Day 8

Comparable and Comparator implementation

- If we want to sort array of value type using Arrays.sort() then sort() method implicitly use "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 same type then reference type must implement Comparable interface.
- compareTo() method returns integer value:
    - Returns a negative integer, zero, or a positive integer as current object is less than, equal to, or greater than the specified object.
- If we use Arrays.sort() method to sort array of instances of reference type then sort() method implicitly use "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 type as well different type then we should use Comparator interface.
- compare method returns integer value.
    - Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.
- If any class implements Comparable interface then it is considered as sortable.
- All the wrapper classes implements Comparable interface.

Collection Framework

- Every value/data stored in data structure is called element.
- In java, data structure class is called collection.
- Framework is library of reusable classes/interafces that is used to develop applicaiton.
- Library of reusable data structure classes that is used to develop java application is called collection framework.
- Main purpose of collection framework is to manage data in RAM efficiently.
- Consider following Example:
    1. Person has-a birthdate
    2. Employee is a person
- In java, collection instance do not contain instances rather it contains reference of instances.
- If we want to use collection framework them we should import java.util package.

Iterable:

- It is a interface delared in java.lang package.
- All the collcetion classes implements Iterable interface hence we can traverse it using for each loop
- Methods of Iterable interface

1. Iterator iterator()

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

Collection

- Collection is interface declared in java.util package.
- It is sub interface of Iterable interface.
- It is root interface in collection framework interface hierarchy.
- 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 java.util.Collection interface.
- It is ordered/sequential collection.
- ArrayList, Vector, Stack, LinkedList etc. implements List interface. It generally refered as "List collections".
- List collection can contain duplicate element as well multiple null elements.
- Using integer index, we can access elements from List collection.
- We can traverse elements of List collection using Iterator as well as ListIterator.
- 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()
7. ListIterator listIterator(int index)

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 resizable array.
- It implements List, RandomAccess, Cloneable, Serializable interfaces.
- It is List collection.
- It is unsynchronized collection. Using "Collections.synchronizedList" method, we can make it synchronized.

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

- Initial capacity of ArrayList is 10. If ArrayList is full then its capacity gets increased by half of its existing capacity.
- It is introduced in jdk 1.2
- Note: If we want to manage elements of non final type inside ArrayList then non final type should override "equals" method.
- Constructor(s) of ArrayList

1. public ArrayList()

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

2. public ArrayList(int initialCapacity)

```
ArrayList<Integer> list =
              new ArrayList<>(15);
List<Integer> list = new ArrayList<>(15);
Collection<Integer> list =
                new ArrayList<>(15)
```

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

```
Collection<Integer> c = ArrayList<>();
List<Integer> list = new ArrayList<>(c);
```

```
Collection<Integer> c = Vector<>();
List<Integer> list = new ArrayList<>(c);
```

```
Collection<Integer> c = TreeSet<>();
List<Integer> list = new ArrayList<>(c);
```

```
Collection<Integer> c = ArrayDeque<>();
List<Integer> list = new ArrayList<>(c);
```

- Methods of ArrayList

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

- Using illegal index, if we try to access element from any List collection then List methods throws IndexOutOfBounds Exception.

```
List<Integer> list = new ArrayList<>();
list.add(10);
list.add(20);
list.add(30);
Integer element = list.get(list.size());
//Output : IndexOutOfBoundsException
```

- If we want to sort elements of array then we should use Arrays.sort() method and to sort elements of List collection, we should use Collections.sort() method.

Vector

- It is resizable array.
- It implements List, RandomAccess, Cloneable, Serializable.
- It is List collection.
- It is synchronized collection.
- Default capacity of vector is 10. If vector is full then its capacity gets increased by its existing capacity.
- We can traverse elements of vector using Iterator, ListIterator as well as Enumeration.
- It is introduced in jdk 1.0.
- Note: If we want to manage elements of non final type inside Vector then non final type should override "equals" method.

Following classes are by default synchronized

1. Vector
2. Stack(Sub class of Vector)
3. Hashtable
4. Properties(Sub class of Hashtable)

Enumeration

- It is interface declared in java.util package.
- Methods of Enumeration I/F
    1. boolean hasMoreElements()
    2. E nextElement()
- It is used to traverse collection only in forward direction. During traversing, we can add, set or remove element from collection.
- It is introduced in jdk 1.0.
- "public Enumeration elements()" is a method of Vector class.

```
Vector<Integer> v = new Vector<Integer>();
v.add(10);
v.add(20);
v.add(30);

Integer element = null;
Enumeration<Integer> e = v.elements();
while( e.hasMoreElements())
{
    element = e.nextElement();
    System.out.println(element);
}
```

Iterator

- It is a interface declared in java.util package.
- It is used to traverse collection only in forward direction. During traversing, we can not add or set element but we can remove element from collection.
- Methods of Iterator

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

- It is introduced in jdk 1.2

```
Vector<Integer> v = new Vector<Integer>();
v.add(10);
v.add(20);
v.add(30);
```

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

ListIterator

- It is subinterface of Iterator interface.
- It is used to traverse only List Collection in bidirection.
- During traversing we can add, set as well as remove element from collection.
- It is introduced in jdk 1.2
- Methods of ListIterator

1. boolean hasNext()
2. E next()
3. boolean hasPrevious()
4. E previous()
5. void add(E e)
6. void set(E e)
7. void remove()

```
Vector<Integer> v = new Vector<Integer>();
v.add(10);
v.add(20);
v.add(30);

Integer element = null;
    ListIterator<Integer> itr = v.listIterator();
while( itr.hasNext())
{
    element = itr.next();
    System.out.print(element+"  ");
}
System.out.println();
while( itr.hasPrevious())
{
    element = itr.previous();
    System.out.print(element+"  ");
}
```