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. But the question arises that we have an array concept in java then why we need collection framework in java? Now let's see that why we need collection framework in java with some valid points of difference between array and collection.
2.What is Collection interface?
Ans:
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.
3.what is package of collection framework?
Ans:
Java.util package.
4.which is root interface of Collection Framework?
util. Collection is the root interface of Collections Framework.
5.List the subinterface of Collection interface.
Subinterfaces are—
Set
List
Queue
6.List out the classes which are implementing the List interface and Set Interface and Collection interface.
The classes which are implementing the List interface and Set Interface
(Array List, Vector, Liked List, Priority Queue, HashSet, LinkedHashSet, TreeSet).

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

```
package collectionframework;

import java.util.ArrayList;

import java.util.List;

public class ListInterface {

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(1.23f);

floatList.add(2.356f);

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("Priyanka");

stringList.add("Ray");

System.out.println(stringList );

}

}
```

9. Diffence between the List and Set
Ans:
List

1. The List is an indexed sequence.
2. 2. List allows duplicate elements
3. Elements by their position can be accessed.
4. Multiple null elements can be stored.
5. List implementations are ArrayList, LinkedList, Vector, Stack
6. Syntax: List = [5, 6, 3, 5, 4]

Set:

1. The Set is an non-indexed sequence.
2. Set doesn't allow duplicate elements.
3. Position access to elements is not allowed.
4. Null element can store only once.
5. Set implementations are HashSet, LinkedHashSet.
6. Syntax: Set = [3, 4, 5, 6]

10. Diffeence between ArrayList and LinkedList
Ans:
ArrayList

1. ArrayList internally uses a **dynamic array** 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.
3. An ArrayList class can **act as a list** only because it implements List only.
4. ArrayList is **better for storing and accessing** data.
5. The memory location for the elements of an ArrayList is contiguous.

LinkedList

1. LinkedList internally uses a **doubly linked list** to store the elements.
2. Manipulation with LinkedList is **faster** than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
3. LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces.
4. LinkedList is **better for manipulating** data.
5. The location for the elements of a linked list is not contagious.

11. Difference between HashMap and HashSet

Ans:

Hashmap

1. Java HashMap is a hash table based implementation of Map interface.
2. HashMap implements **Map, Cloneable, and Serializable** interface es.
3. In HashMap we store a **key-value pair**. It maintains the mapping of key and value.
4. It does not allow **duplicate keys**, but **duplicate values** are **allowed**.
5. It can contain a **single null key** and **multiple null values**.
6. HashMap uses the **put()** method to add the elements in the HashMap.
7. Example: **{a->4, b->9, c->5}** Where **a, b, c** are **keys** and **4, 9, 5** are **values** associated with key.

HasSet

1. HashSet is a Set. It creates a collection that uses a hash table for storage.
2. HashSet implements **Set, Cloneable, Serializable, Iterable** and **Collection** interfaces.
3. In HashSet, we store **objects**.
4. It does not allow **duplicate values**.
5. It can contain **a single null value**.
6. HashSet uses the **add()** method to add elements in the HashSet.
7. Example: **{6, 43, 2, 90, 4}** It denotes a set.

12. Difference between the Iterator and ListIterator

Ans:

Iterator

1. It helps traverse through a map, list and a set.
2. Index can't be obtained with the help of an iterator.
3. The iterator can't modify or replace elements of a Collection.
4. It traverses through elements present in Collection.
5. This iteration can be done in the forward direction only.
6. Elements can't be added, since it would throw ConcurrentModificationException.
7. Methods of the iterator are 'next()', 'remove()', 'hasNext()'.

ListIterator

1. It helps traverse through a list only.
2. It can't traverse through a map and a set.

3. It can traverse through the elements present in Collection.
4. The traversal can be done in both forward and backward directions.
5. Some of the methods of listiterator are ‘nextIndex()’, ‘previousIndex()’, ‘previous()’, ‘next()’.
6. The elements can be modified or replaced.
7. Elements can be added to a collection anytime.

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

```
package collectionframework;

import java.io.FileWriter;

import java.io.IOException;

public class WriteEmployeeFileAndRead {

public static void main(String[] args) {

// TODO Auto-generated method stub

try {

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

fileWriter.write("Hello Priyanka");

fileWriter.close();

}catch(IOException ex)

{

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

}

}

}
```

```
package collectionframework;

import java.io.FileReader;

import java.io.IOException;

public class ReadEmployeeFile {
```

```
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);
/*catch(IOException ex)
{
System.out.println("--->"+ex.getMessage());
}*/
}
}
```

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

```
package collectionframework;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class WriteEmpolyeeArrayFile {
public static void main(String[] args)throws IOException {
// TODO Auto-generated method stub
FileInputStream fileInputStream=new FileInputStream("Employee.txt");
byte[] array=new byte[20];
```

```
fileInputStream.read(array);

for(byte b:array)

{

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

}

fileInputStream.close();

}

}
```