Tutorial of Sort and Map

> Designed by ZHU Yueming in december 14th. 2021

learning target

- Learn how to sort a collection
- Learn hash map

Part 1. Sort

Design a class named `Student` below, and add its three parameter constructor, getter and setter method of all private fields, and toString() method.

```java
public class Student{
    private String group;
    private String name;
    private int grade;
     //todo: add constructor, getter and setter method, toString() method
}
```

Then create a class named `TestSort`, which contains a `List` that contains several students:

```java
public class TestSort {
    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();
        students.add(new Student("01","ZhangSan",90));
        students.add(new Student("01","LiMing",95));
        students.add(new Student("03","XiaoLan",89));
        students.add(new Student("02","Wong",99));
        students.add(new Student("02","LiSi",80));
    }
}
```

Question1: How to sort students by name?

Solution1: Make Student Class Comparable

Change in Student class

```java
public class Student implements Comparable<Student> {
    //......
    @Override
    public int compareTo(Student o) {
        return this.name.compareTo(o.name);
    }
}
```

Change in TestSort class

```
Collections.sort(students);
for (Student s:students) {
    System.out.println(s);
}
```

Result:

```
Student{group='01', name='LiMing', grade=95}
Student{group='02', name='LiSi', grade=80}
Student{group='02', name='Wong', grade=99}
Student{group='03', name='XiaoLan', grade=89}
Student{group='01', name='ZhangSan', grade=90}
```

Solution 2: Create a `Comparator` Class for sorted by name

Add Comparator Class

```
class NameComparator implements Comparator<Student> {
    @Override
    public int compare(Student o1, Student o2) {
        return o1.getName().compareTo(o2.getName());
    }
}
```

Change in TestSort class

```
students.sort(new NameComparator());
for (Student s:students) {
    System.out.println(s);
}
```

Result:

```
Student{group='01', name='LiMing', grade=95}
Student{group='02', name='LiSi', grade=80}
Student{group='02', name='Wong', grade=99}
Student{group='03', name='XiaoLan', grade=89}
Student{group='01', name='ZhangSan', grade=90}
```

Question2: If we have two requirements, the one is sorted by name while the other is sorted by grade? How to design?

If we make `Student` class to be a Comparable, it can only be sorted by one field, so that in this case, using Comparator would be more flexible.

Create another class named `GradeComparator`

```
class GradeComparator implements Comparator<Student> {

    @Override
    public int compare(Student o1, Student o2) {
        return o1.getGrade() - o2.getGrade();
    }
}
```

Change in TestSort class

```
students.sort(new NameComparator());
for (Student s : students) {
    System.out.println(s);
}

students.sort(new GradeComparator());
for (Student s : students) {
    System.out.println(s);
}
```

Result:

```
Student{group='01', name='LiMing', grade=95}
Student{group='02', name='LiSi', grade=80}
Student{group='02', name='Wong', grade=99}
Student{group='03', name='XiaoLan', grade=89}
Student{group='01', name='ZhangSan', grade=90}
Student{group='02', name='LiSi', grade=80}
Student{group='03', name='XiaoLan', grade=89}
Student{group='01', name='ZhangSan', grade=90}
Student{group='01', name='LiMing', grade=95}
Student{group='02', name='Wong', grade=99}
```

Part 2. HashMap

HashMap is a data structure based on `key->value`

Example1: `<Integer, Class>`

Suppose there is a class named `IPhone`, which contains two private fields: name and price.

```
public class IPhone {
    private String name;
    private int price;

    public IPhone(String name, int price) {
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString() {
        return "IPhone{" +
                "name='" + name + '\'' +
                ", price=" + price +
                '}';
    }
}
```

Then different model of IPhone has its different name and price as follows:

model	Name	price
1	iPhone 13 Pro	7999
2	iPhone 13 Pro Max	8999
3	iPhone 13 mini	5199
4	iPhone 13	5999

Now we would to build a map structure that we can get all information about IPhone as long as we know the model of corresponding IPhone.

How to build a Map?

```
public static void main(String[] args) {
        Map<Integer, IPhone> iPhoneMap = new HashMap<>();
        iPhoneMap.put(1,new IPhone("iPhone 13 Pro",7999));
        iPhoneMap.put(2,new IPhone("iPhone 13 Pro Max",8999));
        iPhoneMap.put(3,new IPhone("iPhone 13 mini",5199));
        iPhoneMap.put(4,new IPhone("iPhone 13",5999));

        System.out.println(iPhoneMap.get(1));
    }
```

How to traverse a Map?

```
for (Map.Entry<Integer,IPhone> iPhoneEntry:iPhoneMap.entrySet()) {
    System.out.printf("[%d ->
%s]\n",iPhoneEntry.getKey(),iPhoneEntry.getValue());
}
```

Result:

```
[1 -> IPhone{name='iPhone 13 Pro', price=7999}]
[2 -> IPhone{name='iPhone 13 Pro Max', price=8999}]
[3 -> IPhone{name='iPhone 13 mini', price=5199}]
[4 -> IPhone{name='iPhone 13', price=5999}]
```

Example 2: `<String, List<Class>>`

According to the map relationship about `group` -> `Student`, restore `Student` class into a HashMap structure.

How to build a Map according to the given `List<Student>`?

```
    Map<String, List<Student>> studentMap = new HashMap<>();
         for (Student stu : students) {
             String group = stu.getGroup();
             if (!studentMap.containsKey(group)) {
                 List<Student> studentList = new ArrayList<>();
                 studentMap.put(group, studentList);
             }
             studentMap.get(group).add(stu);
         }
```

How to traverse the map, and print as the format below?

```
[group1]:name1,name2....
[group2]:name3,name4....
.....
```

Source code:

```
         StringBuilder sb = new StringBuilder();
         for (Map.Entry<String, List<Student>> map : studentMap.entrySet()) {
             sb.append(map.getKey()).append(":");
             for (Student s : map.getValue()) {
                 sb.append(s.getName()).append(",");
             }
             sb.setLength(sb.length() - 1);
             sb.append(System.lineSeparator());
         }
         System.out.println(sb);
```

Result:

```
01:ZhangSan,LiMing
02:LiSi,Wong
03:XiaoLan
```