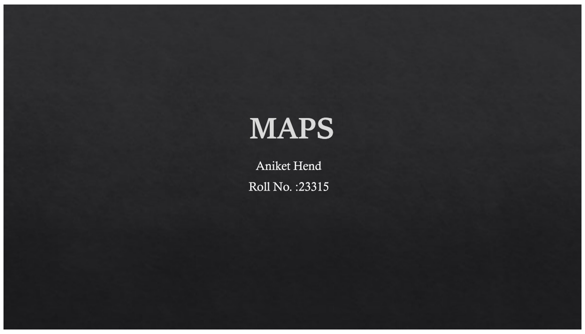
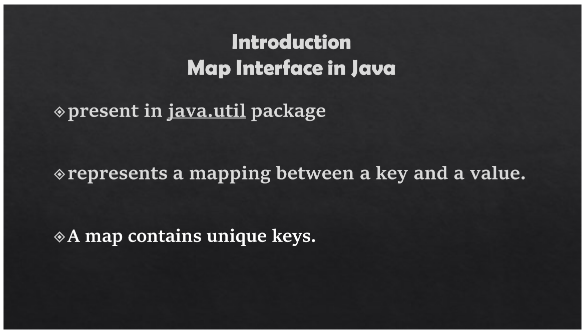


MAPS

Aniket Hend
Roll No. :23315

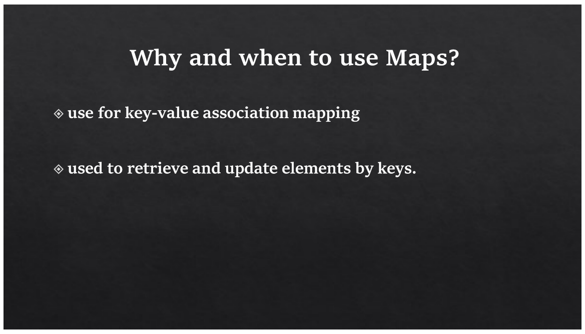


Introduction
Map Interface in Java

⟐ present in java.util package

⟐ represents a mapping between a key and a value.

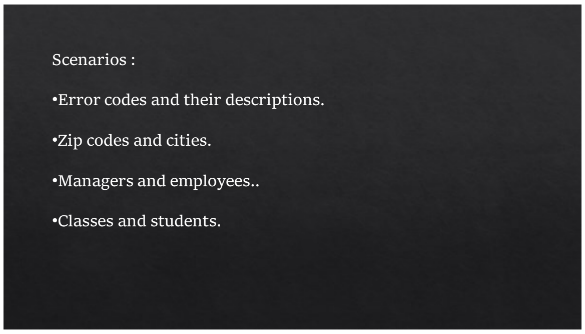
⟐ **A map contains unique keys.**



Why and when to use Maps?

⟐ **use for key-value association mapping**

⟐ **used to retrieve and update elements by keys.**



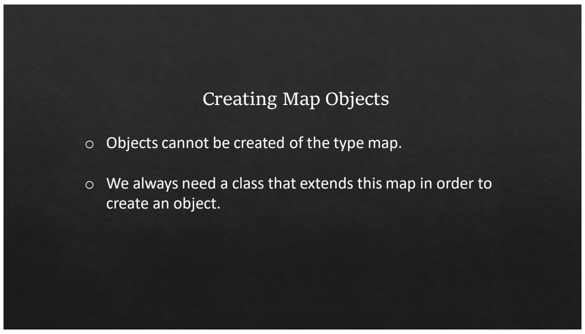
Scenarios :

•Error codes and their descriptions.

•Zip codes and cities.

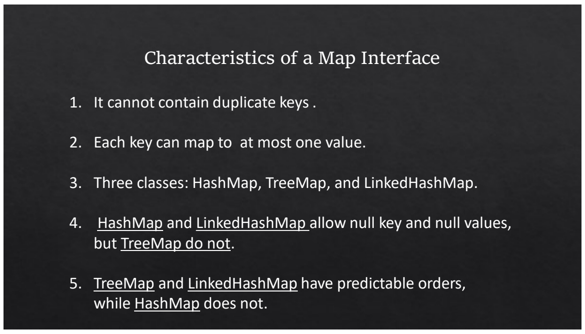
•Managers and employees..

•Classes and students.



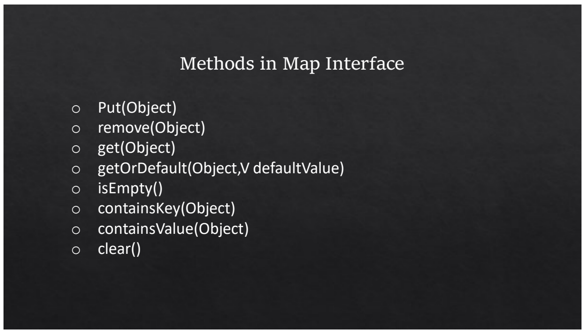
Creating Map Objects

- Objects cannot be created of the type map.
- We always need a class that extends this map in order to create an object.



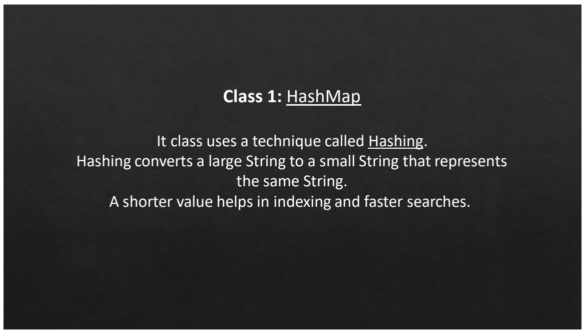
Characteristics of a Map Interface

1. It cannot contain duplicate keys .
2. Each key can map to at most one value.
3. Three classes: HashMap, TreeMap, and LinkedHashMap.
4. HashMap and LinkedHashMap allow null key and null values, but TreeMap do not.
5. TreeMap and LinkedHashMap have predictable orders, while HashMap does not.



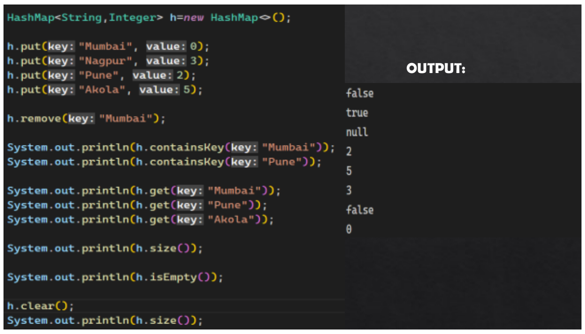
Methods in Map Interface

- Put(Object)
- remove(Object)
- get(Object)
- getOrDefault(Object,V defaultValue)
- isEmpty()
- containsKey(Object)
- containsValue(Object)
- clear()



Class 1: HashMap

It class uses a technique called Hashing.
Hashing converts a large String to a small String that represents the same String.
A shorter value helps in indexing and faster searches.



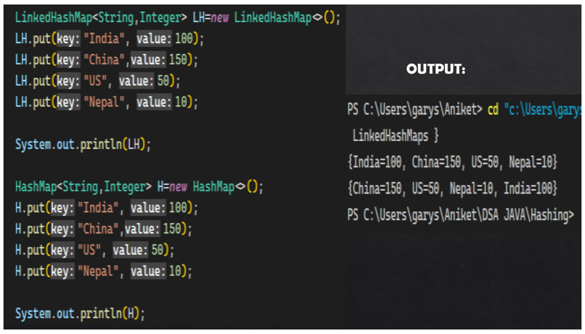
```
HashMap<String,Integer> h=new HashMap<>();

h.put(key: "Mumbai", value: 0);
h.put(key: "Nagpur", value: 3);
h.put(key: "Pune", value: 2);
h.put(key: "Akola", value: 5);

h.remove(key: "Mumbai");

System.out.println(h.containsKey(key: "Mumbai"));
System.out.println(h.containsKey(key: "Pune"));

System.out.println(h.get(key: "Mumbai"));
System.out.println(h.get(key: "Pune"));
System.out.println(h.get(key: "Akola"));

System.out.println(h.size());

System.out.println(h.isEmpty());

h.clear();
System.out.println(h.size());
```

OUTPUT:

```
false
true
null
2
5
3
false
0
```

Class 2: <u>LinkedHashMap</u>

- Maintains an order of elements inserted



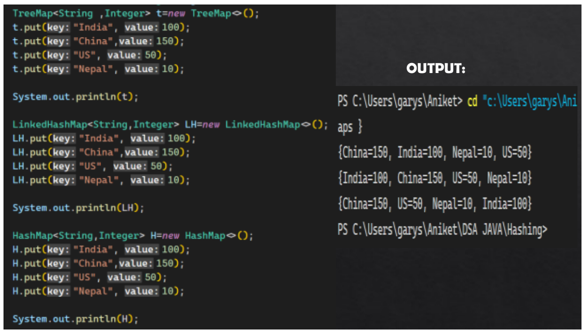
```
LinkedHashMap<String,Integer> LH=new LinkedHashMap<>();
LH.put(key: "India", value: 100);
LH.put(key: "China",value: 150);
LH.put(key: "US", value: 50);
LH.put(key: "Nepal", value: 10);

System.out.println(LH);

HashMap<String,Integer> H=new HashMap<>();
H.put(key: "India", value: 100);
H.put(key: "China",value: 150);
H.put(key: "US", value: 50);
H.put(key: "Nepal", value: 10);

System.out.println(H);
```

OUTPUT:

```
PS C:\Users\garys\Aniket> cd "c:\Users\garys
 LinkedHashMaps }
{India=100, China=150, US=50, Nepal=10}
{China=150, US=50, Nepal=10, India=100}
PS C:\Users\garys\Aniket\DSA JAVA\Hashing>
```

Class 3: <u>TreeMap</u>

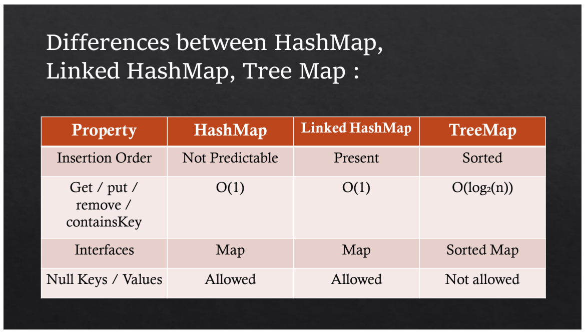
- The map is sorted according to the natural ordering of its keys.
- This proves to be an efficient way of sorting and storing the key-value pairs.



```java
TreeMap<String ,Integer> t=new TreeMap<>();
t.put(key: "India", value: 100);
t.put(key: "China",value: 150);
t.put(key: "US", value: 50);
t.put(key: "Nepal", value: 10);

System.out.println(t);

LinkedHashMap<String,Integer> LH=new LinkedHashMap<>();
LH.put(key: "India", value: 100);
LH.put(key: "China",value: 150);
LH.put(key: "US", value: 50);
LH.put(key: "Nepal", value: 10);

System.out.println(LH);

HashMap<String,Integer> H=new HashMap<>();
H.put(key: "India", value: 100);
H.put(key: "China",value: 150);
H.put(key: "US", value: 50);
H.put(key: "Nepal", value: 10);

System.out.println(H);
```

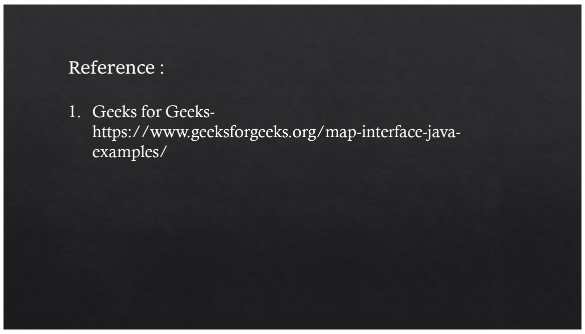
OUTPUT:

```
PS C:\Users\garys\Aniket> cd "c:\Users\garys\Ani
aps }
{China=150, India=100, Nepal=10, US=50}
{India=100, China=150, US=50, Nepal=10}
{China=150, US=50, Nepal=10, India=100}
PS C:\Users\garys\Aniket\DSA JAVA\Hashing>
```



Differences between HashMap, Linked HashMap, Tree Map :

Property	HashMap	Linked HashMap	TreeMap
Insertion Order	Not Predictable	Present	Sorted
Get / put / remove / containsKey	O(1)	O(1)	O(log_2(n))
Interfaces	Map	Map	Sorted Map
Null Keys / Values	Allowed	Allowed	Not allowed



Reference :

1. Geeks for Geeks- https://www.geeksforgeeks.org/map-interface-java-examples/

Thanks…