TreeMap

Features:

1. Sorted Order: keys are maintained in natural order(ascending)
2. Unique Keys: keys must be unique, if we are attempting to insert duplicate keys it will overwrite the existing value.
3. Null handling
    - Keys: TreeMap doesn’t allow null keys
    - Values: It allows multiple null values.
4. Thread Safety
    - It is not synchronized

Common methods:

Put(K key, V value)

Get()

firstKey()

lastKey()

```
package mapExample;

import java.util.TreeMap;

public class TreeMapExample {
    public static void main(String[] args) {
        //Creating a TreeMap
        TreeMap<Integer, String> treeMap = new TreeMap<>();

        //adding elements in treeMap
        treeMap.put(3, "Three");
        treeMap.put(1, "One");
        treeMap.put(4,"Four");
      //  treeMap.put(1, "Duplicate");

        //Displaying the map
        System.out.println("TreeMap: "+ treeMap);

        //accessing the element
        System.out.println("Value for key 1: "+
treeMap.get(1));

        //to remove an element
        treeMap.remove(1);
        System.out.println("After removal: "+treeMap);

        System.out.println("First Key: "+ treeMap.firstKey());
        System.out.println("Last Key: "+ treeMap.lastKey());

    }
}
```

o/p:

```
TreeMap: {1=One, 3=Three, 4=Four}

Value for key 1: One

After removal: {3=Three, 4=Four}

First Key: 3

Last Key: 4
```

Collection and collections:

- Collection Interface:
- Collections Class:
  - It is a utility class in java.util package which provides methods for performing on collection.
  - It is a class, not an interface
  - Provides few imp methods - copy(),shuffle(), min(), max(), replaceAll()

1. Shuffle()

Purpose: randomnly shuffles elements in a list.

```
package collectionsExamples;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class CollectionsShuffleMethod {
    public static void main(String[] args) {
        List<String> cards = Arrays.asList("Ace", "King",
"Jack", "Queen");

        System.out.println("Before Shuffling: "+ cards);

        //shuffling elements of a list
        Collections.shuffle(cards);
        System.out.println("After Shuffling: "+ cards);
    }
}
```

o/p:

```
Before Shuffling: [Ace, King, Jack, Queen]
After Shuffling: [Jack, Queen, King, Ace]
```

2. Min() & Max()

```
package collectionsExamples;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class CollectionsMinMaxExample {
    public static void main(String[] args) {
        List<Integer> numbers =
Arrays.asList(10,2,30,23,45,56);

        int min = Collections.min(numbers);
        int max = Collections.max(numbers);

        System.out.println("Minimum: "+ min);
        System.out.println("Maximum: "+ max);
    }
}
```

o/p:

```
Minimum: 2
Maximum: 56
```

3. Copy()

Purpose: Copies the elements from one list to another

The destination list should have the same size as the source list.

```java
package collectionsExamples;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class CollectionsCopyExample {
    public static void main(String[] args) {
        List<String> source = Arrays.asList("A", "B", "C");

        //Destination list must have the same size
        List<String> destination = new
ArrayList<>(Arrays.asList("","",""));

        Collections.copy(destination,source);

        System.out.println("Source: "+ source);

        System.out.println("Destination: "+ destination);
    }
}
```

o/p:

```
Source: [A, B, C]

Destination: [A, B, C]
```

```java
package collectionsExamples;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class CollectionsCopyExample {
    public static void main(String[] args) {
        List<Integer> source = Arrays.asList(1, 2, 3);

        //Destination list must have the same size
        List<Integer> destination = new
ArrayList<>(Arrays.asList(0,0,0));

       Collections.copy(destination,source);

        System.out.println("Source: "+ source);

        System.out.println("Destination: "+ destination);
    }
}
```

- Without copy():

```java
package collectionsExamples;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class CollectionsCopyExample {
    public static void main(String[] args) {
        List<String> source = Arrays.asList("A", "B", "C");

        //Destination list must have the same size
        List<String> destination = new ArrayList<>(source);

//        Collections.copy(destination,source);

        System.out.println("Source: "+ source);

        System.out.println("Destination: "+ destination);
    }
}
```

- replaceAll():
  Purpose: Replaces all occurrences of a specific element with a new element

```
package collectionsExamples;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class ReplaceAllExample {
    public static void main(String[] args) {
        List<String> names = Arrays.asList("Krishna", "Gopal",
"Govind");

        System.out.println("Real List Before replacement: "+
names);

        Collections.replaceAll(names,"Krishna","Krishna
Yadav");

        System.out.println("After Replacement: "+ names);
    }
}
```

sort(), reverse()

Object Class – Object (Java SE 21 & JDK 21)

- The object class is the root of the class hierarchy
- It is a part of java.lang
- toString(), equals(), getClass()

1. toString()

   It returns the string representation of an object.

```java
package objectClassE;

public class Employee {
    int id;
    String name;

    public Employee(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Employee{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }

    public static void main(String[] args) {
        Employee emp = new Employee(100, "Krishna");

        System.out.println(emp);
    }
}
```

o/p:

Employee{id=100, name='Krishna'}

2. getClass()
   - it returns the runtime class of the object.

```
package objectClassE;

public class Test {
    public static void main(String[] args) {
        String str = "Flynaut";
        System.out.println("Classname: "+
str.getClass().getName());
    }
}
```

o/p:

Classname: java.lang.String

equals()＊