# Collection .sort all methods java.util.Collections.sort() method is present in java.util.Collections class. It is used to sort the elements present in the specified list of Collection in ascending order. It works similar to

# java.util.Arrays.sort() method but it is better than as it can sort the elements of Array as well as linked list, queue and many more present in it.

# Sorting Array : -

Arrays.sort(ArrayName);   
Arrays.sort(name, Collections.reverseOrder());

Sorting ArrayList :-

Collection.sort(ArrayName);   
Collections.sort(a, Collections.reverseOrder());

Comprable ;-

Created a custom class student(having id , name , age , and code );

To print a custom class we need to override to String method

public String toString() {  
 return " id " + id + " Name " + name + " AGE " + age + " Code " + code;  
}

Now Sorting ;-

Implement comparable interface from java.lang to custom class   
 **class** Pair **implements** Comparable<Pair> {

Which will require to implement the method compare to which will have return type s int in all cases

Thus in case of sorting with

* Id

public int compareTo(student o2) {  
 return this.id-o2.id;  
}

* Name
* public int compareTo(student o2) {  
   return this.name.compareTo(o2.name);  
  }
* Code

Code   
public int compareTo(student o2){  
 return this.code.compareTo(o2.code);  
 }

* Reverse id

reverse age   
public int compareTo(student o2){  
 return o2.id-this.id;  
}

* Age if equal then name
* public int compareTo(student o2) {  
   if (this.age != o2.age) {  
   return this.age-o2.age;  
   }  
   else {  
   return this.name.compareTo(o2.name);   
   }  
  }

Code   
package sorting;  
  
  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.Collections;  
  
  
class student implements Comparable<student>{  
 String name;  
 int id;  
 int age;  
 String code;  
  
  
 public student(String name, int id, int age, String code) {  
 this.name = name;  
 this.id = id;  
 this.age = age;  
 this.code = code;  
 }  
  
 public String toString() {  
 return " id " + id + " Name " + name + " AGE " + age + " Code " + code;  
 }  
  
 @Override  
// public int compareTo(student o2) {  
// return this.name.compareTo(o2.name);  
// }  
// public int compareTo(student o2) {  
// return this.id-o2.id;  
// }  
//public int compareTo(student o2){  
// return this.code.compareTo(o2.code);  
// }  
// public int compareTo(student o2){  
// return o2.id-this.id;  
// }  
 public int compareTo(student o2) {  
 if (this.age != o2.age) {  
 return this.age-o2.age;  
 }  
 else {  
 return this.name.compareTo(o2.name);  
 }  
 }  
  
}  
  
 public class Sorting {  
  
 public static void main(String[] args) {  
 ArrayList<student> list = new ArrayList<>();  
 //student student1 = new student()  
 student st1 = new student("rohan" ,001 , 12,"dd001");  
 list.add(st1);  
 list.add( new student("ram" ,002 , 13,"ab001"));  
 list.add( new student("Anupam" ,003 , 14,"aa001"));  
 list.add( new student("aryan" ,000 , 14,"ad001"));  
  
 System.*out*.println(list);  
 Collections.*sort*(list);  
 System.*out*.println(list);  
  
 }  
}

Comparator :- Comparator do not to implemented along with custom class and also comparable has a single method to be overridden and thus comparator can be implemented along side a new custom class or by creating a object of the instance for eg sorting by id .  
  
Comparator<student> idCompare = new Comparator<student>(){  
 @Override  
 public int compare(student o1, student o2) {  
 return o1.id-o2.id;  
 }  
};

System.*out*.println(list);  
Collections.*sort*(list,idCompare);

Now example –

import java.util.ArrayList;

import java.util.Collection;

import java.util.Collections;

import java.util.Comparator;

class student {

String name;

int id ;

int age ;

public student(String name, int id ,int age ) {

this.name = name;

this.id = id;

this.age = age ;

}

public String toString (){

return "id "+id + " NAME "+ name +" Age "+age ;

}

}

public class comparator {

public static void main(String[] args) {

ArrayList<student> list = new ArrayList<>();

list.add(new student("Rohan", 10 , 12 ));

list.add(new student("Anupam", 1,13));

list.add(new student("Shyam", 12,12 ));

list.add(new student("Bro", 9,11));

Comparator<student> idCompare = new Comparator<student>(){

@Override

public int compare(student o1, student o2) {

return o1.id-o2.id;

}

};

Comparator<student> nameCompare = new Comparator<student>(){

@Override

public int compare(student o1, student o2) {

return o1.name.compareTo(o2.name);

}

};

Comparator<student> ageCompare = new Comparator<student>(){

@Override

public int compare(student o1, student o2) {

if(o1.age!= o2.age)return o1.age-o2.age;

else{

return o1.name.compareTo(o2.name);

}

}

};

System.out.println(list);

Collections.sort(list,ageCompare);

System.out.println(list);

}

}