Core Java Topics

Cheke, Sagar

[Year]

# **Object Oriented Programming Concepts:-**

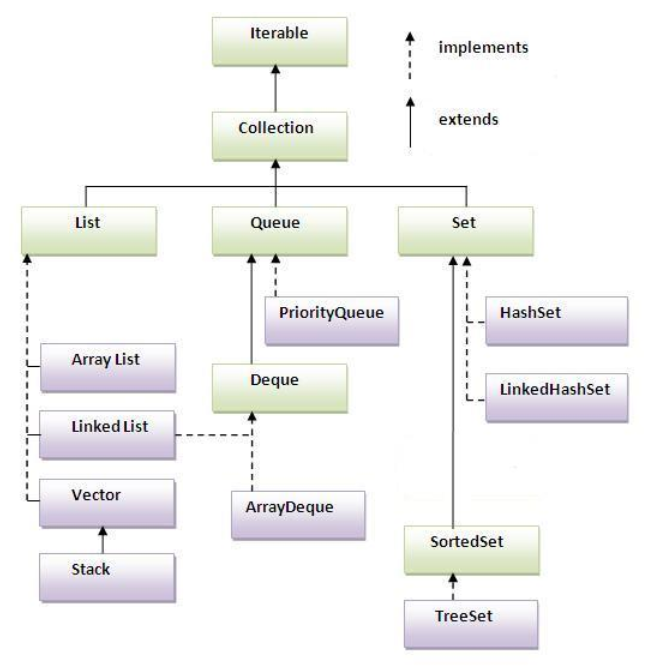
# **Collections:-**

Collections in java is a framework that provides an architecture to store and manipulate the group of objects. All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.

Java Collection simply means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque etc.) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet etc).

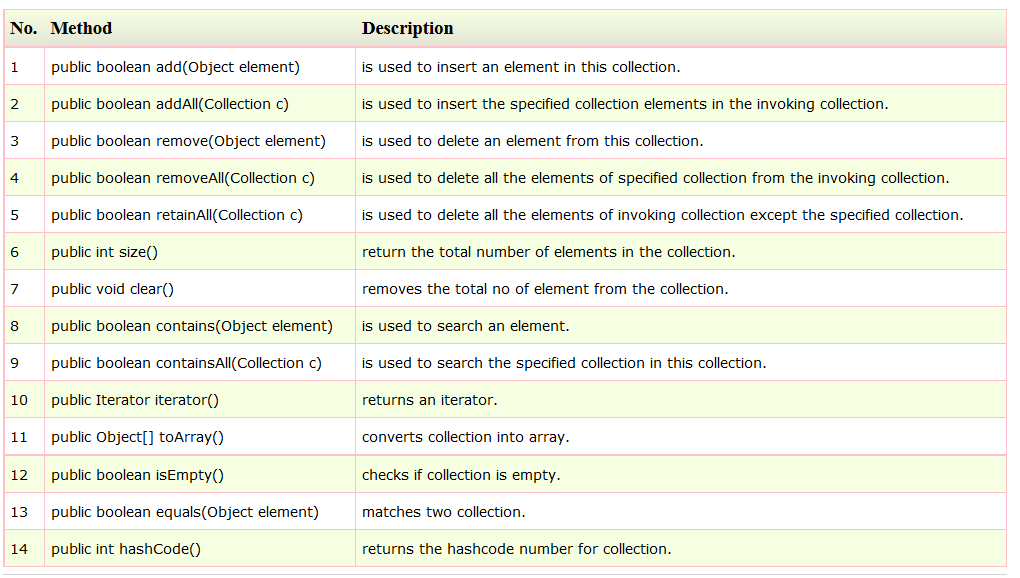
## **Hierarchy of Collection Framework:-**

Let us see the hierarchy of collection framework. The java.util package contains all the classes and interfaces for Collection framework.



## **Methods of Collection interface:-**

There are many methods declared in the Collection interface. They are as follows:



## **Iterator interface:-**

|  |
| --- |
| Iterator interface provides the facility of iterating the elements in forward direction only. |

**Methods of Iterator interface**

There are only three methods in the Iterator interface. They are:

1. **public boolean hasNext()** it returns true if iterator has more elements.
2. **public object next()** it returns the element and moves the cursor pointer to the next element.
3. **public void remove()** it removes the last elements returned by the iterator. It is rarely used.

# **List Collection:-**

## **ArrayList**

## **LinkedList**

# **Set Collection:-**

## **HashSet**

## **LinkedHashSet**

## **TreeSet**

# **Map Collection:-**

A map contains values based on the key i.e. key and value pair. Each pair is known as an entry. Map contains only unique elements.

* **Commonly used methods of Map interface:-**
  1. public Object put(Object key, Object value): is used to insert an entry in this map.
  2. public void putAll(Map map): is used to insert the specified map in this map.
  3. public Object remove(Object key): is used to delete an entry for the specified key.
  4. public Object get(Object key): is used to return the value for the specified key.
  5. public boolean containsKey(Object key): is used to search the specified key from this map.
  6. public boolean containsValue(Object value): is used to search the specified value from this map.
  7. public Set keySet(): returns the Set view containing all the keys.
  8. public Set entrySet(): returns the Set view containing all the keys and values.
* **Entry**

Entry is the sub interface of Map. So we will be accessed it by Map. Entry name. It provides methods to get key and value.

**Methods of Entry interface:**

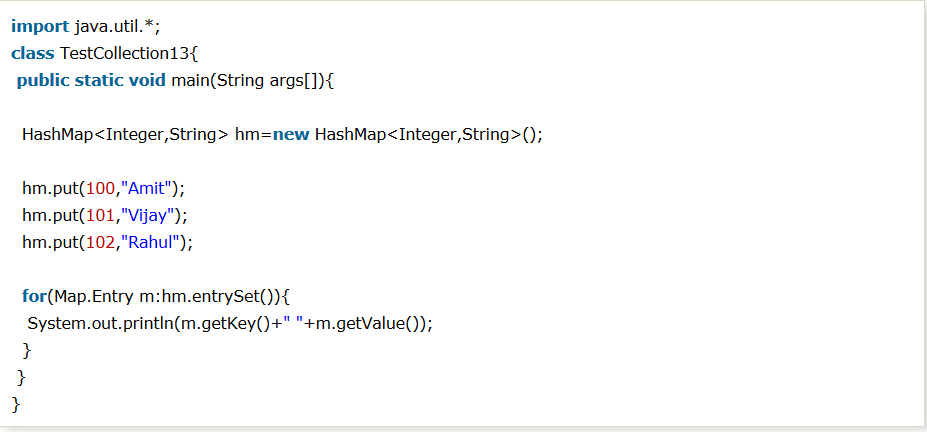
1. public Object getKey(): is used to obtain key.
2. public Object getValue():is used to obtain value.

## **HashMap class:-**

A HashMap contains values based on the key. It implements the Map interface and extends AbstractMap class.

1. It contains only unique elements.
2. It may have one null key and multiple null values.
3. It maintains no order.

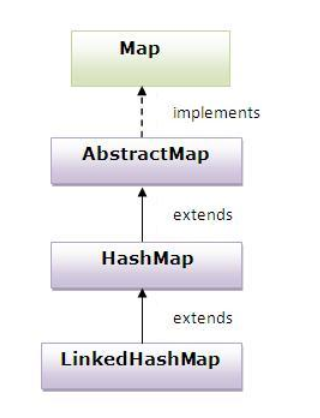
Example of HashMap class:-



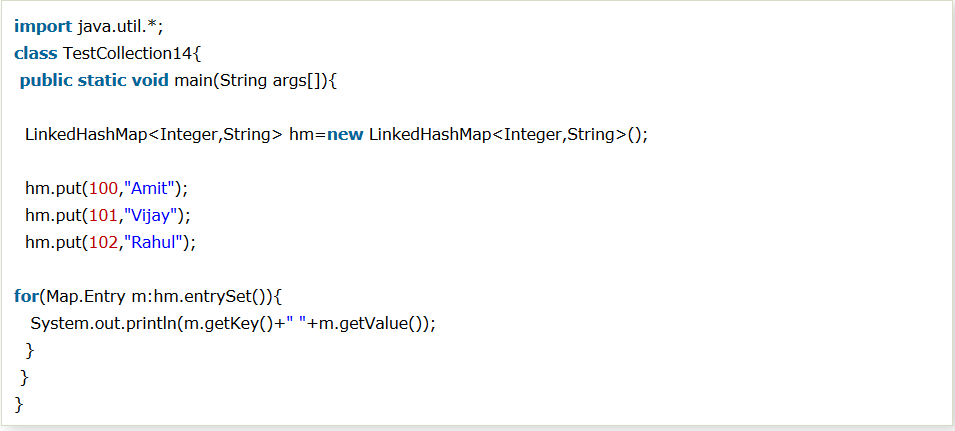
## **LinkedHashMap class:-**

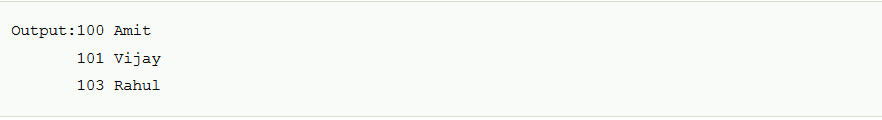
1. A LinkedHashMap contains values based on the key. It implements the Map interface and extends HashMap class.
2. It contains only unique elements.
3. It may have one null key and multiple null values.
4. It is same as HashMap instead maintains insertion order.

* Hierarchy of LinkedHashMap class:



* Example of LinkedHashMap class:

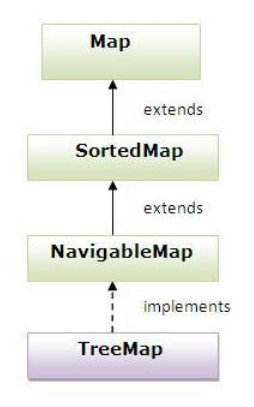




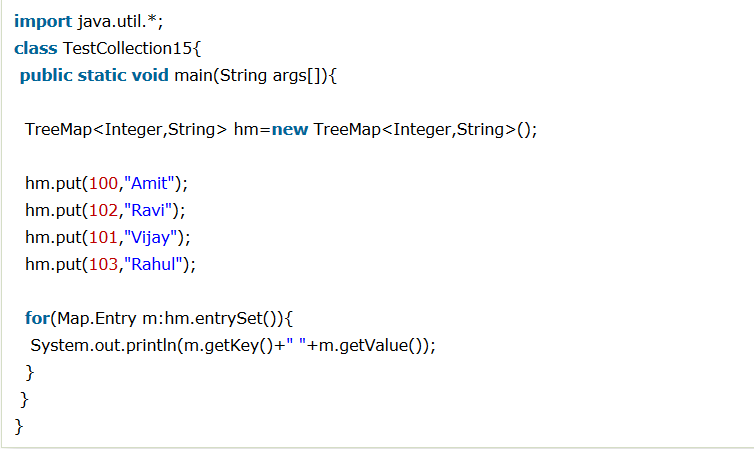
## **TreeMap class:-**

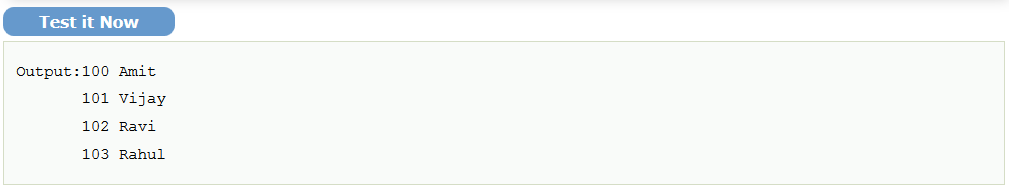
1. A TreeMap contains values based on the key. It implements the NavigableMap interface and extends AbstractMap class.
2. It contains only unique elements.
3. It cannot have null key but can have multiple null values.
4. It is same as HashMap instead maintains ascending order.

* Hierarchy of TreeMap class:



* Example of TreeMap class:

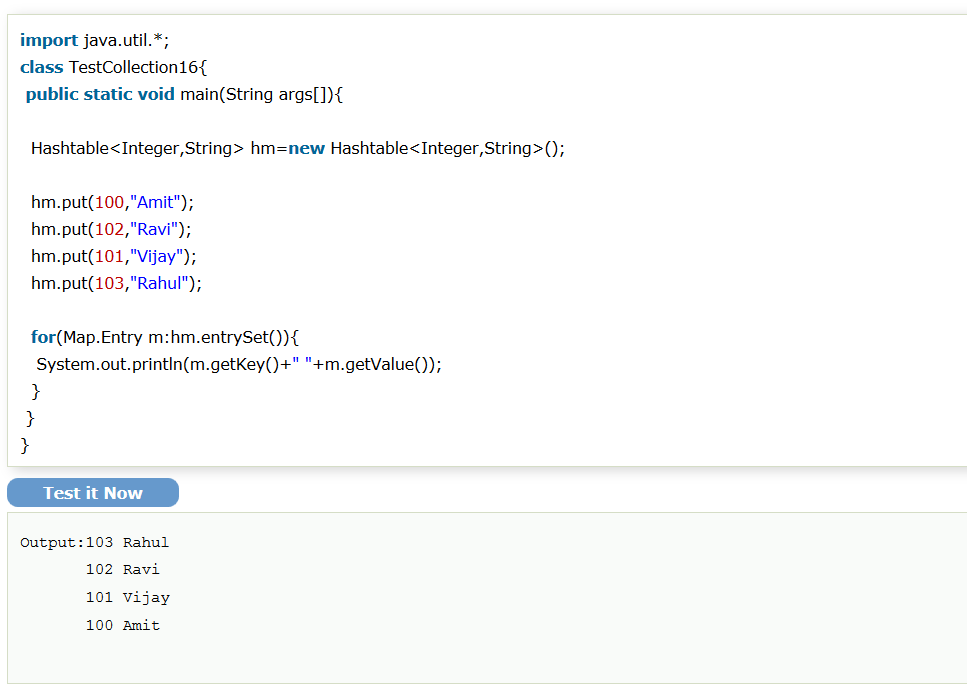




## **Hashtable class:-**

1. A Hashtable is an array of list. Each list is known as a bucket. The position of bucket is identified by calling the hashcode() method. A Hashtable contains values based on the key. It implements the Map interface and extends Dictionary class.
2. It contains only unique elements.
3. It may have not have any null key or value.
4. It is synchronized.

* Example of Hashtable:



* Difference between HashMap and Hashtable:

|  |  |
| --- | --- |
| **Hash Map** | **Hash Table** |
| 1. HashMap is **non synchronized**. It is not-thread safe and can't be shared between many threads without proper synchronization code. | 1. Hashtable is **synchronized**. It is thread-safe and can be shared with many threads. |
| 1. HashMap **allows one null key and multiple null values**. | 1. Hashtable **doesn't allow any null key or value.** |
| 1. HashMap is a **new class introduced in JDK 1.2**. | 1. Hashtable is a **legacy class**. |
| 1. HashMap is **fast**. | 1. Hashtable is **slow**. |
| 1. We can make the HashMap as synchronized by calling this code Map m = Collections.synchronizedMap(hashMap); | 1. Hashtable is internally synchronized and can't be unsynchronized. |
| 1. HashMap is **traversed by Iterator**. | 1. Hashtable is **traversed by Enumerator and Iterator**. |
| 1. Iterator in HashMap is **fail-fast**. | |  |  | | --- | --- | |  | 1. Enumerator in Hashtable is **not fail-fast** | |
| 1. HashMap inherits **AbstractMap** class. | |  |  | | --- | --- | |  | 1. Hashtable inherits **Dictionary** class. | |

## Sorting

## Comparable Interface

## Comparator Interface

## Properties Class

# **Threads:-**

# **Exceptions:-**

# **Java I/O:-**

# **Serialization:-**

# **Java Reflection:-**