**Q1.** Suppose you have a set s containing some strings. Write a code fragment that uses an iterator to print out each element in s.

**Q2.** Again suppose you have a set s containing some strings. Write a code fragment to compute the result of concatenating all of these strings together. (Recall that + is the string concatenation operator.)

**Q3.** Write Java code to define yet another Set s. Insert 3 floating point numbers in s, and using an iterator, find the sum of the numbers in s.

**Q4.** Create your own **HashSet** object with initial capacity of 5. Add the following objects to the newly created **HashSet** object:

* + 2 String objects.
  + 2 MyOwnClass object. (You will have to create MyOwnClass.java first)
  + 3 Integer objects.
  + Display the HashSet object.

**Q5.** Create your own **HashSet** object and Add the following objects with properly initialized values to the newly created **HashSet** object.

* + 2 String objects.
  + 2 MyOwnClass object (You will have to create MyOwnClass.java first.)
    - MyOwnClass class has name (String type) and age (int type).
  + 3 Integer objects .
  + Create **Iterator** object from the **HashSet** object and iterate them to get displayed.
  + Repeat the above for LinkedHashSet, ArrayList.

Collection Framework

**Exercise 1**: Create a method which accepts a hash map and return the values of the map in sorted order as a List.

|  |  |
| --- | --- |
| Method Name | getValues |
| Method Description | Get the values of a map in sorted order |
| Argument | HashMap |
| Return Type | List |
| Logic | Return the values of a hash map in sorted order |

**Exercise2 Collection:** Create a method that accepts a character array and count the number of times each character is present in the array.

|  |  |
| --- | --- |
| Method Name | countChars |
| Method Description | method that accepts a character array and count the number of times each character is present in the array. |
| Argument | char[] arr |
| Return Type | Map<Character, Integer> |
| Logic |  |

**Exercise 3:** Create a method which accepts an array of numbers and returns the numbers and their squares in Hashmap

|  |  |
| --- | --- |
| Method Name | getSquares |
| Method Description | Accepts a list of numbers and return their squares |
| Argument | int[] |
| Return Type | Map |
| Logic | Iterate through the list, find the square of each  number and add the elements to a map object  with the number as the key and the square as the value |

**Exercise 4:**  school offers medals to the students of tenth based on the following criteria

If(Marks>=90) : Gold

If(Marks between 80 and 90) : Silver

If(Marks between 70 and 80) : Bronze

**Note:** Marks between 80 and 90 means marks>=80 and marks<90

Write a function which accepts the marks of students as a Hashmap and return the details of the students eligible for the medals along with type of medal.

The input Hashmap contains the student registration number as key and mark as value.

The output Hashmap should contain the student registration number as key and the medal type as value.

|  |  |
| --- | --- |
| Method Name | getStudents |
| Method Description | Generate the list of students eligible for scholarship |
| Argument | Hashmap |
| Return Type | Hashmap |
| Logic | The method should return the details of the students eligible for the medals along with the medal type. |

**Exercise 5:** Create a method which accepts an array of integer elements and return the second smallest element in the array

|  |  |  |
| --- | --- | --- |
| Method Name |  | getSecondSmallest |
| Method Description |  | Get the second smallest element in the array |
| Argument |  | int[] |
| Return Type |  | int |
| Logic |  | Sort the array and return the second smallest element in the array Hint: 1. Convert to Array List 2. Use sort method in Collections class |

**Exercise 6:** Create a method which accepts the id and the age of people as a Map and decide if they are eligible for vote. A person is eligible for vote if his age is greater than 18. Add the IDs of all the eligible persons to list and return the list.

|  |  |
| --- | --- |
| Method Name | votersList |
| Method Description | Generate the list of voters based on the ages of the people |
| Argument | Map |
| Return Type | List |
| Logic | Accept a map with ID as key and Date of Birth as value and check if the person is eligible to vote. A person is eligible for vote for if his age is greater than 18. If the person is eligible add his ID to the list. |