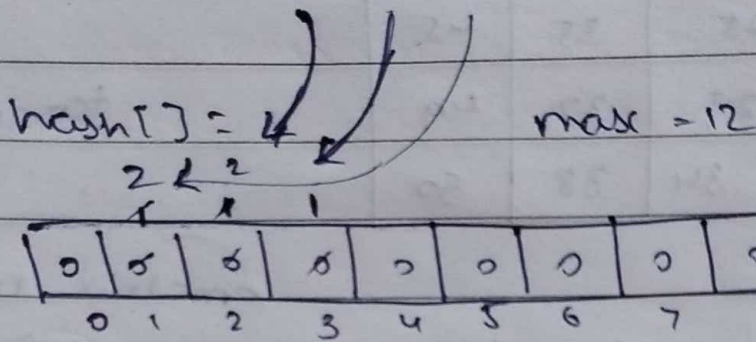


* HashMap

- provides functionality of hash table
- Stores element in Key/Value pair

arr[] = 1 | 2 | 1 | 3 | 2 |



Key	Value
↓	↓
unique	& may / may not be same

① Create a HashMap.

import java.util.HashMap package

Syntax:

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

here K = Key type

V = type of value

eg:

HashMap

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

② Add Elements in Hashmap.

map.put()



exists



update value.

doesn't exist



new pair is inserted

eg:

numbers.put("One", 1)

numbers.put("Two", 2)

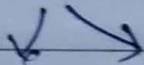
Print("Hashmap", + numbers)

O/P: {One = 1, Two = 2}

③ Access Elements

key ↴

map.get()



~~key~~ exist

↓

value

key does not exist

↓

null

map.containsKey()



✓

↓

true

✗

↓

false

eg: if (map.containsKey("China")) { //true
 }

eg: → this returns true or false

eg: `map.get("China")`

→ returns the value if exists.

`map.keySet()` → returns the keys
`map.keyvalue()` → returns values
`map.entrySet()` → returns key/value

④ Replace value

```
map.put(2, "Python");  
map.replace(2, "C++");
```

Here Python is replaced with C++

⑤ Remove element

```
map.remove(2)
```

⇒ Here, the key-value pair with key 2 will be removed.

⑥ Iterate through a Hash Map.

→ // Iterate through keys

```
for (Integer key : map.keySet()) {  
    cout (key)
```

```
}
```

O/p: 1, 2, 3

→ // Iterate values only

```
for (String value : map.values()) {  
    cout (value)
```

```
}
```

O/p: Java, Python, C++

// Iterate through key/value entries

→ for (Entry < Integer, String > entry : map.entrySet()) {
 cout (entry)

```
}
```

O/p: 1=Java, 2=Python, 3=C++

if optional parameter is not used, then
default capacity $\rightarrow 16$
load factor $\rightarrow 0.75$

Page No.

Date

⑦ Create Hashmap from other maps.

```
TreeMap < String, Integer > evenNumbers =  
    new TreeMap < > ();
```

// Create Hashmap from treemap

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

NOTE

While creating a hashmap, we can include optional parameters : capacity and load factor.

eg:

```
HashMap < K, V > numbers = new HashMap < > (8, 0.6)
```

Here : 8 (capacity) \rightarrow means it can store 8 entries

: 0.6 (load factor 0.6) \rightarrow means whenever our hash table is filled by 60%, the entries are moved to a new hash table double the size of original hash table

* Character Hashing.

$s = \text{"abc d a b e f c"}$

0	1	2	3	4	...	25
---	---	---	---	---	-----	----

ASCII: 'a' \rightarrow 97

'z' \rightarrow 122

$$a = a - a = 0$$

$$b = b - a = 1 - 0 = 1$$

$$f = f - a = 102 - 97 = \underline{\underline{5}}$$

* Time Complexity

\rightarrow unordered-map

Storing $\rightarrow O(1)$ { avg, best }

fetching $\rightarrow O(N)$ { worst }

\downarrow
 $N = \text{no. of elements in map.}$

\rightarrow ordered

Storing $\rightarrow \log(N)$ \rightarrow best

fetching \rightarrow avg

\rightarrow worst