Lab Using the Java Collections Framework

The Java Collections framework allows you to make use of pre-existing collections (data structures) such as sets, lists, maps, queues etc AND to make use of common algorithms such as sorting and searching. The purpose is to maximise **code reuse through providing these capabilities in the API** so that developers don't have to recreate this type of manipulation functionality all the time.

Part 1 Creating and using a Collections - 0.5

Create a class in java called LabHashSet, that has the following:

1. A constructor that creates a HashSet (HashSet is a type of Set) and pre-populates the hashset with counties in Ireland. Just set the counties up as an String array attribute in the class):

"Antrim", "Armagh", "Carlow", "Cavan", "Clare", "Cork", "Derry", "Donegal", "Down", "Dublin", "Fermanagh", "Galway", "Kerry", "Kildare", "Kilkenny", "Laois", "Leitrim", "Limerick", "Longford", "Louth", "Mayo", "Meath", "Monaghan", "Offaly", "Roscommon", "Sligo", "Tipperary", "Tyrone", "Waterford", "Westmeath", "Wexford", "Wicklow" }

2. A printSet() method that prints out the hashset:

Get an **iterator** object to iterate through the set,and print out the contents **on individual lines (look in the inter**.

What order do the counties (elements) appear in? Why?

3. Now, a second method addEntry (String country) that takes in a country (i.e. a String) and adds it to the hashset

4. From a separate control class main method, instantiate a LabHashSet object. Call the methods to print the content and add a new entry "Hibernia".

5. Call the method to print out the contents again. Where does "Hibernia" appear in the elements? Why is it not the last element if it was added last?

6. Add a method matchEntry (String country) to LabHashSet that takes in a country (String) and checks whether it exists in the HashSet – return true or false.

7. From your main method control class, check if the HashSet contains a specific value: "London". Then check if it contains "Louth".

8. Add a method clearSet () to clear out the contents of the HashSet – and check that it works.

Part 2 HashSets – 0.5

Hashsets are unordered collections of elements (as just demonstrated in part 1).

Add a sortSet() method to your LabHashSet class that will sort the hashset.

The Collections class in the java.util package of the Java API contains many common methods for data structures such as shuffling and sorting. Try to "sort" the hashset by passing it to the sort method of the Collections class (() - i.e. Collections.sort(yourhashset)

What happens?

In your sortSet method, instead create an arrayList of your hashset contents (this takes just one line of code!). ArrayLists are "sortable". Use the Collections class sort method to sort this array list. Print out the contents of the arrayList with one element per line, and check that it works.

Now reverse the elements of the HashSet – and verify by printing out as before.

Now shuffle the elements of the HashSet – and verify by printing out as before

Part 2 Beyond strings – full marks

Create a simple person class – attributes first name, last name, date of birth, all strings.

In a separate control class, create a new HashSet that will store Person objects.

Instantiate 5 Person objects. Give two of them identical attribute values : e.g. john , smith, 23/01/2000 .

(1) Check if you can add these two "identical" objects to the Hashset or not.
(2) Call the Collections.sort() Method on the hashset and see if it will work – make sure to understand the answer.