



## ***Java SE 8 Programming Language***

# **Lab Guides**

Document Code	25e-BM/HR/HDCV/FSOFT
Version	1.1
Effective Date	20/11/2012

**RECORD OF CHANGES**

No	Effective Date	Change Description	Reason	Reviewer	Approver
1.	01/Oct/2018	Add the new labs	Create new	DieuNT1	VinhNV
2.	01/Jun/2019	Update template	Fsoft template	DieuNT1	VinhNV

## Contents

Unit 8: Generics and Collections .....	4
Knowledge Summary.....	4
Lab Guide 1: Use HashMap.....	5
Objectives:.....	5
Problem Descriptions:.....	5
Guidelines:.....	5



CODE: JPL.S.L402  
TYPE: SHORT  
LOC:  
DURATION: 60 MINUTES

## Unit 8: Generics and Collections

### Knowledge Summary

#### HashMap

- A HashMap cannot contain duplicate keys.
- Java HashMap allows null values and the null key.
- HashMap is an unordered collection. It does not guarantee any specific order of the elements.
- Java HashMap is not thread-safe. You must explicitly synchronize concurrent modifications to the HashMap.

#### Some methods from HashMap

Method	Description
void clear()	It is used to remove all of the mappings from this map.
boolean isEmpty()	It is used to return true if this map contains no key-value mappings.
Object clone()	It is used to return a shallow copy of this HashMap instance: the keys and values themselves are not cloned.
Set entrySet()	It is used to return a collection view of the mappings contained in this map.
Set keySet()	It is used to return a set view of the keys contained in this map.
V put(Object key, Object value)	It is used to insert an entry in the map.
void putAll(Map map)	It is used to insert the specified map in the map.
V putIfAbsent(K key, V value)	It inserts the specified value with the specified key in the map only if it is not already specified.
V remove(Object key)	It is used to delete an entry for the specified key.
boolean remove(Object key, Object value)	It removes the specified values with the associated specified keys from the map.

## Lab Guide 1: Use HashMap

### Objectives:

This lab guide helps trainees know how to use HashMap in order to perform some operations:

- Create a HashMap
- Access keys and modify their associated value in a HashMap
- Remove keys from a HashMap
- Obtain the entrySet, keySet, and values from a HashMap
- Iterate over a HashMap

### Problem Descriptions:

Create a Java Project named **JPL.S.L402** in Eclipse.

Create package **fa.training.model** that contains:

- Employee class

Create package **fa.training.hashmapdemo** that contains:

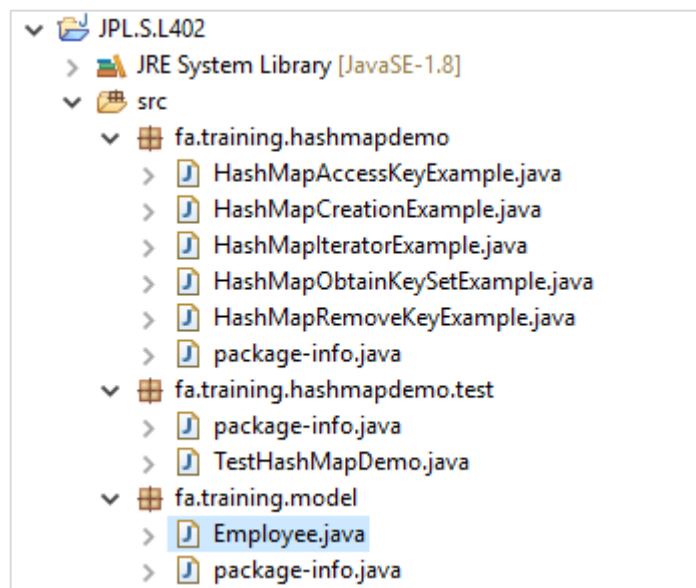
- HashMapCreationExample class
- HashMapAccessKeyExample class
- HashMapRemoveKeyExample class
- HashMapObtainKeySetExample class
- HashMapIteratorExample class

Create package **fa.training.hashmapdemo.test** that contains:

- TestHashMapDemo

### Guidelines:

**Step 1:** Create a project struture like this:



**Step 2: Create Employee class**

```
1. package fa.training.model;
2.
3. /**
4.  * @author hoabt2
5.  *
6.  */
7. public class Employee {
8.
9.     private Integer id;
10.    private String name;
11.    private String city;
12.
13.    /**
14.     * @param id
15.     * @param name
16.     * @param city
17.     */
18.    public Employee(Integer id, String name, String city) {
19.        super();
20.        this.id = id;
21.        this.name = name;
22.        this.city = city;
23.    }
24.    /**
25.     * @return the id
26.     */
27.    public Integer getId() {
28.        return id;
29.    }
30.    /**
31.     * @param id the id to set
32.     */
33.    public void setId(Integer id) {
34.        this.id = id;
35.    }
36.    /**
37.     * @return the name
38.     */
39.    public String getName() {
40.        return name;
41.    }
42.    /**
43.     * @param name the name to set
44.     */
45.    public void setName(String name) {
46.        this.name = name;
47.    }
48.    /**
49.     * @return the city
50.     */
51.    public String getCity() {
52.        return city;
53.    }
54.    /**
55.     * @param city the city to set
56.     */
57.    public void setCity(String city) {
58.        this.city = city;
59.    }
60.
61.    @Override
62.    public String toString() {
63.        return "Employee{" + "name='" + name + '\'' + ", city='" + city + '\'' + '}';
64.    }
65.
66. }
67.
```

**Step 3: Create HashMapCreationExample class**

```
1. package fa.training.hashmapdemo;
2.
3. import java.util.HashMap;
4. import java.util.Map;
5.
6. import fa.training.model.Employee;
7.
8. /**
9.  * Examples of how to create HashMap
10.  *
11.  * @author hoabt2
12.  *
13.  */
14. public class HashMapCreationExample {
15.
16.     /**
17.      * Create a HashMap
18.      *
19.      */
20.     public void createHashMap() {
21.
22.         System.out.println("createHashMap() !!!");
23.
24.         // Creating a HashMap
25.         Map<String, Integer> numberMapping = new HashMap<>();
26.
27.         // Adding key-value pairs to a HashMap
28.         numberMapping.put("One", 1);
29.         numberMapping.put("Two", 2);
30.         numberMapping.put("Three", 3);
31.
32.         // Add a new key-value pair only if the key does not exist in
33.         // the HashMap, or is mapped to `null`
34.         numberMapping.putIfAbsent("Four", 4);
35.
36.         System.out.println(numberMapping);
37.     }
38.
39.     /**
40.      * Create a HashMap that contains employees
41.      *
42.      */
43.     public void createEmployeeMap() {
44.         System.out.println("createEmployeeMap() !!!");
45.
46.         Map<Integer, Employee> employeesMap = new HashMap<>();
47.
48.         employeesMap.put(1001, new Employee(1001, "Peter", "London"));
49.         employeesMap.put(1002, new Employee(1002, "David", "New York"));
50.         employeesMap.put(1003, new Employee(1003, "Jack Ma", "Hong Kong"));
51.
52.         System.out.println(employeesMap);
53.     }
54. }
55.
```

**Step 4: Create HashMapAccessKeyExample class**

```
1. package fa.training.hashmapdemo;
2.
3. import java.util.HashMap;
4. import java.util.Map;
5.
6. /**
7.  * Examples of how to access key from HashMap
8.  *
```

```
9.  * @author hoabt2
10. *
11. */
12. public class HashMapAccessKeyExample {
13.
14. /**
15.  * Access keys from HashMap
16.  *
17.  */
18. public void accessKeys() {
19.
20.     System.out.println("accessKeys() !!!");
21.
22.     Map<String, String> userCityMapping = new HashMap<>();
23.
24.     // Check if a HashMap is empty
25.     System.out.println("is userCityMapping empty? : " +
26.         userCityMapping.isEmpty());
27.
28.     userCityMapping.put("John", "New York");
29.     userCityMapping.put("Rajeev", "Bengaluru");
30.     userCityMapping.put("Steve", "London");
31.
32.     System.out.println("userCityMapping HashMap : " +
33.         userCityMapping);
34.
35.     // Find the size of a HashMap
36.     System.out.println("We have the city information of " +
37.         userCityMapping.size() + " users");
38.
39.     String userName = "Steve";
40.     // Check if a key exists in the HashMap
41.     if(userCityMapping.containsKey(userName)) {
42.         // Get the value assigned to a given key in the HashMap
43.         String city = userCityMapping.get(userName);
44.         System.out.println(userName + " lives in " + city);
45.     } else {
46.         System.out.println("City details not found for user " +
47.             userName);
48.     }
49.
50.     // Check if a value exists in a HashMap
51.     if(userCityMapping.containsValue("New York")) {
52.         System.out.println("There is a user in
53.             the userCityMapping who lives in New York");
54.     } else {
55.         System.out.println("There is no user in
56.             the userCityMapping who lives in New York");
57.     }
58.
59.
60.     // Modify the value assigned to an existing key
61.     userCityMapping.put(userName, "California");
62.     System.out.println(userName + " moved to a new city " +
63.         userCityMapping.get(userName) + ", New userCityMapping : " +
64.         userCityMapping);
65.
66.     // The get() method returns `null` if
67.     // the specified key was not found in the HashMap
68.     System.out.println("Lisa's city : " + userCityMapping.get("Lisa"));
69. }
70. }
```



**Step 5:** Create **HashMapRemoveKeyExample** class

```
1. package fa.training.hashmapdemo;
2.
3. import java.util.HashMap;
4. import java.util.Map;
5.
6. /**
7.  * Examples of how to remove key from HashMap
8.  *
9.  * @author hoabt2
10.  *
11.  */
12. public class HashMapRemoveKeyExample {
13.
14.     /**
15.      * Remove keys from HashMap
16.      *
17.      */
18.     public void removeKeys() {
19.
20.         System.out.println("removeKeys() !!!");
21.
22.         Map<String, String> husbandWifeMapping = new HashMap<>();
23.         husbandWifeMapping.put("Jack", "Marie");
24.         husbandWifeMapping.put("Chris", "Lisa");
25.         husbandWifeMapping.put("Steve", "Jennifer");
26.
27.         System.out.println("Husband-Wife Mapping : " + husbandWifeMapping);
28.
29.         // Remove a key from the HashMap
30.         // Ex - Unfortunately, Chris got divorced.
31.         // Let's remove him from the mapping
32.         String husband = "Chris";
33.         String wife = husbandWifeMapping.remove(husband);
34.
35.         System.out.println("Couple (" +
36.             husband + " => " + wife + ") got divorced");
37.         System.out.println("New Mapping : " +
38.             husbandWifeMapping);
39.
40.         // Remove a key from the HashMap only if it is mapped to
41.         // the given value
42.         // Ex - Divorce "Jack" only if He is married to "Linda"
43.         boolean isRemoved = husbandWifeMapping.remove("Jack", "Linda");
44.         System.out.println("Did Jack get removed from the mapping? : " +
45.             isRemoved);
46.
47.         // remove() returns null if the mapping was not found for
48.         // the supplied key
49.         wife = husbandWifeMapping.remove("David");
50.         if (wife == null) {
51.             System.out.println("Looks like David is not married to anyone");
52.         } else {
53.             System.out.println("Removed David and his wife from
54.                 the mapping");
55.         }
56.     }
57. }
```

**Step 6: Create HashMapObtainKeySetExample class**

```
1. package fa.training.hashmapdemo;
2.
3. import java.util.Collection;
4. import java.util.HashMap;
5. import java.util.Map;
6. import java.util.Set;
7.
8. /**
9.  * Examples of how to obtain entrySet, keySet, values from HashMap
10.  *
11.  * @author hoabt2
12.  *
13.  */
14. public class HashMapObtainKeySetExample {
15.
16.     /**
17.      * Get entry set, key set and values from HashMap
18.      *
19.      */
20.     public void obtainEntryKeySetValues() {
21.
22.         System.out.println("obtainEntryKeySetValues() !!!");
23.
24.         Map<String, String> countryISOCODEMapping = new HashMap<>();
25.
26.         countryISOCODEMapping.put("India", "IN");
27.         countryISOCODEMapping.put("United States of America", "US");
28.         countryISOCODEMapping.put("Russia", "RU");
29.         countryISOCODEMapping.put("Japan", "JP");
30.         countryISOCODEMapping.put("China", "CN");
31.
32.         // HashMap's entry set
33.         Set<Map.Entry<String, String>> countryISOCODEEntries =
34.             countryISOCODEMapping.entrySet();
35.         System.out.println("countryISOCODE entries : " + countryISOCODEEntries);
36.
37.         // HashMap's key set
38.         Set<String> countries = countryISOCODEMapping.keySet();
39.         System.out.println("countries : " + countries);
40.
41.         // HashMap's values
42.         Collection<String> isoCodes = countryISOCODEMapping.values();
43.         System.out.println("isoCodes : " + isoCodes);
44.     }
45. }
46.
```

**Step 7: Create HashMapIteratorExample class**

```
1. package fa.training.hashmapdemo;
2.
3. import java.util.HashMap;
4. import java.util.Iterator;
5. import java.util.Map;
6. import java.util.Set;
7.
8. /**
9.  * Examples of how to iterate over a HashMap
10.  *
11.  * @author hoabt2
12.  *
13.  */
14. public class HashMapIteratorExample {
15.
16.     /**
17.      * Iterate over a HashMap
```

```
18.  *
19.  */
20. public void iterateHashMap() {
21.     System.out.println("iterateHashMap() !!!");
22.
23.     Map<String, Double> employeeSalary = new HashMap<>();
24.     employeeSalary.put("David", 76000.00);
25.     employeeSalary.put("John", 120000.00);
26.     employeeSalary.put("Mark", 95000.00);
27.     employeeSalary.put("Steven", 134000.00);
28.
29.     System.out.println("=== Iterating over a HashMap using Java 8
30.                         forEach and lambda ===");
31.     employeeSalary.forEach((employee, salary) -> {
32.         System.out.println(employee + " => " + salary);
33.     });
34.
35.     System.out.println("\n=== Iterating over
36.                         the HashMap's entrySet using iterator() ===");
37.     Set<Map.Entry<String, Double>> employeeSalaryEntries =
38.         employeeSalary.entrySet();
39.     Iterator<Map.Entry<String, Double>> employeeSalaryIterator =
40.         employeeSalaryEntries.iterator();
41.     while (employeeSalaryIterator.hasNext()) {
42.         Map.Entry<String, Double> entry = employeeSalaryIterator.next();
43.         System.out.println(entry.getKey() + " => " + entry.getValue());
44.     }
45.
46.     System.out.println("\n=== Iterating over
47.                         the HashMap's entrySet using simple for-each loop ===");
48.     for (Map.Entry<String, Double> entry : employeeSalary.entrySet()) {
49.         System.out.println(entry.getKey() + " => " + entry.getValue());
50.     }
51.
52.     System.out.println("\n=== Iterating over the HashMap's keySet ===");
53.     employeeSalary.keySet().forEach(employee -> {
54.         System.out.println(employee + " => " +
55.             employeeSalary.get(employee));
56.     });
57. }
58. }
```

#### Step 8: Create TestHashMapDemo class

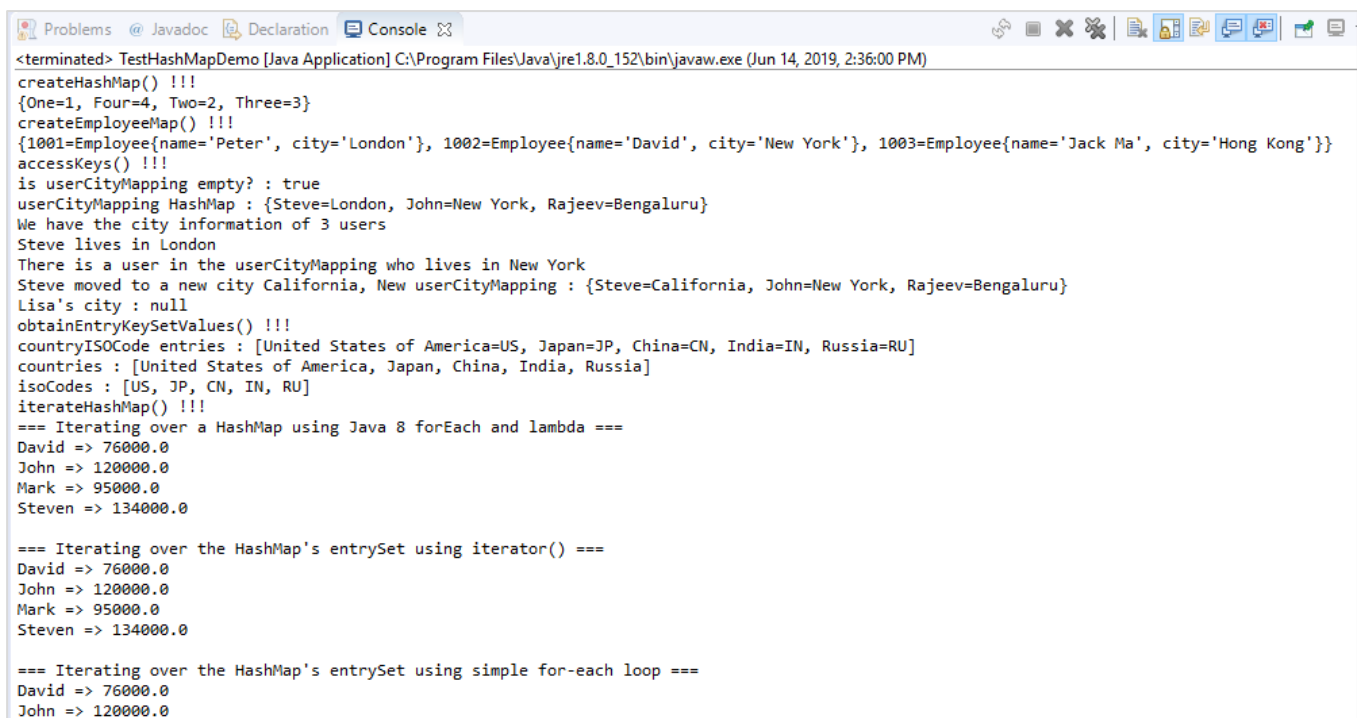
```
1. package fa.training.hashmapdemo.test;
2.
3. import fa.training.hashmapdemo.HashMapAccessKeyExample;
4. import fa.training.hashmapdemo.HashMapCreationExample;
5. import fa.training.hashmapdemo.HashMapIteratorExample;
6. import fa.training.hashmapdemo.HashMapObtainKeySetExample;
7. import fa.training.hashmapdemo.HashMapRemoveKeyExample;
8.
9. /**
10.  * @author hoabt2
11.  *
12.  */
13. public class TestHashMapDemo {
14.
15.     /**
16.      * @param args
17.      */
18.     public static void main(String[] args) {
19.         HashMapCreationExample hashMapCreation = new HashMapCreationExample();
20.         HashMapAccessKeyExample hashMapAccess = new HashMapAccessKeyExample();
21.         HashMapObtainKeySetExample hashMapKeySet = new
22.             HashMapObtainKeySetExample();
```

```
23.     HashMapIteratorExample hashMapIterator = new HashMapIteratorExample();
24.     HashMapRemoveKeyExample hashMapRemove = new HashMapRemoveKeyExample();
25.     hashMapCreation.createHashMap();
26.     hashMapCreation.createEmployeeMap();
27.     hashMapAccess.accessKeys();
28.     hashMapKeySet.obtainEntryKeySetValues();
29.     hashMapIterator.iterateHashMap();
30.     hashMapRemove.removeKeys();
31. }
32. }
33.
```

### Step 9: Run TestHashMapDemo to see the result

You can call corresponding methods separately in order to test the result clearly.

Result:



```
<terminated> TestHashMapDemo [Java Application] C:\Program Files\Java\jre1.8.0_152\bin\javaw.exe (Jun 14, 2019, 2:36:00 PM)
createHashMap() !!!
{One=1, Four=4, Two=2, Three=3}
createEmployeeMap() !!!
{1001=Employee{name='Peter', city='London'}, 1002=Employee{name='David', city='New York'}, 1003=Employee{name='Jack Ma', city='Hong Kong'}}
accessKeys() !!!
is userCityMapping empty? : true
userCityMapping HashMap : {Steve=London, John=New York, Rajeev=Bengaluru}
We have the city information of 3 users
Steve lives in London
There is a user in the userCityMapping who lives in New York
Steve moved to a new city California, New userCityMapping : {Steve=California, John=New York, Rajeev=Bengaluru}
Lisa's city : null
obtainEntryKeySetValues() !!!
countryISOCode entries : [United States of America=US, Japan=JP, China=CN, India=IN, Russia=RU]
countries : [United States of America, Japan, China, India, Russia]
isoCodes : [US, JP, CN, IN, RU]
iterateHashMap() !!!
=== Iterating over a HashMap using Java 8 forEach and lambda ===
David => 76000.0
John => 120000.0
Mark => 95000.0
Steven => 134000.0

=== Iterating over the HashMap's entrySet using iterator() ===
David => 76000.0
John => 120000.0
Mark => 95000.0
Steven => 134000.0

=== Iterating over the HashMap's entrySet using simple for-each loop ===
David => 76000.0
John => 120000.0
```