/\*

Java Comparator example.

This Java Comparator example describes how java.util.Comparator interface is implemented to compare Java user defined classe's objects.

These Java Comparator is passed to Collection's sorting method ( for example Collections.sort method)to perform sorting of Java user defined classe's objects.

\*/

**import** java.util.\*;

/\*

java.util.Comparator interface declares two methods,

1) public int compare(Object object1, Object object2) and

2) boolean equals(Object object)

\*/

/\*

We will compare objects of the Employee class using custom comparators

on the basis of employee age and name.

\*/

**class** Employee{

**private** **int** age;

**private** String name;

**public** **void** setAge(**int** age){

**this**.age=age;

}

**public** **int** getAge(){

**return** **this**.age;

}

**public** **void** setName(String name){

**this**.name=name;

}

**public** String getName(){

**return** **this**.name;

}

}

/\*

User defined java comaprator.

To create custom java comparator Implement Comparator interface and

define compare method.

The below given comparator compares employees on the basis of their age.

\*/

**class** AgeComparator **implements** Comparator{

**public** **int** compare(Object emp1, Object emp2){

//parameter are of type Object, so we have to downcast it to Employee objects

**int** emp1Age = ( (Employee) emp1).getAge();

**int** emp2Age = ( (Employee) emp2).getAge();

**if**( emp1Age > emp2Age )

**return** 1;

**else** **if**( emp1Age < emp2Age )

**return** -1;

**else**

**return** 0;

}

}

/\*

The below given comparator compares employees on the basis of their name.

\*/

**class** NameComparator **implements** Comparator{

**public** **int** compare(Object emp1, Object emp2){

//parameter are of type Object, so we have to downcast it to Employee objects

String emp1Name = ( (Employee) emp1 ).getName();

String emp2Name = ( (Employee) emp2 ).getName();

//uses compareTo method of String class to compare names of the employee

**return** emp1Name.compareTo(emp2Name);

}

}

/\*

This Java comparator example compares employees on the basis of

their age and name and sort it in that order.

\*/

**public** **class** JavaComparatorExample{

**public** **static** **void** main(String args[]){

/\*

Employee array which will hold employees

\*/

Employee employee[] = **new** Employee[2];

//set different attributes of the individual employee.

employee[0] = **new** Employee();

employee[0].setAge(40);

employee[0].setName("Joe");

employee[1] = **new** Employee();

employee[1].setAge(20);

employee[1].setName("Mark");

System.out.println("Order of employee before sorting is");

//print array as is.

**for**(**int** i=0; i < employee.length; i++){

System.out.println( "Employee " + (i+1) + " name :: " + employee[i].getName() + ", Age :: " + employee[i].getAge());

}

/\*

Sort method of the Arrays class sorts the given array.

Signature of the sort method is,

static void sort(Object[] object, Comparator comparator)

IMPORTANT: All methods defined by Arrays class are static. Arrays class

serve as a utility class.

\*/

/\*

Sorting array on the basis of employee age by passing AgeComparator

\*/

Arrays.sort(employee, **new** AgeComparator());

System.out.println("\n\nOrder of employee after sorting by employee age is");

**for**(**int** i=0; i < employee.length; i++){

System.out.println( "Employee " + (i+1) + " name :: " + employee[i].getName() + ", Age :: " + employee[i].getAge());

}

/\*

Sorting array on the basis of employee Name by passing NameComparator

\*/

Arrays.sort(employee, **new** NameComparator());

System.out.println("\n\nOrder of employee after sorting by employee name is");

**for**(**int** i=0; i < employee.length; i++){

                  System.out.println( "Employee " + (i+1) + " name :: " + employee[i].getName() + ", Age :: " + employee[i].getAge());

}

}

}

/\*

OUTPUT of the above given Java Comparable Example would be :

Order of employee before sorting is

Employee 1 name :: Joe, Age :: 40

Employee 2 name :: Mark, Age :: 20

Order of employee after sorting by employee age is

Employee 1 name :: Mark, Age :: 20

Employee 2 name :: Joe, Age :: 40

Order of employee after sorting by employee name is

Employee 1 name :: Joe, Age :: 40

Employee 2 name :: Mark, Age :: 20

\*/