Name: Vangipuram Srinivasa Sarath Chandra
Regd No: 21BCE9853
Title: Digital Assignment Java
Date: 11/June/2022

2. Develp a java program to implement Set interface by using TreeSet class and EnumSet class.[10M]

List of methods used in the follwing TreeSet and EnumSet are:

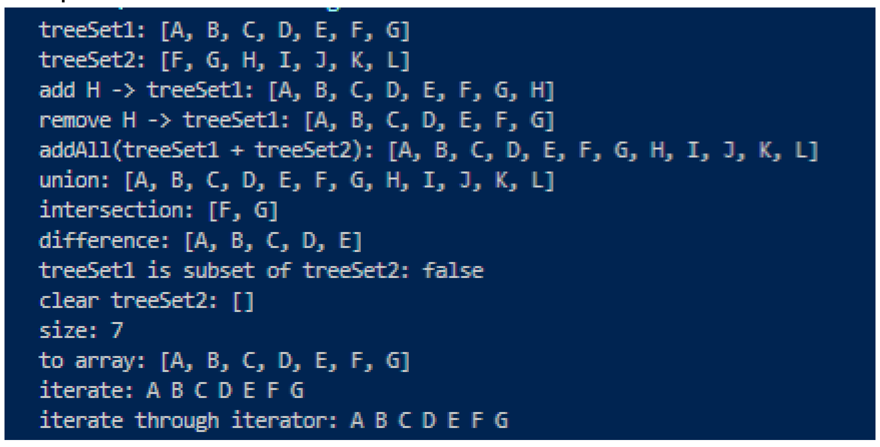
addAll	retainAll	removeAll	containsAll
remove	clear	add	toArray
size	iterator		

Tree Set

Tree Set: uses tree (a non linear data structure) for storing data. The ordering of the elements is maintained by a set using their natural ordering whether or not an explicit compartor is provided. This must be consistent with equals if it is to correctly implement the Set interface.

```java
import java.util.*;

public class TreeSetAssign {

    // creating a stack using a method
    static <T> TreeSet<T> createTreeSet(T... ele) {
        TreeSet<T> newTreeSet = new TreeSet<T>();
        for (T el : ele) {
            newTreeSet.add(el);
        }
        return newTreeSet;
    }

    public static void main(String[] args) {
        TreeSet<String> treeSet1 = createTreeSet("A", "B", "C", "D", "E", "F", "G");
        TreeSet<String> treeSet2 = createTreeSet("F", "G", "H", "I", "J", "K", "L");

        System.out.println("treeSet1: " + treeSet1);
        System.out.println("treeSet2: " + treeSet2);

        // add
        treeSet1.add("H");
        System.out.println("add H -> treeSet1: " + treeSet1);

        // remove
        treeSet1.remove("H");
        System.out.println("remove H -> treeSet1: " + treeSet1);

        // add all
        TreeSet<String> addAll = new TreeSet<String>(treeSet1);
        addAll.addAll(treeSet2);
        System.out.println("addAll(treeSet1 + treeSet2): " + addAll);

        // union
        TreeSet<String> union = new TreeSet<String>(treeSet1);
        union.addAll(treeSet2);
        System.out.println("union: " + union);

        // intersection
        TreeSet<String> intersection = new TreeSet<String>(treeSet1);
        intersection.retainAll(treeSet2);
        System.out.println("intersection: " + intersection);

        // difference
        TreeSet<String> difference = new TreeSet<String>(treeSet1);
        difference.removeAll(treeSet2);
        System.out.println("difference: " + difference);

        // subset
        System.out.println("treeSet1 is subset of treeSet2: " + treeSet1.containsAll(treeSet2));

        // clear
        treeSet2.clear();
        System.out.println("clear treeSet2: " + treeSet2);

        // size
        System.out.println("size: " + treeSet1.size());

        // to array
        String[] array = treeSet1.toArray(new String[treeSet1.size()]);
        System.out.println("to array: " + Arrays.toString(array));

        // iterate
        System.out.print("iterate: ");
        for (String el : treeSet1) {
            System.out.print(el + " ");
        }

        // iterate throught iterator
        Iterator<String> iterator = treeSet1.iterator();
        System.out.print("\niterate through iterator: ");
        while (iterator.hasNext()) {
            System.out.print(iterator.next() + " ");
        }
    }
}
```

Output:



```
treeSet1: [A, B, C, D, E, F, G]
treeSet2: [F, G, H, I, J, K, L]
add H -> treeSet1: [A, B, C, D, E, F, G, H]
remove H -> treeSet1: [A, B, C, D, E, F, G]
addAll(treeSet1 + treeSet2): [A, B, C, D, E, F, G, H, I, J, K, L]
union: [A, B, C, D, E, F, G, H, I, J, K, L]
intersection: [F, G]
difference: [A, B, C, D, E]
treeSet1 is subset of treeSet2: false
clear treeSet2: []
size: 7
to array: [A, B, C, D, E, F, G]
iterate: A B C D E F G
iterate through iterator: A B C D E F G
```

Explanation:
As the execution of the program starts i.e the main method is called,
we create a new Tree Set object and add some elements to it.
then we use different methods like: add, reomve, clear, etc to manipulate the
Tree Set object.

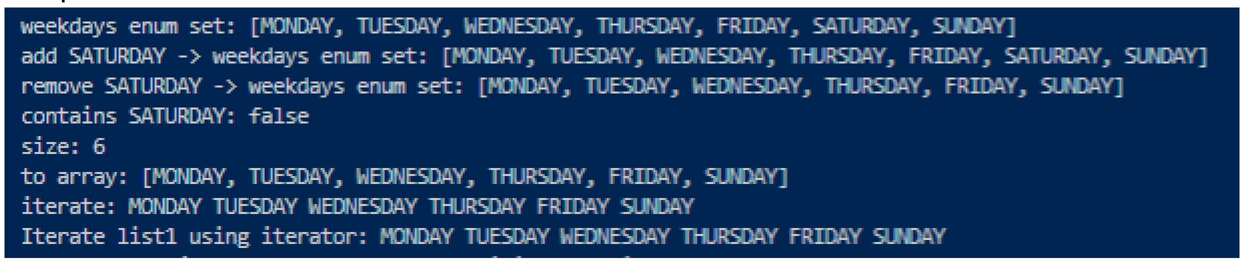
Enum Set

Advantages of enum set:

- Due to its implementation using RegularEnumSet and JumboEnumSet all the methods in an EnumSet are implemented using bitwise arithmetic operations.
- EnumSet is faster than HashSet because we no need to compute any hashCode to find the right bucket.
- The computations are executed in constant time and the space required is very little.

```java
import java.util.*;

enum Weekday {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
};

public class EnumSetAssign {
    public static void main(String[] args) {
        EnumSet<Weekday> weekdays = EnumSet.allOf(Weekday.class);

        System.out.println("weekdays enum set: " + weekdays);

        // add
        weekdays.add(Weekday.SATURDAY);
        System.out.println("add SATURDAY -> weekdays enum set: " + weekdays);

        // remove
        weekdays.remove(Weekday.SATURDAY);
        System.out.println("remove SATURDAY -> weekdays enum set: " + weekdays);

        // contains
        System.out.println("contains SATURDAY: " + weekdays.contains(Weekday.SATURDAY));

        // size
        System.out.println("size: " + weekdays.size());

        // to array
        Weekday[] array = weekdays.toArray(new Weekday[weekdays.size()]);

        System.out.println("to array: " + Arrays.toString(array));

        // iterate
        System.out.print("iterate: ");
        for (Weekday el : weekdays) {
            System.out.print(el + " ");
        }

        // iterate using iterator
        System.out.print("\nIterate list1 using iterator: ");
        Iterator<Weekday> itr = weekdays.iterator();
        while (itr.hasNext()) {
            System.out.print(itr.next() + " ");
        }

    }
}
```

Output:



```
weekdays enum set: [MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY]
add SATURDAY -> weekdays enum set: [MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY]
remove SATURDAY -> weekdays enum set: [MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SUNDAY]
contains SATURDAY: false
size: 6
to array: [MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SUNDAY]
iterate: MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SUNDAY
Iterate list1 using iterator: MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SUNDAY
```

Explanation:
As the execution of the program starts i.e the main method is called,
we create a new Enum Set object and add some elements to it.
then we use different methods like: add, reomve, clear, etc to manipulate the
Enum Set object.