Pre Requisites:

- Basic Java Syntax

List of concepts involved :

- What are hashmaps
- Various functions of hashmap
- Collision in Hashmap
- Types of hashmap
- Two Sum problem
- First unique character in String

What are Hashmaps

First of all let's understand Maps.Maps is a interface in Java which is present in java.util.* package represents a key and value pair in java. A key should always be unique in the map.

Maps can be used on various occasions where we need to have a key -value sort of relationship. Various examples of map usage are.

- Dictionaries having key value relationships. Word is key and meaning is value
- A map of zip codes and cities. Cities are key and list of zip codes are values.
- A map of class of students. Class is a key and list of students are values.

Now HashMap<K,V> is a part of Java's collection. The class is also found in java.util.* package. It provides the basic implementation of the map interface in java. It stores the data in (key, value) pairs, and you can access them by a index of key. A hashmap always contains unique key. If we try to insert a duplicate key, then it replaces the old key. Hashmap also allows us to store NULL keys. But it only contains one NULL key. If we try to insert another, it will update the value of previous null key.

Syntax to declare Hashmap in Java

HashMap<K,V> hashmap = new HashMap<>();

K,V can be of any data structure. For eg.

HashMap<Integer, String> hashmap = new HashMap<>();

Various functions of HashMap

put()

Put is used to insert elements into the hashmap. We need to specify the arguments of key, value type and it will insert into the hashmap.

For eg.

hashmap.put(1, "Piyush");
hashmap.put(2, "Raghav");

Now map contains: {1=Piyush, 2= Raghav}
If we again update put the same key in the map, it will update the value of hashmap key.

hashmap.put(2, “Akash”);
Now map contains: {1=Piyush, 2= Akash}

get()

get() is used to get the value of a particular key inside the map. If the map doesn’t contains the key it will return null as the output.

For eg,

String answer = hashmap.get(2);

gives us the output in String as Akash.

remove()

In order to remove a element from hashmap we use remove function. We provide key as the argument to the function and it removes the key-value pair from the map if it is present in the map. For eg.

hashmap.remove(1);

Now map contains: {2= Akash}

entrySet()

To traverse a complete map we use the entrySet function. It helps in traversing over the map in order of keys.

For eg.

For the previous created hashmap,

```
for (Map.Entry<Integer, String> e : hashmap.entrySet()) {
        System.out.println(“Key :”+ e.getKey() + “, value: ”+ e.getValue());

}
```

Now we have created an iterator e which will traverse every key of map. When we use getKey(), it gives us the element which is acting as a key in map and similarly when we use .getValue(), it gives the value stored for the corresponding key.

containsKey()

This function is used to check if map contains a following key or not. We pass the key inside this function and it will return true if key is present in map.

For eg.

Boolean result = hashmap.containsKey(1);

Code: LP_code1.java

Output

```
Value of Map is: {1=Piyush, 2=Athar, 3=Ajay, 4=Anil}
Value of Map is: {1=Rajesh, 2=Athar, 3=Ajay, 4=Anil}
Value of key = 3 is: Ajay
Value of Map is: {1=Rajesh, 3=Ajay, 4=Anil}
Key: 1 Value: Rajesh
Key: 3 Value: Ajay
Key: 4 Value: Anil
```

Collision in Hashmap

What is the Hash function?

A hash function is basically used to map a data of large size into a data of fixed size. The values returned by a hash function are called hash values.
The hash values are used to store keys in the map.

For eg a simple hash function could be

```
Integer hashFunction(Integer key) {
        return key%10;
}
```

Now if you pass any sort of data to this hash function it will return modulus 10. So all the hash values will be in order of 0-9.

What is collision

When a hash function starts returning same hash value for more than one key, then it results in collision. for eg for the above hash function, if we pass 10, 20 to the hash function it will return as 0 in both cases. Now we won't be able to store value for different keys as both are being mapped to same hash key.So it's very necessary to have a strong hash function.

How to handle collision

- **Separate Chaining.**
  a. Separate Chaining with linked lists.
- **Open addressing**
  a. Linear Probing
  b. Quadratic probing
  c. Double hashing.