

**Tutorial No. 8**

**Title: Implementation of HashMap**

**Batch: SY\_IT(B3)****Roll No.:16010423076****Tutorial No.:8****Aim:** To implement HashMap

---

**Resources needed:** Java SDK

---

**Theory:****Theory:**

A HashMap however, store items in "key/value" pairs, and you can access them by an index of another type(Eg. String).

One object is used as a key (index) to another object (value). It can store different types: String keys and Integer values, or the same type, like: String keys and String values.

**Syntax:**

```
import java.util.HashMap; // import the HashMap class
HashMap<String, String> capitalCities = new HashMap<String, String>();
```

**Add Item in a HashMap:**

The HashMap class has many useful methods. For example, to add items to it, use the put() method:

```
HashMap<String, String> capitalCities = new HashMap<String, String>();
capitalCities.put("England", "London");
capitalCities.put("Germany", "Berlin");
capitalCities.put("Norway", "Oslo");
capitalCities.put("USA", "Washington DC");
System.out.println(capitalCities);
```

**Access an item from HashMap**

```
capitalCities.get("England");
```

**Removing an item in HashMap**

```
capitalCities.remove("England");
```

**To remove an item from the HashMap:**

```
capitalCities.clear();
```

**HashMap size method**

To find out how many items there are, use the size() method:

### **Loops in HashMap:**

```
1. for (String i : capitalCities.keySet()) {
    System.out.println(i);
}

2. for (String i : capitalCities.keySet()) {
    System.out.println("key: " + i + " value: " + capitalCities.get(i));
}

3. for (String i : capitalCities.values()) {
    System.out.println(i);
}

capitalCities.size();
```

---

### **Task:**

1. **Write a program to:**
  - a. **Write a Java program to**
    - i. **associate the specified value with the specified key in a HashMap.**
    - ii. **count the number of key-value (size) mappings in a map.**
    - iii. **copy all mappings from the specified map to another map.**
    - iv. **check whether a map contains key-value mappings (empty) or not.**
    - v. **test if a map contains a mapping for the specified key.**

### **Code :**

```
import java.util.HashMap;
import java.util.Map;

public class Main {

    public static void main(String[] args) {
        Map<String, Integer> map1 = new HashMap<>();
        map1.put("Ritesh", 25);
        map1.put("Akshay", 10);
        map1.put("Dev", 15);
        System.out.println("Initial Map: " + map1);

        System.out.println("Size of the map: " + map1.size());

        Map<String, Integer> map2 = new HashMap<>();
        map2.putAll(map1);
        System.out.println("Map2 (copy of Map1): " + map2);

        if (map1.isEmpty()) {
            System.out.println("Map1 is empty.");
        } else {
```

```

        System.out.println("Map1 is not empty.");
    }

    String keyToCheck = "Varad";
    if (map1.containsKey(keyToCheck)) {
        System.out.println("Map1 contains the key: " + keyToCheck);
    } else {
        System.out.println("Map1 does not contain the key: " + keyToCheck);
    }
}
}
}

```

**Output :**

```

Initial Map: {Akshay=10, Dev=15, Ritesh=25}
Size of the map: 3
Map2 (copy of Map1): {Akshay=10, Dev=15, Ritesh=25}
Map1 is not empty.
Map1 does not contain the key: Varad

```

```

Initial Map: {Akshay=10, Dev=15, Ritesh=25}
Size of the map: 3
Map2 (copy of Map1): {Akshay=10, Dev=15, Ritesh=25}
Map1 is not empty.
Map1 contains the key: Akshay

```

**Outcomes:**

CO4: Illustrate the use of collection classes, functional programming, and GUI programming with Java.

**Conclusion: (Conclusion to be based on the outcomes achieved)**

**From this article, I learned how to use a HashMap in Java to store key-value pairs, count the size of the map, copy its contents to another map, check if the map is empty, and verify if a specific key exists. These basic operations help in efficiently managing data in Java.**

---

**Grade: AA / AB / BB / BC / CC / CD /DD**

Signature of faculty in-charge with date

---

**References Books**

1. Herbert Schildt; JAVA The Complete Reference; Seventh Edition, Tata McGraw- Hill Publishing Company Limited 2007.
2. Java 7 Programming - Black Book : Kogent Learning Solutions Inc.
3. Sachin Malhotra, Saurabh Chaudhary "Programming in Java", Oxford University Press, 2010
4. Jaime Nino, Frederick A. Hosch, 'An introduction to Programming and Object Oriented Design using Java', Wiley Student Edition.