

## What is Hashing?

- Hashing is a technique used for storing, retrieving and removing information as quick as possible.
- It's a process of converting a arbitrary size key into fixed sized value. The conversion is done via special function called as Hash function.
- The operations supported by hashing such as storing, retrieving and removing information have average runtime complexity of  $O(1)$ .

## What are Hash Functions?

- A Hash function simply takes an arbitrary size key and provides fixed size value also called as index.



## Modular Hash Function

- A Modular Hash function simply takes a key and size, returns remainder by dividing key by size.
- The remainder is used as an index to store the key in an array of provided size.

## Modular Hash Function

$$\text{index} = h(\text{key}) = \text{key} \% \text{size}$$

size = 10

Key Space



$$5 \% 10 = 5$$

$$1 \% 10 = 1$$

$$10 \% 10 = 0$$

$$26 \% 10 = 6$$

$$99 \% 10 = 9$$

## What is a Hash Table?

- It is a generalized form of an array.
- It stores the data in form of **key-value** pair.
- It converts **key** to an **index** using **hash** function.
- Taking the **index** we store **key-value** in array.
- The primary operations supported by HashTable are –
  - `put(key, value)` – Adds **key-value** pair against unique key.
  - `get(key)` – Get **value** for the provided **key**.
  - `remove(key)` – Removes the **key-value** pair from HashTable.
- Average running time is of  $O(1)$ .
- Java Collections Framework has `HashMap` class - if we want to deal with key-value pair and `HashSet` class if we want to deal with only keys.

## Simple Hash Table

$\text{index} = h(\text{key}) = \text{key} \% \text{size}$   
 $\text{size} = 10$

Key - Value Space



10, James

1, Tom

5, John

26, Tina

99, Sana

$$5 \% 10 = 5$$

$$1 \% 10 = 1$$

$$10 \% 10 = 0$$

$$26 \% 10 = 6$$

$$99 \% 10 = 9$$

$$105 \% 10 = 5$$

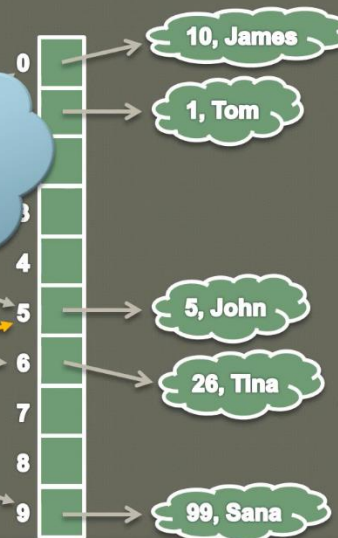
## Simple Hash Table

index =  $h(\text{key}) = \text{key} \% \text{size}$   
size = 10

Key - Value Space

5, John  
1, Tom  
10, James  
26, Tina  
99, Sana  
105, Mary

Collision



$5 \% 10 = 5$

$1 \% 10 = 1$

$10 \% 10 = 0$

$26 \% 10 = 6$

$99 \% 10 = 9$

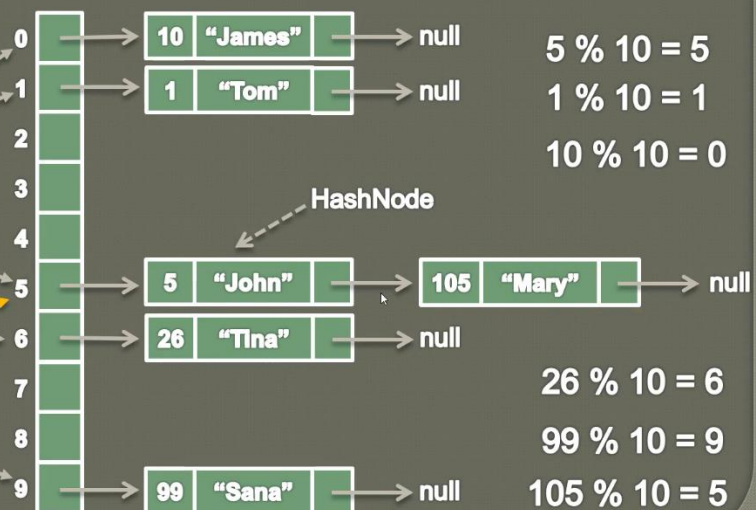
$105 \% 10 = 5$

## Separate Chaining

index =  $h(\text{key}) = \text{key} \% \text{size}$   
size = 10

Key - Value Space

5, John  
1, Tom  
10, James  
26, Tina  
99, Sana  
105, Mary



$5 \% 10 = 5$

$1 \% 10 = 1$

$10 \% 10 = 0$

$26 \% 10 = 6$

$99 \% 10 = 9$

$105 \% 10 = 5$



## Representation of a HashNode in HashTable

A HashNode class in HashTable consists of three data members.

1. **K key** – It is a unique value which helps in storing data. Here, K signifies generic type.
2. **V value** – It is the data that is stored based on location computed by key. Here, V signifies generic type.
3. **HashNode next** – It refers to next HashNode in chain of hash nodes.

```
graph LR; subgraph HashNode_Struct [HashNode]; direction TB; K[K key]; V[V value]; next[HashNode next]; end; subgraph HashNode_Table [HashNode]; direction LR; key[key] --- value[value] --- next_ptr[next] --> next_ptr; end;
```

# HashTable Terminology

size = 6

HashTable Terminology

size = 6

numOfBuckets = length of buckets array, also called as capacity

Diagram illustrating HashTable Terminology:

- HashTable buckets (array of HashNodes):** An array of 10 buckets (0 to 9).
- HashNode structure:** Each HashNode contains a key (integer), a value (string), and a pointer to the next HashNode (null or another HashNode).
- Head pointer:** Points to the first HashNode in the chain for a given bucket.
- Chains:** The sequence of HashNodes linked together for a given bucket.

Example data from the diagram:

Bucket Index	Key	Value	Next Pointer
0	10	"James"	null
1	1	"Tom"	null
2			
3			
4			
5	5	"John"	105
6	26	"Tina"	null
7			
8			
9	99	"Sana"	null

Note: The pointer from bucket 5 points to a HashNode with key 105 and value "Mary", which then points to null.