Collection comparator pgm

Comparator is a generic interface that has this declaration:

interface Comparator<T>

Here, **T** specifies the type of objects being compared.

int compare(T *obj1*, T *obj2*)

obj1 and *obj2* are the objects to be compared.

This method returns zero if the objects are equal. It returns a positive value if *obj1* is greater than *obj2*. Otherwise, a negative value is returned.

The method can throw a **ClassCastException** if the types of the objects are not compatible for comparison.

By overriding **compare( )**, you can alter the way that objects are ordered. For example, to sort in reverse order, you can create a comparator that reverses the outcome of a comparison.

Comparator in array list

```
class Student implements Comparable<Student>{
    int rollno;
    String name;
    int age;
    Student(int rollno,String name,int age){
        this.rollno=rollno;
        this.name=name;
        this.age=age;
    }

    public int compareTo(Student st){
        if(age==st.age)
            return 0;
        else if(age>st.age)
            return 1;
        else
            return -1;
    }

}
```

```
public class TestSort1{
   public static void main(String args[]){
        ArrayList<Student> al=new ArrayList<Student>();
        al.add(new Student(101,"Vijay",23));
        al.add(new Student(106,"Ajay",27));
        al.add(new Student(105,"Jai",21));

        Collections.sort(al);
        for(Student st:al){
            System.out.println(st.rollno+" "+st.name+" "+st.age);
        }
   }

}
```

```
105     Jai     21
101     Vijay   23
106     Ajay    27
```

```
public class MyArrayListSort {

    public static void main(String a[]){

        List<Empl> list = new ArrayList<Empl>();
        list.add(new Empl("Ram",3000));
        list.add(new Empl("John",6000));
        list.add(new Empl("Crish",2000));
        list.add(new Empl("Tom",2400));
        Collections.sort(list,new MySalaryComp());
        System.out.println("Sorted list entries: ");
        for(Empl e:list){
            System.out.println(e);
        }
    }
}

class MySalaryComp implements Comparator<Empl>{

    @Override
```

```
    public int compare(Empl e1, Empl e2) {
        if(e1.getSalary() < e2.getSalary()){
            return 1;
        } else {
            return -1;
        }
    }
}

class Empl{

    private String name;
    private int salary;

    public Empl(String n, int s){
        this.name = n;
        this.salary = s;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```
        this.name = name;
    }
    public int getSalary() {
        return salary;
    }
    public void setSalary(int salary) {
        this.salary = salary;
    }
    public String toString(){
        return "Name: "+this.name+"-- Salary: "+this.salary;
    }
}
```

```
Sorted list entries:
Name: John-- Salary: 6000
Name: Ram-- Salary: 3000
Name: Tom-- Salary: 2400
Name: Crish-- Salary: 2000
```