```
 class Student implements Comparable<Student> {
        // public class Student {

        String name;
        String address;

        public String getName() {
                return name;
        }

        public Student(String name, String address) {
                this.name = name;
                this.address = address;

        }
        @Override
        public String toString() {
                return name + " :" + address;
        }


  @Override
  public int compareTo(Student o) {

          return this.name.compareTo(o.name);

  }
}

public class CollectionSortTest {
        public static void main(String[] args) {
                ArrayList<Student>   arrys   = new ArrayList<>();
                Student a= new Student("김길동","하남시" );
                Student  b= new Student("강길동","부산시" );
                Student c= new Student("나길동","제주도" );

                arrys.add(a);
                arrys.add(b);
                arrys.add(c);

                //list 정렬
                // Collections 클래스 사용해 보기
                Collections.sort(arrys);

                for(Student s : arrys)
                        System.out.println(s);


                // 지역내부클래스 , 이름이 있는 클래스작성
                 class AddressSort implements Comparator<Student>
                   {

                                                @Override
                                                public int compare(Student o1, Student o2) {

                                                        return   o1.address.compareTo(o2.address);
                                                }

                   }

                // 방법1
                Collections.sort(arrys , new AddressSort() );

                //  방법2 , 익명클래스로 익명객체 생성
                Collections.sort(arrys, new Comparator<Student>() {

                        @Override
                        public int compare(Student o1, Student o2) {

                                return   o1.address.compareTo(o2.address);
                        }

                });
                //방법3, 람다식 (익명클래스가 만들어지고 익명객체가 생성됨)
                Collections.sort(arrys, (o1,o2) ->    o1.address.compareTo(o2.address) );

                for(Student s : arrys)
                        System.out.println(s);



        }

}
```