Sử dụng Comparator trong Java

Cả TreeSet và TreeMap đều lưu giữ các phần tử trong thứ tự đã được sắp xếp. Tuy nhiên, chính Comparator định nghĩa chính xác ý nghĩa của *sorted order*.

Comparator Interface định nghĩa 2 phương thức: **compare()** và **equals()**. Phương thức compare() so sánh 2 phần tử về thứ tự, được hướng dẫn tiếp theo:

Phương thức compare trong Java

```
int compare(Object obj1, Object obj2)
```

Trong đó, obj1 và obj2 là các đối tượng để được so sánh. Phương thức này trả về 0 nếu các đối tượng là cân bằng. Nó trả về một giá trị dương nếu obj1 lớn hơn obj2. Nếu không thì, một giá trị âm được trả về.

Bằng việc ghi đè compare(), bạn có thể lọc theo cách mà các đối tượng được xếp thứ tự. Ví dụ, để sắp xếp trong thứ tự đảo ngược, bạn có thể tạo một Comparator mà đảo ngược kết quả của lần so sánh.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài: **Ví dụ về Collection trong Java**.

Phương thức equals trong Java

Phương thức equals trong Java kiểm tra có hay không một đối tượng là cân bằng với comparator đang triệu hồi:

```
boolean equals(Object obj)
```

obj là đối tượng để được kiểm tra về sự cân bằng. Phương thức trả về true nếu cả obj và đối tượng đang triệu hồi là các đối tượng Comparator và có cùng thứ tự. Nếu không thì, nó trả về false.

Việc ghi đè equals() là không cần thiết, và các comparator đơn giản nhất sẽ không làm điều này.

Ví dụ

```
class Dog implements Comparator<Dog>, Comparable<Dog>{
   private String name;
   private int age;
```

```
   Dog(){
   }

   Dog(String n, int a){
      name = n;
      age = a;
   }

   public String getDogName(){
      return name;
   }

   public int getDogAge(){
      return age;
   }

   // Overriding the compareTo method
   public int compareTo(Dog d){
      return (this.name).compareTo(d.name);
   }

   // Overriding the compare method to sort the age
   public int compare(Dog d, Dog d1){
      return d.age - d1.age;
   }
}

public class Example{

   public static void main(String args[]){
      // Takes a list o Dog objects
      List<Dog> list = new ArrayList<Dog>();
```

```
      list.add(new Dog("Shaggy",3));
      list.add(new Dog("Lacy",2));
      list.add(new Dog("Roger",10));
      list.add(new Dog("Tommy",4));
      list.add(new Dog("Tammy",1));
      Collections.sort(list);// Sorts the array list

      for(Dog a: list)//printing the sorted list of names
         System.out.print(a.getDogName() + ", ");

      // Sorts the array list using comparator
      Collections.sort(list, new Dog());
      System.out.println(" ");
      for(Dog a: list)//printing the sorted list of ages
         System.out.print(a.getDogName() +"  : "+
                    a.getDogAge() + ", ");
   }
}
```

Nó sẽ cho kết quả sau:

```
Lacy, Roger, Shaggy, Tammy, Tommy,
Tammy  : 1, Lacy  : 2, Shaggy  : 3, Tommy  : 4, Roger  : 10,
```

Ghi chú: Xếp thứ tự các lớp Array là tương tự như các Collection trong Java.