

# Chapter *11*

컬렉션 프레임워크

Collections Framework

## [ 연습문제 ]

**[11-1]** 다음은 정수집합 1,2,3,4와 3,4,5,6의 교집합, 차집합, 합집합을 구하는 코드이다. 코드를 완성하여 실행결과와 같은 결과를 출력하시오.

**[Hint]** ArrayList클래스의 addAll(), removeAll(), retainAll()을 사용하라.

**[연습문제]/ch11/Exercisel1\_1.java**

```
import java.util.*;

class Exercisel1_1 {
    public static void main(String[] args) {
        ArrayList list1 = new ArrayList();
        ArrayList list2 = new ArrayList();
        ArrayList kyo = new ArrayList(); // 교집합
        ArrayList cha = new ArrayList(); // 차집합
        ArrayList hap = new ArrayList(); // 합집합

        list1.add(1);
        list1.add(2);
        list1.add(3);
        list1.add(4);

        list2.add(3);
        list2.add(4);
        list2.add(5);
        list2.add(6);

        /*
            (1) 알맞은 코드를 넣어 완성하시오.
        */

        System.out.println("list1="+list1);
        System.out.println("list2="+list2);
        System.out.println("kyo="+kyo);
        System.out.println("cha="+cha);
        System.out.println("hap="+hap);
    }
}
```

```
kyo.addAll(list1); // list1의 모든 요소를 kyo에 저장한다.
kyo.retainAll(list2); // list2와 kyo의 공통요소만 남기고 삭제한다.

cha.addAll(list1);
cha.removeAll(list2); // cha에서 list2와 공통된 요소들을 모두 삭제한다.

hap.addAll(list1); // list1의 모든 요소를 hap에 저장한다.
hap.removeAll(kyo); // hap에서 kyo와 공통된 모든 요소를 삭제한다.
hap.addAll(list2); // list2의 모든 요소를 hap에 저장한다.
```

**[실행결과]**

```
list1=[1, 2, 3, 4]
list2=[3, 4, 5, 6]
kyo=[3, 4]
cha=[1, 2]
hap=[1, 2, 3, 4, 5, 6]
```

**[11-2]** 다음 코드의 실행결과를 적으시오.

**[연습문제]/ch11/Exercise11\_2.java**

```
import java.util.*;

class Exercise11_2 {
    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(3);
        list.add(6);
        list.add(2);
        list.add(2);
        list.add(2);
        list.add(7);

        HashSet set = new HashSet(list); // 중복요소들이 제거되고 순서유지 안됨 2,6,3,7
        TreeSet tset = new TreeSet(set); // 오름차순으로 정렬된다. 2,3,6,7
        Stack stack = new Stack(); // Stack에 넣었다 꺼내면 저장순서가 반대가 된다.
        stack.addAll(tset); // TreeSet의 저장된 모든 요소를 stack에 담는다.

        while(!stack.empty())
            System.out.println(stack.pop());
    }
}
```

**[실행 결과]**

7  
6  
3  
2

**[11-3]** 다음 중 ArrayList에서 제일 비용이 많이 드는 작업은? 단, 작업도중에 ArrayList의 크기 변경이 발생하지 않는다고 가정한다.

- a. 첫 번째 요소 삭제
- b. 마지막 요소 삭제
- c. 마지막에 새로운 요소 추가
- d. 중간에 새로운 요소 추가

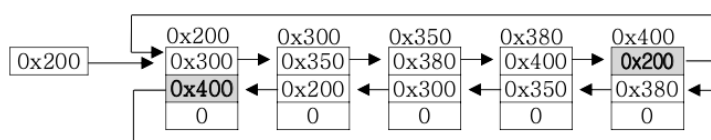
배열은 크기를 변경할 수 없기 때문에 저장할 공간이 부족하면 새로운 배열을 만들고 내용을 복사해야 하므로 많은 비용이 든다. 배열의 중간이나 마지막에 새로운 요소를 찾거나 삭제 등을 하는 작업보다 첫 번째 요소를 찾아서 작업하는 것이 더 많은 비용이 든다.

**[11-4]** LinkedList클래스는 이름과 달리 실제로는 이중 원형 연결리스트(doubly circular linked list)로 구현되어 있다. LinkedList인스턴스를 생성하고 11개의 요소를 추가했을 때, 이 11개의 요소 중 접근시간(access time)이 가장 오래 걸리는 요소는 몇 번째 요소인가?

6번째 요소(LinkedList에서 제일 가운데에 위치한 요소)

원형 연결리스트는 첫 번째 요소와 마지막 요소를 연결해서 LinkedList의 단점인 접근성을 향상시켰다. 마지막 요소에 접근하기 위해서는 첫 번째 요소에서 한 번만 이동하면 된다. 마지막 요소에서 첫 번째 요소에 접근하기 위해서도 한 번만 이동하면 된다.

마치 TV리모콘의 첫 번째 채널에서 감소시키면 마지막 채널로 이동하고 마지막 채널에서 채널을 증가시키면 첫 번째 채널로 이동하는 것과 같다고 할 수 있다.



**[11-5]** 다음에 제시된 Student클래스가 Comparable인터페이스를 구현하도록 변경해서 이름(name)이 기본 정렬기준이 되도록 하시오.

**[연습문제]/ch11/Exercisel1\_5.java**

```
import java.util.*;

class Student implements Comparable {
    String name;
    int    ban;
    int    no;
    int    kor, eng, math;

    Student(String name, int ban, int no, int kor, int eng, int math) {
        this.name = name;
        this.ban  = ban;
        this.no   = no;
        this.kor  = kor;
        this.eng  = eng;
        this.math = math;
    }

    int getTotal() {
        return kor+eng+math;
    }

    float getAverage() {
        return (int)((getTotal()/ 3f)*10+0.5)/10f;
    }

    public String toString() {
        return name + "," +ban + "," +no + "," +kor  + "," +eng + "," +math
            + "," +getTotal() + "," +getAverage();
    }
}

class Exercisel1_5 {
    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(new Student("홍길동",1,1,100,100,100));
        list.add(new Student("남궁성",1,2,90,70,80));
        list.add(new Student("김자바",1,3,80,80,90));
        list.add(new Student("이자바",1,4,70,90,70));
        list.add(new Student("안자바",1,5,60,100,80));

        Collections.sort(list);
        Iterator it = list.iterator();

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

```
public int compareTo(Object o) {
    if(o instanceof Student) {
        Student tmp = (Student)o;

        // String클래스의 compareTo()를 호출
        return name.compareTo(tmp.name);
    } else {
        return -1;
    }
}
```

**[실행결과]**

```
김자바,1,3,80,80,90,250,83.3
남궁성,1,2,90,70,80,240,80.0
안자바,1,5,60,100,80,240,80.0
이자바,1,4,70,90,70,230,76.7
홍길동,1,1,100,100,100,300,100.0
```

Collections.sort(List list)를 이용하면 ArrayList에 저장된 요소들을 쉽게 정렬할 수 있다.

List객체를 파라미터로 받는 경우는 제네릭 타입인 클래스가 Comparable를 구현하고 있어야 사용 가능

Comparator를 파라미터로 받는 경우에는 제네릭 타입 클래스가 Comparable을 구현하고 있지 않아도 사용할 수 있다. 이러한 경우에는 Comparator 인터페이스에 구현된 정렬 기준으로 정렬이 수행된다.

**[11-6]** 다음의 코드는 성적평균의 범위별로 학생 수를 세기 위한 것이다. TreeSet이 학생들의 평균을 기준으로 정렬하도록 compare(Object o1, Object o2)와 평균점수의 범위를 주면 해당 범위에 속한 학생의 수를 반환하는 getGroupCount()를 완성하라.

**[Hint]** TreeSet의 subSet(Object from, Object to)를 사용하라.

**[연습문제]/ch11/Exercise11\_6.java**

```
import java.util.*;

class Student implements Comparable {
    String name;
    int    ban;
    int    no;
    int    kor;
    int    eng;
    int    math;

    Student(String name, int ban, int no, int kor, int eng, int math) {
        this.name = name;
        this.ban  = ban;
        this.no   = no;
        this.kor  = kor;
        this.eng  = eng;
        this.math = math;
    }

    int getTotal() {
        return kor+eng+math;
    }

    float getAverage() {
        return (int)((getTotal()/ 3f)*10+0.5)/10f;
    }

    public String toString() {
        return name
            +", "+ban
            +", "+no
            +", "+kor
            +", "+eng
            +", "+math
            +", "+getTotal()
            +", "+getAverage()
            ;
    }

    public int compareTo(Object o) {
        if(o instanceof Student) {
            Student tmp = (Student)o;

            return name.compareTo(tmp.name);
        } else {
            return -1;
        }
    }
} // class Student
```

```

class Exercisel1_6 {
    static int getGroupCount(TreeSet tset, int from, int to) {
        /*
            (1) 알맞은 코드를 넣어 완성하시오.
        */
    }

    public static void main(String[] args) {
        TreeSet set = new TreeSet(new Comparator() {
            public int compare(Object o1, Object o2) {
                /*
                    (2) 알맞은 코드를 넣어 완성하시오.
                */
            }
        });

        set.add(new Student("홍길동", 1, 1, 100, 100, 100));
        set.add(new Student("남궁성", 1, 2, 90, 70, 80));
        set.add(new Student("김자바", 1, 3, 80, 80, 90));
        set.add(new Student("이자바", 1, 4, 70, 90, 70));
        set.add(new Student("안자바", 1, 5, 60, 100, 80));

        Iterator it = set.iterator();

        while(it.hasNext())
            System.out.println(it.next());

        System.out.println("[60~69] :"+getGroupCount(set, 60, 70));
        System.out.println("[70~79] :"+getGroupCount(set, 70, 80));
        System.out.println("[80~89] :"+getGroupCount(set, 80, 90));
        System.out.println("[90~100] :"+getGroupCount(set, 90, 101));
    }
}

```

```

Student s1 = new Student("", 0, 0, from, from, from);
Student s2 = new Student("", 0, 0, to, to, to);
return tset.subSet(s1, s2).size();

```

```

if(o1 instanceof Student && o2 instanceof Student) {
    Student s1 = (Student)o1;
    Student s2 = (Student)o2;
    return (int)(s1.getAverage() - s2.getAverage());
}
return -1;

```

**[실행결과]**

```

이자바, 1, 4, 70, 90, 70, 230, 76.7
남궁성, 1, 2, 90, 70, 80, 240, 80.0
김자바, 1, 3, 80, 80, 90, 250, 83.3
홍길동, 1, 1, 100, 100, 100, 300, 100.0
[60~69] :0
[70~79] :1
[80~89] :2
[90~100] :1

```

**[11-7]** 다음에 제시된 BanNoAscending클래스를 완성하여, ArrayList에 담긴 Student인스턴스들이 반(ban)과 번호(no)로 오름차순 정렬되게 하시오.(반이 같은 경우 번호를 비교해서 정렬한다.)

**[연습문제]/ch11/Exercisel1\_7.java**

```
import java.util.*;

class Student {
    String name;
    int    ban;
    int    no;
    int    kor;
    int    eng;
    int    math;

    Student(String name, int ban, int no, int kor, int eng, int math) {
        this.name = name;
        this.ban  = ban;
        this.no   = no;
        this.kor  = kor;
        this.eng  = eng;
        this.math = math;
    }

    int getTotal() {
        return kor+eng+math;
    }

    float getAverage() {
        return (int)((getTotal()/ 3f)*10+0.5)/10f;
    }

    public String toString() {
        return name
            +", "+ban
            +", "+no
            +", "+kor
            +", "+eng
            +", "+math
            +", "+getTotal()
            +", "+getAverage()
            ;
    }
} // class Student

class BanNoAscending implements Comparator {
    public int compare(Object o1, Object o2) {
        /*
            (1) 알맞은 코드를 넣어 완성하시오.
        */
    }
}

class Exercisel1_7 {
    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(new Student("이자바", 2, 1, 70, 90, 70));
    }
}
```

```
if(o1 instanceof Student && o2 instanceof Student) {
    Student s1 = (Student)o1;
    Student s2 = (Student)o2;
    int result = s1.ban - s2.ban;

    if(result==0) { // 반이 같으면, 번호를 비교한다.
        return s1.no - s2.no;
    }
    return result;
}
return -1;
```

```
list.add(new Student("안자바", 2, 2, 60, 100, 80));  
list.add(new Student("홍길동", 1, 3, 100, 100, 100));  
list.add(new Student("남궁성", 1, 1, 90, 70, 80));  
list.add(new Student("김자바", 1, 2, 80, 80, 90));  
  
Collections.sort(list, new BanNoAscending());  
  
Iterator it = list.iterator();  
  
while(it.hasNext())  
    System.out.println(it.next());  
}  
}
```

**[실행결과]**

남궁성, 1, 1, 90, 70, 80, 240, 80.0  
김자바, 1, 2, 80, 80, 90, 250, 83.3  
홍길동, 1, 3, 100, 100, 100, 300, 100.0  
이자바, 2, 1, 70, 90, 70, 230, 76.7  
안자바, 2, 2, 60, 100, 80, 240, 80.0



**[11-8]** 문제11-7의 Student클래스에 총점(total)과 전교등수(schoolRank)를 저장하기 위한 인스턴스변수를 추가하였다. Student클래스의 기본정렬을 이름(name)이 아닌 총점(total)을 기준으로 한 내림차순으로 변경한 다음, 총점을 기준으로 각 학생의 전교등수를 계산하고 전교등수를 기준으로 오름차순 정렬하여 출력하시오.

**[연습문제]/ch11/Exercise11\_8.java**

```
import java.util.*;

class Student implements Comparable {
    String name;
    int    ban;
    int    no;
    int    kor;
    int    eng;
    int    math;

    int    total;          // 총점
    int    schoolRank;     // 전교등수

    Student(String name, int ban, int no, int kor, int eng, int math) {
        this.name = name;
        this.ban  = ban;
        this.no   = no;
        this.kor  = kor;
        this.eng  = eng;
        this.math = math;

        total = kor+eng+math;
    }

    int getTotal() {
        return total;
    }

    float getAverage() {
        return (int)((getTotal()/ 3f)*10+0.5)/10f;
    }

    public int compareTo(Object o) {
        /*
            (1) 알맞은 코드를 넣어 완성하시오.
        */
    }

    public String toString() {
        return name
            +", "+ban
            +", "+no
            +", "+kor
            +", "+eng
            +", "+math
            +", "+getTotal()
            +", "+getAverage()
            +", "+schoolRank // 새로추가
    }
}
```

```
if (o instanceof Student) {
    Student tmp = (Student)o;
    return tmp.total - this.total; // 총점 기준(내림차순)으로 정렬한다.
} else {
    return -1;
}
```

```

        ;
    }
} // class Student

class Exercise11_8 {
    public static void calculateSchoolRank(List list) {
        Collections.sort(list); // 먼저 list를 총점기준 내림차순으로 정렬한다.

        int prevRank = -1; // 이전 전교등수
        int prevTotal = -1; // 이전 총점
        int length = list.size();

        // 1. 반복문을 이용해서 list에 저장된 Student객체를 하나씩 읽는다.
        for (int i=0; i < length; i++) {
            Student s = (Student)list.get(i);

            // 1.1 총점(total)이 이전총점(prevTotal)과 같으면 이전 등수(prevRank)를 등수(schoolRank)로 한다.
            if (s.total==prevTotal) {
                s.schoolRank = prevRank;
            } else {
                // 1.2 총점이 서로 다르면, 등수(schoolRank)의 값을 알맞게 계산해서 저장한다.
                // 이전에 동점자였다면, 그 다음 등수는 동점자의 수를 고려해야한다.
                s.schoolRank = i + 1;

                // 1.3 현재 총점과 등수를 이전총점(prevTotal)과 이전등수(prevRank)에 저장한다.
                prevRank = s.schoolRank;
                prevTotal = s.total;
            }
        }

        public static void main(String[] args) {
            ArrayList list = new ArrayList();
            list.add(new Student("이자바", 2, 1, 70, 90, 70));
            list.add(new Student("안자바", 2, 2, 60, 100, 80));
            list.add(new Student("홍길동", 1, 3, 100, 100, 100));
            list.add(new Student("남궁성", 1, 1, 90, 70, 80));
            list.add(new Student("김자바", 1, 2, 80, 80, 90));

            calculateSchoolRank(list);

            Iterator it = list.iterator();

            while(it.hasNext())
                System.out.println(it.next());
        }
    }
}

```

#### [실행결과]

```

홍길동,1,3,100,100,100,300,100.0,1
김자바,1,2,80,80,90,250,83.3,2
안자바,2,2,60,100,80,240,80.0,3
남궁성,1,1,90,70,80,240,80.0,3
이자바,2,1,70,90,70,230,76.7,5

```

**[11-9]** 문제11-8의 Student클래스에 반등수(classRank)를 저장하기 위한 인스턴스변수를 추가하였다. 반등수를 계산하고 반과 반등수로 오름차순 정렬하여 결과를 출력하시오. (1)~(2)에 알맞은 코드를 넣어 완성하시오.

**[연습문제]/ch11/Exercise11\_9.java**

```
import java.util.*;

class Student implements Comparable {
    String name;
    int    ban;
    int    no;
    int    kor;
    int    eng;
    int    math;

    int    total;
    int    schoolRank;    // 전교등수
    int    classRank;    // 반등수

    Student(String name, int ban, int no, int kor, int eng, int math) {
        this.name = name;
        this.ban  = ban;
        this.no   = no;
        this.kor  = kor;
        this.eng  = eng;
        this.math = math;

        total = kor+eng+math;
    }

    int getTotal() {
        return total;
    }

    float getAverage() {
        return (int)((getTotal()/ 3f)*10+0.5)/10f;
    }

    public int compareTo(Object o) {
        if(o instanceof Student) {
            Student tmp = (Student)o;

            return tmp.total - this.total;
        } else {
            return -1;
        }
    }

    public String toString() {
        return name
            +","+"ban
            +","+"no
            +","+"kor
            +","+"eng
            +","+"math
    }
}
```

```

        +", "+getTotal()
        +", "+getAverage()
        +", "+schoolRank
        +", "+classRank    // 새로추가
    };
}
} // class Student

class ClassTotalComparator implements Comparator {
    public int compare(Object o1, Object o2) {
        /*
            (1) 알맞은 코드를 넣어 완성하십시오.
        */
    }
}

class Exercisel1_9 {
    public static void calculateClassRank(List list) {
        // 먼저 반별 총점기준 내림차순으로 정렬한다.
        Collections.sort(list, new ClassTotalComparator());

        int prevBan = -1;
        int prevRank = -1;
        int prevTotal = -1;
        int length = list.size();

        for(int i=0, n=0; i < length; i++, n++) {
            // 1. 반복문을 이용해서 list에 저장된 Student객체를 하나씩 읽는다.
            Student s = (Student)list.get(i);

            // 1.1 반이 달라지면, (ban과 prevBan이 다르면) 이전 등수(prevRank)와 이전 총점(prevTotal)을 초기화 한다.
            if(s.ban!=prevBan) { prevRank = -1;
                                prevTotal = -1;
                                n=0;
            }

            // 1.2 총점(total)이 이전총점(prevTotal)과 같으면 이전 등수(prevRank)를 등수(classRank)로 한다.
            if (s.total==prevTotal) {
                s.classRank = prevRank;
            } else {
                // 1.3 총점이 서로 다르면, 등수(classRank)의 값을 알맞게 계산해서 저장한다.
                // 이전에 등점자였다면, 그 다음 등수는 등점자의 수를 고려해야한다.
                s.classRank = n + 1;
            }

            // 1.4 현재 반과 총점과 등수를 이전 반(prevBan), 이전 총점(prevTotal), 이전 등수(prevRank)에 저장한다.
            prevBan = s.ban;
            prevRank = s.classRank;
            prevTotal = s.total;
        }
    }

    // public static void calculateClassRank(List list) {

    public static void calculateSchoolRank(List list) {
        /* 내용 생략 */
    }

    public static void main(String[] args) {
        ArrayList list = new ArrayList();
        list.add(new Student("이자바", 2, 1, 70, 90, 70));
        list.add(new Student("안자바", 2, 2, 60, 100, 80));
        list.add(new Student("홍길동", 1, 3, 100, 100, 100));
        list.add(new Student("남궁성", 1, 1, 90, 70, 80));
        list.add(new Student("김자바", 1, 2, 80, 80, 90));
    }
}

```

```

        calculateSchoolRank(list);
        calculateClassRank(list);

        Iterator it = list.iterator();

        while(it.hasNext())
            System.out.println(it.next());
    }
}

```

#### [실행결과]

```

홍길동,1,3,100,100,100,300,100.0,1,1
김자바,1,2,80,80,90,250,83.3,2,2
남궁성,1,1,90,70,80,240,80.0,3,3
안자바,2,2,60,100,80,240,80.0,3,1
이자바,2,1,70,90,70,230,76.7,5,2

```

**[11-10]** 다음 예제의 빙고판은 1~30사이의 숫자들로 만든 것인데, 숫자들의 위치가 잘 섞이지 않는다는 문제가 있다. 이러한 문제가 발생하는 이유와 이 문제를 개선하기 위한 방법을 설명하고, 이를 개선한 새로운 코드를 작성하시오.

#### [연습문제]/ch11/Exercisell\_10.java

```

import java.util.*;

class Exercisell_10 {
    public static void main(String[] args) {
        Set set = new HashSet();
        int[][] board = new int[5][5];

        for(int i=0; set.size() < 25; i++) {
            set.add((int) (Math.random()*30)+1+"");
        } ArrayList list = new ArrayList(set);
        Collections.shuffle(list);
        Iterator it = set.iterator(); list.iterator();

        for(int i=0; i < board.length; i++) {
            for(int j=0; j < board[i].length; j++) {
                board[i][j] = Integer.parseInt((String)it.next());
                System.out.print((board[i][j] < 10 ? " " : " ")
                                + board[i][j]);

                }
            System.out.println();
        }
    } // main
}

```

중복된 값을 허용하지 않는 HashSet의 특성을 이용해 서로 다른 임의 값을 저장한다.  
 하지만 해싱알고리즘의 특성상 같은 값은 같은 자리에 저장되기 때문에 빙고판의 숫자들이 잘 섞이지 않는 문제 발생.

저장 순서를 유지하는 ArrayList에 HashSet의 데이터를 옮겨담아, Collections.shuffle()을 이용해서 데이터의 순서를 섞는다.

**[11-11]** 다음은 SutdaCard 클래스를 HashSet에 저장하고 출력하는 예제이다. HashSet에 중복된 카드가 저장되지 않도록 SutdaCard의 hashCode()를 알맞게 오버라이딩하시오.

**[Hint]** String 클래스의 hashCode()를 사용하라.

**[연습문제]/ch11/Exercisel1\_11.java**

```
import java.util.*;

class SutdaCard {
    int num;
    boolean isKwang;

    SutdaCard() {
        this(1, true);
    }

    SutdaCard(int num, boolean isKwang) {
        this.num = num;
        this.isKwang = isKwang;
    }

    public boolean equals(Object obj) {
        if(obj instanceof SutdaCard) {
            SutdaCard c = (SutdaCard)obj;
            return num==c.num && isKwang==c.isKwang;
        } else {
            return false;
        }
    }

    public String toString() {
        return num + ( isKwang ? "K":"" );
    }
    public int hashCode() {
        return toString().hashCode(); // String 클래스의 hashCode()를 호출한다.
    }
}

class Exercisel1_11 {
    public static void main(String[] args) {
        SutdaCard c1 = new SutdaCard(3, true);
        SutdaCard c2 = new SutdaCard(3, true);
        SutdaCard c3 = new SutdaCard(1, true);

        HashSet set = new HashSet();
        set.add(c1);
        set.add(c2);
        set.add(c3);

        System.out.println(set);
    }
}
```

**[실행결과]**

[3K, 1K]

**[11-12]** 다음은 섯다게임에서 카드의 순위를 결정하는 등급목록(족보)이다. HashMap에 등급과 점수를 저장하는 registerJokbo()와 게임참가자의 점수를 계산해서 반환하는 getPoint()를 완성하시오.

**[참고]** 섯다게임은 두 장의 카드의 숫자를 더한 값을 10으로 나눈 나머지가 높은 쪽이 이기는 게임이다. 그 외에도 특정 숫자로 구성된 카드로 이루어진 등급(족보)이 있어서 높은 등급의 카드가 이긴다.

카드1	카드2	점수	카드1	카드2	점수
K	K	4000	1	2	2060
10	10	3100	2	1	2060
9	9	3090	1	4	2050
8	8	3080	4	1	2050
7	7	3070	1	9	2040
6	6	3060	9	1	2040
5	5	3050	1	10	2030
4	4	3040	10	1	2030
3	3	3030	4	10	2020
2	2	3020	10	4	2020
1	1	3010	4	6	2010
-	-	-	6	4	2010

**[연습문제]**/ch11/Exercise11\_12.java

```
import java.util.*;

class Exercise11_12 {
    public static void main(String args[]) throws Exception {
        SutdaDeck deck = new SutdaDeck();

        deck.shuffle();
        Player p1 = new Player("타짜", deck.pick(), deck.pick());
        Player p2 = new Player("고수", deck.pick(), deck.pick());

        System.out.println(p1+" "+deck.getPoint(p1));
        System.out.println(p2+" "+deck.getPoint(p2));
    }
}

class SutdaDeck
{
    final int CARD_NUM = 20;
    SutdaCard[] cards = new SutdaCard[CARD_NUM];

    int pos = 0; // 다음에 가져올 카드의 위치
    HashMap jokbo = new HashMap(); // 족보를 저장할 HashMap

    SutdaDeck() {
        for(int i=0;i < cards.length;i++) {
            int num = i%10+1;
            boolean isKwang = i < 10 && (num==1 || num==3 || num==8);

            cards[i] = new SutdaCard(num,isKwang);
        }
    }
}
```

```

        registerJokbo(); // 족보를 등록한다.
    }

    void registerJokbo() {
        /*
            (1) 아래의 로직에 맞게 코드를 작성하시오.
            1. jokbo(HashMap)에 족보를 저장한다.
                두 카드의 값을 문자열로 붙여서 key로, 점수를 value로 저장한다.
            */
    }

    int getPoint(Player p) {
        if(p==null) return 0;

        SutdaCard c1 = p.c1;
        SutdaCard c2 = p.c2;

        Integer result = 0;

        // 1. 카드 두 장이 모두 같으면, jokbo에서 카를 "KK"로 해서 점수를 조회한다.
        if (c1.isKwang && c2.isKwang) {
            result = (Integer)jokbo.get("KK");
        } else {
            // 2. 두 카드의 숫자(num)로 jokbo에서 등급을 조회한다.
            result = (Integer)jokbo.get(""+c1.num+c2.num);

            // 3. 해당하는 등급이 없으면, 아래의 공식으로 점수를 계산한다. (c1.num + c2.num) % 10 + 1000
            if (result == null) {
                result = new Integer((c1.num + c2.num) % 10 + 1000);
            }

            // 4. Player의 점수(point)에 계산한 값을 저장한다.
            p.point = result.intValue();
        }
        return result.intValue();
    }

    SutdaCard pick() throws Exception {
        SutdaCard c = null;

        if(0 <= pos && pos < CARD_NUM) {
            c = cards[pos];
            cards[pos++] = null;
        } else {
            throw new Exception("남아있는 카드가 없습니다.");
        }

        return c;
    }

    void shuffle() {
        for(int x=0; x < CARD_NUM * 2; x++) {
            int i = (int) (Math.random() * CARD_NUM);
            int j = (int) (Math.random() * CARD_NUM);

            SutdaCard tmp = cards[i];
            cards[i] = cards[j];
            cards[j] = tmp;
        }
    }
} // SutdaDeck

```

```

jokbo.put("KK", 4000);
jokbo.put("1010", 3100);
jokbo.put("99", 3090);
jokbo.put("88", 3080);
jokbo.put("77", 3070);
jokbo.put("66", 3060);
jokbo.put("55", 3050);
jokbo.put("44", 3040);
jokbo.put("33", 3030);
jokbo.put("22", 3020);
jokbo.put("11", 3010);
jokbo.put("12", 2060);
jokbo.put("21", 2060);
jokbo.put("14", 2050);
jokbo.put("41", 2050);
jokbo.put("19", 2040);
jokbo.put("91", 2040);
jokbo.put("110", 2030);
jokbo.put("101", 2030);
jokbo.put("104", 2020);
jokbo.put("410", 2020);
jokbo.put("46", 2010);
jokbo.put("64", 2010);

```

"KK"로 해서 점수를 조회한다.

조회한다.

계산한다.

한다.



```
class Player {
    String name;
    SutdaCard c1;
    SutdaCard c2;

    int point; // 카드의 등급에 따른 점수 - 새로 추가

    Player(String name, SutdaCard c1, SutdaCard c2) {
        this.name = name ;
        this.c1 = c1 ;
        this.c2 = c2 ;
    }

    public String toString() {
        return "["+name+"]"+ c1.toString() +"," + c2.toString();
    }
} // class Player

class SutdaCard {
    int num;
    boolean isKwang;

    SutdaCard() {
        this(1, true);
    }

    SutdaCard(int num, boolean isKwang) {
        this.num = num;
        this.isKwang = isKwang;
    }

    public String toString() {
        return num + ( isKwang ? "K":"" );
    }
}
```

**[실행결과]**

[타짜] 5, 9 1004

[교수] 1, 1K 3010

**[11-13]** 다음 코드는 문제11-12를 발전시킨 것으로 각 Player들의 점수를 계산하고, 점수가 제일 높은 사람을 출력하는 코드이다. TreeMap의 정렬기준을 점수가 제일 높은 사람부터 내림차순이 되도록 아래의 코드를 완성하시오. 단, 동점자 처리는 하지 않는다.

**[연습문제]/ch11/Exercisel1\_13.java**

```
import java.util.*;

class Exercisel1_13 {
    public static void main(String args[]) throws Exception
    {
        SutdaDeck deck = new SutdaDeck();

        deck.shuffle();

        Player[] pArr = {
            new Player("타짜", deck.pick(), deck.pick()),
            new Player("고수", deck.pick(), deck.pick()),
            new Player("물주", deck.pick(), deck.pick()),
            new Player("종수", deck.pick(), deck.pick()),
            new Player("하수", deck.pick(), deck.pick())
        };

        TreeMap rank = new TreeMap(new Comparator() {
            public int compare(Object o1, Object o2) {
                if(o1 instanceof Player && o2 instanceof Player) {
                    Player p1 = (Player)o1;
                    Player p2 = (Player)o2;
                    return p2.point - p1.point; // 점수기준의 내림차순으로 정렬
                }
                return -1;
            }
        });

        for(int i=0; i < pArr.length; i++) {
            Player p = pArr[i];
            rank.put(p, deck.getPoint(p));
            System.out.println(p+" "+deck.getPoint(p));
        }

        System.out.println();
        System.out.println("1위는 "+rank.firstKey()+"입니다.");
    }
}

class SutdaDeck
{
    final int CARD_NUM = 20;
    SutdaCard[] cards = new SutdaCard[CARD_NUM];

    int pos = 0; // 다음에 가져올 카드의 위치
    HashMap jokbo = new HashMap(); // 족보를 저장할 HashMap

    SutdaDeck() {
        for(int i=0; i < cards.length; i++) {
            int num = i%10+1;
            boolean isKwang = i < 10 && (num==1 || num==3 || num==8);

            cards[i] = new SutdaCard(num, isKwang);
        }
    }
}
```

```

        registerJokbo(); // 족보를 등록한다.
    }

    void registerJokbo() {
        jokbo.put("KK", 4000);

        jokbo.put("1010", 3100); jokbo.put("12", 2060);
        jokbo.put("99", 3090);   jokbo.put("21", 2060);
        jokbo.put("88", 3080);   jokbo.put("14", 2050);
        jokbo.put("77", 3070);   jokbo.put("41", 2050);
        jokbo.put("66", 3060);   jokbo.put("19", 2040);
        jokbo.put("55", 3050);   jokbo.put("91", 2040);
        jokbo.put("44", 3040);   jokbo.put("110", 2030);
        jokbo.put("33", 3030);   jokbo.put("101", 2030);
        jokbo.put("22", 3020);   jokbo.put("104", 2020);
        jokbo.put("11", 3010);   jokbo.put("410", 2020);
                                   jokbo.put("46", 2010);
                                   jokbo.put("64", 2010);
    }

    int getPoint(Player p) {
        if(p==null) return 0;

        SutdaCard c1 = p.c1;
        SutdaCard c2 = p.c2;

        Integer result = 0;

        if(c1.isKwang && c2.isKwang) {
            result = (Integer)jokbo.get("KK");
        } else {
            result = (Integer)jokbo.get(""+c1.num+c2.num);

            if(result==null) {
                result = new Integer((c1.num + c2.num) % 10 + 1000);
            }
        }

        p.point = result.intValue();

        return result.intValue();
    }

    SutdaCard pick() throws Exception {
        SutdaCard c = null;

        if(0 <= pos && pos < CARD_NUM) {
            c = cards[pos];
            cards[pos++] = null;
        } else {
            throw new Exception("남아있는 카드가 없습니다.");
        }

        return c;
    }
}

```

```
void shuffle() {
    for(int x=0; x < CARD_NUM * 2; x++) {
        int i = (int) (Math.random() * CARD_NUM);
        int j = (int) (Math.random() * CARD_NUM);

        SutdaCard tmp = cards[i];
        cards[i] = cards[j];
        cards[j] = tmp;
    }
}

} // SutdaDeck

class Player {
    String name;
    SutdaCard c1;
    SutdaCard c2;

    int point;

    Player(String name, SutdaCard c1, SutdaCard c2) {
        this.name = name ;
        this.c1 = c1 ;
        this.c2 = c2 ;
    }

    public String toString() {
        return "["+name+"]"+ c1.toString() +"," + c2.toString();
    }
} // class Player

class SutdaCard {
    int num;
    boolean isKwang;

    SutdaCard() {
        this(1, true);
    }

    SutdaCard(int num, boolean isKwang) {
        this.num = num;
        this.isKwang = isKwang;
    }

    public String toString() {
        return num + ( isKwang ? "K":"" );
    }
}
```

**[실행결과]**

[타짜] 7, 2 1009

[고수] 2, 5 1007

[물주] 1, 7 1008

[중수] 10, 4 2020

[하수] 9, 6 1005

1위는 [중수] 10, 4입니다.

**[11-14]** 다음은 성적처리 프로그램의 일부이다. Scanner 클래스를 이용해서 화면으로부터 데이터를 입력하고 보여주는 기능을 완성하시오.

**[연습문제]**/ch11/Exercise11\_14.java

```
import java.io.*;
import java.util.*;

class Exercise11_14
{
    static ArrayList record = new ArrayList(); // 성적데이터를 저장할 공간
    static Scanner s = new Scanner(System.in);

    public static void main(String args[]) {
        while(true) {
            switch(displayMenu()) {
                case 1 :
                    inputRecord();
                    break;
                case 2 :
                    displayRecord();
                    break;
                case 3 :
                    System.out.println("프로그램을 종료합니다.");
                    System.exit(0);
            }
        } // while(true)
    }

    // menu를 보여주는 메서드
    static int displayMenu() {
        System.out.println("*****");
        System.out.println("*          성적 관리 프로그램          *");
        System.out.println("*****");
        System.out.println();
        System.out.println(" 1. 학생성적 입력하기 ");
        System.out.println();
        System.out.println(" 2. 학생성적 보기");
        System.out.println();
        System.out.println(" 3. 프로그램 종료 ");
        System.out.println();
        System.out.print("원하는 메뉴를 선택하세요. (1~3) : ");

        int menu = 0;

        /*
        (1) 아래의 로직에 맞게 코드를 작성하시오
        1. 화면으로부터 1~3사이의 정수를 입력받는다
        2. 1~3사이의 정수가 아니면 다시 입력받는다
        */

        return menu;
    } // public static int displayMenu() {

    // 데이터를 입력받는 메서드

```

```
do {
    try {
        menu = Integer.parseInt(s.nextLine().trim());
        if (1 <= menu && menu <= 3) {
            break;
        } else {
            throw new Exception();
        }
    } catch (Exception e) {
        System.out.println("메뉴를 잘못 선택하셨습니다. 다시 입력해주세요.");
        System.out.print("원하는 메뉴를 선택하세요. (1~3) : ");
    }
} while (true);
```

```

static void inputRecord() {
    System.out.println("1. 학생성적 입력하기");
    System.out.println("이름,반,번호,국어성적,영어성적,수학성적'의 순서로 공백없이 입력하세요.");
    System.out.println("입력을 마치려면 q를 입력하세요. 메인화면으로 돌아갑니다.");

    while(true) {
        System.out.print(">>");

        /*
        (2) 아래의 로직에 맞게 코드를 작성
        1. Scanner를 이용해서 화면으로 입력받는다.
        2. 입력받은 값이 q 또는 Q이면 메서드를 종료한다.
           그렇지 않으면 입력받은 값으로 s2에 저장한다.
        3. 입력받은 데이터에서 예외가 발생하면 다시 입력받는다.
        4. q 또는 Q가 입력될 때까지 반복한다.
        */
    } // end of while
} // public static void inputRecord() {

// 데이터 목록을 보여주는 메서드
static void displayRecord() {
    int koreanTotal = 0;
    int englishTotal = 0;
    int mathTotal = 0;
    int total = 0;

    int length = record.size();

    if(length > 0) {
        System.out.println();
        System.out.println("이름 반 번호 국어 영어 수학 총점 평균 전교등수 반등수");

        System.out.println("=====");

        for (int i = 0; i < length ; i++) {
            Student student = (Student)record.get(i);
            System.out.println(student);
            koreanTotal += student.kor;
            mathTotal += student.math;
            englishTotal += student.eng;
            total += student.total;
        }

        System.out.println("=====");
        System.out.println("총점: "+koreanTotal+" "+englishTotal
            +" "+mathTotal+" "+total);

        System.out.println();
    } else {
        System.out.println("=====");
        System.out.println(" 데이터가 없습니다.");
        System.out.println("=====");
    }
} // static void displayRecord() {
}

```

```

try {
    String input = s.nextLine().trim();
    if (!input.equalsIgnoreCase("q")) {
        // Scanner를 이용해서 화면으로 부터 데이터를 입력받는다. (','를 구분자로)
        Scanner s2 = new Scanner(input).useDelimiter(",");

        // 입력받은 값으로 Student인스턴스를 생성하고 record에 추가한다.
        record.add(new Student(s2.next(), s2.nextInt(), s2.nextInt(),
            s2.nextInt(), s2.nextInt(), s2.nextInt()));
        System.out.println("잘입력되었습니다. 입력을 마치려면 q를 입력하세요.");
    } else {
        // 입력받은 값이 q 또는 Q이면 메서드를 종료한다.
        return;
    }
} catch (Exception e) {
    // 입력받은 데이터에서 예외가 발생하면, "입력오류입니다."를 보여주고 다시 입력받는다.
    System.out.println("입력오류입니다. 이름, 반, 번호, 국어성적, 영어성적, 수학성적'의 순서로 입력하세요.");
}

```

```
class Student implements Comparable {
    String name;
    int    ban;
    int    no;
    int    kor;
    int    eng;
    int    math;

    int    total;
    int    schoolRank;
    int    classRank; // 반등수

    Student(String name, int ban, int no, int kor, int eng, int math) {
        this.name = name;
        this.ban = ban;
        this.no = no;
        this.kor = kor;
        this.eng = eng;
        this.math = math;

        total = kor+eng+math;
    }

    int getTotal() {
        return total;
    }

    float getAverage() {
        return (int)((getTotal()/ 3f)*10+0.5)/10f;
    }

    public int compareTo(Object o) {
        if(o instanceof Student) {
            Student tmp = (Student)o;

            return tmp.total - this.total;
        } else {
            return -1;
        }
    }

    public String toString() {
        return name
            +", "+ban
            +", "+no
            +", "+kor
            +", "+eng
            +", "+math
            +", "+getTotal()
            +", "+getAverage()
            +", "+schoolRank
            +", "+classRank
            ;
    }
} // class Student
```

**[실행결과]**

```

*****
*                               *
*          성적 관리 프로그램          *
*                               *
*****

1. 학생성적 입력하기

2. 학생성적 보기

3. 프로그램 종료

원하는 메뉴를 선택하세요. (1~3) : 5
메뉴를 잘못 선택하셨습니다. 다시 입력해주세요.
원하는 메뉴를 선택하세요. (1~3) : 2
=====
데이터가 없습니다.
=====
*****
*                               *
*          성적 관리 프로그램          *
*                               *
*****

1. 학생성적 입력하기

2. 학생성적 보기

3. 프로그램 종료

원하는 메뉴를 선택하세요. (1~3) : 1
1. 학생성적 입력하기
이름, 반, 번호, 국어성적, 영어성적, 수학성적'의 순서로 공백없이 입력하세요.
입력을 마치려면 q를 입력하세요. 메인화면으로 돌아갑니다.
>>
입력오류입니다. 이름, 반, 번호, 국어성적, 영어성적, 수학성적'의 순서로 입력하세요.
>>자바짱,1,1,100,100,100
잘입력되었습니다. 입력을 마치려면 q를 입력하세요.
>>김자바,1,2,80,80,80
잘입력되었습니다. 입력을 마치려면 q를 입력하세요.
>>q
*****
*                               *
*          성적 관리 프로그램          *
*                               *
*****

1. 학생성적 입력하기

2. 학생성적 보기

3. 프로그램 종료

원하는 메뉴를 선택하세요. (1~3) : 2

이름 반 번호 국어 영어 수학 총점 평균 전교등수 반등수
=====
자바짱,1,1,100,100,100,300,100.0,0,0
김자바,1,2,80,80,80,240,80.0,0,0
=====

```



총점: 180 180 180 540

```
*****
*                               *
*          성적 관리 프로그램          *
*****
```

1. 학생성적 입력하기
2. 학생성적 보기
3. 프로그램 종료

원하는 메뉴를 선택하세요. (1~3) : 3  
프로그램을 종료합니다.