**Guided LAB - 303.11.5 - HashSet and TreeSet Processing**

**Objective**

In this lab, we will explore and demonstrate HashSet and TreeSet. We will utilize the built-in methods.

**Learning Objective**

By the end of this lab, learners will be able to explain and use HashSet and TreeSet.

# 

# HashSet Examples

## Example One: Insert Elements to HashSet using addAll().

**addAll()** - Inserts all of the elements of the specified collection to the set.

Create a new Java project and create a new Class named **“exampleOne.”** Write the code below in the class.

| import java.util.HashSet;  public class exampleOne {  public static void main(String[] args) {  HashSet<Integer> evenNumber = new HashSet<>();  // Using add() method  evenNumber.add(2);  evenNumber.add(4);  evenNumber.add(6);  System.out.println("HashSet: " + evenNumber);  HashSet<Integer> numbers = new HashSet<>();  // Using addAll() method  numbers.addAll(evenNumber);  numbers.add(5);  System.out.println("New HashSet: " + numbers);  }  } |
| --- |

**Output:**

HashSet: [2, 4, 6]

New HashSet: [2, 4, 5, 6]

## Example Two: Union of Sets.

To perform the union between two sets, we can use the addAll() method.

Create a new Class named **“exampletwo,”** and then write the code below in the class.

| import java.util.HashSet;  public class exampletwo {  public static void main(String[] args) {  HashSet<Integer> evenNumbers = new HashSet<>();  evenNumbers.add(2);  evenNumbers.add(4);  System.out.println("HashSet1: " + evenNumbers);  HashSet<Integer> numbers = new HashSet<>();  numbers.add(1);  numbers.add(3);  System.out.println("HashSet2: " + numbers);  // Union of two set  numbers.addAll(evenNumbers);  System.out.println("Union is: " + numbers);  }  } |
| --- |

**Output**

HashSet1: [2, 4]

HashSet2: [1, 3]

Union is: [1, 2, 3, 4]

## Example Three: Difference of Sets.

To calculate the difference between the two sets, we can use the removeAll() method:

Create a new Class named **“examplethree,”** and then write the code below in the class.

| import java.util.HashSet;  public class examplethree {  public static void main(String[] args) {  HashSet<Integer> primeNumbers = new HashSet<>();  primeNumbers.add(2);  primeNumbers.add(3);  primeNumbers.add(5);  System.out.println("HashSet1: " + primeNumbers);  HashSet<Integer> oddNumbers = new HashSet<>();  oddNumbers.add(1);  oddNumbers.add(3);  oddNumbers.add(5);  System.out.println("HashSet2: " + oddNumbers);  // Difference between HashSet1 and HashSet2  primeNumbers.removeAll(oddNumbers);  System.out.println("Difference : " + primeNumbers);  }} |
| --- |

**Output:**

HashSet1: [2, 3, 5]

HashSet2: [1, 3, 5]

Difference: [2]

## Example Four: Enhanced for() Loop

Create a new Class named **“examplefour,”** and then write the code below in the class.

| import java.util.HashSet; public class Examplefour {   public static void main(String args[]) {  // HashSet declaration  HashSet<String> hset = new HashSet<String>();   // Adding elements to the HashSet  hset.add("Apple");  hset.add("Mango");  hset.add("Grapes");  hset.add("Orange");  hset.add("Fig");  //Addition of duplicate elements  hset.add("Apple");  hset.add("Mango");  //Addition of null values  hset.add(null);  hset.add(null); // Using advanced for loop  for (String str : hset) {  System.out.println(" ---> " + str);  }  } } |
| --- |

**Output:**

---> null

---> Apple

---> Grapes

---> Fig

---> Mango

---> Orange

Remember: HashSet DOES NOT maintain any order, so this order might be changed if you run your code a second time.

# TreeSet Examples

## Example one: Iterate Through TreeSet.

To access the individual elements of TreeSet, we need to iterate through the TreeSet —— traverse through the TreeSet. We do this by declaring an Iterator for the TreeSet, and then use that Iterator to access each element. For this, we use the next() method of an iterator that returns the next element in the TreeSet.

**The following Java program demonstrates the use of the Iterator to iterate through TreeSet.**

| **import** java.util.TreeSet;  **import** java.util.Iterator;  **public class** TreesetExampleone {  **public static void** main(String[] args) {  TreeSet<Integer> num\_Treeset = **new** TreeSet<>();  num\_Treeset.add(20);  num\_Treeset.add(5);  num\_Treeset.add(15);  num\_Treeset.add(25);  num\_Treeset.add(10);  *// Call iterator() method to define Iterator for TreeSet*  Iterator<Integer> iter\_set = num\_Treeset.iterator();  System.***out***.print(**"TreeSet using Iterator: "**);  *// Access TreeSet elements using Iterator*  **while**(iter\_set.hasNext()) {  System.***out***.print(iter\_set.next());  System.***out***.print(**", "**);  }  }  } |
| --- |

**Output;**

TreeSet using Iterator: 5, 10, 15, 20, 25,

## Example two: Remove Elements

* remove() - removes the specified element from the set.
* removeAll() - removes all of the elements from the set.

| import java.util.TreeSet; public class TreesetExampletwo {  public static void main(String[] args) {  TreeSet<Integer> numbers = new TreeSet<>();  numbers.add(2);  numbers.add(5);  numbers.add(6);  System.out.println("TreeSet: " + numbers);   // Using the remove() method  boolean value1 = numbers.remove(5);  System.out.println("Is 5 removed? " + value1);   // Using the removeAll() method  boolean value2 = numbers.removeAll(numbers);  System.out.println("Are all elements removed? " + value2);  } } |
| --- |

**Output:**

TreeSet: [2, 5, 6]

Is 5 removed? true

Are all elements removed? true

## Example three: Methods for Navigation.

Since the TreeSet class implements NavigableSet, it provides various methods to navigate over the elements of the TreeSet.

### **1. first() and last() Methods**

* first() - returns the first element of the set.
* last() - returns the last element of the set.

| import java.util.TreeSet;  public class TreesetExampletwo {  public static void main(String[] args) {  TreeSet<Integer> numbers = new TreeSet<>();  numbers.add(2);  numbers.add(5);  numbers.add(6);  System.out.println("TreeSet: " + numbers);   // Using the first() method  int first = numbers.first();  System.out.println("First Number: " + first);   // Using the last() method  int last = numbers.last();  System.out.println("Last Number: " + last);  } } |
| --- |

**Output:**

TreeSet: [2, 5, 6]

First Number: 2

Last Number: 6

## Examplefour: sort the given TreeSet alphabetically in reverse order.

In this example, we will implement a **Comparator** class to sort the given TreeSet alphabetically in **reverse order**. By default, the TreeSet sorts data in **ascending** order.

We can also sort TreeSet in a customized order by defining a new comparator class. In this comparator class, we need to override the ‘compare’ method to sort the elements of the TreeSet. This comparator object is then passed to the TreeSet constructor.

Create a class named “**cities\_Comparator”** and add the code below:

| **import** java.util.Comparator;  **public class** cities\_Comparator **implements** Comparator<String> {  *//override compare method to compare two elements of the TreeSet*  @Override  **public int** compare(String cities\_one, String cities\_two) {  **int** value = cities\_one.compareTo(cities\_two);  *// sort elements in reverse order*  **if** (value > 0) {  **return** -1;  }  **else if** (value < 0) {  **return** 1;  }  **else** {  **return** 0;  }  }  } |
| --- |

Create a class named “**TreeSetExampleCom”** and add the code below:

| **import** java.util.TreeSet;  **public class** TreeSetExampleCom {  **public static void** main(String[] args) {  *// Create a TreeSet with user-defined comparator*  TreeSet<String> cities = **new** TreeSet<>(**new** cities\_Comparator());  *//add elements to the comparator*  cities.add(**"UAE"**);  cities.add(**"Mumbai"**);  cities.add(**"NewYork"**);  cities.add(**"Hyderabad"**);  cities.add(**"Karachi"**);  cities.add(**"Xanadu"**);  cities.add(**"Lahore"**);  cities.add(**"Zagazig"**);  cities.add(**"Yingkou"**);  *//print the contents of TreeSet*  System.***out***.println(**"TreeSet: "** + cities);  }  } |
| --- |

**Output:**

TreeSet: [Zagazig, Yingkou, Xanadu, UAE, NewYork, Mumbai, Lahore, Karachi, Hyderabad]

**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.