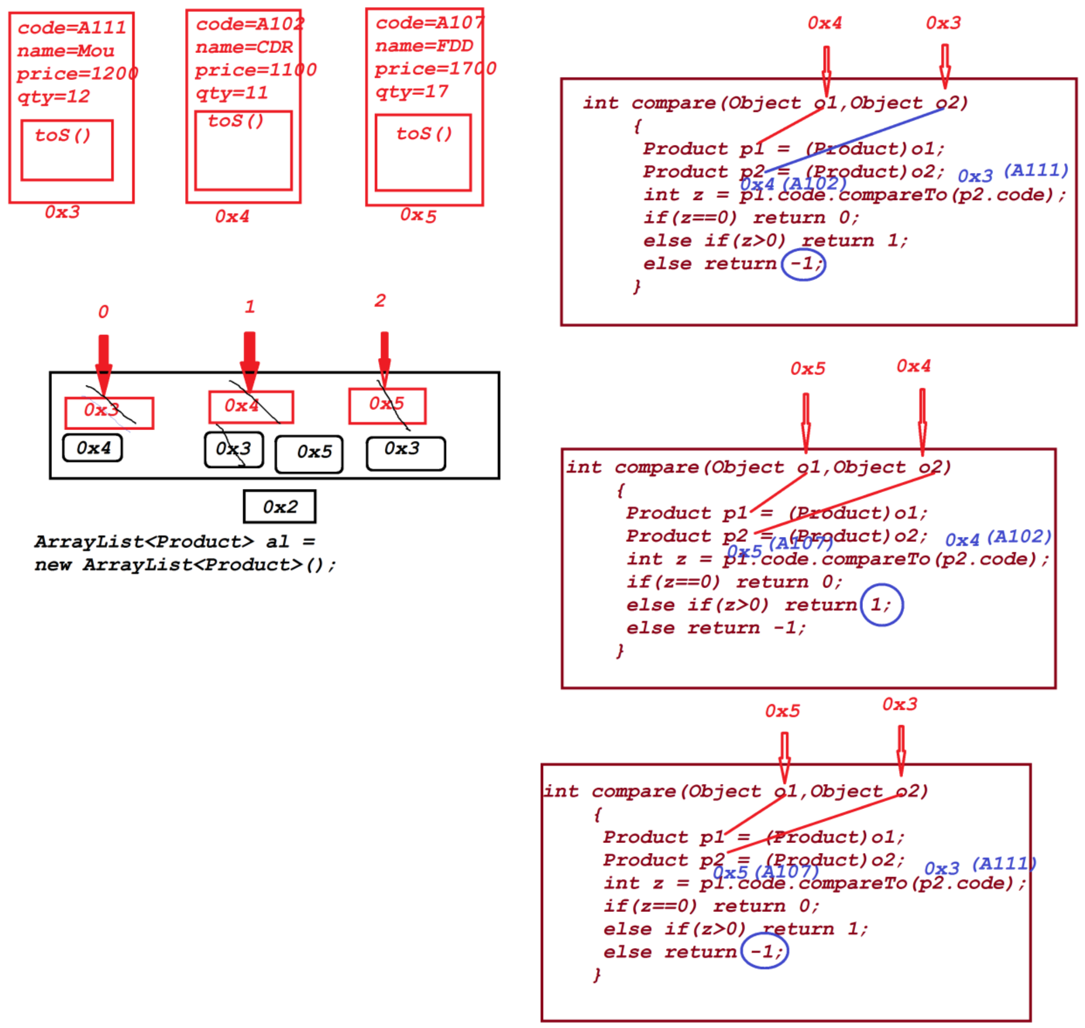
Dt : 9/10/2023

Diagram:



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

```
int compare(Object o1,Object o2)
    {
      Product p1 = (Product)o1;
      Product p2 = (Product)o2;
      int z = p1.code.compareTo(p2.code);
      if(z==0) return 0;
      else if(z>0) return 1;
      else return -1;
    }
```

```
int compare(Object o1,Object o2)
    {
      Product p1 = (Product)o1;
      Product p2 = (Product)o2;
      int z = p1.code.compareTo(p2.code);
      if(z==0) return 0;
      else if(z>0) return 1;
      else return -1;
    }
```

```
int compare(Object o1,Object o2)
    {
      Product p1 = (Product)o1;
      Product p2 = (Product)o2;
      int z = p1.code.compareTo(p2.code);
      if(z==0) return 0;
      else if(z>0) return 1;
      else return -1;
    }
```

===

faq:

define Sorting process?

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

=>we perform sorting process on List<E> objects in the following two ways:

(i)Using sort() method available from List<E>

=>This Method introduced by Java8 version

Ex:

above program

(ii)Using sort() method available from 'java.util.Collections' class

Method Signature:

public static <T> void sort(java.util.List<T>);

syntax:

Collections.sort(list_var);

Ex-program : DemoList3.java

```
package p2;
import java.util.*;
public class DemoList3 {
     @SuppressWarnings("removal")
     public static void main(String[] args) {
          ArrayList<Integer> al1 = new
ArrayList<Integer>();
          ArrayList<String> al2 = new
ArrayList<String>();

          al1.add(new ~~Integer~~(12));
          al1.add(new ~~Integer~~(19));
          al1.add(new ~~Integer~~(11));
          al1.add(new ~~Integer~~(9));

          al2.add(new String("bat"));
          al2.add(new String("apple"));
          al2.add(new String("egg"));
```

```
        al2.add(new String("cat"));

        System.out.println("----Before Sorting----");
        System.out.println(al1.toString());
        Collections.sort(al1);//Sorting process
        System.out.println("----After Sorting----");
        System.out.println(al1.toString());

        System.out.println("----Before Sorting----");
        System.out.println(al2.toString());
        Collections.sort(al2);//Sorting process
        System.out.println("----After Sorting----");
        System.out.println(al2.toString());

    }
}
```

o/p:

```
----Before Sorting----
[12, 19, 11, 9]
----After Sorting----
[9, 11, 12, 19]
----Before Sorting----
[bat, apple, egg, cat]
----After Sorting----
[apple, bat, cat, egg]
```

Note:

=>To perform Sorting process on User defined class objects using

'Collections.sort()' method,we following the following two steps:

step-1 : The user defined class must be implemented from

"java.lang.Comparable" interface.

structure of Comparable interface:

```
public interface java.lang.Comparable<T>
{
 public abstract int compareTo(T);
}
```

step-2 : The User defined class must construct body for abstract method

compareTo() and the method must hold sorting-specification

logic.

Program :

User.java

```
package p1;
@SuppressWarnings("rawtypes")
public class User extends Object implements Comparable
{
  public String name;
  public long phNo;
  public User(String name,long phNo) {
       this.name=name;
       this.phNo=phNo;
  }
  @Override
  public String toString() {
      return name+"\t"+phNo;
  }
  @Override
  public int compareTo(Object o)
  {
      User u = (User)o;
      int k = name.compareTo(u.name);
      if(k==0) return 0;
      else if(k>0) return 1;
      else return -1;
  }
}
```

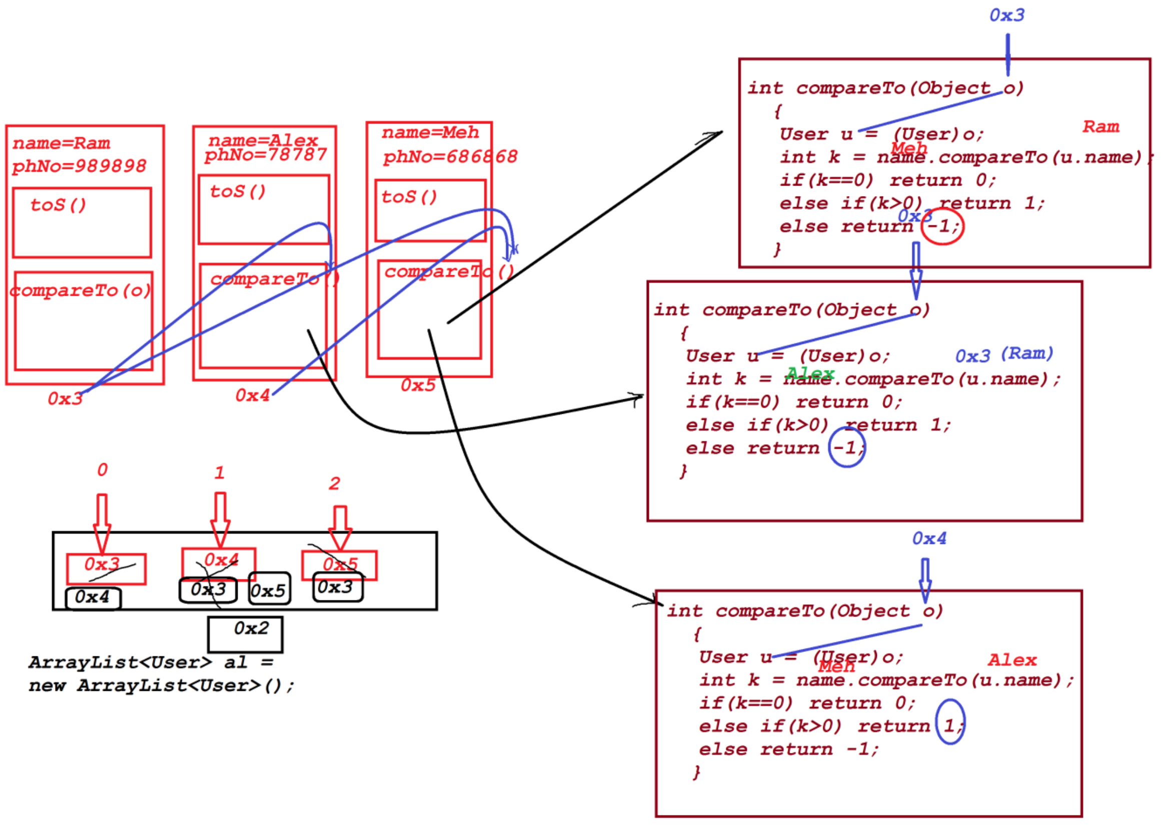
DemoList4.java(MainClass)

```
package p2;

import java.util.*;

import p1.*;

public class DemoList4 {

        @SuppressWarnings("unchecked")

        public static void main(String[] args) {

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

    al.add(new User("Ram",9898981234L));

    al.add(new User("Alex",7878781234L));

    al.add(new User("Mah",6868681234L));

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

    al.spliterator().forEachRemaining((k)->
    {
        System.out.println(k.toString());
    });
    System.out.println("----After Sorting----");
    Collections.sort(al);
    al.spliterator().forEachRemaining((k)->
    {
        System.out.println(k.toString());
    });
      }
}
```

o/p:

```
----Before Sorting----
Ram   9898981234
Alex  7878781234
Mah   6868681234
----After Sorting----
Alex  7878781234
Mah   6868681234
Ram   9898981234
```

Diagram:



```
ArrayList<User> al =
new ArrayList<User>();
```

```
int compareTo(Object o)
  {
   User u = (User)o;
   int k = name.compareTo(u.name);
   if(k==0) return 0;
   else if(k>0) return 1;
   else return -1;
  }
```

```
int compareTo(Object o)
  {
   User u = (User)o;
   int k = name.compareTo(u.name);
   if(k==0) return 0;
   else if(k>0) return 1;
   else return -1;
  }
```

```
int compareTo(Object o)
  {
   User u = (User)o;
   int k = name.compareTo(u.name);
   if(k==0) return 0;
   else if(k>0) return 1;
   else return -1;
  }
```

===