**package** com.java2novice.treemap;

**import** java.util.Comparator; **import** java.util.Set; **import** java.util.TreeMap;

**public** **class** MyTMCompUserDefine {

**public** **static** **void** main(String a[]) {

// By using name comparator (String comparison)

TreeMap<Empl, String> tm = **new** TreeMap<Empl, String>(**new** MyNameComp());

tm.put(**new** Empl("Ram", 3000), "RAM");

tm.put(**new** Empl("John", 6000), "JOHN");

tm.put(**new** Empl("Crish", 2000), "CRISH");

tm.put(**new** Empl("Tom", 2400), "TOM");

Set<Empl> keys = tm.keySet();

**for**(Empl key : keys){System.***out***.println(key + " ==> " + tm.get(key)); }

System.***out***.println("===================================");

// By using salary comparator (int comparison)

TreeMap<Empl, String> trmap = **new** TreeMap<Empl, String>(**new** MySalaryComp());

trmap.put(**new** Empl("Ram", 3000), "RAM");

trmap.put(**new** Empl("John", 6000), "JOHN");

trmap.put(**new** Empl("Crish", 2000), "CRISH");

trmap.put(**new** Empl("Tom", 2400), "TOM");

Set<Empl> ks = trmap.keySet();

**for** (Empl key : ks){System.***out***.println(key + " ==> " + trmap.get(key));}

} }

**class** MyNameComp **implements** Comparator<Empl> {

@Override

**public** **int** compare(Empl e1, Empl e2) {

**return** e1.getName().compareTo(e2.getName());

}

}

**class** MySalaryComp **implements** Comparator<Empl> {

@Override

**public** **int** compare(Empl e1, Empl e2) {

**if** (e1.getSalary() > e2.getSalary()) {**return** 1;}

**else** **if** (e1.getSalary() < e2.getSalary()) {**return** -1;}

**return** 0;

} }

**class** Empl { **private** String name; **private** **int** salary;

**public** Empl(String n, **int** s) { **this**.name = n; **this**.salary = s; }//**getter & setter**

**public** String toString(){**return** "Name: " + **this**.name + "-- Salary: " + **this**.salary; }

} **Output:**

Name: Crish-- Salary: 2000 ==> CRISH

Name: John-- Salary: 6000 ==> JOHN

Name: Ram-- Salary: 3000 ==> RAM

Name: Tom-- Salary: 2400 ==> TOM

===================================

Name: Crish-- Salary: 2000 ==> null

Name: Tom-- Salary: 2400 ==> null

Name: Ram-- Salary: 3000 ==> null

Name: John-- Salary: 6000 ==> null

**Whatif MySalaryComp is not returning 0 as below And trmap’s JOHN has same3000 salary.**

**class** MySalaryComp **implements** Comparator<Empl> {

@Override

**public** **int** compare(Empl e1, Empl e2) {

**if** (e1.getSalary() > e2.getSalary()) { **return** 1; }

**else** {**return** -1; } } }

AA {Name: Crish-- Salary: 2000=CRISH, Name: John-- Salary: 6000=JOHN, Name: Ram-- Salary: 3000=RAM, Name: Tom-- Salary: 2400=TOM}

Name: Crish-- Salary: 2000 ==> CRISH

Name: John-- Salary: 6000 ==> JOHN

Name: Ram-- Salary: 3000 ==> RAM

Name: Tom-- Salary: 2400 ==> TOM

===================================

AA {Name: Crish-- Salary: 2000=CRISH, Name: Tom-- Salary: 2400=TOM, Name: John-- Salary: 3000=JOHN, Name: Ram-- Salary: 3000=RAM}

Name: Crish-- Salary: 2000 ==> null

Name: Tom-- Salary: 2400 ==> null

Name: John-- Salary: 3000 ==> null

Name: Ram-- Salary: 3000 ==> null

It will store duplicate salary because its DS is linklist unlike hashTable DS but fail to when sysout, it is unable to decide.

<http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap>

# [Sorting custom data structure on Key in TreeMap](http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap)

|  |  |  |  |
| --- | --- | --- | --- |
| up vote1down vote[favorite](http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap) | I am trying to sort a TreeMap on key. Key is some custom DataStructure having int, List, String, etc. The member on which I am expecting a sort has some duplicates. Let's say that member is Rank. More than 1 object can have same rank.  Simplified version example:  NOTE: in the CompareTo method below 0 is not returned intentionally to NOT ignore duplicates.(Please correct me if this is not the right way to avoid duplicates)  import java.util.TreeMap;  public class TreeTest {  public static void main(String[] args) {  TreeMap<Custom,String> t = new TreeMap<Custom,String>();  Custom c1 = new Custom();  c1.setName("a");  c1.setRank(0);  Custom c2 = new Custom();  c2.setName("b");  c2.setRank(1);  Custom c3 = new Custom();  c3.setName("c");  c3.setRank(0);  t.put(c1, "first");  t.put(c2, "Second");  t.put(c3, "Third");  System.out.println(t.keySet());  for(Custom c:t.keySet()){  System.out.println(t.get(c));  }  }  }  And Custom Object  package com.example.ui;  public class Custom implements Comparable<Custom>{  int rank;  String name;  public int getRank() {  return rank;  }  public void setRank(int rank) {  this.rank = rank;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  @Override  public int hashCode() {  final int prime = 31;  int result = 1;  result = prime \* result + ((name == null) ? 0 : name.hashCode());  result = prime \* result + rank;  return result;  }  @Override  public boolean equals(Object obj) {  if (this == obj)  return true;  if (obj == null)  return false;  if (getClass() != obj.getClass())  return false;  Custom other = (Custom) obj;  if (name == null) {  if (other.name != null)  return false;  } else if (!name.equals(other.name))  return false;  if (rank != other.rank)  return false;  return true;  }  // 0 is not returned intentionally to NOT ignore duplicates.  public int compareTo(Custom o) {  if(o.rank>this.rank)  return 1;  if(o.rank==this.rank)  return -1;  return -1;  }  }  Output::  [com.example.ui.Custom@fa0, com.example.ui.Custom@fbe, com.example.ui.Custom@f80]  null  null  null  Expected: First, Second, Third based on Rank 0,1,0 respectively.  I looked at couple of examples on Google. Most of them were basic usage on TreeMap sort using keys or values with primitive datatypes, but none with duplicates when sorting member is a part of custom key DataStructure.  Please help?  [java](http://stackoverflow.com/questions/tagged/java) [sorting](http://stackoverflow.com/questions/tagged/sorting) [treemap](http://stackoverflow.com/questions/tagged/treemap" \o ")   |  |  | | --- | --- | | [share](http://stackoverflow.com/q/7385189/6707834)[edit](http://stackoverflow.com/posts/7385189/edit) | asked Sep 12 '11 at 8:35  [Kunal](http://stackoverflow.com/users/467444/kunal)  **709**3917 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | First thing, your print statement should be System.out.println(c.getName()); – [medopal](http://stackoverflow.com/users/208446/medopal" \o "6,003 reputation) [Sep 12 '11 at 8:40](http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap#comment8917409_7385189) |   add a comment |

## 2 Answers

[active](http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap?answertab=active#tab-top)[oldest](http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap?answertab=oldest#tab-top)[votes](http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap?answertab=votes#tab-top)

|  |  |  |  |
| --- | --- | --- | --- |
| up vote5down voteaccepted | The problem is that your implementation of compareTo is not consistent with equals, which is required by [TreeMap](http://download.oracle.com/javase/6/docs/api/java/util/TreeMap.html). From the API docs:  Note that the ordering maintained by a sorted map (whether or not an explicit comparator is provided) must be consistent with equals if this sorted map is to correctly implement the Map interface.  One possible consistent implementation would be to first compare by rank and then by name if the rank values are equal. For two instances of Custom with equal ranks and identical names you should not expect to be able to store them both as keys within the same Map - **This violates the contract of Map**.  public int compareTo(Custom o) {  int ret = this.rank - o.rank;  // Equal rank so fall back to comparing by name.  if (ret == 0) {  ret = this.name.compareTo(o.name);  }  return ret;  }   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/7385234/6707834)[edit](http://stackoverflow.com/posts/7385234/edit) | answered Sep 12 '11 at 8:40  [Adamski](http://stackoverflow.com/users/127479/adamski)  **38.5k**980130 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 1 |  | | +1. Another problem is that you return -1 for two objects having the same rank, and this means that you are in a situation where you have A < B and B < A at the same time, which violates the contract of Comparable. – [JB Nizet](http://stackoverflow.com/users/571407/jb-nizet) [Sep 12 '11 at 8:54](http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap#comment8917592_7385234) | | |  |  | | --- | --- | |  |  | | This assumes you can't get an overflow from this.rank - o.rank. Its possibly a safe assumption here. Even though its an int it probably doesn't need to be. – [Peter Lawrey](http://stackoverflow.com/users/57695/peter-lawrey) [Sep 12 '11 at 9:04](http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap#comment8917740_7385234) | | |  |  | | --- | --- | |  |  | | @Peter - Fair point and to be honest I did it this way to keep the example code small. – [Adamski](http://stackoverflow.com/users/127479/adamski" \o "38,479 reputation) [Sep 12 '11 at 10:30](http://stackoverflow.com/questions/7385189/sorting-custom-data-structure-on-key-in-treemap#comment8919087_7385234) | |