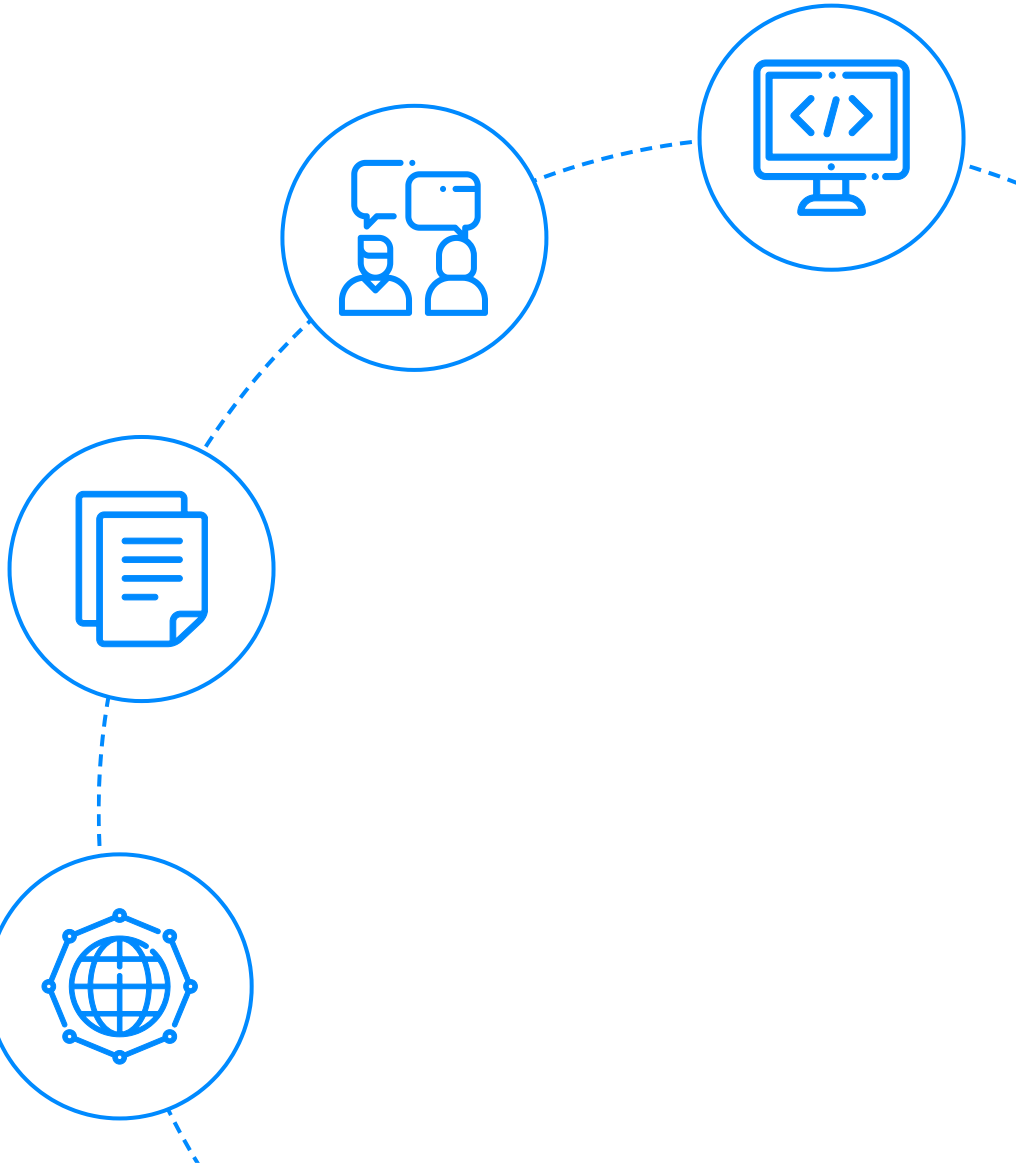
InterviewBit

HashMap Interview Questions



To view the live version of the page, click here.

Contents

HashMap Interview Questions for Freshers

HashMap Interview Questions for Experienced

HashMap Interview Questions for Experienced

(.....Continued)

19. Distinguish between a hashmap and a Treemap.
20. How can a hashmap be used to check whether two given arrays are equal or not i.e they contain the same elements or not? Given the arrays can be unsorted.
21. What factors determine the performance of a hashmap?

HashMap Coding Interview Questions

22. You will be given an unsorted array of integers and a target t. How will you implement a program using a Hashmap to find the position of two numbers whose sum equals target t?
23. Write a program to show how can we retrieve values from a hashmap and state which method is used for it.
24. Write a program to find the highest frequency of a character in a given string
25. 4 sum using hashmap: You will be given four arrays arr1, arr2, arr3, and arr4 all of length n,you have to return the count of tuples (i, j, k, l) satisfying the conditions: 0 <= i, j, k, l < n nums1[i] + nums2[j] + nums3[k] + nums4[l] == 0
26. Which method is used for checking whether a particular key or a value is present in a HashMap or not? Write a program for checking whether a particular key or a value is present in a HashMap or not. If present return true else return false.
27. Which method is used for finding out the total number of key-value pairs present in a hashmap?Give example.
28. Design and implement a class which contains two function find() as well as add()
29. Which method is used for removing or deleting all the key-value pairs? Write a program for removing all the key-value pairs from a HashMap.
30. Write a program to make a hashmap synchronized in java.

Let's get Started

In the world of programming, hashing is a method of storing key-value pairs and this is achieved by making use of HashMaps in Java. In addition to storing key-val pairs, HashMaps are extensively used for solving a range of problems in information technology. By reducing the time complexity of various operations, hashmaps can make problem-solving much quicker when used in combination with different data structures.

What is a HashMap?

A collection based on the map which is used for storing key-value pairs in Java is known as a HashMap. All the keys which are stored in a Hashmap are unique.

Let's see the syntax for how we can create a HashMap in Java:

```
HashMap<Key, Value_mapped_to_the_key> name_of_your_map = new HashMap<>();
```

To give you an idea of HashMap let's consider an example:

Suppose you want to store the roll number of students in a class along with their names then you can easily do so with the help of HashMaps.The name serves as the key for the hashmap and the roll number as the value which can be mapped to the names.

Dave	16
Robin	19
Selena	32
Mark	10

As you can observe there is no ordering of names because hashmaps do not follow any particular ordering for storing the elements.

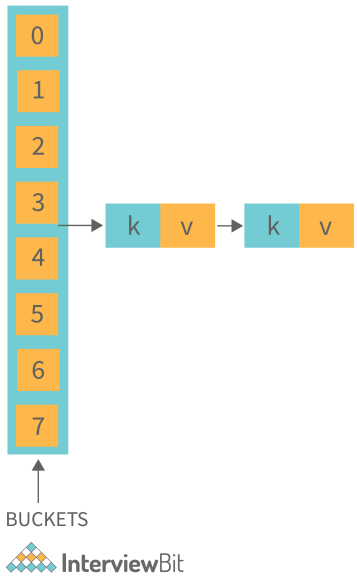
Some important properties of a Hashmap in Java to remember:

- Hashmaps are basically used for storing key-value pairs using a hashtable
- Hashmap stores only unique keys
- The is no particular ordering of elements in a HashMap
- Import java.util.HashMap in your program for using a HashMap

Now in this article, we will be covering the most important and frequently asked **Java HashMap Interview Questions and Answers** to help you ace the interviews. We will cover different varieties of questions that cater to both freshers as well as experienced programmers.

HashMap Interview Questions for Freshers

1. Explain the internal working of a HashMap.



- A hashmap uses a hashtable but what many people don't know is that it is internally created using two data structures namely an array and a linked list. Whenever you declare a hashmap what is going to happen internally is that it will create an array of buckets. The buckets are referred to as nodes or you can say a linked list. These nodes contain mainly:
  - The key,
  - Value
  - Hashcode
  - Address of the next node.
- Now, when you insert values in a key like this:

```
map.put("hashmap implementation in Java",1);
```

Then hashcode will be calculated by the put method. This hashcode will help us to store the key at a particular index. Also, hashcode will make the process of retrieving the values faster.

```
hashcode=hash(“hashmap implementation in Java”);
```

This hash code is further computed and will generate an index for storing the value. The value of our key will be stored at the index given by the hashcode in the form of a LinkedList.

- It's important to note that Java 8 introduced BST in place of a linked list to make retrieval of the key-value pairs more efficient.

2. What is the time complexity in terms of big o notation of pushing and retrieving an element from a hashmap?

- Time complexity is the measure of the number of CPU cycles utilized by a program to execute.
- Hashmap time complexity for pushing and retrieving the elements is the order of 1 i.e O(1) using put and get methods respectively.

3. State the differences between a Hashmap and a Hashtable in Java.

HashMap Vs HashTable

Parameter	HashMap	HashTable
Synchronization	Hashmap and HashTable are very similar to each other except for the fact that a hashmap does not allow synchronization	Unlike a Hashmap, synchronization is permitted in a Hashtable
Performance	You won't face any performance issues in a hashmap.	Synchronization in a hashtable might sound like a good idea that a particular element would be thread-safe if there's concurrent access with multiple threads however, the problem with synchronizing every method that allows access to the underlying collection is that it creates performance issues. So you will have performance issues while using a hashtable

4. Can you store multiple keys with the same value in a hashmap?

- No, multiple keys with the same value can't be stored in a hashmap.
- When you try to insert a new key that is already present in the hashmap then overriding will happen and the old key will be replaced with the new key and a new value.

5. Specify different methods of creating a hashmap in java along with implementation.

Different methods of creating a hashmap are:

1) Constructing a hashmap with default capacity

- **Syntax:**

```
HashMap<String, Integer> InterviewBIt_map1 = new HashMap<String, Integer>();
```

2) Constructing a hashmap with a defined capacity i.e 50 in our example

- **Syntax:**

```
HashMap<String, Integer> InterviewBIt_map2 = new HashMap<String, Integer>(50);
```

3) Specifying lead factor along with the capacity

- **Syntax:**

```
HashMap<String, Integer> InterviewBIt_map3= new HashMap<String, Integer>(50, 0.5f);
```

4) By copying another map into our map

- **Syntax:**

```
HashMap<String, Integer> InterviewBIt_map4 = new HashMap<String, Integer>( InterviewBIt
```

6. Specify whether hashmaps in Java are thread-safe or not?

- The answer is "NO" i.e `java.lang.HashMap` does not support thread-safety. If several threads are altering the HashMap, for example, insertions or removals, then a HashMap should not be shared with other threads.
- If you want to implement thread-safety then you can either use a `ConcurrentHashMap` or by using the `Collections.synchronizedMap()` method.
- When data consistency is crucial, `Collections.synchronizedMap()` should be used, and ConcurrentHashMap should be used where the number of reads and write operations to be performed are less.
- Since `Collections.synchronizedMap()` needs every thread to attain a lock on the whole object to accomplish both read and write operations, this results in slow performance.
- With `ConcurrentHashMap` , threads can simultaneously modify sections while maintaining a lock on individual segments.

7. Does hashmap allow you to store null values?

Yes, We can store as many null values in a hashmap as we want but only one null key can be stored not more than that.

Let's see a program that depicts how we can store null values in a hashmap.

```
import java.util.*;
import java.io.*;
public class interviewBit {
        public static void main(String[] args)
        {
            HashMap hmap=new HashMap();
            hmap.put(1,"Program to store null value");
            hmap.put(null,"InterviewBit");
            System.out.println(hmap);
        }
    }
```

OUTPUT:

```
{null=InterviewBit, 1=Program to store null value}
```

8. What is the order in which keys are stored in a hashmap?

- There is no particular order for storing keys in a hashmap. All the keys are stored in an unsorted order.
- For storing the elements in a particular order a Treemap can be used in Java but a hashmap does not guarantee the ordering of your elements.

9. How can iteration be performed in a HashMap?

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

public class example_____iteration
{
    public static void main(String[] arg)
    {
        Map<String,String> INTERVIEW_BIT = new HashMap<String,String>();


        INTERVIEW_BIT.put("hashmap coding interview questions", "AT INTERVEWBIT");
        INTERVIEW_BIT.put("hashmap programming interview questions", "AT INTERVEWBIT");


        for (Map.Entry<String,String> entry : INTERVIEW_BIT.entrySet())
            System.out.println("Key = " + entry.getKey() +
                               ", Value = " + entry.getValue());
    }
}
```

Output:

```
Key = hashmap coding interview questions, Value = AT INTERVEWBIT
Key = hashmap programming interview questions, Value = AT INTERVEWBIT
```

10. How Does Java's Capacity And Size Of Hashmap Differ?

Sometimes it's confusing for students to understand the difference between capacity is size.

So basically, HashMap's capacity indicates the number of entries it can hold, while its length indicates how many key-value pairs are presently in existence.

11. Discuss the approach for calculating all the equal combinations formed by a pair of numbers in an array using a hashmap i.e The numbers must satisfy these two conditions: nums[i] == nums[j] and i < j Answer)

- You all must have studied combinations, so the concept of combinations states that every time we just have to select a pair from n numbers in an array that satisfies the given conditions and the formula for doing so is NCR which is equivalent to **n*(n-1)/2**.
- Now hashmap will be used for calculating the frequency of all the numbers in the array.
- While iterating the array we will store the frequency of every element in a hashmap.
- The final step would be to iterate over the map and apply the formula **n*(n-1)/2**, where n is the frequency of each element in the array.

12. What is collision in HashMap?

- A collision occurs when more than one key generates the same hashCode() value.
- An inefficient hashCode() algorithm causes considerable collisions in a hashmap.
- An increased number of collisions can affect the performance of a hashmap.

13. Which property of a hashmap is used to find the intersection of two arrays?

- Since a hashmap stores only unique elements, this property of a hashmap can be used to find intersections i.e common elements of two arrays.
- The approach to this problem is: first we will create a hashmap and fill the elements of the first array in it.
- The next step is to iterate over the second array and check whether the element is present in the hashmap or not.
- In this way, we can find all common elements of two arrays in O(n) time complexity.

HashMap Interview Questions for Experienced

14. As we know that hashcodes are generated for each and every key but what happens when the same hashcode is generated for distinct keys?

- When the same hashcode is generated for distinct keys then a collision will occur as the bucket address of the two keys will be the same.
- We have discussed that internally hashmap uses a linked list therefore the linked list will store them all together.

15. Distinguish between a Hashmap and ConcurrentHashMap in Java along with the implementation of both.

Hashmap vs ConcurrentHashMap:

1. **ThreadSafe:** One of the most significant differences between both is that a Cocurrenthashmap is synchronized internally and thread-safe which makes it suitable for a multithreaded environment whereas a hashmap is non-synchronized as well non-thread-safe which makes it unsuitable for a multithreaded environment.
2. **Null Keys And Null Values:** As we all know, data is kept in the form of key and value pairs in a hashmap and we can store as many null values in a hashmap as we want but only 1 null key can be stored whereas in a concurrent hashmap null keys, as well as null values, are not permitted.

Implementations

- Hashmap Implementation:

```
import java.util.*;
import java.io.*;
public class interviewBit {
        public static void main(String[] args)
        {
            HashMap hmap=new HashMap();
            hmap.put(91,"Hashmap Implementation");
            hmap.put(92,"in ");
            hmap.put(93,"Java");
            hmap.put(null,"InterviewBit");
            System.out.println(hmap);
        }
    }
```

OUTPUT:

```
java -cp /tmp/ZPv88JJ0D2 interviewBit
{null=InterviewBit, 91=Hashmap Implementation, 92=in , 93=Java}
```

- ConcurrentHashMap implementation:

For implementing ConcurrentHashmap we have to import the concurrent package.

```
import java.util.concurrent.ConcurrentHashMap;
public class interviewBit {
        public static void main(String[] args)
        {
            ConcurrentHashMap hmap=new ConcurrentHashMap();
            hmap.put(91,"Hashmap Implementation");
            hmap.put(92,"in ");
            hmap.put(93,"Java");
            hmap.put(null,"InterviewBit");
            System.out.println(hmap);
        }
    }
```

OUTPUT:

```
java -cp /tmp/ZPv88JJ0D2 interviewBit
Exception in thread "main" java.lang.NullPointerException
at java.base/java.util.concurrent.ConcurrentHashMap.putVal(ConcurrentHashMap.java:1011)
at java.base/java.util.concurrent.ConcurrentHashMap.put(ConcurrentHashMap.java:1006)
    at interviewBit.main(interviewBit.java:12)
```

As we can observe from the above outputs we can store the null value in the hashmap but it can't be stored in a ConcurrentHashMap as it gives NullPinterException.

16. What is the maximum number of entries you can store in HashMap?

HashMap does not have a maximum number of entries because when the bucket is full, the keys will be added to a linked list, which can store forever until all the memory you have is consumed.

17. Define LinkedHashMap in Java

- A linked HashMap is implemented in a similar way to a hashmap but contains a doubly LinkedList for querying through all the key-value pairs.
- LinkedHashMap stores the key-value pair in order, unlike a hashmap.
- You can retrieve the data in the same order as they were inserted in the map i.e the key which was inserted first can be taken out first.

Implementation of LinkedHashMap for understanding how we can insert and retrieve key and value pairs easily:

```
import java.util.LinkedHashMap;
public class InterviewBit {

public static void main(String[] args) {

LinkedHashMap m = new LinkedHashMap();

m.put("Linked", new Integer(1));
m.put("Map", new Integer(2));

Object lobj = m.get("Linked");
System.out.println(lobj);
}
}
```

The output of the above program will be 1.

18. Which is the best technique to handle collision in a hashmap.

As part of its collision handling, HashMap employs chaining. In chaining, a linked list is used for placing the key-value pairs inserted into the map with the value already present to avoid collision in the map at a bucket location as the newly inserted value is placed in front of the linked list.

19. Distinguish between a hashmap and a Treemap.

Hashmap	Treemap
In a hashmap, no ordering of elements is maintained	The ordering, of elements, is maintained in a treemap.
We can store as many null values in a hashmap as we want but only 1 null key can be stored	We cannot store any null values or keys in a treemap.
Hashmap is Fast since no ordering of elements is maintained therefore elements can be inserted and retrieved in constant time.	Treemap is slow and most of the functions take logarithmic time.
Hashmap uses arrays and LinkedList which forms a hashtable for its implementation internally.	Treemap is implemented with the help of red-black trees.

20. How can a hashmap be used to check whether two given arrays are equal or not i.e they contain the same elements or not? Given the arrays can be unsorted.

- A hashmap can be easily used to check whether the given arrays are equal or not.
- First, create a hashmap and using a for loop fill the frequency of each and every element of array 1 in the hashmap.
- Now, make another loop and decrement the frequency of each and every element of the second array in the hashmap.
- In the next step, iterate over the hashmap and check whether all the frequencies are zero or not. Even if any frequency is 1 return false else return true.

21. What factors determine the performance of a hashmap?

The performance or efficiency of a hashmap depends basically on two things that are:

- **Capacity -** Capacity is defined as the total number of buckets present in a Hashmap and Sixteen is the default capacity when a Hashmap is created by the user. The capacity can be increased when more key-value pairs are added to the hashmap.
- **Load Factor -** As discussed in the previous point the capacity of the map can be increased but when should the hashmap increase its capacity?

This is decided by the load factor and its default value is seventy-five per cent.

HashMap Coding Interview Questions

22. You will be given an unsorted array of integers and a target t. How will you implement a program using a Hashmap to find the position of two numbers whose sum equals target t?

Approach:

- A general approach for solving such kinds of problems is that first, you have to create a map of Integers.
- Then iterate through the given array and subtract the current value in the array from the given target.
- Look for that value in the hashmap simultaneously.
- If the value is found in the map then return the index of these 2 numbers else insert the current element of the array as the key and its index as the value in the Hashmap.

Implementation:

```
class InterviewBit{
    public int[] target(int[] nums, int t) {
        HashMap<Integer, Integer> find_complement = new HashMap<>();

        for (int i = 0; i < nums.length; i++) {
            if (find_complement.containsKey(t - nums[i]) == true) {
                return new int[] { find_complement.get(t - nums[i]), i };
            }
            find_complement.put(nums[i], i);
        }
        return null;
    }
}
```

23. Write a program to show how can we retrieve values from a hashmap and state which method is used for it.

The get() method is used for retrieving the values from the hashmap.

Implementation:

```
import java.util.HashMap;
public class InterviewBit
{
    public static void main(String[] args)

    {
         HashMap<String, Integer> InterviewBit_map = new HashMap<String, Integer>();
        InterviewBit_map.put("X", 1);
        InterviewBit_map.put("Y", 2);
        InterviewBit_map.put("Z", 3);
        InterviewBit_map.put("W", 4);
        int value = InterviewBit_map.get("Y");
        System.out.println(value);       //Output : 2
    }
}
```

24. Write a program to find the highest frequency of a character in a given string

You will be given a string and you are required to find the character which occurs the most number of times using a hashmap.

Approach:

- First of all, you are required to create a Hashmap.
- After that iterate over the string and check for each and every character whether it is present in the hashmap or not.
- If the character is not present in the hashmap then insert it else increase the frequency of the character.
- Once the hashmap is filled with the frequency of all the characters in the string, loop over the map and checks for the highest frequency character present on the map.

Implementation:

```
import java.io.*;
import java.util.*;

public class InterviewBit {

    public static void main(String[] args) {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str = br.readLine();

        //Creating  a hashmap
        HashMap<Character, Integer> frequency_map = new HashMap<>();
        for(int i = 0; i < str.length(); i++){
            Character ch = str.charAt(i);
            if(frequency_map.containsKey(ch) == true){
                int old_freq = frequency_map.get(ch);
                int new_freq = old_freq + 1;
                frequency_map.put(ch, new_freq);
            } else {
                frequency_map.put(ch, 1);
            }
        }

        Set<Character> keys = frequency_map.keySet();
        Character c = str.charAt(0);
        for(Character key : keys){
            if(frequency_map.get(key) > frequency_map.get(c)){
                c = key;
            }
        }

        System.out.println(c);
    }
}
```

25. 4 sum using hashmap: You will be given four arrays arr1, arr2, arr3, and arr4 all of length n,you have to return the count of tuples (i, j, k, l) satisfying the conditions: 0 <= i, j, k, l < n nums1[i] + nums2[j] + nums3[k] + nums4[l] == 0

Approach:

- 2 nested loops will be required for implementing this.
- We will start by creating a hashmap
- Next, all possible sums of 2 elements will be calculated and stored in the map using the nested loops we created earlier.
- Now again the nested loop will be used for calculating the sum of 2 elements of the other two arrays.
- Once we find out all the combinations of the sum, we take the complement of that sum and find that in our hashmap.
- If the complement is found then the count of tuples will be increased by 1 every time
- The time complexity will be O(N2)

Implementation:

```
import java.util.*;

class Solution {
    public int fourSumCount(int[] arr1, int[] arr2, int[] arr3, int[] arr4) {
        HashMap<Integer, Integer> AB = new HashMap<>();
        for (int val1 : arr1)
            for (int val2 : arr2)
                AB.put(val1 + val2, AB.getOrDefault(val1 + val2, 0) + 1);

        int count = 0;
        for (int val3 : arr3)
            for (int val4 : arr4)
                count += AB.getOrDefault(-val3 - val4, 0);

        return count;
    }
}
```

26. Which method is used for checking whether a particular key or a value is present in a HashMap or not? Write a program for checking whether a particular key or a value is present in a HashMap or not. If present return true else return false.

ContainsKey() and **ContainsValue()** methods are used for checking whether a particular key or a value is present in a map or not.

```
import java.util.HashMap;
public class InterviewBit
{
    public static void main(String[] args)
    {
         HashMap<String, Integer> InterviewBit_map = new HashMap<String, Integer>();

        InterviewBit_map.put("X", 1);

        InterviewBit_map.put("Y", 2);

        InterviewBit_map.put("Z", 3);

        InterviewBit_map.put("W", 4);

        int value = InterviewBit_map.get("Y");

        System.out.println(InterviewBit_map.containsKey("X"));    //Output : true
    }
}
```

27. Which method is used for finding out the total number of key-value pairs present in a hashmap?Give example.

Size() method is used for finding out the total number of key-value pairs present in a hashmap.

Example:

```
import java.util.HashMap;

public class InterviewBit
{
    public static void main(String[] args)
    {
         HashMap<String, Integer> InterviewBit_map = new HashMap<String, Integer>();

        InterviewBit_map.put("X", 1);

        InterviewBit_map.put("Y", 2);

        InterviewBit_map.put("Z", 3);

        InterviewBit_map.put("W", 4);

        System.out.println(InterviewBit_map.size());       //Output : 4
    }
}
```

28. Design and implement a class which contains two function find() as well as add()

The find function will check whether any pair exists whose sum is equal to a given value in O(N) average time whereas add function will be used for adding any element in O(1) average time.

A hashmap will be the best choice for meeting our requirements as adding an element can be performed in O(1) time whereas searching or finding an element can be performed in O(N) time.

Implementation:

```
public class TwoSum {
    HashMap<Integer, Integer> our_map;

    public TwoSum() {
        our_map = new HashMap<>();
    }

    public void add(int number) {
        our_map.put(number, our_map.getOrDefault(number, 0) + 1);
    }

    public boolean find(int data) {
        for (Integer key : our_map.keySet()) {
            int complement = data - key;
            int freq_comp = freq.getOrDefault(complement, 0);

            if (data - key == key) {
                if (freq_comp >= 2)
                    return true;
            } else {
                if (freq_comp >= 1)
                    return true;
            }
        }
        return false;
    }
}
```

29. Which method is used for removing or deleting all the key-value pairs? Write a program for removing all the key-value pairs from a HashMap.

Clear() method is used for removing or deleting all the key-value pairs.

Implementation:

```
import java.util.HashMap;

public class InterviewBit
{
    public static void main(String[] args)
    {
         HashMap<String, Integer> InterviewBit_map = new HashMap<String, Integer>();

        InterviewBit_map.put("X", 1);

        InterviewBit_map.put("Y", 2);

        InterviewBit_map.put("Z", 3);

        InterviewBit_map.put("W", 4);

        InterviewBit_map.clear();

        System.out.println(InterviewBit_map.size());       //Output : 0
    }
}
```

30. Write a program to make a hashmap synchronized in java.

`Collections.synchronizedMap()` method is used for making a hashmap synchronized in java.

Here is the implementation:

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

public class JavaHashMapPrograms
{
    public static void main(String[] args)
    {

        HashMap<String, Integer> InterviewBit_map = new HashMap<String, Integer>();
        Map<String, Integer> syncMap = Collections.synchronizedMap(InterviewBit_map);
    }  }
```

Useful Resources

- Data Structure Interview Questions
- DSA Tutorial
- Java String Interview Questions
- Java Interview Questions for 5 Years Experience
- Java Interview Questions
- Java Cheat Sheet
- Array Interview Questions
- Java Collections Interview Questions
- Complete Interview Preparation Guide

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions