**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **17-06-2020** | **Name:** | **Kiran N** |
| **Course:** | **JAVA** | **USN:** | **4al16ec031** |
| **Topic:** | **The java collections framework**  **1.ArrayList: Arrays the Easy Way**  **2.Linked Lists**  **3.HashMap: Retrieving Objects via a**  **Key**  **4.Sorted Maps**  **5.Sets**  **6.Using Custom Objects in Sets and as Keys in Maps**  **7.Sorting Lists** | **Semester & Section:** | **8th and A** |
| **Github Repository:** | **Kiran-course** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Programming**      **Array of ArrayList in Java**  We often come across 2D arrays where most of the part in the array is empty. Since space is a huge  problem, we try different things to reduce the space. One such solution is to use jagged array  when we know the length of each row in the array, but the problem arises when we do not specifically know the length of each of the rows. Here we use ArrayList since the length is unknown.  Following is a Java program to demonstrate the above concept.  **LinkedList in Java**  Linked List are linear data structures where the elements are not stored in contiguous locations and  every element is a separate object with a data part and address part. The elements are linked using  pointers and addresses. Each element is known as a node. Due to the dynamicity and ease of insertions and deletions, they are preferred over the arrays. It also has few disadvantages like the nodes cannot be accessed directly instead we need to start from the head and follow through the link to reach to a node we wish to access.  To store the elements in a linked list we use a doubly linked list which provides a linear data structure  and also used to inherit an abstract class and implement list and deque interfaces.  In Java, LinkedList class implements the list interface. The LinkedList class also consists of various  constructors and methods like other java collections.  Constructors for Java LinkedList:  1.LinkedList(): Used to create an empty linked list.  2.LinkedList(Collection C): Used to create a ordered list which contains all the elements of a  specified collection, as returned by the collection’s iterator. SortedMap Interface in Java with Examples  SortedMap is an interface in collection framework . This interface extends Map interface and provides a total ordering of its elements (elements can be traversed in sorted order of keys). Exampled class that implements this interface is TreeMap  The main characteristic of a SortedMap is that, it orders the keys by their natural ordering, or by a  specified comparator. So consider using a TreeMap when you want a map that satisfies the following criteria:  Collections.sort() in Java with Examples java.util.Collections.sort()  method is present in java.util.Collections class. It is used to sort the elements present in the specified list of Collection in ascending order. It works similar to java.util.Arrays.sort() method but it is better then as it can sort the elements of Array as well as linked list, queue and many more present in it.  public static void sort(List myList)  myList : A List type object we want to sort.  This method doesn't return anything  Set in Java  •Set is an interface which extends Collection. It is an unordered collection of objects in which  duplicate values cannot be stored.  •Basically, Set is implemented by HashSet, LinkedHashSet or TreeSet (sorted representation).  •Set has various methods to add, remove clear, size, etc to enhance the usage of this interface  filter\_none  edit  play\_arrow  brightness\_4  import  java.util.\*;  public  class  Set\_example  {  public  static  void  main(String[] args)  {  // Set deonstration using HashSet  Set<String> hash\_Set = new  HashSet<String>();  hash\_Set.add("Geeks");  hash\_Set.add("For");  hash\_Set.add("Geeks");  hash\_Set.add("Example");  hash\_Set.add("Set");  System.out.print("Set output without the duplicates");  System.out.println(hash\_Set);  // Set deonstration using TreeSet  System.out.print("Sorted Set after passing into TreeSet");  Set<String> tree\_Set = new  TreeSet<String>(hash\_Set);  System.out.println(tree\_Set);  }  }  Java Stream interface  Java Stream interface provides two methods for sorting the list:  sorted() method  Stream interface provides a sorted() method to sort a list. It is defined in Stream interface which is  present in java.util package. It returns a stream sorted according to the natural order. If the elements are not comparable, it throws java.lang.ClassCastException. The signature of the method is:  1.Stream<T>sorted()  Parameter  T: It is a type of stream element.  Java Stream.sorted(Comparator comparator)  It also returns a stream sorted according to the provided comparator. It is stable for an ordered stream.  The signature of the method is:  1.Stream<T>sorted(Comparator<?  Super T> comparator)  Parameters  T is the type of stream element.  comparator to be used to compare elements  Example  In the following example, we have used the following methods:  In Java 8, stream() is an API used to process collections of objects.  The collect() method is used to receive elements from a stream and stored them in a collection.  The toList() return the collector which collects all the input elements into a list, in encounter  order. |