Collection Interface Vs Collections Class

1. Collection Interface:
   - Root interface in java.util package which represents group of objects as a single unit
   - It is an interface
   - Implemented by various collection classes like ArrayList, Hashset, TreeSet, etc
   - It provides methods for adding, removing, and iterating over elements in a collection.
2. Collections Class:
   - Is a utility class in java.util package that provides static methods for performing various operations on collections.
   - It is a class
   - It is final class, so it cannot be inherited.
   - Methods like sort(), reverse(), shuffle(), etc.

- Collections.shuffle(List<?> list)
  This randomly shuffles the elements in a list.

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

public class CollectionsShuffleExample {
    public static void main(String[] args) {
        // Creating a list of playing cards
        List<String> cards = Arrays.asList("Ace", "King",
"Queen", "Jack");

        //Displaying cards before shuffling
        System.out.println("Before Shuffling: "+ cards);

        //To shuffle elements randomly
        Collections.shuffle(cards);

        //Displaying list after shuffle
        System.out.println("After Shuffling: "+ cards);
    }
}
```

o/p:
Before Shuffling: [Ace, King, Queen, Jack]
After Shuffling: [Queen, Jack, Ace, King]

- Min() and Max()

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

public class CollectionsMinMaxEx {
    public static void main(String[] args) {
        //Creating a list of numbers
        List<Integer> numbers =
Arrays.asList(10,2,20,40,56,59);

        //Finding the minimum and maximum value in tthe
list
        int min = Collections.min(numbers);
        int max = Collections.max(numbers);

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

o/p:
Minimum : 2
Maximum : 59

- Collections.copy()
  This method copies elements from one list to another

```
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) {
        //creating a source list
        List<String> source = Arrays.asList("A","B","C");
```

```
        //creating a destination list with the same size as of
source
        List<String> destination = new
ArrayList<>(Arrays.asList("","",""));
//      List<String> destination = new ArrayList<>(source);

        //Copying elements from source to destination
        Collections.copy(destination, source);

        //Displaying the list
        System.out.println("Source: "+ source);

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


    }
}
```

o/p:
Source: [A, B, C]
Destination: [A, B, C]

- replaceAll():

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

public class ReplaceAllEx {
    public static void main(String[] args) {
        List<String> names = Arrays.asList("Krishna",
"Gopal" , "Govind");
        System.out.println("Real: "+ names);
        Collections.replaceAll(names, "Krishna",
"Krish");
        System.out.println("After Replacement: "+ names);
    }
}
```

- Collections.synchronizedList(List<T> list)
  It will return the thread safe version of the given list
- Collections.synchronizedMap()
- Collections.synchronizedSet()
- Sort() – sorts elements in ascending order
- Reverse() – reverses the order of elements

- Object Class:
  - Root of the class hierarchy in java
  - Every java class inherits from object class either directly or indirectly.
  - It is a part of java.lang package.
  - It provides commonly used methods such as toString(), equals(), hashCode(), wait(), notify()

1. toString() Method:
   - it returns string representation of an object
   - by default, it prints classname and object's hashcode
   - It can be overridden to provide meaningful representation.

Task: create employee class with id and name, override the toString create an object in main class and display the object.

2. equals(Object obj) Method
   - compares two objects
   - default implementation checks if the references are the same.
   - We can override it to compare object values.

```
- package exampleEquals;

  public class Employee {
      int id;
      String name;

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

      //overriding the equals() to compare employee
  objects
      @Override
      public boolean equals(Object obj){
          if (this == obj) return true; // if both
  references point to the same object
```

```
        if(obj == null || getClass() != obj.getClass()
) return false;

        Employee employee = (Employee)obj; // Typecast
to Employee
        return id == employee.id &&
name.equals(employee.name);     //Compare values
    }

    public static void main(String[] args) {
        Employee emp1 = new Employee(101,"Krishna");
        Employee emp2 = new Employee(101, "Krishna");

        //Comparing two employee objects
        System.out.println("Are they equal? "+
emp1.equals(emp2));

    }

}
```

o/p:

Are they equal? true

getClass() : returns the runtime class of object.

Will discuss in threads:

wait()

notify()

notifyAll()

Object (Java SE 21 & JDK 21)
Collection (Java SE 21 & JDK 21)
Collections (Java SE 21 & JDK 21)