Assignment-12

Part -A

Que-1)What is LinkedList?

LinkedList is the implementation class of List and Deque interfaces, present in the java.util package.
It is used to store a group of objects where insertion order is maintained and duplicate elements are allowed.
The underlying data structure of LinkedList is a doubly linked list.
It is the best choice when frequent insertion and deletion operations are required.

Que-2)Difference between ArrayList and LinkedList.

ArrayList - ArrayList uses a dynamic array to store elements.
In ArrayList, elements are stored in contiguous memory locations, which allows fast random access with O(1) time complexity.
ArrayList consumes less memory since it stores only data.
ArrayList is not best choice if our frequent operation are insertion and deletion.

Insertion and deletion operations are slower in ArrayList because elements need to be shifted, whereas they are faster in LinkedList since only pointers are updated
LinkedList - LinkedList uses a doubly linked list
LinkedList does not support fast random access and requires O(n) time to access an element because traversal is needed.
LinkedList consumes more memory because each node stores data along with previous and next references.
LinkedList is best choice if our frequent operation are insertion and deletion.

Que-3)What is a HashSet?

HashSet is an implementation class of the Set interface, present in the java.util package.
It is used to store a group of objects where duplicate elements are not allowed and insertion order is not maintained.
The underlying data structure of HashSet is HashMap, which uses hashing.
he default capacity of a HashSet is 16 and the load factor is 0.75.
HashSet is the best choice when frequent searching, insertion, and deletion operations are required.
HashSet implements Serializable and Cloneable interfaces.
HashSet is not Thread safe, to make thread use Collection.synchronizedSet().

Que-4)What is LinkedHashSet?

LinkedHashSet is an implementation class of the Set interface, present in the java.util package.
It is used to store a group of objects where duplicate elements are not allowed and insertion order is maintained.
The underlying data structure of LinkedHashSet is LinkedHashMap, which maintains a doubly linked list along with hashing.
The default initial capacity is 16 and the load factor is 0.75.
LinkedHashSet is the best choice when uniqueness and insertion order must be preserved, with efficient searching operations.
LinkedHashSet implements Serializable and Cloneable interfaces.
LinkedHashSet is not thread-safe; to make it thread-safe, we use Collections.synchronizedSet().

Que-5)What is TreeSet?

TreeSet is an implementation class of the NavigableSet (and SortedSet) interface, present in the java.util package.
It is used to store a group of objects where duplicate elements are not allowed and elements are stored in sorted order (natural order by default).
Insertion order is not maintained.

The underlying data structure of TreeSet is a Red-Black Tree.
TreeSet uses Comparable or Comparator for sorting and implements Serializable and Cloneable interfaces.
TreeSet is the best choice when sorted data is required.

Part -B

Que-1)Write a program to reverse a LinkedList.

```
import java.util.Collections;
import java.util.LinkedList;
import java.util.Scanner;

public class ReverseLinkedList{
public static void main(String[] args) {
LinkedList<Integer> ll = new LinkedList<Integer>();
Scanner sc = new Scanner(System.in);

System.out.println("Enter a N :");
int n = sc.nextInt();

System.out.println("Enter Elements :");
for (int i = 0; i < n; i++) {
ll.add(sc.nextInt());
}
Collections.reverse(ll);
System.out.println("Reversed Linked List is : " + ll);
}
}
```

Output -

```
Enter a N :
5
Enter Elements :
1 2 3 4 5
Reversed Linked List is : [5, 4, 3, 2, 1]
```

Que-2)Write a program to iterate ArrayList using Iterator.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class IteratorArray {
public static void main(String[] args) {
ArrayList<Integer> arr = new ArrayList<Integer>();
arr.addAll(List.of(1, 2, 3, 4, 5, 6, 7, 8, 9, 10));

Iterator<Integer> itr = arr.iterator();
System.out.println("Given data is : ");
while (itr.hasNext()) {
System.out.print(itr.next() + " ");
}

}
}
```

Output -

Given data is :
1 2 3 4 5 6 7 8 9 10

Que-3)Write a program to sort an ArrayList of integers.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.Scanner;

public class SortArrayList {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
ArrayList<Integer> arr = new ArrayList<Integer>();
System.out.println("Enter a N :");
int n = sc.nextInt();

System.out.println("Enter elements :");
for (int i = 0; i < n; i++) {
arr.add(sc.nextInt());
}
Collections.sort(arr);
Iterator<Integer> itr = arr.iterator();
System.out.println("After Sorting Arraylisyt : ");
while (itr.hasNext()) {
System.out.print(itr.next() + " ");
}

}
}
```

Output -

```
Enter elements :
12
32
1
4
32
After Sorting Arraylisyt :
1 4 12 32 32
```

Que-4)Write a program to remove duplicates from ArrayList.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.Scanner;

public class RemoveDuplicateArrayList {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
ArrayList<Integer> arr = new ArrayList<Integer>();
System.out.println("Enter a N :");
int n = sc.nextInt();

System.out.println("Enter elements :");
for (int i = 0; i < n; i++) {
arr.add(sc.nextInt());
}
```

```
LinkedHashSet lh = new LinkedHashSet(arr);
arr.clear();
arr.addAll(lh);

Iterator<Integer> itr = arr.iterator();
System.out.println("After Removing Duplicate from Arraylisyt : ");
while (itr.hasNext()) {
System.out.print(itr.next() + " ");
}

}
}
```

Output -

```
Enter a N :
5
Enter elements :
1 2 3 2 1
After Removing Duplicate from Arraylisyt :
1 2 3
```

Que-5)Write a program to merge two ArrayLists.

```
import java.util.ArrayList;
import java.util.List;

public class MergeTwoArrayList {
public static void main(String[] args) {
ArrayList<Integer> list1 = new ArrayList<Integer>(List.of(1, 2, 3, 4, 5));
ArrayList<Integer> list2 = new ArrayList<Integer>(List.of(6, 7, 8, 9, 10));

ArrayList<Integer> mergedList = new ArrayList<Integer>();
mergedList.addAll(list1);
mergedList.addAll(list2);

System.out.println("Merged List: " + mergedList);
}
}
```

Output -

```
Merged List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Github Link - https://github.com/YogeshPathade01/Java_Coding_Questtions.git