Sort a HashMap by Key or Value using Java Stream

**Map.Entry.*comparingByKey*()🡸 Sort by key**

**Map.Entry.comparingByValue() 🡸 Sort by Value**

**Sort by Key**

**Simplest way**

Map<String, Integer> map = Map.*of*("Ram", 10, "Shyam",20,"Sophie", 9, "Hari", 23, "Radheshyam", 13);

map.forEach( (k,v) -> System.***out***.println("Key: "+k+"<===>"+"Value: "+v));

*//* ***Sort a Map by Key*****Map<String, Integer> lm = new LinkedHashMap<>();**  
map.entrySet().stream()**.sorted(Map.Entry.*comparingByKey*())** **.forEach(a -> lm.put(a.getKey(), a.getValue()));**  
  
System.***out***.println(**"LM: "** + lm);

**Complex way**  
  
*//* ***Sort a Map by key***LinkedHashMap<String, Integer> lm1 = map.entrySet().stream() **.sorted(Map.Entry.*comparingByKey*())  
 .collect(Collectors.*toMap*(e -> e.getKey(), e -> e.getValue(),  
 (e1, e2) -> e1, LinkedHashMap::new));**  
System.***out***.println(**"LM 1: "** + lm1);

**Sort by Value**

**Simplest way**

*// Sort a Map by value***Map<String, Integer> lm = new LinkedHashMap<>();**  
map.entrySet().stream() **.sorted(Map.Entry.*comparingByValue*())** .**forEach(a -> lm.put(a.getKey(), a.getValue()));**  
  
System.***out***.println(**"LM: "** + lm);  
  
**Complex way**

LinkedHashMap<String, Integer> lm1 = map.entrySet().stream()  
 **.sorted(Map.Entry.*comparingByValue*())** .collect(Collectors.*toMap*(e -> e.getKey(), e -> e.getValue(),  
 (e1, e2) -> e1, LinkedHashMap::**new**));  
System.***out***.println(**"LM 1: "** + lm1);

**Sort a Map by Key as Object**

**public class** Employee {  
  
 **private** String **name**;  
 **private int age**;  
 **private** String **city**;  
 **private** String **gender**;  
  
 **public** constructor(String name, **int** age, String city, String gender) {}

//get()/set() Methods

}

Map<Employee,Integer> map = Map.*of*(

**new** Employee("Ram",11, "Hyderabad", "Male"), 9,

**new** Employee("Shyam",12, "Bangalore", "Male"), 8,

**new** Employee("Sophie",13, "Cuttack", "FeMale"), 13,

**new** Employee("Hari",14, "Chennai", "Male"), 50,

**new** Employee("Radheshyam",15, "Bhubaneswar", "Male"), 41,

**new** Employee("Chitra",16, "Coimbatore", "FeMale"), 23

);

**Sort Map by Key by City Simplest Way**Map<Employee,Integer> linkedMap = **new** LinkedHashMap<>();

**Comparator<Employee> byCity = (Employee e1, Employee e2) -> e1.getCity().compareTo(e2.getCity());**

map.entrySet().stream().sorted(Map.Entry.*comparingByKey*(byCity))

.forEach(m -> linkedMap.put(m.getKey(), m.getValue()));

linkedMap.forEach( (k,v) -> System.***out***.println(k+"<===>"+v));

**Sort Map by Key by Name Complex Way****Comparator<Employee> byName = (Employee e1, Employee e2) -> e1.getName().compareTo(e2.getName());** LinkedHashMap lm1 = map.entrySet().stream()  
 **.sorted(Map.Entry.*comparingByKey*(byName))** .collect(Collectors.*toMap*(e -> e.getKey(), e -> e.getValue(),  
 (e1,e2) -> e1, LinkedHashMap::**new**));  
 System.***out***.println(lm1);  
}

**Sum of Integer in an array or list**

**int**[] a = **new int**[]{1,2,3,4,5};  
**int** sum = **Arrays.*stream*(a).reduce((x,y) -> x+y).getAsInt();**  
System.***out***.println(sum);

**int** sumVal = **Arrays.*stream*(a).sum();**  
System.***out***.println(**"Sum Val : "**+sumVal);  
  
List<Integer> list = Arrays.*asList*(1,2,3,4,5);  
**int** sum1 = **list.stream().reduce((x,y) -> x+y).get();**  
System.***out***.println(sum1);  
  
**sum = (int) list.stream().collect(Collectors.*summingInt*( e -> e.intValue()));**System.***out***.println(sum);

**Max Min using Java Stream**

**int**[] a = **new int**[]{55, 45, 23, 12, 34, 78};  
**int** max = Arrays.*stream*(a).max().getAsInt();  
System.***out***.println(**"Max Value: "** + max);  
  
List<Integer> list = Arrays.*asList*(55, 45, 23, 12, 34, 78);  
  
**int** maxVal = **list.stream().max(Comparator.*naturalOrder*()).get();**  
System.***out***.println(**"Max: "** + maxVal);  
  
maxVal = **list.stream().max(Comparator.*comparing*(Integer::*valueOf*)).get();**  
System.***out***.println(**"Max: "** + maxVal);  
  
maxVal = **list.stream().max(Comparator.*comparing*(e -> e)).get();**System.***out***.println(**"Max: "** + maxVal);

**Comparator<Integer> cmp = (Integer a, Integer b) -> a.compareTo(b);  
maxVal = list.stream().max(cmp).get();**  
System.***out***.println(maxVal);

*// For List of Strings*List<String> slist = Arrays.*asList*(**"a"**, **"b"**, **"c"**, **"d"**);  
String maxStr = **slist.stream().max(Comparator.*naturalOrder*()).get();**  
System.***out***.println(maxStr);  
maxStr = **slist.stream().max(Comparator.*comparing*(e -> e)).get();**  
System.***out***.println(maxStr);  
maxStr = **slist.stream().max(Comparator.*comparing*(String::*valueOf*)).get();**  
System.***out***.println(maxStr);  
  
List<Person> plist = Arrays.*asList*(**new** Person(**"Ram"**, 23, 1000),  
 **new** Person(**"Hari"**, 63, 3000),  
 **new** Person(**"Shyam"**, 23, 7000));

If we make from int to Integer, it can be written lik this.

**Comparator<Person> bySal = (Person p1, Person p2) -> p1.getSal().compareTo(p2.getSal());**plist.stream().max(bySal).get();

*// Who is drawing highest salary***Comparator<Person> bySal = Comparator.*comparing*(p -> p.getSal());**  
Person maxSalPerson = **plist.stream().max(bySal).get();**  
System.***out***.println(maxSalPerson);  
  
**Comparator<Person> bySal1 = (Person p1, Person p2) ->  
 Integer.*valueOf*(p1.getSal()).compareTo(Integer.*valueOf*(p2.getSal()));**  
Person maxSalPerson1 = **plist.stream().max(bySal1).get();**  
System.***out***.println(maxSalPerson1);  
  
*// Who is seniorMost***Comparator<Person> byAge = Comparator.*comparing*(e -> e.getAge());**  
Person senior1 = **plist.stream().max(byAge).get();**  
System.***out***.println(senior1);

**Sort Map by key and Value**

**Sort a Map<String, String> by Key**

Map<String, String> lmap = **new** LinkedHashMap<>();

Map<String, String> map = Map.*of*("Ram","Bangalore", "Hari", "Chennai",  
 "Shyam","Hyderabad", "Pili", "Balangir");

**Sort Key in Ascending Order**  
Comparator<String> byStr = Comparator.*comparing*(e->e);  
**map.entrySet().stream().sorted(Map.Entry.*comparingByKey*(byStr))**.forEach(e->{  
 lmap.put(e.getKey(),e.getValue());  
});  
System.***out***.println(lmap);

**Sort Key in Descending Order**  
Comparator<String> byStr = Comparator.*comparing*(e->e, Comparator.*reverseOrder*());  
**map.entrySet().stream().sorted(Map.Entry.*comparingByKey*(byStr)).**forEach(e->{  
 lmap.put(e.getKey(),e.getValue());  
});  
System.***out***.println(lmap);

The above can also be written as below, since the data type is String

**map.entrySet().stream().sorted(Map.Entry.*comparingByKey*(Comparator.*reverseOrder*()))**

.forEach(e->{  
 lmap.put(e.getKey(),e.getValue());  
});  
System.***out***.println(lmap);

**For Better understanding**

**Comparator<String> byStr = Comparator.*comparing*(e->e);**  
**map.entrySet().stream().sorted(Map.Entry.*comparingByKey*(byStr))**.forEach(e->{  
 lmap.put(e.getKey(),e.getValue());  
});  
System.***out***.println(lmap);

**Sort a Map<String, String> by Value**

Map<String, String> lmap = **new** LinkedHashMap<>();

Map<String, String> map = Map.*of*("Ram","Bangalore", "Hari", "Chennai",  
 "Shyam","Hyderabad", "Pili", "Balangir");  
  
// Sort Map<String, String> by value  
**map.entrySet().stream().sorted(Map.Entry.*comparingByValue*())**.forEach(e-> {  
 lmap.put(e.getKey(),e.getValue());  
});  
System.***out***.println(lmap);

map.entrySet().stream().sorted(Map.Entry.*comparingByKey*(byCity)).forEach(e -> lmap.put(e.getKey(), e.getValue()));

**In case of custom object like Person**

**public class** Person {  
 **private** String name;  
 **private int** age;  
 **private** String city;  
 **private** Integer sal;  
  
 **public** Person(String name, **int** age, Integer sal, String city) {  
 **this**.name = name;  
 **this**.age = age;  
 **this**.sal = sal;  
 **this**.city = city;  
 }

}

**Sort Map by Key as per person's city**

Map<Person, String> map = Map.*of*(  
 **new** Person("Ram", 23, 1000, "Bangalore"), "Enginer",  
 **new** Person("Nimi", 27, 2000, "Hyderabd"), "Doctor",  
 **new** Person("Pili", 18, 1000, "Secunderabd"), "Nurse",  
 **new** Person("Shyam", 47, 1000, "Delhi"), "Architect"  
);  
  
Map<Person, String> lmap = **new** LinkedHashMap<>();  
Map<Person, String> map = Map.*of*(  
 **new** Person("Ram", 23, 1000, "Bangalore"), "Enginer",  
 **new** Person("Nimi", 27, 2000, "Hyderabd"), "Doctor",  
 **new** Person("Pili", 18, 1000, "Secunderabd"), "Nurse",  
 **new** Person("Shyam", 47, 1000, "Delhi"), "Architect"  
);

**Sort key in Ascending order as per Person's city**  
Comparator<Person> byCity = Comparator.*comparing*(e->e.getCity());  
map.entrySet().stream().sorted(Map.Entry.*comparingByKey*(byCity)).forEach(e-> {  
 lmap.put(e.getKey(),e.getValue());  
});  
System.***out***.println(lmap);

**Sort key in Descending order as per Person's city**Comparator<Person> byCity = Comparator.*comparing*(e->e.getCity(), Comparator.*reverseOrder*());  
map.entrySet().stream().sorted(Map.Entry.*comparingByKey*(byCity)).forEach(e-> {  
 lmap.put(e.getKey(),e.getValue());  
});  
System.***out***.println(lmap);

**Sort map by Key as per person's age**  
**Comparator<Person> byAge = (Person p1, Person p2) -> Integer.*valueOf*(p1.getAge()).compareTo(Integer.*valueOf*(p2.getAge()));**  
**map.entrySet().stream().sorted(Map.Entry.*comparingByKey*(byAge))**.forEach(e-> {  
 lmap.put(e.getKey(),e.getValue());  
});  
System.***out***.println(lmap);

**forEach()** method performs an action for each element of this stream. For parallel stream, this operation does not guarantee to maintain order of the stream.

**forEachOrdered()** method performs an action for each element of this stream, guaranteeing that each element is processed in encounter order for streams that have a defined encounter order.

String str = "Radheshyam Hari Gopal";  
System.***out***.println("\*\*\*\*forEach without using parallel\*\*\*\*");  
str.chars().forEach(s -> System.***out***.print((**char**) s)); // **Radheshyam Hari Gopal**  
System.***out***.println("**\n**\*\*\*\*forEach with using parallel\*\*\*\*");  
  
str.chars().parallel().forEach(s -> System.***out***.print((**char**) s)); // **airH phlasG maRyoahed**  
System.***out***.println("**\n**\*\*\*\*forEachOrdered with using parallel\*\*\*\*");  
  
str.chars().parallel().forEachOrdered(s -> System.***out***.print((**char**)s));// **Radheshyam Hari Gopal**