Dt : 10/10/2023

Note:

=>Sorting process on WrapperClass Objects and String-Objects uses Quick sorting technique.

=>Sorting process on User defined class Objects uses Merge Sorting technique.

===

faq:

wt is the diff b/w

(i)Collection<E>

(ii)Collections

=>Collection<E> is an interface from java.util package and which is root of Java Collection<E> framework

=>Collections is a class from java.util package and which privide "sort()" method to perform Sorting process.

==

faq:

wt is the diff b/w

(i)Comparable<T>

(ii)Comparator<T>

(i)Comparable<T>:

=>Comparable<T> is an interface from java.lang package and which provide 'compareTo(T)' method to perform Sorting process.

(ii)Comparator<T>:

=>Comparator<T> is an interface from java.util package and which provide 'compare(T,T)' method to perform Sorting process.

==

**imp*

define Predicate<T>? (Java8 - version Component)

=>Predicate<T> is a Functional Interface from java.util package introducted by Java8 version and which is used to perform Conditional operation on Collection<E> objects.

Structure of Predicate<T>:

```
public interface java.util.function.Predicate<T>
{
  public abstract boolean test(T);
  ...
}
```

syntax of LambdaExpression for Predicate<T>:

```
Predicate<T> pd = (T)->
            {
             ...
            };
```

Program : DemoList5.java

```
package p2;
import java.util.*;
import java.util.function.*;//Java8 version
public class DemoList5 {
        @SuppressWarnings("removal")
        public static void main(String[] args) {
     ArrayList<Integer> al = new ArrayList<Integer>();
     for(int i=11;i<=20;i++) {
           al.add(new Integer(i));
     }
     System.out.println("====List<E>====");
     System.out.println(al.toString());
```

```
        Predicate<Integer> pd = (z)->
        {
          if(z%2==0)
          {
             return true;
          }
          else
          {
             return false;
          }
        };
        System.out.println("====Display Odd Elements====");
        al.spliterator().forEachRemaining((k)->
        {
            if(!pd.test(k))
            {
                    System.out.print(k+" ");
            }
        });
           }
}
```

o/p:

```
====List<E>====
[11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
====Display Odd Elements====
11 13 15 17 19
```

===

Assignment:

wap to read n Product details and display products which are having price less than or equal to 1000?(Use Predicate<T>)

==

****imp***

define Function<T,R>?

=>Function<T,R> is a functional interface from java.util.function package introduced by Java8 version and which is used to perform functional-operation on Collection<E> objects.

structure of Function<T,R>:

```
public interface java.util.function.Function<T, R>
{
  public abstract R apply(T);
  ...
}
```

syntax of LambdaExpression for Function<T,R>:

```
Function<T,R> fc = (T)->
                {
                  ....
                };
```

faq:

define UnaryOperator<T>?

=>UnaryOperator<T> is a interface from java.util.function package introduced Java8 version and which extend from "java.util.functiom.Function<T,R>" and which is a parameter to replaceAll() method of List<E>

Method Signature of replaceAll():

```
public default void replaceAll(java.util.function.UnaryOperator<E>);
```

Program : DemoList6.java

```
package p2;
import java.util.*;
import java.util.function.*;

public class DemoList6 {

          @SuppressWarnings("removal")

          public static void main(String[] args) {

                    ArrayList<Integer> al = new ArrayList<Integer>();

                    for(int i=11;i<=20;i++) {

                              al.add(new Integer(i));

                    }

                    System.out.println("====List<E>====");

                    al.spliterator().forEachRemaining((k)->

                    {

                              System.out.print(k+" ");

                    });

                    al.replaceAll((z)->z+10);

                    System.out.println("\n====Ele in List<E> add by 10==");

       al.spliterator().forEachRemaining((k)->

       {

          System.out.print(k+" ");

       });

          }

}
```

o/p:

```
====List<E>====

11 12 13 14 15 16 17 18 19 20

====Ele in List<E> add by 10==

21 22 23 24 25 26 27 28 29 30

================================================================
```