key){
		int length = a.length;
		int i = 0;
		for(i=0; i < length; i++){
			if(key == a[i]) // 찾은 경우
				break;
		}
		return i == length ? -(length + 1):i;
		// i의 값이 배열의 크기와 같다면 -(length+1) 리턴, 같지않으면 i를 리턴
	}
}
```

검색 위치 반환

```
 static int[] findKeys(int[] a, int key){
        int length = a.length;
        int[] positions = new int[length];
        //배열의 크기보다 작거나 같을때까지 반복...
        int j = 0;
        for(i=0; i<n; i++){
          if(a[i] == key)
              positions[j++] = i;
        }
        return positions;
}
```

java.util.ArrayList

주요 특징

- 순서를 가지는 요소들의 모임
- 중복을 허용하고, 인덱스를 이용한 접근을 허용함
- 저장되는 요소들의 개수에 따라 자동적으로 크기가 변경됨
- 추가 연산을 실행하는 경우 오버헤드가 발생함 vs. LinkedList
- 스레드 안전 하지 않음(thread-unsafe) vs. Vector

상속 관계

- public class ArrayList<E>
  extends AbstractList<E>
  implements LIst<E>, RandomAccess, Cloneable, Serializable

java.util.ArrayList

- **주요 메소드**
  - public boolean add(Element E) : 리스트의 마지막에 지정한 요소를 추가
  - public E set(int index, Element E) : 리스트의 지정한 위치에 지정한 요소를 추가
  - public E get(int index) : 리스트의 지정한 위치에 있는 요소를 반환
  - public boolean remove(Object o) : 리스트에서 지정한 요소가 첫번째로 발견되는 위치의 요소를 제거
  - public E remove(int index) : 리스트의 지정한 위치에 있는 요소를 제거
    - IndexOutOfBoundsException - if the index is out of range (index < 0 || index >= size())

ArrayListLoopTest

```java
package iducs.java.jcf;
import java.util.*;
public class ArrayListLoopTest{
    public static void main(String[] args) {
        ArrayList<Integer> arrlist = new ArrayList<Integer>();
        arrlist.add(11);      arrlist.add(15);      arrlist.add(12);      arrlist.add(17);
        for (int counter = 0; counter < arrlist.size(); counter++) {
            System.out.print(arrlist.get(counter) + ", ");
        }
        System.out.print("\n");
        int count = 0;
        while (arrlist.size() > count) {     System.out.print(arrlist.get(count++) + ", ");}
        System.out.print("\n");
        // Enhanced Loop Statement 사용
        for (Integer num : arrlist) {   System.out.print(num + ", ");     }
        System.out.print("\n");
        // Iterator 사용
        Iterator iter = arrlist.iterator();
        while (iter.hasNext()) {        System.out.print(iter.next() + ", "); }

    }
}
```

ArrayListAddRemoveTest.java

```java
package iducs.java.jcf;

import java.util.*;
public class ArrayListAddRemoveTest {
    public static void main(String args[]) {
        ArrayList<String> arrayList = new ArrayList<String>();
        System.out.println("Initial size of al: " + arrayList.size());

        arrayList.add("C"); // index is 0
        arrayList.add("A");          arrayList.add("E");      arrayList.add("B");
        arrayList.add("D");      arrayList.add("F");      arrayList.add( index: 1,  element: "A2");
        System.out.println("추가 후 ArrayList 사이즈 : " + arrayList.size());
        System.out.println("ArrayList 내용 : " + arrayList);

        arrayList.remove( o: "F");
        arrayList.remove( index: 2);
        System.out.println("삭제 후 ArrayList 사이즈 : " + arrayList.size());
        System.out.println("ArrayList 내용 : " + arrayList);
    }
}
```

계속

- **크기 비교**
  - ArrayList 형 객체의 크기 : arrayList.size()
  - 배열형 객체의 크기 : arrayName.length
  - String형 객체의 크기 : strName.length()

ArrayListSizeTester.java

```java
package iducs.java.jcf;
import java.util.ArrayList;

public class ArrayListSizeTester {
    public static void main(String [] args)    {
        ArrayList<Integer> arrayList=new ArrayList<Integer>();
        System.out.println("최초 ArrayList 사이즈 : "+arrayList.size());
        arrayList.add(1);
        arrayList.add(2); // index 1
        arrayList.add(3);
        arrayList.add(4); // index  3
        arrayList.add(5);
        System.out.println("요소 추가 후 ArrayList 크기 : "+arrayList.size());
        arrayList.remove( index: 1); // 현재 ArrayList중 2번째 인덱스 요소 제거, ArrayList의 크기가 5에서 4로 작아짐
        arrayList.remove( index: 3); // 현재 ArrayList중 4번째 인덱스 요소 제거, 최초 ArrayList의 경우 마지막 요소
        System.out.println("요소 제거 후 ArrayList 크기: "+arrayList.size());
        System.out.println("ArrayList 요소 출력 : ");
        for(int num: arrayList){
            System.out.println(num);
        }
    }
}
```

java.util.Collections 활용

- **정의**
  - 컬렉션들에 대한 다양한 연산을 수행하거나 반환해주는 정적 메소드를 제공하는 클래스
- **상속 관계**
  - public class Collections extends Object

계속

▪주요 메소드

- public static <T extends Comparable<? super T>> void sort(List<T> list) : 대상 리스트를 요소들의 자연스러운 순서 매기기에 따라 오름차순 정렬
- public static <T> void sort(List<T> list, Comparator<? super T> c) : 대상 리스트를 지정한 비교자에 의해 유도된 오름차순으로 정렬

ArrayListStringSort.java

```java
package iducs.java.jcf;

import java.util.*;
public class ArrayListStringSort {
    public static void main(String args[]){
        ArrayList<String> arrayList= new ArrayList<String>();
        arrayList.add("Korea");
        arrayList.add("Denmark");
        arrayList.add("France");
        arrayList.add("India");
        System.out.println("정렬 전 :");
        for(String countryName: arrayList){
            System.out.println(countryName);
        }
        Collections.sort(arrayList);
        System.out.println("정렬 후:");
        for(String countryName: arrayList){
            System.out.println(countryName);
        }
    }
}
```

ArrayListSortDescending.java

```java
package iducs.java.jcf;

import java.util.*;
public class ArrayListStringSortDesc  {
    public static void main(String args[]){
        ArrayList<String> arrayList = new ArrayList<String>();
        arrayList.add("Korea");
        arrayList.add("Denmark");
        arrayList.add("France");
        arrayList.add("India");
        System.out.println("정렬 전:");
        for(String countryName: arrayList){
            System.out.println(countryName);
        }
        Collections.sort(arrayList, Collections.reverseOrder());
        System.out.println("정렬 후 (내림차순):");
        for(String countryName: arrayList){
            System.out.println(countryName);
        }
    }
}
```

ArrayListIntergerSort.java

```java
package iducs.java.jcf;
import java.util.*;
public class ArrayListIntergerSort  {
    public static void main(String args[]){
        ArrayList<Integer> arraylist = new ArrayList<Integer>();
        arraylist.add(11);
        arraylist.add(5);
        arraylist.add(7);
        arraylist.add(3);
        System.out.println("정렬 전:");
        for(int counter: arraylist){
            System.out.println(counter);
        }
        Collections.sort(arraylist);
        System.out.println("정렬 후:");
        for(int counter: arraylist){
            System.out.println(counter);
        }
    }
}
```

비교를 위한 인터페이스

- **Comparable<T> interface**
  - int compareTo(T o) : 현재 객체 c와 o를 비교
    - negative : 오름 차순 정렬 시 c, o2 순서
    - zero : 오름 차순 정렬 시 c, o2 순서 유지
    - positive : 오름 차순 정렬 시 o2, c 순서
- **Comparator<T> interface**
  - int compare(T o1, T o2) : 순서를 정하기 위해서 o1과 o2를 비교
    - negative : 오름 차순 정렬 시 o1, o2 순서
    - zero : 오름 차순 정렬 시 o1, o2 순서 유지
    - positive : 오름 차순 정렬 시 o2, o1 순서
  - boolean equals(Object obj)

계속

- **활용**
  - comparable 인터페이스는 객체의 특정 속성을 기준으로 정렬이 가능하도록 하는 클래스 정의에 활용
  - comparator 인터페이스는 객체에 대하여 다양한 속성기준으로 정렬이 가능하도록

ComparableMember.java 1

```
public class ComparableMember<Object> implements Comparable<Object> {
	private String memberName;
	private int rollno;
	private int memberPhone;
	public ComparableMember(int rollno, String memberName, int memberPhone) {
		this.rollno = rollno;
		this.memberName = memberName;
		this.memberPhone = memberPhone;
	}
	public String getmemberName() {
		return memberName;
	}
	public void setmemberName(String memberName) {
		this.memberName = memberName;
	}
	public int getRollno() {
		return rollno;
	}
	public void setRollno(int rollno) {
		this.rollno = rollno;
	}
```

ComparableMember.java 2

```
    public int getmemberPhone() {
        return memberPhone;
    }
    public void setmemberPhone(int memberPhone) {
        this.memberPhone = memberPhone;
    }
    public String toString() {
        return "[ 번호 =" + rollno + ", name=" + memberName + ", 전화번호 =" + memberPhone + "]";
    }

    @Override
    public int compareTo(Object arg0) {
        int comparePhone = ((ComparableMember<Object>) arg0).getmemberPhone();
        // 오름차순
        return this.memberPhone - comparePhone;
        // 내림차순
        // return comparePhone - this.memberPhone;
    }
}
```

ComparableMemberTest.java

```
import java.util.*;


public class ComparableMemberTest  {
	@SuppressWarnings({ "rawtypes", "unchecked" })
	public static void main(String args[]){
		ArrayList<ComparableMember> arraylist = new ArrayList<ComparableMember>();
		arraylist.add(new ComparableMember(223, "apple", 31));
		arraylist.add(new ComparableMember(245, "mango", 30));
		arraylist.add(new ComparableMember(209, "banana", 32));
		arraylist.add(new ComparableMember(215, "kiwi", 28));
		arraylist.add(new ComparableMember(233, "melon", 29));
		// 전화번호로 정렬
		Collections.sort(arraylist);

		for(ComparableMember str: arraylist){
			System.out.println(str);
		}
	}
}
```

ComparatorMember.java 1

```
package induk.soft.collection;


import java.util.Comparator;
public class ComparatorMember  {
	private String memberName;
	private int rollno;
	private int memberPhone;

	public ComparatorMember(int rollno, String memberName, int memberPhone) {
		this.rollno = rollno;
		this.memberName = memberName;
		this.memberPhone = memberPhone;
	}
	public String getmemberName() {
		return memberName;
	}
	public void setmemberName(String memberName) {
		this.memberName = memberName;
	}
	public int getRollno() {
		return rollno;
	}
	public void setRollno(int rollno) {
		this.rollno = rollno;
	}
```

ComparatorMember.java 2

```
public int getmemberPhone() {
        return memberPhone;
}
public void setmemberPhone(int memberPhone) {
        this.memberPhone = memberPhone;
}

public static Comparator<ComparatorMember> MemNameComparator =
        new Comparator<ComparatorMember>() {
        public int compare(ComparatorMember s1, ComparatorMember s2) {
                String memberName1 = s1.getmemberName().toUpperCase();
                String memberName2 = s2.getmemberName().toUpperCase();
                //오름 차순(ascending order)
                return memberName1.compareTo(memberName2);

                //내림 차순 (descending order)
                //return memberName2.compareTo(memberName1);
        }
};
```

ComparatorMember.java 3

```
    public static Comparator<ComparatorMember> MemNoComparator =
        new Comparator<ComparatorMember>() {

        public int compare(ComparatorMember s1, ComparatorMember s2) {

            int rollno1 = s1.getRollno();

            int rollno2 = s2.getRollno();

            //오름 차순(ascending order)

            return rollno1-rollno2;

            //내림 차순 (descending order)

            //rollno2-rollno1;

        }

    };

    public String toString() {

        return "[ 번호=" + rollno + ", name=" + memberName + ", 전화번호=" + memberPhone + "]";

    }


}
```

ComparatorMemberTest.java

```
import java.util.ArrayList;

import java.util.Collections;

public class ComparatorMemberTest {

	public static void main(String args[]){

		ArrayList<ComparatorMember> arraylist = new ArrayList<ComparatorMember>();

		arraylist.add(new ComparatorMember(11, "yeyeong", 14));

		arraylist.add(new ComparatorMember(12, "junyeong", 10));

		arraylist.add(new ComparatorMember(23, "jeongwon", 18));

		arraylist.add(new ComparatorMember(24, "joowon", 17));

		arraylist.add(new ComparatorMember(35, "soogyeum", 19));

		arraylist.add(new ComparatorMember(36, "minha", 15));


		System.out.println("이름으로 오름 차순:");

		Collections.sort(arraylist, ComparatorMember.MemNameComparator);

		for(ComparatorMember str: arraylist){

			System.out.println(str);

		}

		System.out.println("번호로 오름 차순:");

		Collections.sort(arraylist, ComparatorMember.MemNoComparator);

		for(ComparatorMember str: arraylist){

			System.out.println(str);

		}

	}

}
```