1. What are the main differences between array and collection?

Array:

* Arrays are collection of homogeneous data.
* Array can hold both object and primitive data types.
* Arrays are fixed in size i.e. we cannot able to increase or decrease the size of an array.
* Arrays doesn’t have an underlying data structures and inbuilt methods are not available.
* With respect to memory arrays are not recommended to use.

Collection:

* Collection can have both homogeneous and heterogeneous data.
* Collection can hold only object types but primitive.
* Collection are grow able in nature i.e. we can increase or decrease the size of the object.
* Collection have underlying data structure for every classes and inbuilt methods are present which we can use directly without implementing it.
* With respect to memory collection are recommended to use.

1. Explain various interfaces used in Collection framework?

Interfaces used in collection framework are:

* List
* Set
* Queue
* Sorted set
* Navigable set

1. What is the difference between ArrayList and Vector?

ArrayList:

1. Arraylist is not synchronized
2. It is not a legacy class because it is introduced in jdk 1.2
3. It is fast because it is not synchronized
4. Arraylist increases 50% if the default size exceeds
5. Arraylist uses iterator interface to traverse the elements

Vector:

1. Vector is synchronized
2. It is a legacy class because it is introduced in jdk 1.0
3. It is slow because it is synchronized.
4. Vector increases by 100% if the default size exceeds
5. Vector uses iterator and enumeration interface to traverse the elements.
6. What is the difference between ArrayList and LinkedList?

Arraylist:

* Arraylist internally uses dynamic array to store the elements.
* Arraylist manipulation is slow because it internally uses array .If any elements are removed from the array,all the bits are shifted from the memory.
* Arraylist can act only as list because it implements from list interface.
* Arraylist is best for accessing and storing the data.

Linkedlist:

* Linkedlist internally uses doubly linkedlist to store the elements.
* Manipulation in linkedlist is fast because it uses doublylinkedlist so no more bits are to be shifted.
* Linkedlist can act as both list and queue because it implements from both list and deque interfaces.
* Linkedlist is best for manipulating the data.

1. What is the difference between Iterator and ListIterator?

Iterator:

Iterator interface can traverse the elements in forward directions only

It has three methods:

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.*

List iterator:

List iterator can traverse the elements in both forward and backward directions.

By using ListIterator we can perform replacement and addition of new objects in addition to read and remove operations.

Listiterator is a child interface of iterator.

It has following methods:

* Public Boolean hasNext();
* Public object next();
* Public int nextIndex();
* Public Boolean hasPrevious();
* Public Object previous();
* Public int previousIndex();
* Public void remove();
* Public void set(object new);
* Public void add(object new);

1. What is the difference between List and Set?

List:

* It is a child interface of collection.
* List is a type of collection in which we can store duplicate values.
* Null insertion is allowed
* List elements preserves index order by using index.
* We can differentiate duplicates using list

Set:

* It is a child interface of Collection
* Set is a type of collection which can store only unique elements
* Set collection doesn’t allow duplicate value
* Null insertion is allowed
* Set elements are store non linearly without index
* Set elements are retrieve randomly since it is not store with index
* Set it does not contain any new method
* For empty TreeSet as the first element null insertion is possible .But after inserting that null if we are trying to insert any another element we will get NullpointerException.

1. What is the difference between HashSet and TreeSet?

Hashset:

* The underlying data structure is hash table.
* Duplicates are not allowed.
* Null insertion is allowed.
* Doesnot preserves insertion order.
* Heterogeneous objects are accepted.
* Hashset is used for search operation.

Treeset:

* The underlying data structure is binary structure.
* Duplicates are not allowed.
* Doesnot preserves insertion order but arranges in ascending order.
* Null insertion is allowed only once.
* Heterogeneous objects are not allowed.

1. What is the difference between HashSet and HashMap?

Hashset:

* In hashset we store objects.
* HashSet is a Set. It creates a collection that uses a hash table for storage.
* Single null value is allowed.
* Hashset Implements cloneable,serializable,set,collection and iterable.
* Hashset internally uses haspmap for storing objects
* Hashset uses add() method to add the elements
* It is used when we need to maintain the uniqueness of the data.

Hashmap:

* In hashmap we store key value pair
* Hashmap is a map. It is a hashtable based implementation of map interface.
* Single null key is allowed and multiple null values are allowed.
* Hashmap implements cloneable,serializable,map interface.
* Hashmap internally uses hashing for storing objects.
* Hashmap uses put() method to add the elements.
* It is used when we neednot to maintain the uniqueness of the data.

1. What is the difference between HashMap and Hashtable?

Haspmap:

* It is non-synchronized. It is not-thread safe and can't be shared between many threads without proper synchronization code.
* Hashmap allows one null key and multiple null values.
* Hashmap is fast
* Hashmap is traversed by iterator
* Iterator in hashmap is failfast
* It inherits abstractmap class
* It is introduced in java 1.2 version.

Hashtable:

* It is synchronized .it is thread safe and can be shared between many threads with proper synchronization code.
* Hashtable doesnot allow any null values.
* Hashtable is slow
* It is traversed by enumerator and iterator
* Iterator in hashtable is not fail fast.
* It inherits dictionary class.
* It is a legacy class .

1. What is the difference between Comparable and Comparator?

Comparable:

* Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.
* It affects the actual class i.e. the original class is modified
* Comparable provides compareTo() method to sort elements.
* It is present in java.lang package
* We can sort the elements of comparable type by collections.sort(list) method.

Comparator:

* Comparator provides a multiple sorting sequence.
* It does not affect the actual class i.e. the original class is not modified.
* Comparator provides compare() method to sort the elements.
* It is present in java.util package.
* We can sort the elements of comparator type by Collections.sort(list,comparator) method.

1. How to synchronize List, Set and Map elements?

1.Synchronizing list elements:

**package** com.te.collections.arraylist;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Iterator;

**import** java.util.List;

**public** **class** SynchronizeArray {

**public** **static** **void** main(String[] args) {

List<String> list = **new** ArrayList<String>();

list.add("synchronizing");

list.add("the");

list.add("list");

list.add("elements");

list.add(".");

list = Collections.*synchronizedList*(list);

**synchronized** (list) {

Iterator<String> itr = list.iterator();

**while** (itr.hasNext()) {

System.***out***.print(itr.next()+" ");

}

}

}

}

2.Synchronizing set elements:

**package** com.te.collections.arraylist;

**import** java.util.\*;

**import** java.util.Collections;

**public** **class** SynchronizeSet {

**public** **static** **void** main(String[] args) {

Set<Integer> set =Collections.*synchronizedSet*( **new** HashSet<Integer>());

set.add(10);

set.add(12);

set.add(21);

set.add(11);

set.add(16);

**synchronized** (set) {

Iterator<Integer> itr = set.iterator();

**while** (itr.hasNext()) {

System.***out***.println( itr.next());

}

}

}

}

3.synchronizing map elements:

**package** com.te.collections.arraylist;

**import** java.util.\*;

**public** **class** SynchronizeSet {

**public** **static** **void** main(String[] args) {

Map<Integer, String> map = Collections.*synchronizedMap*(**new** TreeMap<Integer, String>());

map.put(1, "hello");

map.put(2, "all");

map.put(3, "bye");

System.***out***.println(map);

}

}

1. What do you understand by fail-fast?

The Fail fast iterator aborts the operation as soon it exposes failures and stops the entire operation. Comparatively, Fail Safe iterator doesn't abort the operation in case of a failure. Instead, it tries to avoid failures as much as possible.

The Fail Fast iterators immediately throw ConcurrentModificationException in case of structural modification of the collection. Structural modification means adding, removing, updating the value of an element in a data collection while another thread is iterating over that collection. Some examples of Fail Fast iterator are iterator on ArrayList, HashMap collection classes.

1. What is the difference between Array and ArrayList?

Array:

* Arrays are collection of homogeneous data.
* Array can hold both object and primitive data types.
* Arrays are fixed in size i.e. we cannot able to increase or decrease the size of an array.
* Arrays doesn’t have an underlying data structures and inbuilt methods are not available.

Arraylist:

* Arraylist can have both homogeneous and heterogeneous data.
* Arraylist can hold only object types but primitive.
* arraylist are grow able in nature i.e. we can increase or decrease the size of the object.
* Arraylist have underlying data structure for every classes and inbuilt methods are present which we can use directly without implementing it.

1. How to remove duplicates from ArrayList?

Removing duplicates from arraylist:

**package** com.te.collections.arraylist;

**import** java.util.\*;

**public** **class** RemoveDuplicates {

**public** **static** ArrayList<Integer> removeDuplicates(ArrayList<Integer> list){

ArrayList<Integer> list2 = **new** ArrayList<Integer>();

**for** (Integer integer : list) {

**if**(!list2.contains(integer)) {

list2.add(integer);

}

}

**return** list2;

}

**public** **static** **void** main(String[] args) {

ArrayList<Integer> list = **new** ArrayList<Integer>();

list.add(10);

list.add(12);

list.add(12);

list.add(15);

list.add(10);

list.add(19);

System.***out***.println(list);

ArrayList<Integer> list2 = *removeDuplicates*(list);

System.***out***.println(list2);

}

}

1. Write a Java program to copy one array list into another.

Copy arraylist:

**package** com.te.collections.arraylist;

**import** java.util.ArrayList;

**public** **class** CopyArrayList {

**public** **static** **void** main(String[] args) {

ArrayList<Integer> list = **new** ArrayList<Integer>();

list.add(10);

list.add(12);

list.add(12);

list.add(1);

list.add(5);

System.***out***.println(list);

ArrayList<Integer> list2 = **new** ArrayList<Integer>();

list2.addAll(list);

System.***out***.println(list2);

}

}

1. Write a Java program of swap two elements in an array list.

Swaping in list:

**package** com.te.collections.arraylist;

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** SwapList {

**public** **static** **void** main(String[] args) {

ArrayList<Integer> list = **new** ArrayList<Integer>();

list.add(12);

list.add(11);

list.add(10);

list.add(9);

System.***out***.println("Before swaping");

System.***out***.println(list);

System.***out***.println("after swaping");

Collections.*swap*(list, 1, 3);

System.***out***.println(list);

}

}

1. Write a Java program to iterate through all elements in a linked list starting at the specified position.

Iteration in specified position:

**package** com.te.collections.arraylist;

**import** java.util.Iterator;

**import** java.util.LinkedList;

**import** java.util.ListIterator;

**public** **class** LinkedListIteration {

**public** **static** **void** main(String[] args) {

LinkedList<Integer> list = **new** LinkedList<Integer>();

list.add(12);

list.add(10);

list.add(15);

list.add(14);

list.add(19);

Iterator<Integer> itr = list.iterator();

**while** (itr.hasNext()) {

Integer integer = (Integer) itr.next();

System.***out***.print(integer+" ");

}

System.***out***.println();

ListIterator<Integer> itr2 = list.listIterator(0);

**while** (itr2.hasNext()) {

System.***out***.print(itr2.next()+" ");

}

System.***out***.println();

ListIterator<Integer> itr3 = list.listIterator(list.size()-2);

**while** (itr3.hasPrevious()) {

System.***out***.print(itr3.previous()+" ");

}

}

}

1. Write a Java program to get the first and last occurrence of the specified elements in a linked list.

**package** com.te.collections;

**import** java.util.\*;

**public** **class** LinkedListRetrive {

**public** **static** **void** main(String[] args) {

java.util.LinkedList<Integer> list = **new** LinkedList<Integer>();

list.add(12);

list.add(11);

list.add(10);

list.add(15);

list.add(16);

System.***out***.println(list);

System.***out***.println("the first occurence " +list.getFirst());

System.***out***.println("the last occurence " +list.getLast());

}

}

1. Write a Java program to retrieve but does not remove, the first element of a linked list.

**package** com.te.collections;

**import** java.util.\*;

**public** **class** LinkedListRetrive {

**public** **static** **void** main(String[] args) {

java.util.LinkedList<Integer> list = **new** LinkedList<Integer>();

list.add(12);

list.add(11);

list.add(10);

list.add(15);

list.add(16);

System.***out***.println(list);

System.***out***.println(list.peekFirst());

System.***out***.println(list.peekLast());

}

}

1. Write a Java program to convert a linked list to array list.

**package** com.te.collections;

**import** java.util.\*;

**public** **class** ListConversion {

**public** **static** **void** main(String[] args) {

LinkedList<String> list1 = **new** LinkedList<String>();

list1.add("converting");

list1.add("LinkedList");

list1.add("to");

list1.add("arraylist");

System.***out***.println(list1);

ArrayList<String> list2 = **new** ArrayList<String>();

list2.addAll(list1);

Iterator<String> itr = list1.iterator();

**while** (itr.hasNext()) {

String string = (String) itr.next();

System.***out***.print(string+" ");

}

}

}