Day 8

Iterator and Iterable implementation

- LinkedList is a class declared in java.util package.

```
LinkedList<Integer> list  =  new LinkedList<>();
list.addLast(10);
list.addLast(20);
list.addLast(30);
```

- Traversal Method#1

```
Integer element = null;
Iterator<Integer> itr = list.iterator();
while( itr.hasNext())
{
    element = itr.next();
    System.out.println(element);
}
```

- Traversal Method#2

```
for( Integer element : list )
    System.out.println(element);
```

- Using foreach loop we can traverse elements of array and any instance whose type implements java.lang.Iterable interface.
- Iterable is a interface declared in java.lang package.
- Methods of Iterable interface

1. Iterator iterator()
2. default Spliterator spliterator()
3. default void forEach(Consumer<? super T> action)

- Implementing this interface allows an object to be the target of the "for-each loop" statement.
- iterator() is method of java.lang.Iterable interface which returns reference of java.util.Iterator interface.
- Iterator is a interface declared in java.util package.
- Methods of Iterator interface

1. boolean hasNext()

2. E next()
3. default void remove()
4. default void forEachRemaining(Consumer<? super E> action)

- If we use object as a pointer then such object is called smart pointer.
- Iterator is smart pointer which is used to traverse collection.

Comparable and Comparator Implementation

- If we want to sort array then we should use "Arrays.sort()" method.

```
int[] arr = { 50, 10, 20, 40, 30 };
System.out.println(Arrays.toString( arr ) );
Arrays.sort(arr);
System.out.println(Arrays.toString( arr ) );
```

- If we use Arrays.sort() method to sort array of value type then it implicitly uses Dual-Pivot Quicksort algorithm.
- Comparable is interface declared in java.lang package.
- "int compareTo(T other)" is a method of Comparable interface.
- If we want to sort array of instances of reference type which is having all the elements of same type then reference type must implement Comparable interface.
- compareTo method returns integer value. if(this < other) return -1; else if(this > other) return 1 else return 0;
- This interface is a member of the Java Collections Framework.
- If any type implements Comparable interface then it is considered as Sortable.
- All the wrapper classes implements Comparable interface.
- If we sort array of instances of reference type using Arrays.sort() then it implicitly uses iterative mergesort algorithm.
- Comparator is interface declared in java.util package.
- "int compare(T o1, T o2)" is a method of Comparator interface.
- If we want to sort array of instances of same / different type then we should use Comparator interface.
- This interface is a member of the Java Collections Framework.
- compare method returns integer value if(o1 < o2) return -1; else if(o1 > o2) return 1; else return 0;

Collection Framework

- Framework is a library of reusable classes that we can use develop application

- AWT, Swing : GUI framework
- RMI : Distributed application development framework
- Struts : MVC based web application framework
- Hibernate : Atomatic persistence framework
- In Java, data structure class is also called as collection.
- Collection framework is a library of reusable data structure classes that we can use to develop any java application.
- If we want to use collection framework then we must import java.util package.

Collection Framework Interface Hierarchy

- java.lang.Iterable
    - java.util.Collection
        - java.util.List
        - java.util.Queue
            - java.util.Deque
        - java.util.Set
            - java.util.SortedSet
                - java.util.NavigableSet

Iterable

- It is a interface declared in java.lang package.
- Implementing this interface allows an object to be the target of the "for-each loop" statement.
- Methods:
    1. Iterator iterator()
    2. default Spliterator spliterator()
    3. default void forEach(Consumer<? super T> action)

Collection

- It is a interaface declared in java.util package.
- It is a root interface in collection framework hierarchy.
- This interface is a member of the Java Collections Framework.
- It is introduced in jdk 1.2
- Abstract Methods of Collection interface.

1. boolean add(E e)
2. boolean addAll(Collection<? extends E> c)
3. void clear()
4. boolean contains(Object o)
5. boolean containsAll(Collection<?> c)
6. boolean isEmpty()
7. boolean remove(Object o)
8. boolean removeAll(Collection<?> c)

9. boolean retainAll(Collection<?> c)
10. int size()
11. Object[] toArray()
12. T[] toArray(T[] a)

- Default Methods of Collection interface.

1. default Stream stream()
2. default Stream parallelStream()
3. default boolean removeIf(Predicate<? super E> filter)

List

- It is sub interface of Collection interface.
- ArrayList, Vector, Stack, LinkedList etc implements List interface. These collections are also called as List collectons.
- List collections are ordered/sequential collections.
- List collections can contain duplicate elements as well as multiple null elements.
- Using integer index, we can access elements from any List collection.
- We can traverse elements of List Collection using Iterator as well as ListIterator.
- This interface is a member of the Java Collections Framework.
- It is introduced in jdk 1.2
- Note : If we want to manage elements of non final type inside List collection then non final type should override equals method.
- Abstract Methods of List interface

1. void add(int index, E element);
2. boolean addAll(int index, Collection<? extends E> c);
3. E get(int index);
4. int indexOf(Object o);
5. int lastIndexOf(Object o);
6. ListIterator listIterator(int index);
7. ListIterator listIterator();
8. E remove(int index);
9. E set(int index, E element);
10. List subList(int fromIndex, int toIndex)

- Default Methods of List interface

1. default void sort(Comparator<? super E> c)
2. default void replaceAll(UnaryOperator operator)

ArrayList

- It is a list collection which implements List, RandomAccess, Cloneable and Serializable interface.
- It is resizable array.
- It is unsychronized collection. Using Collections.synchronizedList () method we can make it synchronized

```
List list = Collections.synchronizedList(new ArrayList(...));
```

- Default capacity of ArrayList is 10. If ArrayList is full then its capacity gets increased by half of its existing capacity.
- This class is a member of the Java Collections Framework.
- It is introduced in jdk 1.2
- Methods of ArrayList class

1. public void ensureCapacity(int minCapacity)
2. protected void removeRange(int fromIndex, int toIndex);
3. public void trimToSize()

- Constructor of ArrayList

1. public ArrayList()

```
ArrayList<Integer> list = new ArrayList<>( );
```

2. public ArrayList(int initialCapacity)

```
int initialCapacity = 15;
ArrayList<Integer> list = new ArrayList<>( initialCapacity );
```

3. public ArrayList(Collection<? extends E> c)

```
ArrayList<Integer> list1 = new ArrayList<>( );
list1.add( 10 );
list1.add( 20 );
list1.add( 30 );
ArrayList<Integer> list2 = new ArrayList<>( list1 );
```