Dt : 25/11/2022

Sorting Process:

=>The process of organizing elements in Ascending order or Descending order is known as Sorting process.

=>we use "sort()" method from 'java.util.Arrays class' to perform sorting process on Array-Objects.

=>we use "sort()" method from 'java.util.Collections' class to perform Sorting process on List-Objects.

=>we have "TreeSet<E>" to sort elements in Set<E> objects.

=>we have "TreeMap<K,V>" to sort elements in Map<K,V> objects.

=>we must not perform sorting process on Queue<E> objects because elements are organized based on algorithm "First-In-First-Out".

Ex-program1 : Sorting process of Array objects.

Product.java

```
package test;
@SuppressWarnings("rawtypes")
public class Product extends Object implements Comparable
{
     public int code;
     public String name;
     public Product(int code,String name) {
      this.code=code;
      this.name=name;
     }
     @Override
     public String toString() {
      return code+"\t"+name;
     }
     @Override
     public int compareTo(Object o) {
      Product p = (Product)o;
      if(code==p.code) return 0;
```

```
        else if(code>p.code) return 1;
        else return -1;
      }
}
```

ArraySort.java(MainClass)

```
package maccess;

import java.util.*;

import test.*;

public class ArraySort {

        @SuppressWarnings("removal")

        public static void main(String[] args) {

    Integer a[] = new Integer[5];

    a[0] = new Integer(12);

    a[1] = new Integer(11);

    a[2] = new Integer(10);

    a[3] = new Integer(7);

    a[4] = new Integer(8);

    System.out.println("====Before Sorting====");

    for(Integer k : a) {

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

    }

    Arrays.sort(a);//Sorting Process

    System.out.println("\n====After Sorting====");

    for(Integer k : a) {

          System.out.print(k.toString()+" ");
    }
    System.out.println("\n====Descending Order=====");
    for(int i=a.length-1;i>=0;i--) {
          System.out.print(a[i].toString()+" ");
    }
    System.out.println("\n*****Product Objects*****");
    Product p[] = new Product[5];
    p[0] = new Product(121,"Mouse");
    p[1] = new Product(120,"Keyboard");
    p[2] = new Product(119,"CDR");
    p[3] = new Product(122,"ANN");
    p[4] = new Product(101,"Board");

    System.out.println("====Before Sorting====");
    for(Product k : p) {
          System.out.println(k.toString());
    }
    Arrays.sort(p);//Sorting process
    System.out.println("====After Sorting====");
    for(Product k : p) {
          System.out.println(k.toString());
    }
    System.out.println("====Descending Order====");
    for(int i=p.length-1;i>=0;i--)
```

```
        {
                System.out.println(p[i].toString());
        }
            }

}
```

o/p:

```
====Before Sorting====
12 11 10 7 8
====After Sorting====
7 8 10 11 12
====Descending Order=====
12 11 10 8 7
*****Product Objects*****
====Before Sorting====
121     Mouse
120     Keyboard
119     CDR
122     ANN
101     Board
====After Sorting====
101     Board
119     CDR
120     Keyboard
```

```
121     Mouse

122     ANN

====Descending Order====

122     ANN

121     Mouse

120     Keyboard

119     CDR

101     Board
```

==

Ex-program2 : Sorting Process on List<E> Objects

Product.java

```java
package test;
@SuppressWarnings("rawtypes")
public class Product extends Object implements Comparable
{
    public int code;
    public String name;
    public Product(int code,String name) {
     this.code=code;
     this.name=name;
    }
    @Override
    public String toString() {
     return code+"\t"+name;
    }
    @Override
    public int compareTo(Object o) {
     Product p = (Product)o;
     if(code==p.code) return 0;
     else if(code>p.code) return 1;
     else return -1;
    }
}
```

ListSort.java(MainClass)

```
package maccess;

import java.util.*;

import test.Product;

public class ListSort {

  @SuppressWarnings({ "removal", "unchecked" })

        public static void main(String[] args) {

        ArrayList<Integer> ob1 = new ArrayList<Integer>();

        ob1.add(new Integer(12));

        ob1.add(new Integer(10));

        ob1.add(new Integer(11));

        ob1.add(new Integer(7));

        ob1.add(new Integer(8));

        System.out.println("====before Sorting====");

        System.out.println(ob1.toString());

        Collections.sort(ob1);//Sorting Process

        System.out.println("====After Sorting====");

        System.out.println(ob1.toString());

        System.out.println("===Descending Order====");

    for(int i=ob1.size()-1;i>=0;i--)

    {

        System.out.print(ob1.get(i)+" ");

    }

    System.out.println("\n*****Product Objects*****");
```

```
ArrayList<Product> ob2 = new ArrayList<Product>();

ob2.add(new Product(121,"Mouse"));

ob2.add(new Product(120,"Keyboard"));

ob2.add(new Product(101,"Board"));

ob2.add(new Product(119,"ANN"));

ob2.add(new Product(107,"CDR"));

ob2.add(new Product(101,"Board"));

ob2.add(new Product(119,"ANN"));

ob2.add(new Product(107,"CDR"));

System.out.println("====Before Sorting====");

ob2.forEach((k)->

{

    System.out.println(k.toString());

});

Collections.sort(ob2);//Sorting Process

System.out.println("====After Sorting====");

ob2.forEach((k)->

{

    System.out.println(k.toString());

});

System.out.println("====Descending Order====");

for(int i=ob2.size()-1;i>=0;i--)

{

    System.out.println(ob2.get(i));
```

```
        }
    }
}
```

o/p:

```
====before Sorting====
[12, 10, 11, 7, 8]
====After Sorting====
[7, 8, 10, 11, 12]
===Descending Order====
12 11 10 8 7
*****Product Objects*****
====Before Sorting====
121     Mouse
120     Keyboard
101     Board
119     ANN
107     CDR
101     Board
119     ANN
107     CDR
====After Sorting====
101     Board
101     Board
107     CDR
```

```
107     CDR

119     ANN

119     ANN

120     Keyboard

121     Mouse

====Descending Order====

121     Mouse

120     Keyboard

119     ANN

119     ANN

107     CDR

107     CDR

101     Board

101     Board
```

===

faq:

define Comparator<T>?

=>Comparator<T> is an interface from java.util package and which is also used to perform Sorting process on List<E> objects.

=>sort() method from List<E>,which is introduced by Java8 version used to perform sorting process using Comparator<T>

Method Signature of sort():

public default void sort(java.util.Comparator<? super E>);

Ex:

BookDetails.java

```
package test;
public class BookDetails {
   public int code;
   public String name;
   public BookDetails(int code,String name) {
        this.code=code;
        this.name=name;
   }
   public String toString() {
        return code+"\t"+name;
   }
}
```

SortByCode.java

```
package test;
import java.util.*;
@SuppressWarnings("rawtypes")
public class SortByCode implements Comparator{
     @Override
     public int compare(Object ob1,Object ob2)
     {
      BookDetails b1 = (BookDetails)ob1;
      BookDetails b2 = (BookDetails)ob2;
      if(b1.code==b2.code) return 0;
      else if(b1.code>b2.code) return 1;
      else return -1;
     }
}
```

SortbyName.java

```
package test;
import java.util.*;
@SuppressWarnings("rawtypes")
public class SortByName implements Comparator{
     @Override
    public int compare(Object ob1,Object ob2)
    {
        BookDetails b1 = (BookDetails)ob1;
        BookDetails b2 = (BookDetails)ob2;
        int z = b1.name.compareTo(b2.name);
        if(z==0) return 0;
        else if(z>0) return 1;
        else return -1;
      }
}
```

ListSort2.java(MainClass)

```
  package maccess;

import test.*;

import java.util.*;

public class ListSort2 {

        @SuppressWarnings("unchecked")

        public static void main(String[] args) {

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

    al.add(new BookDetails(121,"CoreJava"));

    al.add(new BookDetails(120,"AdvJava"));

    al.add(new BookDetails(101,"c-Lang"));

    al.add(new BookDetails(119,"Py.."));

    System.out.println("====Bofore Sorting===");

    al.forEach((k)->

    {

          System.out.println(k.toString());

    });

    System.out.println("====SortByCode===");

    al.sort(new SortByCode());
    al.forEach((k)->
    {
        System.out.println(k.toString());
    });
    System.out.println("====SortByName===");
    al.sort(new SortByName());
    al.forEach((k)->
    {
        System.out.println(k.toString());
    });
      }

}
```

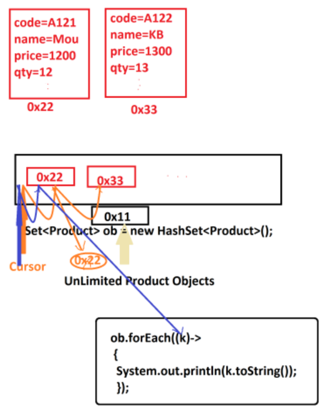
o/p:

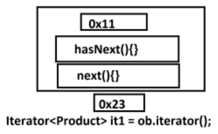
```
====Bofore Sorting===
121    CoreJava
120    AdvJava
101    c-Lang
119    Py..
====SortByCode===
101    c-Lang
119    Py..
120    AdvJava
121    CoreJava
```

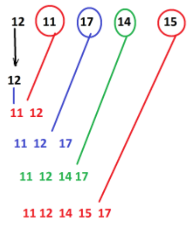
====SortByName===

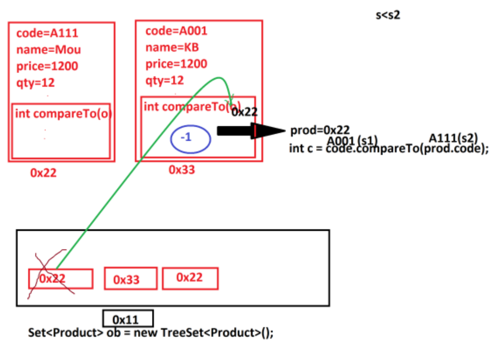
120	*AdvJava*
121	*CoreJava*
119	*Py..*
101	*c-Lang*

Diagram:









===

Dt : 26/11/2022

Object Oriented Programming Levels:

1.Object definition

2.Object Creation

3.Object Location

4.Object Components

5.Object Types

(i)User Defined Class Objects

(ii)String Objects

(iii)WrapperClass Objects

(iV)Array Objects

(v)Collection<E> Objects

(vi)Map<K,V> Objects

(vii)Enum<E> Objects

6.Object Serialization

7.Object Collection

8.Object Locking

9.Object Cloning

10.Object Sorting

*

11.Object holding Database table data(AdvJava)

==

Note:

=>Some methods related to Set<E> and List<E>.

SetMethods.java

```
package maccess;
import java.util.*;
public class SetMethods {
      @SuppressWarnings("removal")
```

```
	public static void main(String[] args) {
	  HashSet<Integer> hs1 = new HashSet<Integer>();
	  hs1.add(new ~~Integer~~(12));
	  hs1.add(new ~~Integer~~(13));
	  hs1.add(new ~~Integer~~(14));
	  hs1.add(new ~~Integer~~(15));
	  System.out.println("*****hs1****");
	  System.out.println(hs1.toString());
	  HashSet<Integer> hs2 = new HashSet<Integer>();
	  hs2.add(new ~~Integer~~(16));
	  hs2.add(new ~~Integer~~(17));
	  hs2.add(new ~~Integer~~(18));
	  hs2.add(new ~~Integer~~(19));
	  System.out.println("*****hs2****");
	  System.out.println(hs2.toString());
	  System.out.println("****addAll()****");
	  hs1.addAll(hs2);
	  System.out.println(hs1.toString());
	  System.out.println("****removeAll()****");
	  hs1.removeAll(hs2);
	  System.out.println(hs1.toString());
	  HashSet<Integer> hs3 = new HashSet<Integer>();
	  hs3.add(new ~~Integer~~(12));
	  hs3.add(new ~~Integer~~(13));
	  System.out.println("****contains(Object)****)");
	  System.out.println(hs1.contains(new ~~Integer~~(12)));
System.out.println("****containsAll(Collection<E>)****)");
	  System.out.println(hs1.containsAll(hs3));
	  HashSet<Integer> hs4 = new HashSet<Integer>();
	  hs4.add(new ~~Integer~~(121));
	  hs4.add(new ~~Integer~~(13));
	  hs4.add(new ~~Integer~~(141));
	  hs4.add(new ~~Integer~~(15));
	  hs1.retainAll(hs4);//Common elements are displayed
	  System.out.println("****retainAll(Collection<E>)*****");
	  System.out.println(hs1);
	}

}
```

ListMethods.java

```
package maccess;
import java.util.*;
public class ListMethods {
	@SuppressWarnings("removal")
```

```
    public static void main(String[] args) {
      ArrayList<Integer> al = new ArrayList<Integer>();
      al.add(new ~~Integer~~(12));
      al.add(new ~~Integer~~(13));
      al.add(new ~~Integer~~(14));
      al.add(new ~~Integer~~(15));
      System.out.println(al.toString());
      System.out.println("****subList(index,index)****");
      List<Integer> al2 = al.subList(1, 3);
      al2.forEach((k)->
      {
       System.out.print(k.toString()+" ");
      });
    }
}
```

==