Guided LAB - 303.11.6 - HashMap and TreeMap Processing

Objectives:

In this lab, we will explore and demonstrate HashMap and TreeMap, and we will utilize the built-in methods.

Learning Objectives:

By the end of this lab, learners will be able to use Hashmap and TreeMap in Java.

HashMap Examples

Example One: Remove HashMap Elements.

To remove elements from a hashmap, we can use the remove() method as shown below.

Create a new Java project and create a new Class named **"ExampleOneHashMap ,"** and then write the code below in the class.

```
import java.util.HashMap;

public class ExampleOneHashMap {
  public static void main(String[] args) {

    HashMap<Integer, String> languages = new HashMap<>();
    languages.put(1, "Java");
    languages.put(2, "Python");
    languages.put(3, "JavaScript");
    languages.put(4, "C Sharp");

    System.out.println("HashMap: " + languages);
```

```
    // remove element associated with key 2
    String value = languages.remove(2);
    System.out.println("Removed value: " + value);
    System.out.println("Updated HashMap: " + languages);
  }
}
```

Output

```
HashMap: {1=Java, 2=Python, 3=JavaScript, 4=C Sharp}
Removed value: Python
Updated HashMap: {1=Java, 3=JavaScript, 4=C Sharp}
```

Here, the remove() method takes the key as its parameter. It then returns the value associated with the key and removes the entry. We can also remove the entry only under certain conditions. For example: **`remove(4, "C Sharp");`**

Here, the remove() method only removes the entry if the key 4 is associated with the value **`"C Sharp."`**

Example Two - Create Hashmap and Remove HashMap Elements

Here, we will do multiple things. We will first create a Hashmap. We will then get its values one by one. After that, we will copy all data of the HashMap to a brand-new HashMap. And finally, we will remove one item and get Hashmap sizes. If the size is lower by one, the decrease of size by removal is confirmed.

Create a new Class named **"ExampletwoHashMap,"** and then write the code below in the class.

```
import java.util.HashMap;
public class ExampletwoHashMap {
   public static void main(String[] args) {
       HashMap<String, String> newHashMap = new HashMap<>();
// Addition of key and value
       newHashMap.put("Key1", "Lenovo");
       newHashMap.put("Key2", "Motorola");
       newHashMap.put("Key3", "Nokia");
       newHashMap.put("Key4", null);
       newHashMap.put(null, "Sony");
       System.out.println("Original map contains:" + newHashMap);
//getting size of Hashmap
       System.out.println("Size of original Map is:" + newHashMap.size());
//copy contains of one Hashmap to another
       HashMap<String, String> copyHashMap = new HashMap<>();
       copyHashMap.putAll(newHashMap);
       System.out.println("copyHashMap mappings= " + copyHashMap);
//Removal of null key
       String nullKeyValue = copyHashMap.remove(null);
       System.out.println("copyHashMap null key value = " + nullKeyValue);
       System.out.println("copyHashMap after removing null key = " + copyHashMap);
       System.out.println("Size of copyHashMap is:" + copyHashMap.size());
   }
}
```

Output:

```
Original map contains:{Key2=Motorola, null=Sony,
Key1=Lenovo, Key4=null, Key3=Nokia}
Size of original Map is:5
copyHashMap mappings= {Key2=Motorola, null=Sony,
Key1=Lenovo, Key4=null, Key3=Nokia}
copyHashMap null key value = Sony
```

```
copyHashMap after removing null key = {Key2=Motorola,
Key1=Lenovo, Key4=null, Key3=Nokia}
Size of copyHashMap is:4
```

TreeMap Examples

Example One: Remove TeeMap Elements.

- **remove(key) -** returns and removes the entry associated with the specified key from a TreeMap.
- **remove(key, value) -** removes the entry from the map only if the specified key is associated with the specified value, and returns a boolean value.

Create a new Class named **"exampleTreemapOne,"** and then write the code below in the class.

```
import java.util.TreeMap;

public class exampleTreemapOne {
    public static void main(String[] args) {

        TreeMap<String, Integer> numbers = new TreeMap<>();
        numbers.put("One", 1);
        numbers.put("Two", 2);
        numbers.put("Three", 3);
        System.out.println("TreeMap: " + numbers);

        // remove method with single parameter
        int value = numbers.remove("Two");
        System.out.println("Removed value: " + value);

        // remove method with two parameters
        boolean result = numbers.remove("Three", 3);
        System.out.println("Is the entry {Three=3} removed? " + result);
        System.out.println("Updated TreeMap: " + numbers);
    }
}
```

Output:

```
TreeMap: {One=1, Three=3, Two=2}
Removed value = 2
Is the entry {Three=3} removed? True
Updated TreeMap: {One=1}
```

Example Two: Methods for Navigation.

TreeMap class also implements NavigableMap; it provides various methods to navigate over the elements of the treemap.

- firstKey() - returns the first key of the map.
- firstEntry() - returns the key/value mapping of the first key of the map.
- lastKey() - returns the last key of the map.
- lastEntry() - returns the key/value mapping of the last key of the map.

Create a new Class named **"exampleTreemapTwo,"** and then write the code below in the class.

```java
import java.util.TreeMap;
public class exampleTreemapTwo {
    public static void main(String[] args) {
        TreeMap<String, Integer> numbers = new TreeMap<>();
        numbers.put("First", 1);
        numbers.put("Second", 2);
        numbers.put("Third", 3);
        System.out.println("TreeMap: " + numbers);

        // Using the firstKey() method
        String firstKey = numbers.firstKey();
        System.out.println("First Key: " + firstKey);
```

```
        // Using the lastKey() method
        String lastKey = numbers.lastKey();
        System.out.println("Last Key: " + lastKey);

        // Using firstEntry() method
        System.out.println("First Entry: " + numbers.firstEntry());

        // Using the lastEntry() method
        System.out.println("Last Entry: " + numbers.lastEntry());
    }
}
```

Output

```
TreeMap: {First=1, Second=2, Third=3}
First Key: First
Last Key: Third
First Entry: First=1
Last Entry: Third=3
```

Example Three: TreeMap Comparator.

TreeMap elements are sorted in ascending order. However, we can also customize the ordering of keys. For this, we need to create our comparator class based on which keys in a TreeMap are sorted.

Create a new Class named **"CustomComparator,"** and then write the below code in the class.

```
import java.util.Comparator;
public  class CustomComparator implements Comparator<String> {

    @Override
    public int compare(String number1, String number2) {
        int value =  number1.compareTo(number2);
```

```
        // elements are sorted in reverse order
        if (value > 0) {
            return -1;
        }
        else if (value < 0) {
            return 1;
        }
        else {
            return 0;
        }
    }
}
```

Create a new Class named **"maincomparator,"** and then write the code below in the class.

```
import java.util.TreeMap;
public class maincomparator {

   public static void main(String[] args) {

        // Creating a treemap with a customized comparator
        TreeMap<String, Integer> numbers = new TreeMap<>(new
CustomComparator());

        numbers.put("First", 1);
        numbers.put("Second", 2);
        numbers.put("Third", 3);
        numbers.put("Fourth", 4);
        System.out.println("TreeMap: " + numbers);
}
}
```

Output

```
TreeMap: {Third=3, Second=2, Fourth=4, First=1}
```

Submission Instructions:

Include the following deliverables in your submission -

- Submit your source code using the Start Assignment button in the top-right corner of the assignment page in Canvas.

CANVAS STAFF USE ONLY: Canvas Submission Guideline:

Instructions for Canvas Assignment Creation
Assignment Name: GLAB - 303.11.6 - HashMap and TreeMap Processing **Points:** 100 **Assignment Group:** Module 303: Java SE Review (Not Graded) **Display Grade As:** Complete/Incomplete **Do not count this assignment towards the final grade:** Checked **Submission Types:** File uploads **Everything else is the default.**