**Grouping And Sorting Of Objects**

Grouping and sorting of objects is a very general problem and no matter whichever language or platform you are using, you are going to face it sooner or later. This article is an approach to provide an efficient solution for the same.

**Grouping of objects:**

Take a scenario where you have class like this:



And there is an array of objects of this class like:

Data[] dataObjects;

Now suppose we want to create groups in this array of objects on the basis of the field Description which means the objects with the same description field should be in one group. The typical approach would be to write the business logic involving complex manipulations. The other smarter approach would be to use "MultiMaps". Now let's solve this problem using multimaps. The language we will use is Java and in case you are working on some different language then you can google how to use multimaps specific to your platform. So in java to use multimaps we will use library named "Guava" provided by Google to work in a faster and efficient manner on collections. Just download the jar from this link <http://code.google.com/p/guava-libraries/> and include it in your project.

Now before going directly to the final answer it will be better to be aware of the concept of multimaps which will make it much easier to grasp the solution. So multimaps is like a key-value pair concept where each key is mapped to some value. In addition to this key-value pair mapping it can also contain duplicate keys which a simple map cannot. Thus multimap groups values on the basis of these duplicate keys. So to understand this let's take a simple example.

**Key Value**

**Orange Fruit**

**Apple Vegetable**

**Brinjal**

**Map with no duplicate keys**

The above example is about key-value pairs with no duplicate keys. This example demonstrates the concept of a simple map with **no duplicate keys allowed**.

**Key Value**

**Orange Fruit**

**Apple Vegetable**

**Brinjal Red**

**Apple**

**Map with duplicate keys**

The above example is about key-value pairs with duplicate keys. This example demonstrates the concept of how the **duplicate keys are allowed in a multimap**.

Now the Guava multimap can thus store a map with duplicate keys **and it groups the values having the same key.** **This concept of grouping of values based on keys is actually is the solution of this problem.** Here is the diagram demonstrating the grouping of values based on the duplicate keys.

**Key Value**

**Apple Fruit**

**Orange Vegetable**

**Brinjal Red**

**Grouping of values based upon duplicate keys**

**Note: Please notice in the above diagram how the values with the same key "apple" are grouped together**

Now let's use this concept to solve our above problem where we have to group the objects which have the same description field. So to solve it first step would be to create a multimap with the value of description field of object as a key and this corresponding object as a value. As a result objects with the same description will be grouped together in one group. Ok now let us try it practically.

Here is the code to create a multimap:

****

In the above code line a multimap provided by guava library is initialized whose key is of type string and value is of type Data class. Next step is to fill this multimap from the array of objects of data class. Here is the code for it:



As a result a map will be created with key as the value of the description field of an object and the value as this corresponding object for this description field. Thus all the objects with the same description field will be grouped together.

Now in order to access these groups we can simply use the value of the field description as a key to get the corresponding group. Here is the code for it:



**Sorting of Objects:**

Another general problem is of sorting of objects. Again the typical approach to solve this problem will involve algorithms doing complex manipulations. But to solve this problem we can use "Comparators" and "Collections" to sort objects in an efficient and faster manner. If you are working on a different technology please google how to use comparators and collections to sort objects specific to your technology. Ok so let's look quickly on how we can use comparators and collections to sort objects. Only the basic code is provided here for introduction of sorting of objects using comparators and collections and the concept of comparators and collections is not explained here. The language we will use is Java.

So suppose we have to sort the objects of class Data described previously in an ascending order according to the value of the field Id. For this we have to define a comparator which compares two objects based upon the value of the field Id. Thereafter we will use this comparator to sort this array of objects using sort method of class Collections provided by java. Here is the code for it:

Here only the basic code is provided to introduce the concept of sorting using comparators and collections. Please read from some good resources about comparators and collections for their conceptual understanding. Thereafter you can apply them in your code to sort objects according to the need.