Assignment-12

1.Why collection framework in java

* Reduces programming effort: By providing useful data structures and algorithms, the Collections Framework frees you to concentrate on the important parts of your program rather than on the low-level "plumbing" required to make it work. By facilitating interoperability among unrelated APIs, the Java Collections Framework frees you from writing adapter objects or conversion code to connect APIs.

* Increases program speed and quality: This Collections Framework provides high-performance, high-quality implementations of useful data structures and algorithms. The various implementations of each interface are interchangeable, so programs can be easily tuned by switching collection implementations. Because you're freed from the drudgery of writing your own data structures, you'll have more time to devote to improving programs' quality and performance.

* Allows interoperability among unrelated APIs: The collection interfaces are the vernacular by which APIs pass collections back and forth. If my network administration API furnishes a collection of node names and if your GUI toolkit expects a collection of column headings, our APIs will interoperate seamlessly, even though they were written independently.

* Reduces effort to learn and to use new APIs: Many APIs naturally take collections on input and furnish them as output. In the past, each such API had a small sub-API devoted to manipulating its collections. There was little consistency among these ad hoc collections sub-APIs, so you had to learn each one from scratch, and it was easy to make mistakes when using them. With the advent of standard collection interfaces, the problem went away.

* Reduces effort to design new APIs: This is the flip side of the previous advantage. Designers and implementers don't have to reinvent the wheel each time they create an API that relies on collections; instead, they can use standard collection interfaces.

* Fosters software reuse: New data structures that conform to the standard collection interfaces are by nature reusable. The same goes for new algorithms that operate on objects that implement these interfaces.

2.What is Collection interface?

The Collection interface is the root interface of the Java collections framework.

There is no direct implementation of this interface. However, it is implemented through its subinterfaces like List, Set, and Queue.

Assignment-12

3.what is package of collection framework?

java.util package

4.which is root interface of Collection Framework?

util. Collection is the root interface of Collections Framework. It is on the top of the Collections framework hierarchy.

5 List the subinterface of Collection interface.

List

Set

Queue

Sortedset

Deque

6.List out the classes which are implementing the List interface and Set Interface and Collection interface.

List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

1. List <data-type> list1= new ArrayList();

2. List <data-type> list2 = new LinkedList();

3. List <data-type> list3 = new Vector();

4. List <data-type> list4 = new Stack();

Set is implemented by HashSet, LinkedHashSet, and TreeSet.

1. Set<data-type> s1 = new HashSet<data-type>();

2. Set<data-type> s2 = new LinkedHashSet<data-type>();

3. Set<data-type> s3 = new TreeSet<data-type>();

The Collection interface is the interface which is implemented by all the classes in the collection framework. It declares the methods that every collection will have.

Assignment-12

7.Write a small program for adding one integer, float, double, char, string, short, boolean, byte object to list and set interface.

```
package listinterfacepoack;

	import java.util.ArrayList;

	import java.util.List;

public class Ques7 {



	public static void main(String[] args) {

	// TODO Auto-generated method stub

	List<Integer> intList = new ArrayList<Integer>();

	intList .add(1);

	intList .add(2);

	intList .add(3);

	System.out.println(intList );

	List<Float> floatList = new ArrayList<>();

	floatList.add(1.53f);

	floatList.add(2.35f);

	System.out.println(floatList );

	List<Short> shortList = new ArrayList<>();

	shortList.add((short) 2);

	shortList.add((short) 2);

	System.out.println(shortList );

	List<String> stringList = new ArrayList<>();

	stringList.add("Apoorva");

	stringList.add("Sajjan");

	System.out.println(stringList );
```

```
List<Double> doubleList = new ArrayList<>();
        doubleList.add(1.5334);
        doubleList.add(2.35567);
List<Char> charList = new ArrayList<>();
        charList.add('c');
        charList.add('A');

        }
```

9. Diffence between the List and Set

List	Set
1. The List is an indexed sequence.	1. The Set is an non-indexed sequence.
2. List allows duplicate elements	2. Set doesn't allow duplicate elements.
3. Elements by their position can be accessed.	3. Position access to elements is not allowed.
4. Multiple null elements can be stored.	4. Null element can store only once.
5. List implementations are ArrayList, LinkedList, Vector, Stack	5. Set implementations are HashSet, LinkedHashSet.

10. Diffeence between ArrayList and LinkedList

ArrayList	LinkedList
1) ArrayList internally uses a **dynamic array** to store the elements.	LinkedList internally uses a **doubly linked list** to store the elements.
2) Manipulation with ArrayList is **slow** because it internally uses an array. If any element is removed from the array, all the other elements are shifted in memory.	Manipulation with LinkedList is **faster** than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
3) An ArrayList class can **act as a list** only because it implements List only.	LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces.
4) ArrayList is **better for storing and accessing** data.	LinkedList is **better for manipulating** data.
5) The memory location for the elements of an ArrayList is contiguous.	The location for the elements of a linked list is not contagious.
6) Generally, when an ArrayList is initialized, a default capacity of 10 is assigned to the ArrayList.	There is no case of default capacity in a LinkedList. In LinkedList, an empty list is created when a LinkedList is initialized.
7) To be precise, an ArrayList is a resizable array.	LinkedList implements the doubly linked list of the list interface.

11. Difference between HashMap and HashSet

Sr. No.	Key	HashMap	HashSet
1	Implementation	Hashmap is the implementation of Map interface.	Hashset on other hand is the implementation of set interface.
2	Internal implementation	Hashmap internally do not implements hashset or any set for its implementation.	Hashset internally uses Hashmap for its implementation.

Sr. No.	Key	HashMap	HashSet
3	Storage of elements	HashMap Stores elements in form of key-value pair i.e each element has its corresponding key which is required for its retrieval during iteration.	HashSet stores only objects no such key value pairs maintained.
4	Method to add element	Put method of hash map is used to add element in hashmap.	On other hand add method of hashset is used to add element in hashset.
5	Index performance	Hashmap due to its unique key is faster in retrieval of element during its iteration.	HashSet is completely based on object so compared to hashmap is slower.
6	Null Allowed	Single null key and any number of null value can be inserted in hashmap without any restriction.	On other hand Hashset allows only one null value in its collection,after which no null value is allowed to be added.

12. Difference between the Iterator and ListIterator

Iterator	ListIterator
Can traverse elements present in Collection only in the forward direction.	Can traverse elements present in Collection both in forward and backward directions.
Helps to traverse Map, List and Set.	Can only traverse List and not the other two.
Indexes cannot be obtained by using Iterator.	It has methods like nextIndex() and previousIndex() to obtain indexes of elements at any time while traversing List.

Assignment-12

Iterator	ListIterator
Cannot modify or replace elements present in Collection	We can modify or replace elements with the help of set(E e)
Cannot add elements and it throws ConcurrentModificationException.	Can easily add elements to a collection at any time.
Certain methods of Iterator are next(), remove() and hasNext().	Certain methods of ListIterator are next(), previous(), hasNext(), hasPrevious(), add(E e).

13.Write a program to write employee object to the file and read it.

```
package listinterfacepoack;

import java.io.FileReader;

import java.io.IOException;

public class Ques13 {

public static void main(String[] args) throws IOException {

// TODO Auto-generated method stub

FileReader reader =new FileReader("Employee.txt");

int res=0;

do {

res=reader.read();

System.out.print((char)res+" ");

}while(res!=-1);

}

}
```

Assignment-12

14.Write a program to write employee array of objects to the file and read it.

```
package listinterfacepoack;

import java.io.FileWriter;

import java.io.IOException;

public class WriteEmployeeEx {

public static void main(String[] args) {

// TODO Auto-generated method stub

try {

FileWriter fileWriter=new FileWriter("Employee.txt");

fileWriter.write("Hello world");

fileWriter.close();

}catch(IOException ex)

{

System.out.println("---->"+ex.getMessage());

}

}

}
```

```
package listinterfacepoack;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

public class WriteExArrayFile {

public static void main(String[] args)throws IOException {

// TODO Auto-generated method stub

FileInputStream fileInputStream=new FileInputStream("Employee.txt");
```

Assignment-12

```
byte[] array=new byte[20];

fileInputStream.read(array);

for(byte b:array)

{

System.out.print((char)b);

}

fileInputStream.close();

}

}
```