Dt : 11/11/2022

Note:

=>TreeSet<E> internally uses,

=>QuickSorting technique on WrapperClass objects and String Objects.

=>MergeSorting technique on User defined class objects.

=>To perform Sorting process on User defined class objects,we must follow the following two steps:

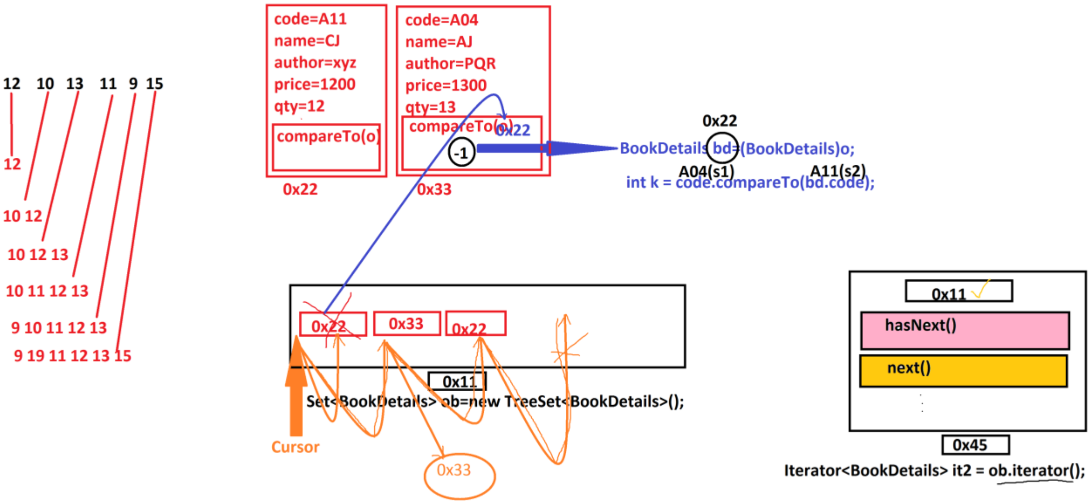
step-1 : The User defined class must be implemented from java.lang.Comparable interface

structure of Comparable<T>:

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

step-2 : Construct the body for compareTo(T) method of Comparable interface (compareTo(T) method will hold sorting-Specification logic)

Diagram:



===

define Iterator<E>?

=>Iterator<E> is an interface from java.util package and which is used to retrieve elements from Collection<E> objects in forward direction.

=>The following are some important methods of Iterator<E>:

public abstract boolean hasNext();

public abstract E next();

public default void forEachRemaining

(java.util.function.Consumer<? super E>);

=>we use iterator() method to create the implementation object for Iterator<E> interface:

Iterator<BookDetails> it2 = ob.iterator();

===

Note:

=>By Java8 version Iterator<E> is added with forEachRemaining() method and which internally uses LambdaExpression.

==

Modified Programs:

BookDetails.java

```
package test;
@SuppressWarnings("rawtypes")
public class BookDetails extends Object implements Comparable
{
     //Instance Variables
   public String code,name,author;
   public float price;
   public int qty;

   //Constructor to initialize Instance variables
   public BookDetails(String code,String name,String
author,float price,int qty){
        this.code=code;
        this.name=name;
        this.author=author;
        this.price=price;
        this.qty=qty;
   }
   @Override
   public String toString()
   {
        return code+"\t"+name+"\t"+author+"\t"+price+"\t"+qty;
   }

   @Override
   public int compareTo(Object o)
   {
        BookDetails bd = (BookDetails)o;//DownCasting process
        int k = code.compareTo(bd.code);
        if(k==0) return 0;
        else if(k>0) return 1;
        else  return -1;
   }
}
```

DemoSet3.java(MainClass)

```
package maccess;

import java.util.*;

import test.*;

public class DemoSet3 {

          public static void main(String[] args) {

      Scanner s = new Scanner(System.in);

      String name=null;

      Set<BookDetails> ob = null;

      try(s;){

          try {

                    while(true) {

                              System.out.println("****Choice*****");

                              System.out.println("1.HashSet\n2.LinkedHashSet\n3.TreeSet\n4.exit");

                              System.out.println("Enter the Choice:");

                              switch(Integer.parseInt(s.nextLine())) {

                              case 1:

                                        ob = new HashSet<BookDetails>();

                                        name="HashSet";

                                        break;

                              case 2:

                                        ob = new LinkedHashSet<BookDetails>();

                                        name="LinkedHashSet";

                                        break;
```

```
case 3:
        ob = new TreeSet<BookDetails>();
        name="TreeSet";
        break;
case 4:
        System.out.println("Operations stopped of Set");
        System.exit(0);
        break;
default:
        System.out.println("Invalid Choice...");
}//end of switch
System.out.println("****Operations on "+name+"****");
xyz:
while(true) {
        System.out.println("****Choice****");
System.out.println("1.add\n2.remove\n3.display\n4.exit");
System.out.println("Enter the Choice:");
switch(Integer.parseInt(s.nextLine())) {
case 1:
        System.out.println("Enter the code:");
        String bC=s.nextLine();
        System.out.println("Enter the name:");
        String bN=s.nextLine();
        System.out.println("Enter the author:");
```

```
    String bA=s.nextLine();

    System.out.println("Enter the price:");

    float bP = Float.parseFloat(s.nextLine());

    System.out.println("Enter the qty:");

    int bQ = Integer.parseInt(s.nextLine());

    ob.add(new BookDetails(bC,bN,bA,bP,bQ));

    System.out.println("BookDetails added Successfully..");

    break;

case 2:

    if(ob.isEmpty()) {

        System.out.println("Set is empty...");

    }else {

        System.out.println("Enter the ele(code) to be removed:");

      String code2 = s.nextLine();

      boolean p=false;

      Iterator<BookDetails> it = ob.iterator();

      while(it.hasNext())

      {

        BookDetails bd = (BookDetails)it.next();

        if(bd.code.equals(code2)) {

            p=true;

            ob.remove(bd);

            System.out.println("Ele removed Successfully..");

            break;
```

```
                        }
                  }//end of loop
                  if(!p)
                  {
                        System.out.println("Element Not found...");
                  }
                }
                break;
        case 3:
                System.out.println("****Iterator<E>****");
                Iterator<BookDetails> it2 = ob.iterator();
                while(it2.hasNext()) {
                        System.out.println(it2.next());
                }//end of loop
                System.out.println("****Iterator<E>(Java8)****");
                Iterator<BookDetails> it3 = ob.iterator();
//LambdaExpresssion attached with accept() method Consumer<T>
                it3.forEachRemaining((x)->
                {
                        System.out.println(x.toString());
                });
                System.out.println("****Spliterator<T>****");
                Spliterator<BookDetails> sp=ob.spliterator();
//LambdaExpresssion attached with accept() method Consumer<T>
```

```
                                sp.forEachRemaining((y)->
                                {
                                        System.out.println(y.toString());
                                });
                                System.out.println("****forEach()****");
                //LambdaExpresssion attached with accept() method Consumer<T>
                                ob.forEach((k)->
                                {
                                        System.out.println(k.toString());
                                });
                                break;
                        case 4:
                                System.out.println("Operations Stopped on "+name);
                                break xyz;
                        default:
                                System.out.println("Invalid Choice...");
                        }//end of switch
                        }//end of while


                }//end of loop
        }catch(Exception e) {e.printStackTrace();}
    }//end of try
        }
}
```

===