Assignment-13

1.Why collection framework in java Ans:

*The Collection interface is a member of the Java Collections Framework. It is a part of java. util package. It is one of the root interfaces of the Collection Hierarchy.

Collection Interfaces

- util. Set.
- util. SortedSet.
- util. NavigableSet.
- util. Queue.
- util. concurrent. BlockingQueue.
- util. concurrent. TransferQueue.
- util. Deque.
- util. concurrent. BlockingDeque.

*The collection framework also provides interfaces and classes with defined methods to perform these functions. Inside the Collection framework is the Collection interface, Map interface and Iterable interface. The Collection interface has three more interfaces, List, Set and Queue.

*The Collection interface is used to pass around collections of objects where maximum generality is desired. For example, by convention all general-purpose collection implementations have a constructor that takes a Collection argument

*The Collection interface is the least common denominator that all collections implement and is used to pass collections around and to manipulate them when maximum generality is desired. Some types of collections allow duplicate elements, and others do not. Some are ordered and others are unordered. *Reduces programming effort by providing data structures and algorithms so you don't have to write them yourself. Increases performance by providing highperformance implementations of data structures and algorithms

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**.
* It is one of the root interfaces of the Collection Hierarchy.
*The Collection interface is not directly implemented by any class.
*However, it is implemented indirectly via its subtypes or subinterfaces like List, Queue, and Set.

3 .what is package of collection framework?

Ans:

The java.util package contains all the classes and interfaces for the Collection framework.

4.which is root interface of Collection Framework?

Ans:

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

interface is a root interface for all collection classes:

*The Iterable interface is the root interface for all the collection classes.
*The Collection interface extends the Iterable interface and therefore all the subclasses of Collection interface also implement the Iterable interface.
*It contains only one abstract method.

5.List the subinterface of Collection interface.

Ans:

- The sub-interfaces of Collection are BeanContext, BeanContextServices, BlockingDeque<E>,
- BlockingQueue<E>,
- Deque<E>,

- EventSet, List<E>,
- NavigableSet<E>,
- Queue<E>,
- Set<E>,
- SortedSet<E>, • TransferQueue<E> .

Set: A set is an unordered collection of objects in which duplicate values cannot be stored.

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

Ans:

List interface is the child interface of Collection interface.

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

To instantiate the List interface, we must use :

- List <data-type> list1= new ArrayList();
- List <data-type> list2 = new LinkedList();
- List <data-type> list3 = new Vector();
- List <data-type> list4 = new Stack();

Queue interface maintains the first-in-first-out order. It can be defined as an ordered list that is used to hold the elements which are about to be processed. There are various classes like PriorityQueue, Deque, and ArrayDeque which implements the Queue interface.

Queue interface can be instantiated as:

- Queue<String> q1 = new PriorityQueue();
- Queue<String> q2 = new ArrayDeque();

Set Interface in Java is present in java.util package. It extends the Collection interface. It represents the unordered set of elements which doesn't allow us to

store the duplicate items. We can store at most one null value in Set. Set is implemented by HashSet, LinkedHashSet, and TreeSet.

Set can be instantiated as:

- Set<data-type> s1 = new HashSet<data-type>();
- Set<data-type> s2 = new LinkedHashSet<data-type>();
- Set<data-type> s3 = new TreeSet<data-type>();

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

Ans:

```
package assignment13;

import java.util.ArrayList;

import java.util.Collections;

public class AddList {

public static void main(String[] args) {

ArrayList<Integer> list1 = new
ArrayList<Integer>();

Collections.addAll(list1, 11, 12, 1, 13, 14,
15);

System.out.println("IntegerList: " + list1);

ArrayList<Float> list2 = new
ArrayList<Float>();
```

```
Collections.addAll(list2, 11.2f, 12.3f, 13.6f,
14.5f, 15.9f);

System.out.println("FloatList: " + list2);

ArrayList<Byte> list3 = new ArrayList<Byte>();

list3.add((byte) 23);

list3.add((byte) 24);

list3.add((byte) 25);

list3.add((byte) 26);

System.out.println("ByteList: " + list3);

ArrayList<Short> list4 = new
ArrayList<Short>();

list4.add((short) 12);

list4.add((short) 13);

list4.add((short) 14);

list4.add((short) 15);

System.out.println("ShortList: " + list4);

ArrayList<Double> list5 = new
ArrayList<Double>();

Collections.addAll(list5, 11.3, 12.2, 1.3,
13.14, 15.9);
```

```
System.out.println("DoubleList: " + list5);

ArrayList<String> list6 = new
ArrayList<String>();

list6.add("Vijay");

list6.add("Ragul");

list6.add("Aswin");

list6.add("Dhanush");

System.out.println("StringList: " +
list6.toString());

ArrayList<Boolean> list7 = new
ArrayList<Boolean>();

list7.add(true);

list7.add(false);

System.out.println("BooleanList: " + list7);

}

}
```

OUTPUT:

```
IntegerList: [11, 12, 1, 13, 14, 15]
```

```
FloatList: [11.2, 12.3, 13.6, 14.5, 15.9]
ByteList: [23, 24, 25, 26]
ShortList: [12, 13, 14, 15]
DoubleList: [11.3, 12.2, 1.3, 13.14, 15.9]
StringList: [Vijay, Ragul, Aswin, Dhanush]
BooleanList: [true, false]
```

8.Write a program to read and print the file properties like name of the file, size,author,dateofcreation and dateofupdation using Properties class of collection framework.

```
package assignment13;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Properties;
public class FileRead {
public static void main(String[] args) {
// TODO Auto-generated method stub
```

```
Properties p=new Properties();

try {

FileReader reader=new
FileReader("D:\\java\\book\\notss\\Demo.txt");

p.load(reader);

System.out.println(p.getProperty("Name"));

System.out.println(p.getProperty("add"));

System.out.println(p.getProperty("mblNO"));

System.out.println(p.getProperty("Study"));

//System.out.println(p.getProperty("Date_of_upd
ation"));

reader.close();

}catch(FileNotFoundException ex)

{

System.out.println(ex.getMessage());

}catch(IOException ex)

{

System.out.println(ex.getMessage());

}
```

```
}
}
```

OUTPUT:

```
santhanam
tiruppur
9698958057
Msc.cs
```

9. Difference between the List and Set Ans:

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. Difference between Array List and LinkedList

Ans:

Array List	Linked List
1. ArrayList internally uses a dynamic array to store the elements.	1. 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	2. 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.	3. 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.	4. LinkedList is better for manipulating data.
5. The memory location for the elements of an ArrayList is contiguous.	5. 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.	6. 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.	7. LinkedList implements the doubly linked list of the list interface.

11. Difference between HashMap and HashSet

Ans:

SI.No	Key	Hash Map	Hash Set

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.
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	On other hand Hashset allows only one null value in its collection,after which no null value is allowed to be added.	Single null key and any number of null value can be inserted in hashmap without any restriction.

12. Difference between the Iterator and ListIterator

Ans:

Iterator	List Iterator

1. An Iterator is an interface in Java and we can traverse the elements of a list in a forward direction	**1.** ListIterator is an interface that extends the Iterator interface and we can traverse the elements in both forward and backward direction
2. An Iterator can be used in these collection types like List, Set, and Queue	**2.** ListIterator can be used in List collection only
3. The important methods of Iterator interface are hasNext(), next() and remove()	**3.** ListIterator interface are add(), hasNext(), hasPrevious() and remove().