PAGE No.
DATE / /

Hashing

Hashing is a technique used in data structures to store and retrieve data efficiently. It is basically transforming one type of data into another.

- Hashing
  - Map
    - HashMap
    - LinkedHashMap
    - TreeMap
  - Set
    - Hashset
    - LinkedHashset
    - TreeSet

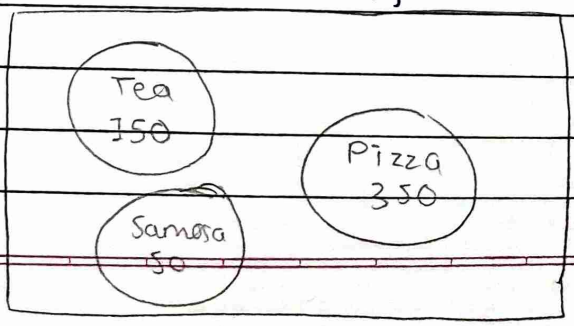
HashMap :-

Java HashMap is similar to HashTable, but it is unsynchronized. It allows to store the null keys as well, but there should be only one null key object and there can be any number of null values.
To use this class and its methods we need to import `java.util.HashMap` or its superclass.

(key, value)
↳ always unique.

Menu	
Tea	150
Pizza	350
Samosa	50

In memory :-



HashMap Syntax :-

```
HashMap<String, Integer> hm = new HashMap<>();
```
(String → key, Integer → value)

HashMap Operations :-

put(key, value)	O(1)	inserts a new key,value pair in Hashmap
get(key)	O(1)	gets the value of specified key
containsKey(key)	O(1)	returns true if Hashmap contains key
remove(key)	O(1)	removes specified key value pair

keyset() → set of keys

entryset() → key, value pairs

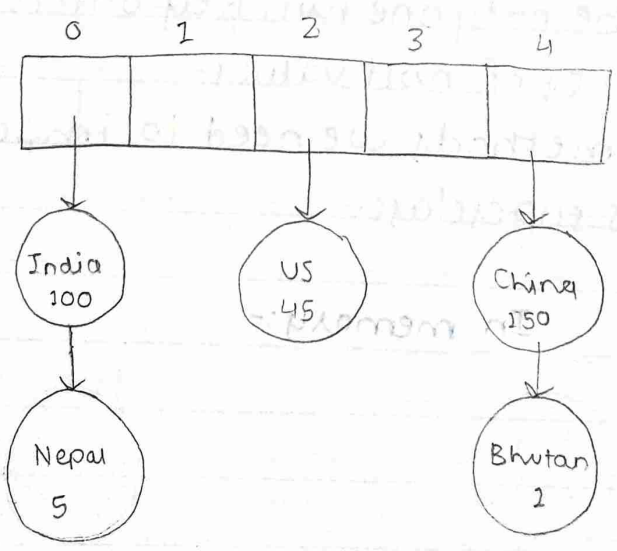
Iteration on HashMap :-

```
Set<Integer> keys = hm.keyset();
```
{TC :- O(1)}

↳ This will return a set of keys.
We can also use for each loop to iterate through it.

Implementation of HashMap :-

Hashmap is internally implemented in Java in the form of Array of Linked list [Buckets]



elements are sorted in the form of Nodes.

n = nodes/pairs

N = buckets.

generics → parameterized types
(it could be of any type)

PAGE No.
DATE / /

② put (key, value)

"Canada", 30 ✗ → we shouldn't use this as it has bad time complexity.

↓ key exists

↓ key doesn't exists

"India" → hashcode() → 1245.

we have certain hashing algorithms which don't transforms the key into another value.

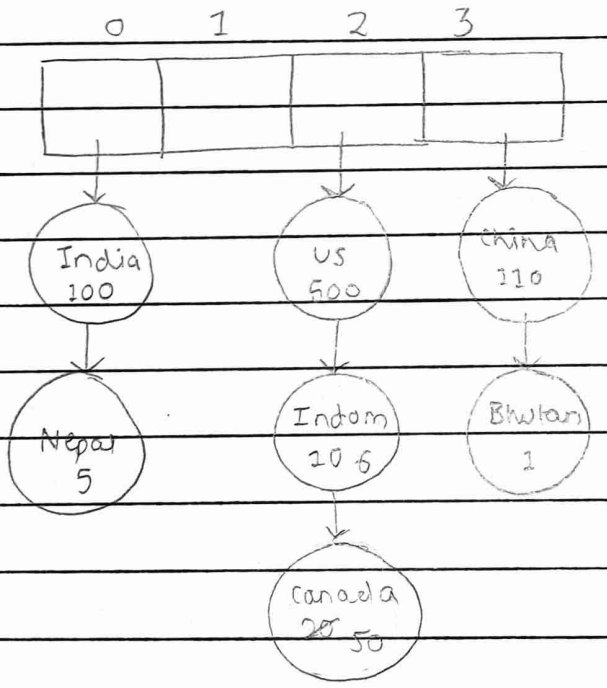
key → [hash function] → bucket index

① hash f(x) → bi

② loop LL in the bucket

if found → update

else → add new node in LL.



N = 4

n = 7

λ (lambda) $= \frac{n}{N} = \frac{7}{4} = 1.75$

$O(\lambda) = O(1.75)$

↳ constant

$\lambda \leq k$

↳ constant threshold value.

k = 2

λ should always be less than or equal to k.

Rehashing :- Making new array if λ is less than two

newArray = 2 * oldsize

```
private LinkedList<Node> buckets[];
```

↳ Array of LinkedList of nodes.

Other types of HashMaps :-

→ **LinkedHashMap** :-

- Keys are insertion ordered

Syntax ↘

```
LinkedHashMap<K,V> hm = new LinkedHashMap<>();
```

→ **TreeMap** :-

- Keys are sorted.
- put get remove are O(log n).

Syntax ↘

```
TreeMap<K,V> hm = new TreeMap<>();
```

TreeMap Data Structure is implemented using Red black Trees.