|  |  |
| --- | --- |
|  | **What is a Collection API?**  It’s a collection of classes, which are used to store and manage multiple elements. |
|  | **What are the limitations of an Array?**  There are mainly 3 limitations.  i)Array size can’t be modified.  ii)Arrays are only for one type of elements. We can’t store multiple elements in same  Array  iii)Managing elements of an Array is very difficult. |
| 1. **Why** | **Why do we require collection framework? (or) What is the need of collection framework to the Industry?**  In order to address limitations of Arrays collection framework is introduced. By using collection framework we can manage multiple elements very easily and effectively. We can add any number of elements, any type of elements also. There are several ways to add elements, several ways to remove elements, several ways to read and update the elements. Searching and Sorting is very easy and effective.Identifying uniqueness is also very easy and effective. |
| **do we require collection framework? (or) What is the need of collection framework to the Industry?**  In order to address limitations of Arrays collection framework is introduced. By using collection framework we can manage multiple elements very easily and effectively. We can add any numberof elements,any type of elements also.There are several ways to add elements, several ways to remove elements, several ways to read and update the elements. Searching and Sorting is very easy and effective.Identifying uniqueness is also very easy and effective. | **What are the different sections available under collection API? (or) What are the different streams available in collection API?**  Four sections are available in collection API:  i)List  ii)Queue  iii)Set  iv)Map |
|  | **What is the significance of List?**  The main significance of List is Auto indexing. While adding an element automatically index is providing to that element. In which ever order you are adding the elements into the List in the same order we can able to read. |
|  | **What is the significance of Queue?**  Queue is mainly used for maintaining First In First Out(FIFO). In case of Queue type collection we can able to read or remove only first element. |
|  | **What is the significance of Set?**  The main significance of Set is maintaining uniqueness. It will not be allowing duplicates elements. In order to store unique elements use Set collection stream. |
|  | **What is the significance of Map?**  In case of Map every element should be stored by choosing a unique key. We can’t store any element without a key. (Key, element) pair should be added in the Map. |
|  | **What are the different ways available to add elements into List type collection object?**  i) Adding an element into add() i.e. It is adding Single element at the end.  ii)Inserting an element in the specified location or index. Here we are using add() which is  taking two arguments one is index and another one is element.  iii)Adding elements of one collection into another collection by using addAll().  iv)Adding elements of one collection into another collection by using constructor. |
|  | **What are the different ways available to read the elements from List type collection object?**  i)By printing reference of the List type collection object. (By using toString() method)  ii)We can read single element by using get(). To read all the elements use regular for loop  along with size and get().  iii)We can read all the elements by using enhanced for loop.  iv)We can read element by element by using Iterator.  v) We can read element by element by using ListIterator. |
|  | **What are the different ways available to remove the elements from List type collection object?**  i)Using remove () by specifying index.  ii)Using remove() by specifying object itself.  iii)Using removerange() in order to remove range of elements from one index to another index.  iv)clear().In order to remove every element from the List type collection object.  v)We can remove elements of one collection from another collection by using removeAll().  vi)We can retain elements of one collection from another collection by calling retainAll(). |
|  | **How to replace an existing element with a new element?**  We can replace an existing element with a new element by using set(). |
|  | **What is the return type of remove() which is taking an index as an argument?**  return type is object. |
|  | **What is the return type of remove() which is taking an argument as a object?**  return type is boolean. |
|  | **What is the return type of set()?**  set () return type is object. |
|  | **What is the significance of return value of set()?**  Whichever object is replacing same object is returning from set(). |
|  | **List type collections allowing duplicates or not?**  Yes, List type collections allowing duplicates. |
|  | **List type collections allowing null values or not?**  Yes, Allowing any number of nulls. |
|  | **What is the return type of contains() from the List type collections?**  boolean |
|  | **How to avoid duplicates while adding elements into the List type collections?**  While adding an element check it out whether same element already added or not by using contains(), if not added then only try to add.  ***Simple Program for avoding Duplicates:***  package com.pryce;  import java.util.ArrayList;  import java.util.Scanner;  public class WithoutDuplicates  {  public static void main(String[] args)  {  ArrayList <String> list= new ArrayList<>();  Scanner sc = new Scanner(System.in);  String decider = null;  do  {  System.out.println("Enter Element");  String element = sc.next();  if (!list.contains(element))  {  list.add(element);  }  System.out.println("Again(y/n)");  decider = sc.next();  }  while ("y".equals(decider));  System.out.println(list);  }  } |
|  | **Which utility class is used to achieve sorting under List type collections?**  collections class. |
|  | **In which package comparable interface is available?**  comparable interface is available in java.lang package. |
|  | **What is the name of the method inside a comparable interface?**  compareTo() method is avaliable inside comparable interface. |
|  | **What is the argument of the compareTo()?**  Arugument type is object type. |
|  | **What is the return type of compareTo() inside a comparable interface?**  return type of compareTo() is int. |
|  | **Which two objects are comparing by comparable interface?**  this and argument of the compareTo() method. |
|  | **In which scenario comparable interface is preferable?**  If sorting is limited to only one attribute and also comparison should be done in the same entity class. |
|  | **In which scenario comparable interface can’t be used?**  If you have a requirement of comparing different attributes in the different scenarios. |
|  | **Why do we require comparator interface?**  If you have a requirement of comparing different attributes in the different scenarios. |
|  | **In which package comparator interface is available?**  java.util package. |
|  | **What is the method available inside a comparator interface?**  compare(). |
|  | **What are the arguments to the compare() of comparator interface?**  Two arguments both are object type. |
|  | **What Is the return type of compare() from comparator interface?**  return type is int. |
|  | **Which two objects are comparing by comparator interface?**  Which ever two objects supplying to compare() as an arguments. |
|  | **What are the differences between comparable and comparator interface?**   |  |  | | --- | --- | | **comparable interface** | **comparator interface** | | 1. comparable interface is available inside java.lang package no need of importing explicitly. | 1. comparator interface is available inside java.util package we have to import explicitly. | | 2. comparable interface is used for comparing current element with element. | 2. comparator is used for comparing any two objects. | | 3. comparable is used only for comparing only one attribute. | 3. comparator can be used for comparing any attributes. | |
|  | **What is the argument type to the sort()of collections class?**  data type is a List type. |
|  | **What is the second argument type to the sort method of the collections class?**  comparator type. |
|  | **What are the conditions to be followed while sorting elements of List type?**  i)All elements should be same type.  ii)No null elements.  iii)Either elements should be comparable type or separate comparator should be specified. |
|  | **Differences between Iterator and ListIterator?**   |  |  | | --- | --- | | **Iterator** | **ListIterator** | | 1. Iterator we can iterate elements only one time that to in the forward direction. | 1. Listiterator we can iterate the elements any number of times in any direction. | | 2. By using Iterator we can just read or remove the elements. | 2. By using ListIterator not only reading or removing we can add and update also. | | 3. By using Iterator we can’t find out what could be the next index and what was the previous index. | 3. By using ListIterator we can find out what could be the next index and what was the previous index. | | 4. Iterator it can be used for any type of collection. | 4. ListIterator only for List type collection. | |
|  | **What are the differences between Iterator and Enumeration?**   |  |  | | --- | --- | | **Iterator** | **Enumeration** | | 1. By using iterator we can read or remove the elements. | 1. By using Enumeration we can just read the elements we can’t remove the elements. | | 2. Iterator is a failfast iterator if collection class is not from the concurrent modification. | 2. Enumeration always failsafe. | | 3. Iterator can be used for all the collection type objects. | 3.Enumeration limited to some collection types only not for all the collection types.(For Eg: Incase of ArrayList only Iterator is possible, Where as in Vector both Enumeration and also Iterator is possible) | |
|  | **Differences between ArrayList and Vector?**   |  |  | | --- | --- | | **ArrayList** | **Vector** | | 1. ArrayList is a non-synchronized | 1. Vector is a synchronized. | | 2.ArrayList is not advisiblein the multi-threaded environment. | 2. Vector is most preferrable in the multi-threaded environment. | | 3.ArrayList gives more performance because of non-synchronization. | 3. Vector gives less performance because of synchronization. | | 4. ArrayList capacity increases 50% every time | 4.Vector it increases 100% once existing capacity got over. | | 5. ArrayList is introduced in JDK1.2 along with the collection framework. | 5. Vector introduced in JDK1.0 itself. | | 6. ArrayList is a non-legacy class | 6. Vector is a legacy class. | |
|  | **Differences between ArrayList and LinkedList?**   |  |  | | --- | --- | | **ArrayList** | **LinkedList** | | 1. ArrayList is only List type. | 1. LinkedList is a both List and Queue. | | 2. ArrayList is very effective incase of frequent random access. | 2. LinkedList is not advisible in place of frequent random access you will be get less performance. | | 3. ArrayList is not advisible incase of frequent insertions you will be getting less performance. | 3. LinkedList is advisible incase of frequent insertions you will be getting more performance. | | 4. ArrayList is the best incase of Sorting. | 4. Sorting is not at all effective incase of LinkedList. | |
|  | **Which remove() method of List type collection can’t be used straightaway by using List type reference variable?**  removerange() method as it is protected. We can’t use straightaway by using List type reference variable. |
|  | **What is the difference between peak() and poll()?**   |  |  | | --- | --- | | **peak()** | **poll()** | | peak() is only reading the head element but it’s not removing that head element | poll() is used for reading and removing the head element. | |
|  | **What are the differences between LinkedList and Priority Queue?**   |  |  | | --- | --- | | **LinkedList** | **Priority Queue** | | 1. LinkedList is a list type of Queue because of this we can able to read, insert, and remove any elements. | 1. Priority Queue is a pure Queue type. We can read (or) remove only the head element. We can’t read (or) remove any other elements. | | 2. LinkedList is not having auto sorting. Inorder to sort we have to use Collection.sort (). | 2. Priority Queue is a auto sorted Queue. Only head element getting sorted always. | | 3. We can add any type of elements. | 3. In the Priority Queue we can add only one type of elements. | | 4. We can able to add null values. | 4. We can’t able to add null values. | | 5. While adding elements into the LinkedList, we don’t require considering either comparable type or separate comparator object. | 5. In the Priority Queue we have to add either elements or comparable type or separate comparable object should be supplied. | |
|  | **What are the conditions to be followed while sorting elements in the Priority Queue?**  i)All elements should be same type.  ii)No element can be a null.  iii)Either all elements should be comparable (or) separately we have to  supply comparator object. |
|  | **How to specify comparator object to the Priority Queue?**  We have to supply comparator object to the constructor of the Priority Queue. |
|  | **What is the return type of the add() of the Set type collection?**  return type is a boolean |
|  | **While adding duplicates to the Set type collection will you get compile time error or Runtime error?**  No, compile time error  No, Runtime error  It won’t add duplicates. |
|  | **What is a significance of return value of add () of a Set type collection?**  If add () returns true then adding is very success. If add () returns false addition isn’t success because of duplicates. |
|  | **Which methods are using by Set type collection to identify the duplicates?**  hashCode() & equals() |
|  | **In which class hashCode () & equals () available (or) incorporated?**  Object class. |
|  | **While adding elements into Set type collection which methods of object class should be overrided in the element class?**  hashCode () & equals(). |
|  | **While overriding hashCode () and equals () inside an element class to avoid duplicates, What are the contracts to be followed?**  i)We have to override both hashCode() and equals(). Not one method.  ii) What ever properties consider in the hashCode () same properties should be consider in the equals () also. |
|  | **While adding first element in the Set type collection, which method will be called from the element?**  It calls only hashCode (). |
|  | **Which method of object class will be called for every addition of the element inside a Set type collections?**  Only hashCode () is guarentee to execute. |
|  | **In which conditions equals () method will be called from the element while adding element into Set type collections?**  hashCode of current element is same as hashCode to any existing element from the Set type collection. |
|  | **Is it possible to read one particular element from Set type collection?**  We can’t read any specified element from the Set type collection. |
|  | **Why we can’t read any specific element from a Set type collection?**  Element is not assigned with any index or key. So that we can’t read any element from the Set type collection. |
|  | **What are the ways available to read all the elements of Set type collection?**  There are 2 ways available to read all the elements:  i)By printing reference of Set type collection(Here internally toString() is calling)  ii)By using iterator (Here reading element by element) |
|  | **Is it possible to read elements of Set type collection in the order of adding?**  Not possible.  ***Reason:*** No indexing (or) No key to a particular element. |
|  | **Is it possible to add null values inside a HashSet?**  Absolutely possible. One null will be added successfully. |
|  | **How to sort elements of a Set type collection?**  By using TreeSet. |
|  | **How to supply comparator type to the TreeSet object?**  By using constructor of the keyset. |
|  | **What are the conditions to be followed while adding elements into a TreeSet which isn’t specified with any comparator objects?**  i) All the elements should be same type.  ii)No element should be null  iii) Either elements should be comparable (or) separate comparator should be specified.  IV) Overriding hashCode () & equals () for avoding duplicates and to achieve Sorting. |
|  | **How to remove duplicates in the given array?**  ***Program:***  package com.pryce;  import java.util.Arrays;  import java.util.List;  import java.util.Set;  import java.util.HashSet;  public class Z1  {  public static void main(String[] args)  {  String[] x = {"abc","xyz","abc","xyz","abc"};  System.out.println(Arrays.toString(x));  List<String> y = Arrays.asList(x);  Set<String> z = new HashSet<>(y);  x = (String[])z.toArray(new String[]{});  System.out.println(Arrays.toString(x));  }  } |
|  | **How can we convert array into List?**  Arrays.asList () converting array into List type. |
|  | **How to convert List type into Set type?**  Supply List type to the constructor of Set type. new HashSet<>() |
|  | **How to convert Set into Array?**  Set type elements we have to store into Array type. Inside a Set, there is a method called toArray. We need to supply argument to toArray (). |
|  | **How to remove duplicates in the ArrayList?**  ***Program:***  package com.pryce;  import java.util.ArrayList;  import java.util.Set;  import java.util.HashSet;  public class Z2  {  public static void main(String[] args)  {  ArrayList <String> x = new ArrayList<>();  x.add("abc");  x.add("xyz");  x.add("abc");  x.add("xyz");  x.add("abc");  x.add("xyz");  System.out.println(x);  Set <String> z = new HashSet<>(x);  x.clear();  x.addAll(z);  System.out.println(x);  }  } |
|  | **Why Map interface is not extending with collection interface?**  Map is not compatable to collection type. The reason is Map required key & element. Collection required only element. |
|  | **How to add elements into the Map type collections?**  By using put () method of the Map type. put () taking two arguments. 1st argument is a Key and 2nd argument is a value (or) element. |
|  | **Is it possible to choose user defined class type as a key while storing elements in the Map?**  Yes, Absolutely possible. |
|  | **Inside a Map is it possible to insert duplicate elements?**  Yes, Absolutely possible. |
|  | **Is it possible to choose duplicate keys while storing multiple elements inside a Map?**  No It’s not possible. All Keys should be unique inside a Map. |
|  | **How Map is identifying duplications incase of keys?**  By calling hashCode () and equals (). |
|  | **What are the differences between HashMap and HashTable?**   |  |  | | --- | --- | | **HashMap** | **HashTable** | | **1.** HashMap allows any number of null elements and it can allow one null key. | **1.** Incase of HashTable no null elements no null as a key.While adding null in the HashTable you will be get null Pointer exception. | | **2.** HashMap is non-synchronized because of this we will get more performance and not adivisible to use in the multi-threaded environment. | **2.** HashTable is synchronized because of this you will be get less performance and advisible to use in the multi-threaded environment. | | **3.** By using HashMap we can get failfast iterator but not failsafe Enumeration. | **3.** By usingHashTable not only getting failfast iterator we can get even failsafe Enumerator also. | |
|  | **How data adding under Map stream?**  One entry object is creating with a key and element that entry object is adding inside a Map. |
|  | **What is an entry under Map type collections?**  entry is a interface inside a Map interface. It is used for storing an element along with it’s key. Inside an entry interface there are two methods :  i) **getKey ():-** Which is returing key object from the entry object.  ii)**getValue():-** Which is returing value object (element object) from the entry object |
|  | **Is it possible to read a specific entry object from the Map object?**  No, It’s not possible.We can’t read one entry from the Map object. |
|  | **How to read all entry objects from a Map object?**  Call entrySet () from the Map it returns setof entry objects from the Map. After getting set use iterator to read entry objects one by one from the set. |
|  | **Is it possible to read a specific value (Element object) from the Map object?**  Yes, Absolutely possible call get () by supplying assigned key. |
|  | **Is it possible to read all the key objects from a Map object?**  Yes, Absolutely possible. Call keySet () from the Map object it returns all the key objects into a Set. Use iterator to read key objects one by one from the Set(). |
|  | **Is it possible to read all the value objects from a Map object?**  No it’s not possible. There is no valueset () like a keyset (). |
|  | **How to copy all the value objects of Map into ArrayList?**  ***Program***  package com.pryce;  import java.util.ArrayList;  import java.util.Set;  import java.util.HashMap;  import java.util.Map.Entry;  public class A  {  public static void main(String[] args)  {  HashMap <String,Integer> map=new HashMap<>();  map.put("k1",100);  map.put("k2",200);  map.put("k2",200);  Set<Entry<String,Integer>> set = map.entrySet();  ArrayList <Integer> list=new ArrayList<>();  for (Entry<String,Integer> entry:set)  {  list.add(entry.getValue());  }  System.out.println(list);  }  } |
|  | **How to sort content of the Map type collections?**  In order to sort content of Map type collections use TreeMap. |
|  | **Sorting of Map content is based on key object or value object?**  Based on key Object. |
|  | **What are the conditions to be followed while storing entries into TreeMap which is not specified with any comparator?**  i) All should be same type.  ii) No key can be a null.  iii) key type should be comparable type  iv) Inside a key class we have to override hashCode() and equals(). |
|  | **What is the root interface in collection hierarchy?**  Collection interface. collection interface is extending with iterable interface but iterable interface is a part of java.lang package.It can’t be consider as a part of the collection framework |
|  | **How to reverse elements of List type collection?**  By using collections.reverse().collections class reverse method taking one argument that is List type.Which ever list type elements we want to reverse that list should supply as an argument to reverse(). |
|  | **What are the differences among peak(), poll(), and remove() in the Queue stream?**   |  |  |  | | --- | --- | --- | | **peak()** | **poll()** | **remove()** | | peak () only reading the head element but not removing. If Queue is a empty it returns a null. | poll () used to reading and removing the head element. If Queue is a empty it returns a null. | remove () used to reading and removing the head element. If Queue is a empty it is throwing NoSuchElementException. | |
|  | **What is an identity HashMap?**  It is a HashMap type where equality of the keys is based on references not the object content. |
|  | **What is a week HashMap?**  Week HashMap is a type of HashMap where Keys are week. If Key not referring from outside week HashMap then that key is eligible for garbage collection. |
|  | **How we can make any collection as read only?**  By using unmodifiable methods of collections class. |
|  | **What are the differences between fail-fast Iterator and fail-safe Iterator?**   |  |  | | --- | --- | | **fail-fast Iterator** | **fail-safe Iterator** | | **1.** fail-fast Iterator throwing concurrent modification exception while modifying concurrently. | **1.** fail-safe Iterator allows concurrent modification without any exception. | | **2.** fail-fast Iterator willn’t be cloning objects into another place of the memory. | **2.** fail-safe Iterator cloning all objects into another place of memory for every modifications. | | **3.** fail-fast Iterator is very effective incaseof memory management. | **3.** fail-safe Iterator produces memory over heads. | |
|  | **What are the differences between ArrayList and CopyonwriteArrayList?**   |  |  | | --- | --- | | **ArrayList** | **CopyonwriteArrayList** | | **1.** ArrayList Iterators are fail-fast Iterators | **1.** CopyonwriteArrayList are fail-safe Iterators. | | **2.** ArrayList is not a thread safed one. | **2.** CopyonwriteArrayList is a thread safed one. | |