**HashMap Examples**

Below example shows how to read add elements from HashMap. The method keySet() returns all key entries as a set object. Iterating through each key, we can get corresponding value object.

import java.util.HashMap;

import java.util.Set;

public class MyHashMapRead {

 public static void main(String a[]){

 HashMap<String, String> hm = new HashMap<String, String>();

 //add key-value pair to hashmap

 hm.put("first", "FIRST INSERTED");

 hm.put("second", "SECOND INSERTED");

 hm.put("third","THIRD INSERTED");

 System.out.println(hm);

 Set<String> keys = hm.keySet();

 for(String key: keys){

 System.out.println("Value of "+key+" is: "+hm.get(key));

 }

 }

}

Output:

{second=SECOND INSERTED, third=THIRD INSERTED, first=FIRST INSERTED}

Value of second is: SECOND INSERTED

Value of third is: THIRD INSERTED

Value of first is: FIRST INSERTED

Below example shows how to copy another collection to HashMap. putAll() method helps us to copy another collections to HashMap object.

import java.util.HashMap;

public class MyHashMapCopy {

 public static void main(String a[]){

 HashMap<String, String> hm = new HashMap<String, String>();

 //add key-value pair to hashmap

 hm.put("first", "FIRST INSERTED");

 hm.put("second", "SECOND INSERTED");

 hm.put("third","THIRD INSERTED");

 System.out.println(hm);

 HashMap<String, String> subMap = new HashMap<String, String>();

 subMap.put("s1", "S1 VALUE");

 subMap.put("s2", "S2 VALUE");

 hm.putAll(subMap);

 System.out.println(hm);

 }

}

Output:

{second=SECOND INSERTED, third=THIRD INSERTED, first=FIRST INSERTED}

{s2=S2 VALUE, s1=S1 VALUE, second=SECOND INSERTED, third=THIRD INSERTED, first=FIRST INSERTED}

Below example shows how to find whether specified value exists or not. By using containsValue() method you can find out the value existance.

import java.util.HashMap;

public class MyHashMapValueSearch {

 public static void main(String a[]){

 HashMap<String, String> hm = new HashMap<String, String>();

 //add key-value pair to hashmap

 hm.put("first", "FIRST INSERTED");

 hm.put("second", "SECOND INSERTED");

 hm.put("third","THIRD INSERTED");

 System.out.println(hm);

 if(hm.containsValue("SECOND INSERTED")){

 System.out.println("The hashmap contains value SECOND INSERTED");

 } else {

 System.out.println("The hashmap does not contains value SECOND INSERTED");

 }

 if(hm.containsValue("first")){

 System.out.println("The hashmap contains value first");

 } else {

 System.out.println("The hashmap does not contains value first");

 }

 }

}

Output:

{second=SECOND INSERTED, third=THIRD INSERTED, first=FIRST INSERTED}

The hashmap contains value SECOND INSERTED

The hashmap does not contains value first

Below example shows how to find whether specified key exists or not. By using containsKey() method you can find out the key existance.

import java.util.HashMap;

public class MyHashMapKeySearch {

 public static void main(String a[]){

 HashMap<String, String> hm = new HashMap<String, String>();

 //add key-value pair to hashmap

 hm.put("first", "FIRST INSERTED");

 hm.put("second", "SECOND INSERTED");

 hm.put("third","THIRD INSERTED");

 System.out.println(hm);

 if(hm.containsKey("first")){

 System.out.println("The hashmap contains key first");

 } else {

 System.out.println("The hashmap does not contains key first");

 }

 if(hm.containsKey("fifth")){

 System.out.println("The hashmap contains key fifth");

 } else {

 System.out.println("The hashmap does not contains key fifth");

 }

 }

}

Output:

{second=SECOND INSERTED, third=THIRD INSERTED, first=FIRST INSERTED}

The hashmap contains key first

The hashmap does not contains key fifth

Below example shows how to get all keys from the given HashMap. By calling keySet() method, you can get set object with all key values.

import java.util.HashMap;

import java.util.Set;

public class MyHashMapKeys {

 public static void main(String a[]){

 HashMap<String, String> hm = new HashMap<String, String>();

 //add key-value pair to hashmap

 hm.put("first", "FIRST INSERTED");

 hm.put("second", "SECOND INSERTED");

 hm.put("third","THIRD INSERTED");

 System.out.println(hm);

 Set<String> keys = hm.keySet();

 for(String key: keys){

 System.out.println(key);

 }

 }

}

Output:

{second=SECOND INSERTED, third=THIRD INSERTED, first=FIRST INSERTED}

second

third

first

Below example shows how to get all key-value pair as Entry objects. Entry class provides getter methods to access key-value details. The method entrySet() provides all entries as set object.

import java.util.HashMap;

import java.util.Map.Entry;

import java.util.Set;

public class MyHashMapEntrySet {

 public static void main(String a[]){

 HashMap<String, String> hm = new HashMap<String, String>();

 //add key-value pair to hashmap

 hm.put("first", "FIRST INSERTED");

 hm.put("second", "SECOND INSERTED");

 hm.put("third","THIRD INSERTED");

 System.out.println(hm);

 //getting value for the given key from hashmap

 Set<Entry<String, String>> entires = hm.entrySet();

 for(Entry<String,String> ent:entires){

 System.out.println(ent.getKey()+" ==> "+ent.getValue());

 }

 }

}

Output:

{second=SECOND INSERTED, third=THIRD INSERTED, first=FIRST INSERTED}

second ==> SECOND INSERTED

third ==> THIRD INSERTED

first ==> FIRST INSERTED

Below example shows how to avoid duplicate user defined objects as a key from HashMap. You can achieve this by implementing equals and hashcode methods at the user defined objects.

import java.util.HashMap;

import java.util.Set;

public class MyDuplicateKeyEx {

 public static void main(String a[]){

 HashMap<Price, String> hm = new HashMap<Price, String>();

 hm.put(new Price("Banana", 20), "Banana");

 hm.put(new Price("Apple", 40), "Apple");

 hm.put(new Price("Orange", 30), "Orange");

 printMap(hm);

 Price key = new Price("Banana", 20);

 System.out.println("Adding duplicate key...");

 hm.put(key, "Grape");

 System.out.println("After adding dulicate key:");

 printMap(hm);

 }

 public static void printMap(HashMap<Price, String> map){

 Set<Price> keys = map.keySet();

 for(Price p:keys){

 System.out.println(p+"==>"+map.get(p));

 }

 }

}

class Price{

 private String item;

 private int price;

 public Price(String itm, int pr){

 this.item = itm;

 this.price = pr;

 }

 public int hashCode(){

 int hashcode = 0;

 hashcode = price\*20;

 hashcode += item.hashCode();

 return hashcode;

 }

 public boolean equals(Object obj){

 if (obj instanceof Price) {

 Price pp = (Price) obj;

 return (pp.item.equals(this.item) && pp.price == this.price);

 } else {

 return false;

 }

 }

 public String getItem() {

 return item;

 }

 public void setItem(String item) {

 this.item = item;

 }

 public int getPrice() {

 return price;

 }

 public void setPrice(int price) {

 this.price = price;

 }

 public String toString(){

 return "item: "+item+" price: "+price;

 }

}

Output:

item: Apple price: 40==>Apple

item: Orange price: 30==>Orange

item: Banana price: 20==>Banana

Adding duplicate key...

After adding dulicate key:

item: Apple price: 40==>Apple

item: Orange price: 30==>Orange

item: Banana price: 20==>Grape

Below example shows how to delete user defined objects as a key from HashMap. You can achieve this by implementing equals and hashcode methods at the user defined objects.

import java.util.HashMap;

import java.util.Set;

public class MyDeleteKeyObject {

 public static void main(String a[]){

 HashMap<Price, String> hm = new HashMap<Price, String>();

 hm.put(new Price("Banana", 20), "Banana");

 hm.put(new Price("Apple", 40), "Apple");

 hm.put(new Price("Orange", 30), "Orange");

 printMap(hm);

 Price key = new Price("Banana", 20);

 System.out.println("Deleting key...");

 hm.remove(key);

 System.out.println("After deleting key:");

 printMap(hm);

 }

 public static void printMap(HashMap<Price, String> map){

 Set<Price> keys = map.keySet();

 for(Price p:keys){

 System.out.println(p+"==>"+map.get(p));

 }

 }

}

class Price{

 private String item;

 private int price;

 public Price(String itm, int pr){

 this.item = itm;

 this.price = pr;

 }

 public int hashCode(){

 System.out.println("In hashcode");

 int hashcode = 0;

 hashcode = price\*20;

 hashcode += item.hashCode();

 return hashcode;

 }

 public boolean equals(Object obj){

 System.out.println("In equals");

 if (obj instanceof Price) {

 Price pp = (Price) obj;

 return (pp.item.equals(this.item) && pp.price == this.price);

 } else {

 return false;

 }

 }

 public String getItem() {

 return item;

 }

 public void setItem(String item) {

 this.item = item;

 }

 public int getPrice() {

 return price;

 }

 public void setPrice(int price) {

 this.price = price;

 }

 public String toString(){

 return "item: "+item+" price: "+price;

 }

}

Output:

item: Apple price: 40==>Apple

item: Orange price: 30==>Orange

item: Banana price: 20==>Banana

Deleting key...

After deleting key:

item: Apple price: 40==>Apple

item: Orange price: 30==>Orange

Below example shows how to search user defined objects as a key from HashMap. You can achieve this by implementing equals and hashcode methods at the user defined objects.

import java.util.HashMap;

import java.util.Set;

public class MyObjectKeySearch {

public static void main(String a[]){

 HashMap<Price, String> hm = new HashMap<Price, String>();

 hm.put(new Price("Banana", 20), "Banana");

 hm.put(new Price("Apple", 40), "Apple");

 hm.put(new Price("Orange", 30), "Orange");

 printMap(hm);

 Price key = new Price("Banana", 20);

 System.out.println("Does key available? "+hm.containsKey(key));

 }

 public static void printMap(HashMap<Price, String> map){

 Set<Price> keys = map.keySet();

 for(Price p:keys){

 System.out.println(p+"==>"+map.get(p));

 }

 }

}

class Price{

 private String item;

 private int price;

 public Price(String itm, int pr){

 this.item = itm;

 this.price = pr;

 }

 public int hashCode(){

 System.out.println("In hashcode");

 int hashcode = 0;

 hashcode = price\*20;

 hashcode += item.hashCode();

 return hashcode;

 }

 public boolean equals(Object obj){

 System.out.println("In equals");

 if (obj instanceof Price) {

 Price pp = (Price) obj;

 return (pp.item.equals(this.item) && pp.price == this.price);

 } else {

 return false;

 }

 }

 public String getItem() {

 return item;

 }

 public void setItem(String item) {

 this.item = item;

 }

 public int getPrice() {

 return price;

 }

 public void setPrice(int price) {

 this.price = price;

 }

 public String toString(){

 return "item: "+item+" price: "+price;

 }

}

Output:

item: Apple price: 40==>Apple

item: Orange price: 30==>Orange

item: Banana price: 20==>Banana

Does key available? true