Array List:

* Array List uses dynamic array for storing the elements
* It had No size limit
* We can add or remove elements anytime
* It implements List Interface
* Array List contains Duplicate values, null values
* Array List maintains the insertion order
* Array List is non-synchronized, Not Thread safe
* Array List allows random access of elements because it follows index basis.
* **import** java.util.\*;
* **public** **class** ArrayListExample2{
* **public** **static** **void** main(String args[]){
* ArrayList<String> list=**new** ArrayList<String>();//Creating arraylist
* list.add("Mango");//Adding object in arraylist
* list.add("Apple");
* list.add("Banana");
* list.add("Grapes");
* //Traversing list through Iterator
* Iterator itr=list.iterator();//getting the Iterator
* **while**(itr.hasNext()){//check if iterator has the elements
* System.out.println(itr.next());//printing the element and move to next
* }
* }
* }

…………

1. //Sorting the list

Collections.sort(list2);

………

1. System.out.println("Returning element: "+al.get(1));//it will return the 2nd element, because index starts from 0
2. //changing the element
3. al.set(1,"Dates");

……….

Linked List:

* Linked List uses double linked list to store elements
* It Implements List interface
* It contains duplicate elements
* Linked list maintains insertion order
* Linked List is non-synchronized and no thread safe
* Manipulation is fast , no shifting required

1. **public** **class** LinkedList2{
2. **public** **static** **void** main(String args[]){
3. LinkedList<String> ll=**new** LinkedList<String>();
4. System.out.println("Initial list of elements: "+ll);
5. ll.add("Ravi");
6. ll.add("Vijay");
7. ll.add("Ajay");
8. System.out.println("After invoking add(E e) method: "+ll);
9. //Adding an element at the specific position
10. ll.add(1, "Gaurav");
11. System.out.println("After invoking add(int index, E element) method: "+ll);
12. LinkedList<String> ll2=**new** LinkedList<String>();
13. ll2.add("Sonoo");
14. ll2.add("Hanumat");
15. //Adding second list elements to the first list
16. ll.addAll(ll2);
17. System.out.println("After invoking addAll(Collection<? extends E> c) method: "+ll);
18. LinkedList<String> ll3=**new** LinkedList<String>();
19. ll3.add("John");
20. ll3.add("Rahul");
21. //Adding second list elements to the first list at specific position
22. ll.addAll(1, ll3);
23. System.out.println("After invoking addAll(int index, Collection<? extends E> c) method: "+ll);
24. //Adding an element at the first position
25. ll.addFirst("Lokesh");
26. System.out.println("After invoking addFirst(E e) method: "+ll);
27. //Adding an element at the last position
28. ll.addLast("Harsh");
29. System.out.println("After invoking addLast(E e) method: "+ll);
31. }

Hash Map:

* Hashmap implements the Map interface
* It stores elements in the form of key and value pair
* It will replace element based on corresponding key
* Hashmap contains values based on key
* Hashmap contains only unique keys
* Non-synchronized, no thread safe
* It will not maintain order
* It will have one null key and multiple null values
* **public** **static** **void** main(String args[]){
* HashMap<Integer,String> hm=**new** HashMap<Integer,String>();
* System.out.println("Initial list of elements: "+hm);
* hm.put(100,"Amit");
* hm.put(101,"Vijay");
* hm.put(102,"Rahul");
* System.out.println("After invoking put() method ");
* **for**(Map.Entry m:hm.entrySet()){
* System.out.println(m.getKey()+" "+m.getValue());
* }
* hm.putIfAbsent(103, "Gaurav");
* System.out.println("After invoking putIfAbsent() method ");
* **for**(Map.Entry m:hm.entrySet()){
* System.out.println(m.getKey()+" "+m.getValue());
* }
* HashMap<Integer,String> map=**new** HashMap<Integer,String>();
* map.put(104,"Ravi");
* map.putAll(hm);
* System.out.println("After invoking putAll() method ");
* **for**(Map.Entry m:map.entrySet()){
* System.out.println(m.getKey()+" "+m.getValue());
* }
* }
* }

Hash Table:

* It implements map interface and hashtable
* A Hashtable is an array of a list. Each list is known as a bucket. The position of the bucket is identified by calling the hashcode() method. A Hashtable contains values based on the key.
* Hashtable contains unique elements
* It doesn’t allow null key, null values
* It is synchronized and thread safe

1. **class** Hashtable1{
2. **public** **static** **void** main(String args[]){
3. Hashtable<Integer,String> hm=**new** Hashtable<Integer,String>();
5. hm.put(100,"Amit");
6. hm.put(102,"Ravi");
7. hm.put(101,"Vijay");
8. hm.put(103,"Rahul");
10. **for**(Map.Entry m:hm.entrySet()){
11. System.out.println(m.getKey()+" "+m.getValue());
12. }
13. }
14. }

Hash Set:

* Hash Set implements Set interface
* Hash set stores elements using hashing mechanism
* Hashset allows null values
* Hash set is non-synchronized, not thread safe
* It doesnot maintain insertion order, elements inserted based on hashcode
* Hashset is best approach for search operation
* **public** **static** **void** main(String args[]){
* //Creating HashSet and adding elements
* HashSet<String> set=**new** HashSet();
* set.add("One");
* set.add("Two");
* set.add("Three");
* set.add("Four");
* set.add("Five");
* Iterator<String> i=set.iterator();
* **while**(i.hasNext())
* {
* System.out.println(i.next());
* }
* }

API Testing:

* 200 – OK
* 201 – Created
* 400 – Bad Request
* 401 – Unauthorised
* 402 – Payment Required
* 403 – Forbidden
* 404 – Not Found
* 405 – Method Not Allowe
* 500 – Internal Server Error
* 501 – Not Implemented
* 502 – Bad Gateway
* 503 – Service Unavailable
* [504 – Gateway Timeout](https://www.softwaretestinghelp.com/504-gateway-timeout-error-fix/)

1. **200**— For Successful request.
2. **201** — For successful request and data was created.
3. **204** — For Empty Response.
4. **400** — For Bad Request. The request could not be understood or was missing any required parameters.
5. **401** — For Unauthorized access. Authentication failed or user does not have permissions for the requested operation.
6. **403** — For Forbidden, Access denied.
7. **404** — For data not found.
8. **405** — For Method Not Allowed or Requested method is not supported.
9. **500** — For Internal Server Error.
10. **503** — For Service Unavailable.
11. **504**--- Gateway Time out

**Maven**:

1.Maven is a build which provided by apache foundation

2.incase if created java project we need to download all jar files and require to add to project i.e Selenium, Java,TestNG,Junit, Cucumber, Rest Assured etc..

3.Incase new versions came for jar files we need to delete already added jar files , need to download latest one and again need to add to project, these are disadvantages. To overcome this we are using maven.