**How to convert Map into List**

=>We can convert Map into List in many ways as your need. Collection and Set classes given many method to convert Map into List.

**Return Type**  **MethodName**

=>Collection<Values> values() =>

Returns a [**Collection**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html) view of the values contained in this map.

=>Set<Key> keySet() =>

Returns a [**Set**](https://docs.oracle.com/javase/8/docs/api/java/util/Set.html) view of the keys contained in this map.

=>Set< Map.Entry <Key, Value>> entrySet() =>

Returns a [**Set**](https://docs.oracle.com/javase/8/docs/api/java/util/Set.html) view of the mappings contained in this map.

**CASE:=> 1**

**=>**We can Convert Map into List by using these two methods keySet()

and values(). In this approach first we create HashMap object with values.

Map<String,String> map= new HashMap<String,String> ();

map.put(“Hello”,”World”);

map.put(“Apple ”,”3.14”);

map.put(“Another”,”Element”);

s.o.p(map);

=>The keys as a List can be obtained by creating a new ArrayList from a Set

returned by map.keySet() method.

List<String> list1=new ArrayList<String>(map.keySet());

System.out.println(list1);

The result of getting the List of keys

Apple

Another

Hello

=>While the values as a List can be obtained creating a new ArrayList from

a Collection returned by the map.values() method.

List<String> list2 =new ArrayList<String>(m.values());

System.out.printn(list2);

The result of getting the List of values

3.14

Element

World

=>Assuming map is your instance of Map.

* map.values() will return a Collection containing all of the map’s value.
* map.keySet() will return a Set containing all of the map’s key.
* map.entrySet() gives you a Collection of Map.Entry object containing both key and values. We can than transform this into any object like such as new ArrayList(map.entrySet())
* Java 8 stream API

List<values>=map.values().stream().collect(Collectors.toList());

**CASE:=> 2:**

In this approach we use entrySet() method to convert Map into List.

First we have to convert Map into Set then after we can convert Set into List.

Map<String , String > map = new HapshMap<String , String>;

map.put("one","java");

map.put("two" ,"spring");

Set<Entry<String,String>> set = map.entrySet();

List<Entry<String , String>> list = new ArrayList<Entry<String , String>> (set);

for(Entry<String , String> entry : list ) {

System.out.println(entry.getKey());

System.out.println(entry.getValue());

}

Example:=1

import java.util.ArrayList;

import java.util.Collection;

import java.util.HashMap;

 import java.util.Iterator;

 import java.util.List;

import java.util.Map.Entry;

 import java.util.Set;

/\*\*

\* Java Program to Convert HashMap into ArrayList in Java

\*/

 public class MapToList {

 public static void main(String... args) {

HashMap<String, Integer> schoolAgeCriteria =

new HashMap<String, Integer>();

 // preparing HashMap with keys and values schoolAgeCriteria.put("NursuryClass age criteria", 3); schoolAgeCriteria.put("KinderGarden1 age criteria ", 4); schoolAgeCriteria.put("KinderGarden2 age criteria ", 5); schoolAgeCriteria.put("PrimarySchool age criteria", 6);

System.out.println("Size of schoolAgeCriteria Map: " + schoolAgeCriteria.size());

// 1st Example : Converting HashMap keys into ArrayList Set<String> keySet = schoolAgeCriteria.keySet();

 List<String> schoolKeyList = new ArrayList<String>(keySet);

System.out.println("Size of Key list from Map: " + schoolKeyList.size());

// print list element

System.out.println("Printing HashMap keys from converted list : ");

for (String key : schoolKeyList) { System.out.println(key); } // 2nd Example : Converting HashMap Values into

ArrayList Collection<Integer> values = schoolAgeCriteria.values();

 List<Integer> schoolValueList = new ArrayList<Integer>(values);

 System.out.println("Size of Value list from Map: " + schoolValueList.size());

 // print values from list

 System.out.println("Printing HashMap values from converted list :");

for (Integer value : schoolValueList) { System.out.println(value);

}

// 3rd Example : Converting HashMap into ArrayList using Entry Set Set<Entry<String, Integer>> set = schoolAgeCriteria.entrySet();

List<Entry<String, Integer>> schoolAgeCriteriaList = new ArrayList<>(set);

Iterator<Entry<String, Integer>> it = schoolAgeCriteriaList.iterator();

while (it.hasNext()) { Entry<String, Integer> entry = it.next();

 System.out.println("Entry from converted list : " + entry);

} } }

Output Size of schoolAgeCriteria Map: 4

 Size of Key list from Map: 4

 Printing HashMap keys from converted list :

PrimarySchool age criteria

KinderGarden1 age criteria

KinderGarden2 age criteria

NursuryClass age criteria

Size of Value list from Map: 4

Printing HashMap values from converted list : 6

4 5 3

Entry from converted list : PrimarySchool age criteria=6

Entry from converted list : KinderGarden1age criteria =4

 Entry from converted list : KinderGarden2 age criteria =5

 Entry from converted list : NursuryClass age criteria=3

**Program 1 to Sort Map by key in Ascending order by implementing Comparator interface and overriding its compare method in java.**

**Logic to Sort Map by key in Ascending order in java**

=>Implement **Comparator** interface and override its **compare** method in java.

=>Obtain map.entrySet() in **set**, convert it into **list** (we have converted set to

list because Collections’s sort method can accept only **list** type as parameter).

=>Call Collections.sort and pass **list** [i.e. **listOfentrySet**]as parameter.

*=>Collections.sort internally calls* [*Arrays.sort*](http://www.javamadesoeasy.com/2015/04/arrayssort-to-sort-arrays-by.html)*,*

*=>Arrays.Sort() internally calls* [*Merge Sort*](http://javamadesoeasy.blogspot.in/2015/01/merge-sort.html)*.*

=>Merge sort calls overridden **compare** method of **Comparator** interface

for **comparison of** **keys**.

=>Ultimately **listOfentrySet** will contain entry (key-value) pairs **sorted on basis**

**of keys** in java**.**

**Note:** If number of elements is less than 7 then [Insertion Sort](http://www.javamadesoeasy.com/2015/01/insertion-sort.html) is used rather than [*Merge Sort*](http://javamadesoeasy.blogspot.in/2015/01/merge-sort.html). (because in case elements are less than 7 it offers better time complexity)

**Code:**

|  |
| --- |
| **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.Comparator;  **import** java.util.LinkedHashMap;  **import** java.util.List;  **import** java.util.Map;  **import** java.util.Map.Entry;  **import** java.util.Set;  **class** SortByKeyAscending **implements** **Comparator<Map.Entry<Integer, Integer>>**{     @Override  **public** **int** **compare**( Map.Entry<Integer,Integer> entry1, Map.Entry<Integer,Integer> entry2){  **return (entry1.getKey()).compareTo( entry2.getKey() );**     }  }  **public** **class** SortMapByKeyAscendingExample {  **public** **static** **void** main(String...a){         Map<Integer, Integer> map = **new** LinkedHashMap<Integer, Integer>();         map.put(4, 1);         map.put(2, 1);         map.put(3, 1);         map.put(5, 1);           Set<Entry<Integer, Integer>> entrySet = map.entrySet();         List<Entry<Integer, Integer>> listOfentrySet = **new** ArrayList<Entry<Integer, Integer>>(entrySet);           System.*out*.print("Before sorting by key : ");  **for**(Map.Entry<Integer, Integer> entry:**listOfentrySet**){          System.*out*.print(entry.getKey()+"="+entry.getValue()+"  ");         }    **Collections.*sort*(listOfentrySet, new SortByKeyAscending());**           System.*out*.print("\nAfter sorting by key(ascending): ");  **for**(Map.Entry<Integer, Integer> entry:**listOfentrySet**)          System.*out*.print(entry.getKey()+"="+entry.getValue()+"  ");     }  }  /\*OUTPUT  Before sorting by key : 4=1  2=1  3=1  5=1  After sorting by key(ascending): 2=1  3=1  4=1  5=1  \*/ |

*Program 2 to Sort Map by key in Descending order by* ***implementing Comparator*** *interface and* ***overriding*** *its* ***compare*** *method in java*

|  |
| --- |
| **class** SortByKeyDescending **implements** **Comparator<Map.Entry<Integer, Integer>>**{    @Override  **public** **int** **compare**( Map.Entry<Integer,Integer> entry1, Map.Entry<Integer,Integer> entry2){  **return (entry2.getKey()).compareTo( entry1.getKey() );**    }  } |

OUTPUT will be >

|  |
| --- |
| /\*OUTPUT  Before sorting by key : 4=1  2=1  3=1  5=1  After sorting by key(descending) : 5=1  4=1  3=1  2=1  \*/ |

**How to convert List into Map**

**=>**There are many ways we can converts List into Map.

**Before java 8**

Example:->

private Map<String, Choice> toMap(List books) {

final Map hashMap = new HashMap<>();

for (final Book book : books) {

hashMap.put(book.getISBN(), book);

}

return hashMap;

}

**Java 8 using Lambdas**

=>We can convert Map into List in Java 8 by using lambda expression and Stream API.

Map<String, Book> result = books.stream()

.collect(Collectors.toMap(book -> book.getISBN, book -> book));

=>In above code example, the stream() method return a stream of Book object from the List and then I have used collect() method of Stream class to collect all elements. All the magic of how to collect elements happening in this method.  
  
=>I have passed the method Collectors.toMap(), which means elements will be collected in a Map, where the key will be ISBN code and value will be the object itself. We have used a [lambda expression](http://javarevisited.blogspot.com/2015/01/how-to-use-lambda-expression-in-place-anonymous-class-java8.html) to simplify the.

**Using Java 8 method reference**:

=>You can further simply the code in Java 8 by using method reference, as shown below:

Map<String, Book> result = books.stream()

.collect(Collectors.toMap(Book::getISBN, b -> b));

=>Here we have called the getISBN() method using [method reference](http://java67.blogspot.com/2014/11/java-8-comparator-example-using-lambda-expression.html) instead of using a lambda expression.