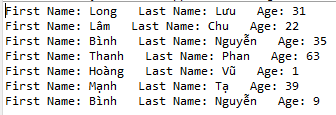
Sau đây là phần báo cáo bài kiểm tra số 2 của Nguyễn Hoàng Long – 2018602388KHMT1

Câu 1: Sắp xếp một danh sách Person, sử dụng interface Comparable, Comparator

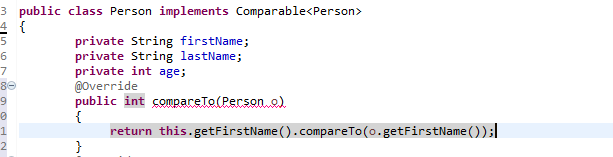
Comparable

Để sắp xếp một danh sách Person theo tên và tên đệm, sử dụng interface Comparable

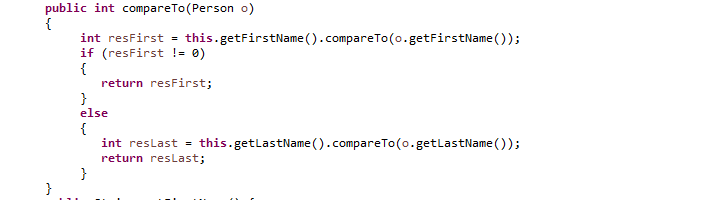
* Sau khi khởi tạo đối tượng Person với các thuộc tính firstname, lastname, age, tiến hành sinh ngẫu nhiên danh sách Person thông qua phương thức **generatePerson()** với tham số truyền vào là số lượng Person của danh sách
* Ghép ngẫu nhiên họ tên, tuổi:



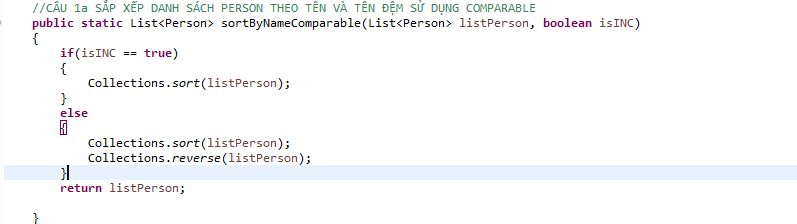
* Khi đã có danh sách Person ta tiến hành sắp xếp sử dụng Interface **Comparable**
* Lớp Person phải implements Interface **Comparable** để cài đặt **phương thức compareTo()**.
* Ta cài đặt tiêu chí để so sánh trong phương thức **compareTo()** để so sánh các đối tượng Person với nhau theo FirstName và LastName



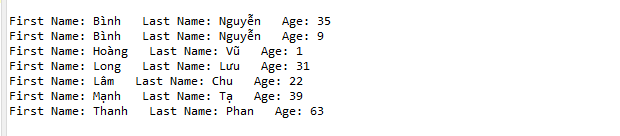
* Với trường hợp 2 FirstName giống nhau ta sử dụng tiếp **compareTo()** với LastName:



* Phương thức **Collections.sort(List list)** được sử dụng để sắp xếp các phần tử của Listperson. Với điều kiện lớp Person phải được implements interface **Comparable**



* Danh sách Person theo chiều alphabet của Firstname và Lastname

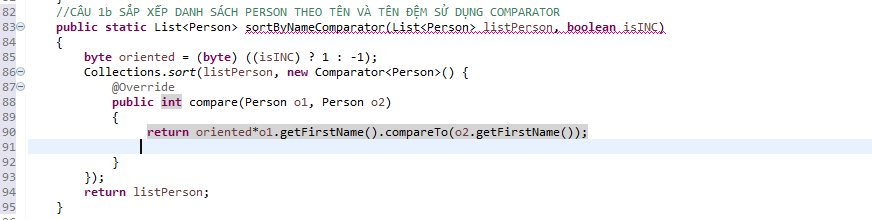


* Note: để sắp xếp danh sách Person theo chiều ngược của Alphabet, ta sử dụng phương thức **Collections.reverse(List list)** sau khi sắp xếp với **Collections.sort(List list)**

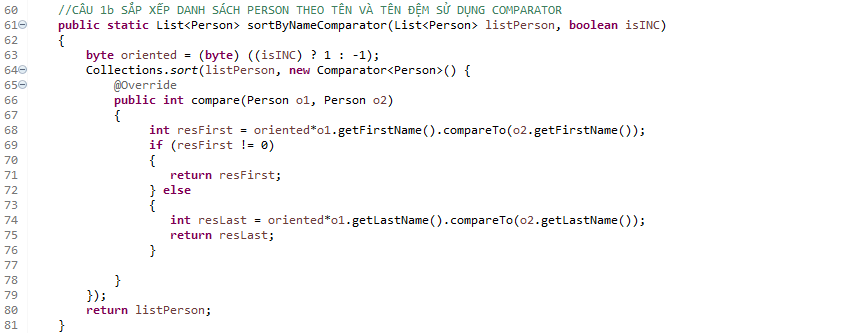
Comparator

Để sắp xếp một danh sách Person, sử dụng interface Comparator

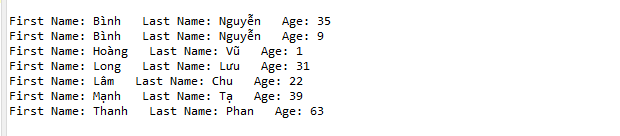
* Khi đã có danh sách Person ta tiến hành sắp xếp sử dụng Interface **Comparator**
* Interface **Comparator trong java** chứa phương thức có tên **compare(Object obj1,Object obj2)**
* Ta tạo đối tượng nặc danh **Comparator** để cài đặt **phương thức compare()**. Cài đặt tiêu chí để so sánh trong phương thức **compare()** để so sánh các đối tượng Person với nhau theo FirstName và LastName



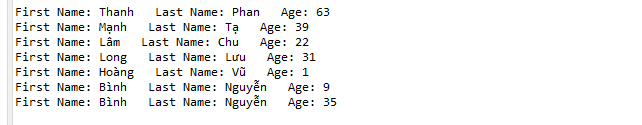
* Với trường hợp 2 FirstName giống nhau ta sử dụng tiếp **compare()** với phương thức **compareTo()** với LastName:



* Danh sách Person theo chiều alphabet của Firstname và Lastname



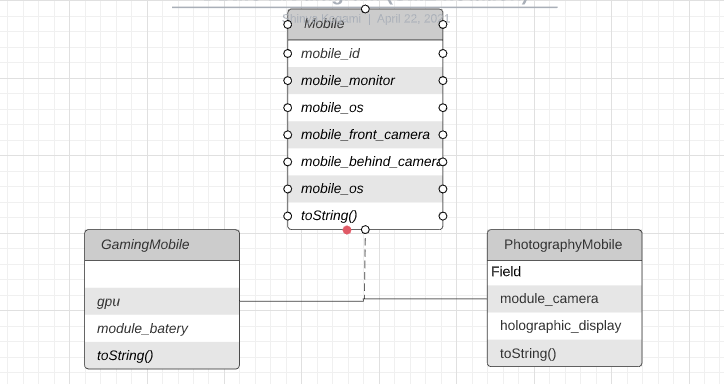
* Note: để sắp xếp danh sách Person theo chiều ngược của Alphabet, ta nhân 1 biến oriented với giá trị = 1 khi True và = -1 khi False với giá trị trả về của phương thức **compareTo()**
* Danh sách Person ngược chiều alphabet của Firstname và Lastname



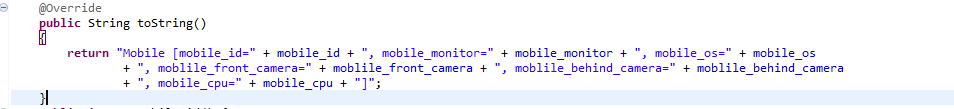
Câu 2 Cài đặt đa hình cho lớp đối tượng Mobile

Đa hình sẽ gắn liền với kế thừa Và gắn liền với ghi đè phương thức (overriding) Đa hình là nói đến một đối tượng nào đó có khả năng nhập vai thành các đối tượng khác. Vậy thì để mà một đối tượng có thể là một đối tượng nào đó, thì nó phải là đối tượng cha. Và để đối tượng cha có thể là một trong các đối tượng con ở từng hoàn cảnh, thì nó phải định nghĩa ra các phương thức để con của nó có thể ghi đè. Điều này giúp hệ thống xác định được đối tượng nào và phương thức nào thực sự đang hoạt động khi ứng dụng đang chạy.

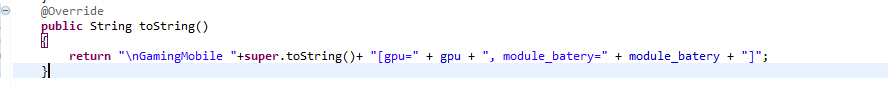
* Lớp **Mobile** là lớp cha, hai lớp con **GamingMobile** và **PhotographicMobile** đều override phương thức **toString()** từ cha.



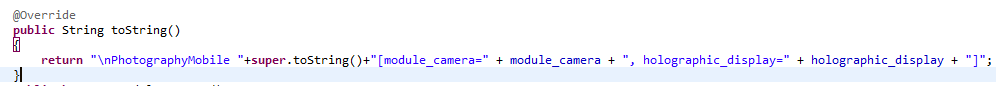
* Phương thức toString() của lớp cha **Mobile**



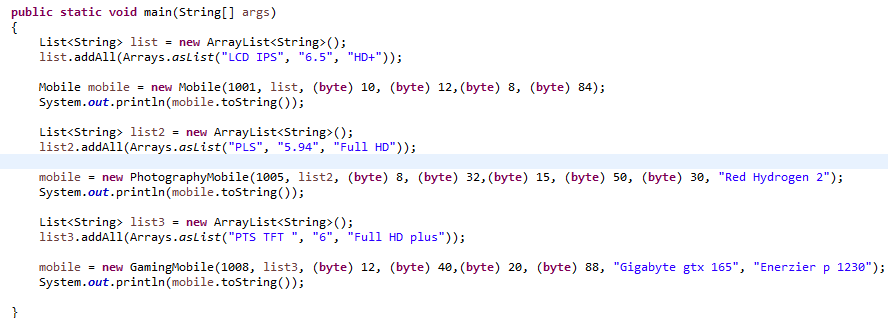
* Phương thức **toString()** của lớp con **GamingMobile** ghi đè phương thức **toString()** của lớp cha **Mobile**



* Phương thức **toString()** của lớp con **PhotographyMobile** ghi đè phương thức **toString()** của lớp cha **Mobile**



* Main Class



Đối tượng **Mobile** bản thân nó có một phương thức **toString()**. Chúng ta cho phép lớp **Mobile** có khả năng đóng vai trò là lớp con(**GamingMobile, PhotographyMobile**), bằng cách khởi tạo lại đối tượng là lớp con của nó

**Khởi tạo Mobile mobile = new PhotographyMobile ()**

Khi gọi phương thức toString() thì đối tượng mobile này sẽ gọi ra toString() của lớp con **PhotographyMobile**

Khởi tạo **Mobile mobile = new GamingMobile()**

Khi gọi phương thức toString() thì đối tượng mobile này sẽ gọi ra toString() của lớp con **GamingMobile**

* Kết quả

