**So sánh và sắp xếp trong Java**

1- Kiểu nguyên thủy và kiểu tham chiếu trong Java.

2- Stack và Heap là gì?

3- Kiểu nguyên thủy lưu trữ thế nào trên bộ nhớ

4- Kiểu tham chiếu lưu trữ thế nào trên bộ nhớ

5- Các kiểu so sánh trong Java

6- So sánh các kiểu nguyên thủy

7- So sánh các kiểu tham chiếu

7.1- Sử dụng toán tử == so sánh các kiểu tham chiếu

7.2- Sử dụng equals(..) so sánh các kiểu tham chiếu

7.3- Ghi đè method equals(Object)

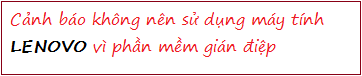
8- Ví dụ sắp xếp một mảng String

9- Các đối tượng có thể so sánh với nhau (Comparable)

10- Sắp xếp một danh sách (List)

11- Sắp xếp sử dụng bộ so sánh (Comparator)

[Công ty Vĩnh Cửu tuyển dụng lập trình viên Java](http://o7planning.org/vi/11139/cong-ty-vinh-cuu-tuyen-dung-lap-trinh-vien-java)

[](http://viettimes.vn/viet-nam/doi-song/hai-phong-dua-khuyen-nghi-khong-dung-may-tinh-do-lenovo-san-xuat-31667.html)



1- Kiểu nguyên thủy và kiểu tham chiếu trong Java.

Trước hết chúng ta cần phân biệt kiểu nguyên thủy (Primitive type) và kiểu tham chiếu (reference type) trong java.

Trong Java chúng ta có 8 kiểu nguyên thủy.

* **Kiểu nguyên thủy (Primitive type)**

|  |  |  |
| --- | --- | --- |
| **Type** | **Bit/Bytes** | **Range** |
| boolean | 1 bit | True or False |
| char | 16 bit/ 2 bytes | 0 to 65535 |
| byte | 8 bit/ 1 byte | -128 to 127 |
| short | 16 bit/ 2 bytes | -32768 to 32767 |
| int | 32 bits/ 4 bytes | -2147483648 to 2147483647 |
| long | 64 bits/ 8 bytes | -9,223,372,036,854,775,808 to -9,223,372,036,854,775,808 (-2^63 to -2^63) |
| float | 32 bits/ 4 bytes | -3.4028235 x 10^38 to 3.4028235 x 10^38 |
| double | 64 bits/ 8 bytes | -1.7976931348623157 x 10^308 to 1.7976931348623157 x 10^308 |

Tất cả các kiểu khác đều mở rộng từ Object, chúng là các kiểu tham chiếu.

2- Stack và Heap là gì?

Biến của bạn được lưu trữ trên Stack (Ngăn xếp) hoặc Heap, vậy Stack và Heap là gì?

* TODO

3- Kiểu nguyên thủy lưu trữ thế nào trên bộ nhớ

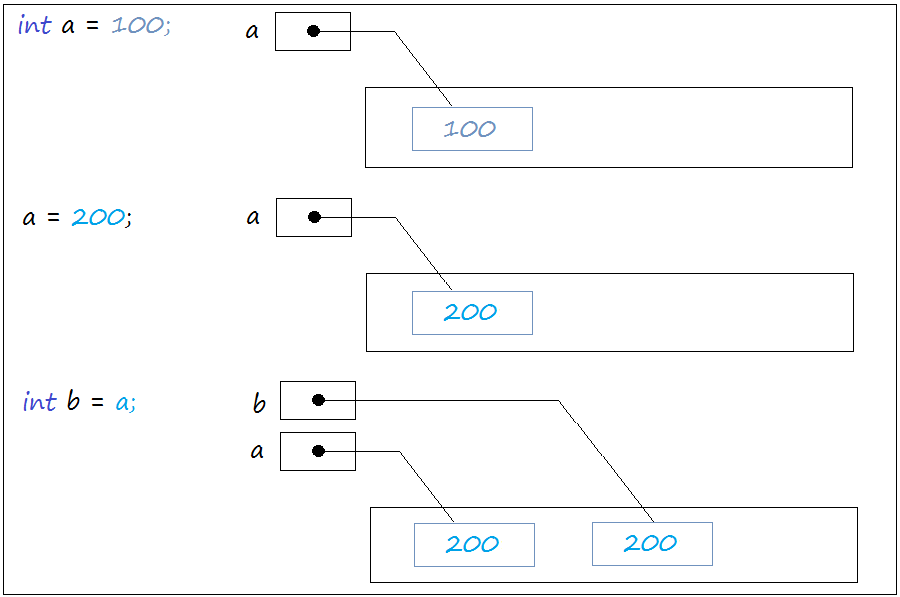
Trước hết bạn phải hiểu rằng, Java không đảm bảo rằng mỗi biến sẽ tương ứng với một vị trí trên bộ nhớ. chẳng hạn Java sẽ tối ưu theo cách biến **'i'** sẽ được lưu trữ trên bộ đăng ký (register), hoặc thậm trí không được lưu trữ ở đâu cả, nếu trình biên dịch nhận thấy rằng bạn không bao giờ sử dụng giá trị của nó, hoặc nó có thể được dõi theo thông qua code và sử dụng các giá trị phù hợp một cách trực tiếp.

Xem một đoạn code:

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // Create a variable 'a', and assin valud to it.  int a = 100;    // Assign new value  a = 200;    // Create variable b, assign value a  int b = a; |

Khi đó Java thực hiện các thao tác sau:



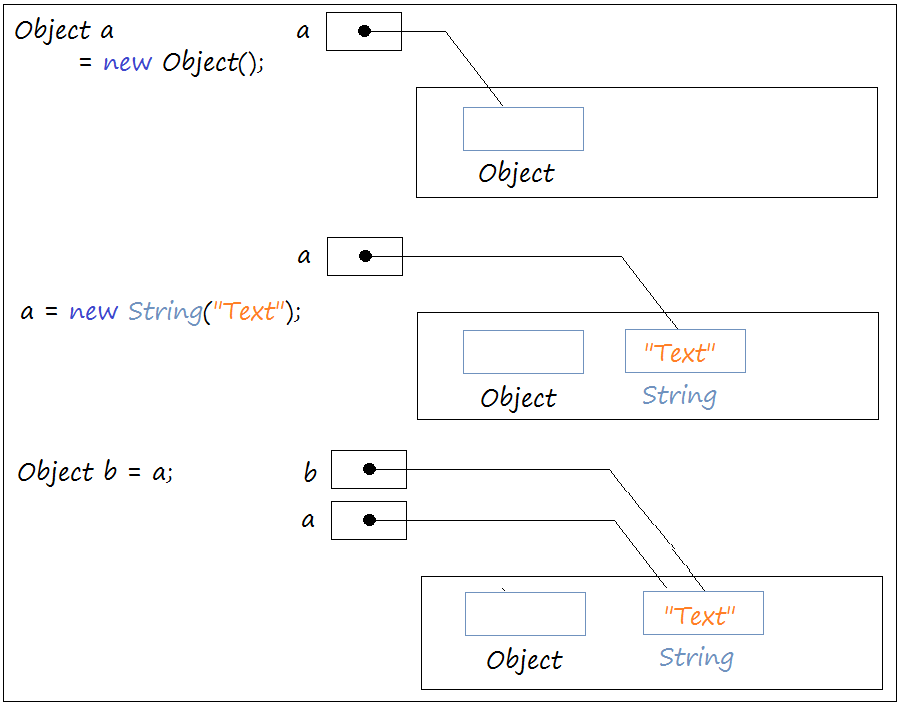
* TODO

4- Kiểu tham chiếu lưu trữ thế nào trên bộ nhớ

Khi bạn sử dụng toán tử new (Ví dụ new Object()), Java sẽ tạo ra một thực thể mới trên bộ nhớ. Bạn khai báo một biến và khởi tạo giá trị của nó thông qua toán tử new, chẳng hạn Object a = new Object(); Java sẽ tạo ra một thực thể mới trong bộ nhớ, và một tham chiếu a trỏ tới vị trí bộ nhớ của thực thể vừa được tạo ra.   
  
Khi bạn khai báo một biến b   Object b = a;  không có thực thể nào được tạo ra trong bộ nhớ, Java chỉ tạo ra một tham chiếu b, trỏ tới vị cùng vị trí mà a đang trỏ tới.

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // Khai báo và khởi tạo đối tượng.  Object a =  new Object();    // Khởi tạo lại đối tượng  a = new String("Text");    // Khai báo đối tượng 'b' và gán nó bằng đối tượng 'a'.  Object b =  a; |



* TODO

5- Các kiểu so sánh trong Java

Trong Java có 2 kiểu so sánh:

* Sử dụng toán tử ==
* Sử dụng method equals(..)

Toán tử == dùng so sánh các kiểu nguyên thủy và các kiểu tham chiếu.   
Toán tử equals(..) là method chỉ dùng cho các kiểu tham chiếu.

6- So sánh các kiểu nguyên thủy

Với kiểu nguyên thủy chúng ta chỉ có duy nhất một cách so sánh bằng toán tử ==, các kiểu nguyên thủy so sánh với nhau thông qua giá trị của chúng.

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | // Tạo ra một biến a, gán bởi giá trị 200  // Một vùng bộ nhớ (1) được tạo ra chứa giá trị 200.  int a = 200;    // Tạo ra một biến b, gán giá trị 200.  // Một vùng bộ nhớ (2) được tạo ra chứa giá trị 200.  int b = 200;    // Mặc dù 'a' và 'b' trỏ tới 2 vùng bộ nhớ khác nhau.  // So sánh a == b sẽ cho kết quả true.  // Vì kiểu nguyên thủy so sánh với nhau là giá trị.  boolean c = (a == b); |

7- So sánh các kiểu tham chiếu

7.1- Sử dụng toán tử == so sánh các kiểu tham chiếu

Khi bạn so sánh 2 đối tượng tham chiếu theo toán tử ==, có nghĩa là so sánh vị trí mà 2 đối tượng tham chiếu này trỏ tới. Về bản chất là kiểm tra xem 2 tham chiếu đó có cùng trỏ tới một thực thể trên bộ nhớ hay không.

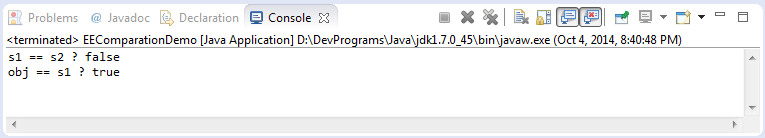
Hãy xem ví dụ:

* **ReferenceEeDemo.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | package org.o7planning.tutorial.comparation;    public class ReferenceEeDemo {     public static void main(String[] args) {         // Chú ý: Với String 2 cách khởi tạo đối tượng sau là không giống nhau:       // Bạn có thể xem trong tài liệu nói về String.       String str1 = "String 1";       String str2 = new String("String 1");         // Toán tử new tạo ra vùng bộ nhớ (1)       // Chứa String "This is text"       // Và s1 là một tham chiếu trỏ đến (1)       String s1 = new String("This is text");         // Toán tử new tạo ra vùng bộ nhớ (2)       // Chứa String "This is text"       // Và s2 là một tham chiếu trỏ đến (2)       String s2 = new String("This is text");         // Sử dụng toán tử == so sánh s1 và s2.       // Kết quả ra false.       // Nó rõ ràng khác với suy nghĩ của bạn.       // Lý do là với kiểu tham chiếu       // toán tử == so sánh vị trí mà chúng trỏ tới.       boolean e1 = (s1 == s2); // false         System.out.println("s1 == s2 ? " + e1);         // Không có toán tử new nào.       // Java tạo ra một tham chiếu có tên 'obj'       // Và trỏ tới vùng bộ nhớ mà s1 trỏ tới.       Object obj = s1;         // 2 tham chiếu 'obj' và 's1' đang cùng trỏ tới 1 vùng bộ nhớ.       // Kết quả trả về true       boolean e2 = (obj == s1); // true         System.out.println("obj == s1 ? " + e2);   }  } |

Kết quả chạy chương trình



7.2- Sử dụng equals(..) so sánh các kiểu tham chiếu

* **StringComparationDemo.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | package org.o7planning.tutorial.comparation;    public class StringComparationDemo {       public static void main(String[] args) {           String s1 = new String("This is text");           String s2 = new String("This is text");           // So sánh s1 và s2 thông qua method equals(..)         boolean e1 = s1.equals(s2);           // Kết quả sẽ là true         System.out.println("first comparation: s1 equals s2 ? " + e1);             s2 = new String("New s2 text");           boolean e2 = s1.equals(s2);           // Kết quả sẽ là false         System.out.println("second comparation: s1 equals s2 ? " + e2);     }    } |

Kết quả chạy chương trình:



7.3- Ghi đè method equals(Object)

Phương thức **equals(Object)** là phương thức có sẵn của class **Object**, mọi class con đều được thừa kế method này. Trong một số tình huống bạn có thể ghi đè method này tại class con.

* **NumberOfMedals.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

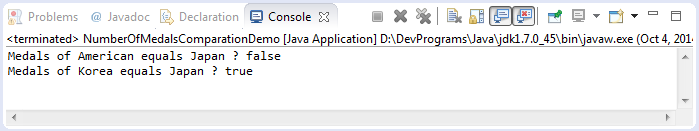
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | package org.o7planning.tutorial.comparation.equals;    // Số lượng Huy chương  public class NumberOfMedals {       // Số huy chương vàng     private int goldCount;     // Số huy chương bạc.     private int silverCount;     // Số huy chương đồng     private int bronzeCount;       public NumberOfMedals(int goldCount, int silverCount, int bronzeCount) {         this.goldCount = goldCount;         this.silverCount = silverCount;         this.bronzeCount = bronzeCount;     }       public int getGoldCount() {         return goldCount;     }       public int getSilverCount() {         return silverCount;     }       public int getBronzeCount() {         return bronzeCount;     }       // Ghi đè method equals(Object) của class Object.     @Override     public boolean equals(Object other) {         // Nếu đối tượng so sánh null trả về false.         if (other == null) {             return false;         }         // Nếu đối tượng cần so sánh không phải kiểu NumberOfMedals         // thì trả về false.         if (!(other instanceof NumberOfMedals)) {             return false;         }         // Ép kiểu         NumberOfMedals otherNoM = (NumberOfMedals) other;         // Nếu số lượng huy chương vàng, bạc, đồng bằng nhau         // trả về true.         if (this.goldCount == otherNoM.goldCount                 && this.silverCount == otherNoM.silverCount                 && this.bronzeCount == otherNoM.bronzeCount) {             return true;         }         return false;     }  } |

* **NumberOfMedalsComparationDemo.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | package org.o7planning.tutorial.comparation.equals;    public class NumberOfMedalsComparationDemo {        public static void main(String[] args) {            // Thành tích của đội Mỹ          NumberOfMedals american = new NumberOfMedals(40, 15, 15);            // Thành tích của đội Nhật          NumberOfMedals japan = new NumberOfMedals(10, 5, 20);            // Thành tích của đội Hàn Quốc          NumberOfMedals korea = new NumberOfMedals(10, 5, 20);            System.out.println("Medals of American equals Japan ? "                  + american.equals(japan));            System.out.println("Medals of Korea equals Japan ? "                  + korea.equals(japan));      }    } |

Kết quả chạy ví dụ



8- Ví dụ sắp xếp một mảng String

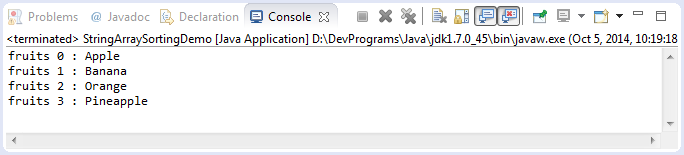
String vốn là một class mà các đối tượng của nó có thể so sánh với nhau, theo quy tắc chữ cái. Ví dụ sau đây minh họa cách sắp xếp một mảng String sử dụng các phương thức tiện ích có sẵn trong Java.

* **StringArraySortingDemo.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package org.o7planning.tutorial.sorting;    import java.util.Arrays;    public class StringArraySortingDemo {       public static void main(String[] args) {         String[] fruits = new String[] { "Pineapple", "Apple", "Orange",                 "Banana" };           // Sử dụng method tĩnh của class Arrays để sắp xếp.         // Arrays.sort(Object[])         Arrays.sort(fruits);           for (int i = 0; i < fruits.length; i++) {             System.out.println("fruits " + i + " : " + fruits[i]);         }     }  } |

Kết quả chạy ví dụ



9- Các đối tượng có thể so sánh với nhau (Comparable)

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | // Bạn viết một class mô phỏng một diễn viên (Actor).  // Bạn muốn sắp xếp thứ tự các diễn viên theo nguyên tắc sắp xếp theo họ trước rồi mới tới tên.  // Lúc này nẩy sinh ra tình huống cần phải so sánh các đối tượng Actor.    public class Actor {       // Tên       private String firstName;         // Họ       private String lastName;  } |

* **Actor.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

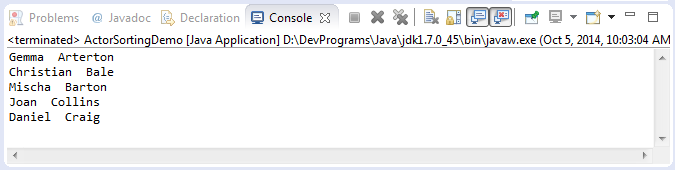
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | package org.o7planning.tutorial.sorting;    // Để so sánh được với nhau class Actor cần thi hành interface Comparable  public class Actor implements Comparable<Actor> {      private String firstName;    private String lastName;      public Actor(String firstName, String lastName) {        this.firstName = firstName;        this.lastName = lastName;    }      public String getFirstName() {        return firstName;    }      public String getLastName() {        return lastName;    }      // So sánh với một đối tượng Actor khác    // Theo nguyên tắc lastName trước, firstName sau.    @Override    public int compareTo(Actor other) {        // String có thể so sánh được với nhau        // Vì String thi hành interface Comparable<String>        // So sánh lastName giữa 2 đối tượng.        int value = this.lastName.compareTo(other.lastName);          // Nếu lastName của 2 đối tượng không bằng nhau.        if (value != 0) {            return value;        }        // Trường hợp lastName của 2 đối tượng giống nhau.        // So sánh tiếp fistName        value = this.firstName.compareTo(other.firstName);        return value;    }    } |

* **ActorSortingDemo.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | package org.o7planning.tutorial.sorting;    public class ActorSortingDemo {      public static void main(String[] args) {          Actor actor1 = new Actor("Mischa", "Barton");        Actor actor2 = new Actor("Christian", "Bale");        Actor actor3 = new Actor("Joan", "Collins");        Actor actor4 = new Actor("Gemma", "Arterton");        Actor actor5 = new Actor("Daniel", "Craig");          Actor[] actors = new Actor[] { actor1, actor2, actor3, actor4, actor5 };          // Chúng ta sử dụng một thuật toán để sắp xếp lại mảng trên.        // Sắp xếp tăng dần các Actor.        // Duyệt trên các phần tử của mảng.        for (int i = 0; i < actors.length; i++) {              for (int j = i + 1; j < actors.length; j++) {                // Nếu Actor sau nhỏ hơn Actor trước                // Thì thực hiện việc tráo đổi vị trí với nhau.                if (actors[j].compareTo(actors[i]) < 0) {                    // Sử dụng một biến tạm thời.                    Actor temp = actors[j];                    actors[j] = actors[i];                    actors[i] = temp;                }            }        }        // Bây giờ thì in ra các phần tử mảng.        for (int i = 0; i < actors.length; i++) {            System.out.println(actors[i].getFirstName() + "  "                    + actors[i].getLastName());        }      }  } |

Kết quả chạy ví dụ



Sử dụng **Arrays.sort(Object[])** để sắp xếp ví dụ trên:

* **ActorSortingDemo2.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package org.o7planning.tutorial.sorting;    import java.util.Arrays;    public class ActorSortingDemo2 {         public static void main(String[] args) {           Actor actor1 = new Actor("Mischa", "Barton");         Actor actor2 = new Actor("Christian", "Bale");         Actor actor3 = new Actor("Joan", "Collins");         Actor actor4 = new Actor("Gemma", "Arterton");         Actor actor5 = new Actor("Daniel", "Craig");           Actor[] actors = new Actor[] { actor1, actor2, actor3, actor4, actor5 };             // Sử dụng Arrays.sort(Object[]) để sắp xếp.         Arrays.sort(actors);             // Bây giờ thì in ra các phần tử mảng.         for (int i = 0; i < actors.length; i++) {             System.out.println(actors[i].getFirstName() + "  "                     + actors[i].getLastName());         }       }  } |

10- Sắp xếp một danh sách (List)

Bạn có thể tham khảo thêm tài liệu **Java Collection Framework** tại:

* <http://o7planning.org/vi/10165/huong-dan-su-dung-nen-tang-tap-hop-trong-java>

Ví dụ:

* **ListSortingDemo.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | package org.o7planning.tutorial.sorting;    import java.util.ArrayList;  import java.util.Collections;  import java.util.List;    public class ListSortingDemo {       public static void main(String[] args) {           Actor actor1 = new Actor("Mischa", "Barton");         Actor actor2 = new Actor("Christian", "Bale");         Actor actor3 = new Actor("Joan", "Collins");         Actor actor4 = new Actor("Gemma", "Arterton");         Actor actor5 = new Actor("Daniel", "Craig");           // Một danh sách chứa các phần tử có thể so sánh với nhau.         List<Actor> actors = new ArrayList<Actor>();           actors.add(actor1);         actors.add(actor2);         actors.add(actor3);         actors.add(actor4);         actors.add(actor5);           // Sử dụng Collections.sort(List)         // để sắp xếp một danh sách (List)         Collections.sort(actors);           for (Actor actor : actors) {             System.out.println(actor.getFirstName() + "  "                     + actor.getLastName());         }       }  } |

11- Sắp xếp sử dụng bộ so sánh (Comparator)

Các ví dụ trên chúng ta sắp xếp một mảng hoặc một danh sách. Bản thân các phần tử của nó có khả năng so sánh với nhau (Do thi hành interface ***Comparable***). Câu hỏi đặt ra với các đối tượng mà class của nó không thi hành interface ***Comparable***, thì có thể sắp xếp được không. Trong trường hợp này bạn cần cung cấp 1 bộ so sánh ( ***Comparator***) nó là quy tắc để sắp xếp các đối tượng kia.

* **Person.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package org.o7planning.tutorial.comparator;    public class Person {       private int age;     private String fullName;       public Person(String fullName, int age) {         this.fullName = fullName;         this.age = age;     }       public int getAge() {         return age;     }       public String getFullName() {         return fullName;     }  } |

* **PersonComparator.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

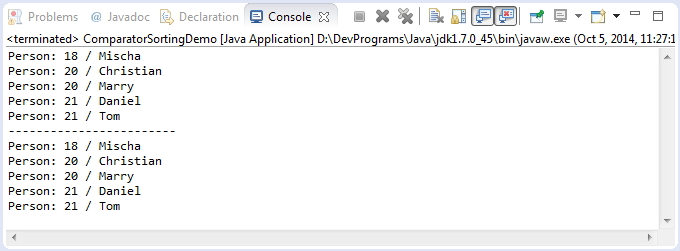
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | package org.o7planning.tutorial.comparator;    import java.util.Comparator;    // Bộ so sánh này thi hành interface Comparator<Person>  // Chuyên so sánh các đối tượng Person.  public class PersonComparator implements Comparator<Person> {      // Ghi đè method compare.    // Nêu rõ quy tắc so sánh giữa 2 đối tượng Person.    @Override    public int compare(Person o1, Person o2) {        // Hai đối tượng null coi như bằng nhau.        if (o1 == null && o2 == null) {            return 0;        }        // Nếu o1 null, coi như o2 lớn hơn        if (o1 == null) {            return -1;        }        // Nếu o2 null, coi như o1 lớn hơn.        if (o2 == null) {            return 1;        }        // Nguyên tắc:        // Sắp xếp tăng dần theo tuổi.        int value = o1.getAge() - o2.getAge();        if (value != 0) {            return value;        }        // Trường hợp bằng tuổi, so sánh tiếp fullName        // So sánh theo Alphabet.        value = o1.getFullName().compareTo(o2.getFullName());        return value;    }    } |

* **ComparatorSortingDemo.java**

[?](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | package org.o7planning.tutorial.comparator;    import java.util.ArrayList;  import java.util.Arrays;  import java.util.Collections;  import java.util.List;    public class ComparatorSortingDemo {       public static void main(String[] args) {         Person person1 = new Person("Marry", 20);         Person person2 = new Person("Tom", 21);         Person person3 = new Person("Daniel", 21);         Person person4 = new Person("Mischa", 18);         Person person5 = new Person("Christian", 20);           // Một mảng cần sắp xếp         Person[] array = new Person[] { person1, person2, person3, person4,                 person5 };           // Sắp xếp mảng sử dụng: <T> Arrays.sort(T[],Comparator<? supers T>)         // Cung cấp rõ một bộ so sánh.         Arrays.sort(array, new PersonComparator());           for (Person person : array) {             System.out.println("Person: " + person.getAge() + " / "                     + person.getFullName());         }           // Trường hợp danh sách:         List<Person> list = new ArrayList<Person>();         list.add(person1);         list.add(person2);         list.add(person3);         list.add(person4);         list.add(person5);           // Sắp xếp danh sách sử dụng:         // <T> Collections.sort(List<T>, Comparator<? supers T>)         // Cung cấp rõ một bộ so sánh.         Collections.sort(list, new PersonComparator());           for (Person person : list) {             System.out.println("Person: " + person.getAge() + " / "                     + person.getFullName());         }     }  } |

Kết quả chạy ví dụ:



**Java cơ bản**

* [Bắt đầu với Java cần những gì?](http://o7planning.org/vi/10159/bat-dau-voi-java-can-nhung-gi)
* [Hướng dẫn cài đặt và cấu hình Java](http://o7planning.org/vi/10377/huong-dan-cai-dat-va-cau-hinh-java)
* [Hướng dẫn cài đặt và cấu hình Eclipse](http://o7planning.org/vi/10379/huong-dan-cai-dat-va-cau-hinh-eclipse)
* [Học nhanh Java cho người mới bắt đầu](http://o7planning.org/vi/10161/hoc-nhanh-java-cho-nguoi-moi-bat-dau)
* [JDK Javadoc định dạng CHM](http://o7planning.org/vi/10317/jdk-javadoc-dinh-dang-chm)
* [Thừa kế và đa hình trong Java](http://o7planning.org/vi/10193/thua-ke-va-da-hinh-trong-java)
* [Abstract class và Interface trong Java](http://o7planning.org/vi/10205/abstract-class-va-interface-trong-java)
* [Access modifier trong Java](http://o7planning.org/vi/10319/access-modifier-trong-java)
* [Hướng dẫn sử dụng Java Enum](http://o7planning.org/vi/10199/huong-dan-su-dung-java-enum)
* [Hướng dẫn sử dụng Java Annotation](http://o7planning.org/vi/10197/huong-dan-su-dung-java-annotation)
* [So sánh và sắp xếp trong Java](http://o7planning.org/vi/10213/so-sanh-va-sap-xep-trong-java)
* [Hướng dẫn sử dụng Java String, StringBuffer và StringBuilder](http://o7planning.org/vi/10217/huong-dan-su-dung-java-string-stringbuffer-va-stringbuilder)
* [Hướng dẫn xử lý ngoại lệ trong Java - Java Exception Handling](http://o7planning.org/vi/10187/huong-dan-xu-ly-ngoai-le-trong-java)
* [Hướng dẫn sử dụng Java Generics](http://o7planning.org/vi/10403/huong-dan-su-dung-java-generics)
* [Hướng dẫn sử dụng nền tảng tập hợp trong Java (Java Collection Framework)](http://o7planning.org/vi/10165/huong-dan-su-dung-nen-tang-tap-hop-trong-java)
* [Hướng dẫn sử dụng luồng vào ra nhị phân - Java IO](http://o7planning.org/vi/10163/huong-dan-su-dung-luong-vao-ra-nhi-phan-java-io)
* [Hướng dẫn sử dụng luồng vào ra ký tự - Java IO](http://o7planning.org/vi/10173/huong-dan-su-dung-luong-vao-ra-ky-tu-java-io)
* [Hướng dẫn sử dụng Date, Time trong Java](http://o7planning.org/vi/10245/huong-dan-su-dung-date-time-trong-java)
* [Cú pháp và các tính năng mới trong Java 5](http://o7planning.org/vi/10103/cu-phap-va-cac-tinh-nang-moi-trong-java-5)
* [Cú pháp và các tính năng mới trong Java 8](http://o7planning.org/vi/10323/cu-phap-va-cac-tinh-nang-moi-trong-java-8)
* [Hướng dẫn sử dụng biểu thức chính quy trong Java](http://o7planning.org/vi/10175/huong-dan-su-dung-bieu-thuc-chinh-quy-trong-java)
* [Hướng dẫn lập trình đa luồng trong Java - Java Multithreading](http://o7planning.org/vi/10269/huong-dan-lap-trinh-da-luong-trong-java)
* [Thư viện điều khiển các loại cơ sở dữ liệu khác nhau trong Java](http://o7planning.org/vi/10227/thu-vien-dieu-khien-cac-loai-co-so-du-lieu-khac-nhau-trong-java)
* [Hướng dẫn sử dụng Java JDBC kết nối cơ sở dữ liệu](http://o7planning.org/vi/10167/huong-dan-su-dung-java-jdbc)
* [Hướng dẫn nén và giải nén trong Java](http://o7planning.org/vi/10195/huong-dan-nen-va-giai-nen-trong-java)
* [Hướng dẫn sử dụng Java Reflection](http://o7planning.org/vi/10155/huong-dan-su-dung-java-reflection)
* [Hướng dẫn gọi phương thức từ xa với Java RMI](http://o7planning.org/vi/10171/huong-dan-goi-phuong-thuc-tu-xa-voi-java-rmi)
* [Hướng dẫn lập trình Java Socket](http://o7planning.org/vi/10393/huong-dan-lap-trinh-java-socket)
* [Các nền tảng nào bạn nên chọn để lập trình ứng dụng Java Desktop?](http://o7planning.org/vi/10177/cac-nen-tang-nao-ban-nen-chon-de-lap-trinh-ung-dung-java-desktop)
* [Hướng dẫn sử dụng Eclipse JFace](http://o7planning.org/vi/10251/huong-dan-su-dung-eclipse-jface)
* [Hướng dẫn sử dụng Java Commons IO](http://o7planning.org/vi/10141/huong-dan-su-dung-java-commons-io)
* [Hướng dẫn sử dụng Java Commons Email](http://o7planning.org/vi/10147/huong-dan-su-dung-java-commons-email)
* [Hướng dẫn sử dụng Java Commons Logging](http://o7planning.org/vi/10151/huong-dan-su-dung-java-commons-logging)

**Tài liệu mới nhất**

* [Tạo ra hệ thống Bảng từ các class Entity trong Hibernate](http://o7planning.org/vi/11223/tao-ra-he-thong-bang-tu-cac-class-entity-trong-hibernate)
* [Tạo ứng dụng Java RESTful Client với Jersey Client](http://o7planning.org/vi/11217/tao-ung-dung-java-restful-client-voi-jersey-client)
* [Ví dụ CRUD đơn giản với Java RESTful Web Service](http://o7planning.org/vi/11207/vi-du-crud-don-gian-voi-java-restful-web-service)
* [Chạy các nhiệm vụ nền theo lịch trình trong Spring MVC](http://o7planning.org/vi/11131/chay-cac-nhiem-vu-nen-theo-lich-trinh-trong-spring-mvc)
* [RESTClient công cụ gỡ rối cho RESTful Web Service](http://o7planning.org/vi/11203/restclient-cong-cu-go-roi-cho-restful-web-service)
* [Hướng dẫn lập trình Java RESTful Web Service cho người mới bắt đầu](http://o7planning.org/vi/11199/huong-dan-lap-trinh-java-restful-web-service-cho-nguoi-moi-bat-dau)
* [Hướng dẫn sử dụng Java SWT Radio Button](http://o7planning.org/vi/11189/huong-dan-su-dung-java-swt-radio-button)
* [Hướng dẫn sử dụng Java SWT Text](http://o7planning.org/vi/11193/huong-dan-su-dung-java-swt-text)
* [Hướng dẫn sử dụng JavaFX HTMLEditor](http://o7planning.org/vi/11113/huong-dan-su-dung-javafx-htmleditor)
* [Kiếm tiền chuyên nghiệp với Ezoic - đối tác của Google](http://o7planning.org/vi/11187/kiem-tien-chuyen-nghiep-voi-ezoic-doi-tac-cua-google)