Assignment No 12

1. Why collection framework in java.

Ans - Collection framework is a powerful framework in java. This framework defines the most common methods that can be used for any collection of objects.

- **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.
- **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.
- **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.

2. What is Collection interface?

Ans – The Collection interface is a member of the Java Collections Framework. It is a part of java.util package. 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 it's subinterfaces like List, Set, and Queue.

3. What is package of collection framework?

Ans - java.util package

4. Which is root interface of Collection Framework?

Ans - 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.

Ans – List, Set, Queue, Sortedset, Deque

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

Ans –

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.

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

Ans –

```
import java.util.ArrayList;
import java.util.List;
public class AddingList
{
	public static void main(String[] args)
	{
		// TODO Auto-generated method stub
		List<Integer> intList = new ArrayList<Integer>();
		intList.add(32);
		intList.add(23);
		System.out.println(intList);

		List<Float> floatList = new ArrayList<>();
		floatList.add(1.73f);
		floatList.add(2.56f);
		System.out.println(floatList );

		List<Short> shortList = new ArrayList<>();
		shortList.add((short) 1);
		shortList.add((short) 1);
		System.out.println(shortList );

		List<String> stringList = new ArrayList<>();
		stringList.add("ABC");
		stringList.add("DEF");
		System.out.println(stringList );
	}
}
```

8. Difference between the List and Set.

Ans -

List	Set
1. The list is an index sequence.	1. The set non-index sequence
2. It allow duplicate elements	2. It doesn't allow duplicate elements
3. Element by their position can be accessed	3. Position access to elements is not allowed
4. List implementations are ArrayList, LinkedList, Vector, Stack	4. Set implementations are HashSet, LinkedHashset
5. Syntax :- List = [1, 2, 3, 4, 5]	5. Syntax :- Set = [6, 7, 8, 9]

9. Difference between ArrayList and LinkedList

Ans –

ArrayList	LinkedList
1.ArrayList internally uses dynamic array to store the elements.	1.LinkedList internally uses doubly linked list to store the elements.
2. An ArrayList class can act as a list only because it implements List only.	2. LinkedList class can act as a list and queue both because it implements List and Deque interfaces.
3. The memory location for the elements of an arrayList is contiguous.	3. The location for the elements of a linked list is not contagious.
4. Generally, when an ArrayList is initialized, a default capacity of 10 is assigned to the ArrayList.	4. There is no case of default capacity in a LinkedList. In LinkedList, an empty list is created when a LinkedList is initialized.
5. To be precise, an ArrayList is a resizable array.	5. LinkedList implements the doubly linked list of the list interface.

10. Difference between Hashmap and Hashset.

Ans -

Hashmap	Hashset
1. Hashmap is the implementation of map interface.	1. Hashset is the implementation of set interface.
2. Hashmap internally do not implements hashset or any set for its implementation.	2. Hashset internally uses Hashmap for its implementation.
3. 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.	3. HashSet stores only objects no such key value pairs maintained.
4. Put method of hash map is used to add element in hashmap.	4. On other hand add method of hashset is used to add element in hashset.
5. Hashmap due to its unique key is faster in retrieval of element during its iteration.	5. HashSet is completely based on object so compared to hashmap is slower.

11. Difference between `Iterator` and `ListIterator`.

Ans -

Iterator	ListIterator
1. Iterator can traverse elements present in Collection only in the forward direction.	1. ListIterator can traverse elements present in Collection both in forward and backward directions.
2. Indexes cannot be obtained by using Iterator.	2. It has methods like nextIndex() and previousIndex() to obtain indexes of elements at any time while traversing List
3. Cannot modify or replace elements present in Collection	3. We can modify or replace elements with the help of set(E e)
4. Certain methods of Iterator are next(), remove() and hasNext().	4. Certain methods of ListIterator are next(), previous(), hasNext(), hasPrevious(), add(E e).

5. Helps to traverse Map, List and Set.	5. Can only traverse List and not the other two

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

Ans –

```
import java.io.FileReader;
import java.io.IOException;
public class EmployeeObj
{
	public static void main(String[] args) throws IOException
	{
	FileReader reader =new FileReader("Employee.txt");
	int res=0;
	do {
	res=reader.read();
	System.out.print((char)res+" ");
	}while(res!=-1);
	}
}
```

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

Ans -

```
import java.io.FileWriter;
import java.io.IOException;

public class WriteEmpEx
{
	public static void main(String[] args) throws IOException
	{
		FileWriter fw = new FileWriter("Emp.txt");
		fw.write("Hellow World");
		fw.close();
	}
}
```

```
import java.io.FileInputStream;
import java.io.IOException;

public class WriteExArrayFile
{
	public static void main(String[] args) throws IOException
	{
		FileInputStream fileInputStream=new FileInputStream("Employee.txt");
		byte[] array=new byte[20];
		fileInputStream.read(array);
		for(byte b:array)
			{
			System.out.print((char)b);
			}
		fileInputStream.close();
}
}
```