Java Interview Questions

- Internal working of HashMap.
- How does put(Object, Object) method works internally.
- What is Load Factor ?
- What is Hash Collision ?
- How does get(Object) method works internally.
- Usage of equals(Object) and hashCode() in HashMap.
- Usage of compareTo(Object) method in HashMap.
- Java 1.8 Enhancement with respect to HashMap.

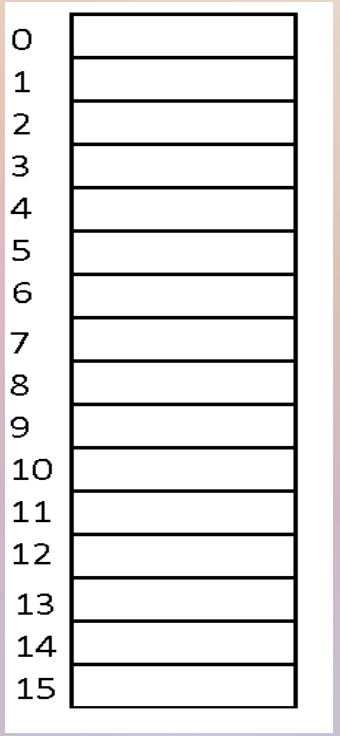
Internal working of HashMap

HashMap stores the data in the form of,
{“Key”:”Value”} pair and algorithm works only based on Key.

What happens internally when, we create new HashMap

```
Map<Key, Value> map = new HashMap<>();
                    or
Map<Object, Object> map = new HashMap<>();
                    or
Map<String, Integer> map = new HashMap<>();
```

JVM creates 16 entries (buckets) in the heap memory



Load Factor

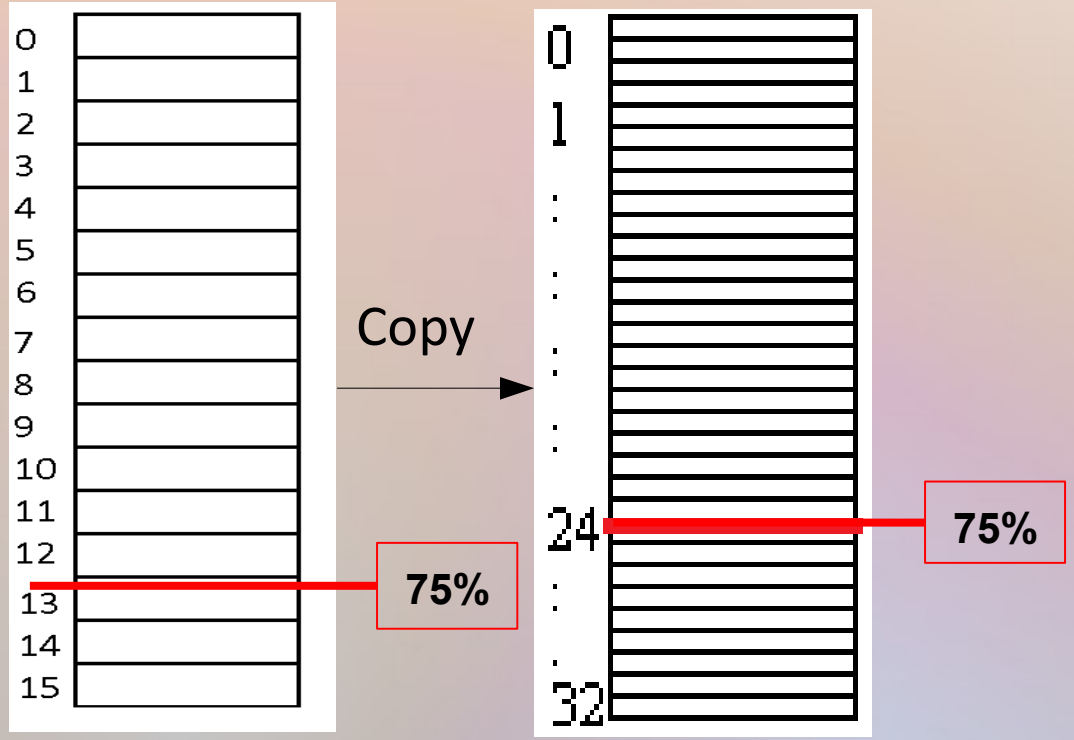
In case, I want to put few more entries or elements to HashMap?

What is Load Factor ?

If HashMap reaches more than 75% of it's capacity, then it doubles the existing capacity.

Load Factor = Threshold

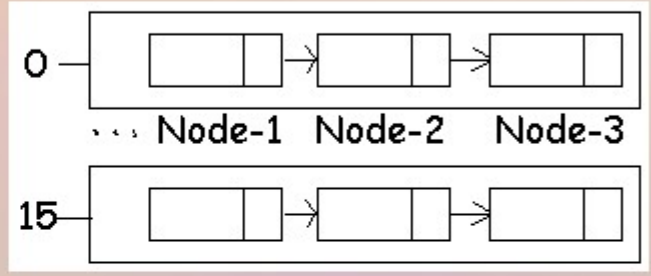
Load Factor = 75% or 0.75



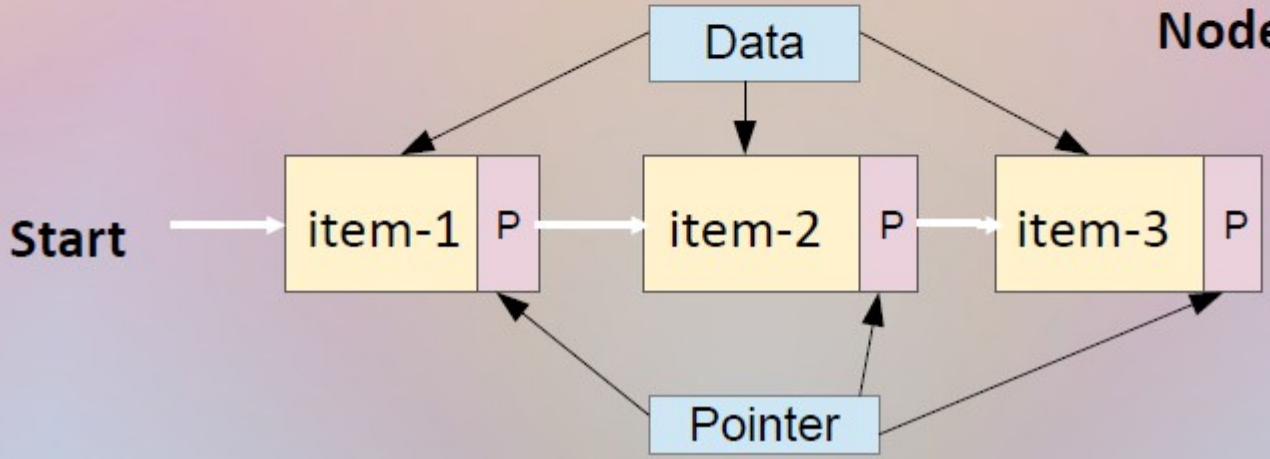
Internal working of HashMap

- Let's see what exactly inside a bucket
- Let's see what happens when we add nodes to Linked List

Bucket = Linked List



Node = Entry



Working of put(Object, Object) method

map.put(key , value);

map.put(“Aa” , 1);

“Aa” => String Object

Step-1: find hashCode

Object.hashCode()

“Aa”.hashCode()

hashcode = 2112

Step-2: find bucket index

HashCode & (length - 1)
= 2112 & 15

bucket index = 0

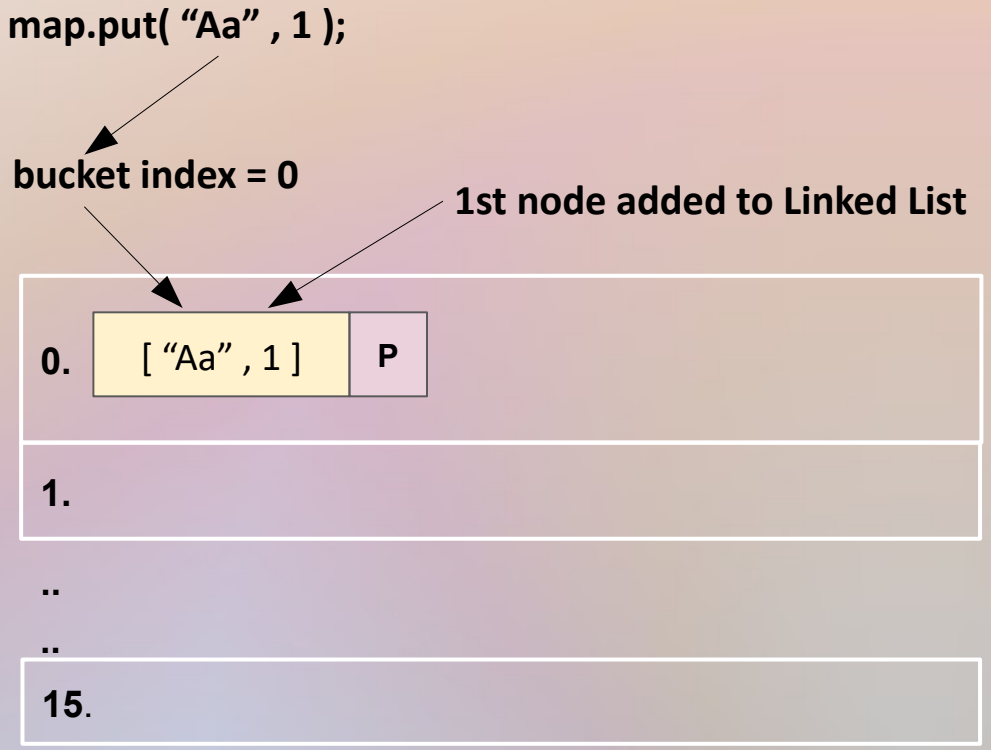
hashCode of Aa = 2112

&

index 0

Working of put(Object, Object) method

Let’s see what happens when we put(Object, Object) items to HashMap



hashCode of Aa = 2112

&

index 0

Use of hashCode() method

Let’s put one more item to the HashMap

map.put(“AB” , 2);

“AB” => String Object

Step-1: find hashCode

Object.hashCode()

“AB”.hashCode()

hashcode = 2081

Step-2: find bucket index

hashcode & (length-1)
2081 & 15

bucket index = 1

hashCode of Aa = 2081

&

index 1

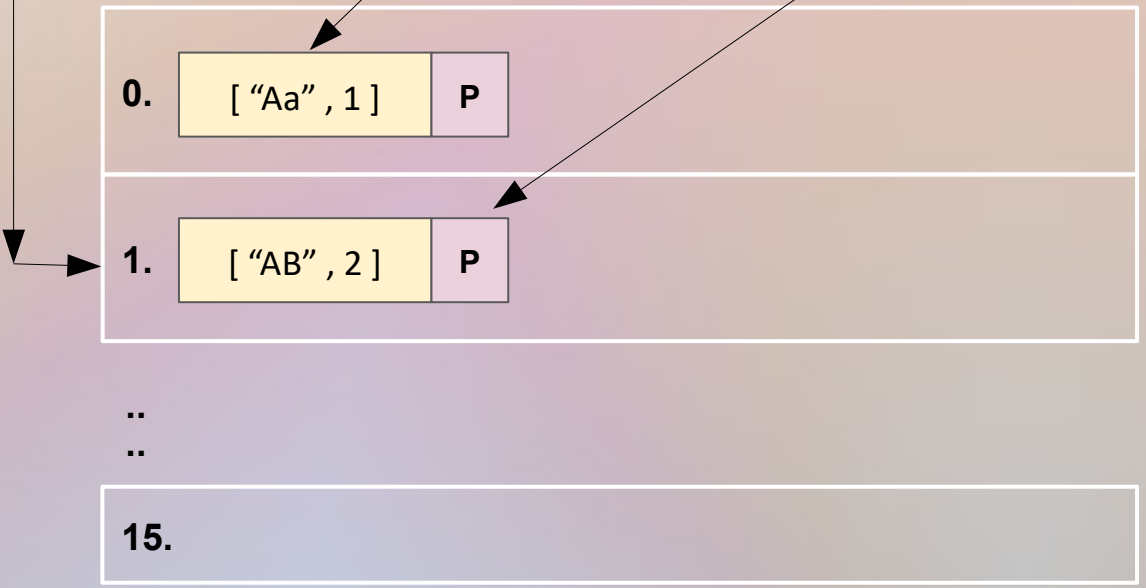
Working of put(Object, Object) method

map.put(“AB” , 2);

bucket index = 1

Already available node in 0 bucket

Newly added node in 1st bucket



hashCode of Aa = 2112 & index 0
hashCode of AB = 2081 & index 1

Hash Collision

Let’s put one more item to the HashMap
hash-collision

map.put(“BB” , 3);

“BB” => String Object

Step-1: find hashCode

Object.hashCode()

“AB”.hashCode()

hashcode = 2112

Step-2: find bucket index

hashcode & (length-1)
2112 & 15

bucket index = 0

hashCode of Aa = 2112 & index 0
hashCode of AB = 2081 & index 1
hashCode of BB = 2112 & index 0

Hash Collision

map.put(“BB” , 3);

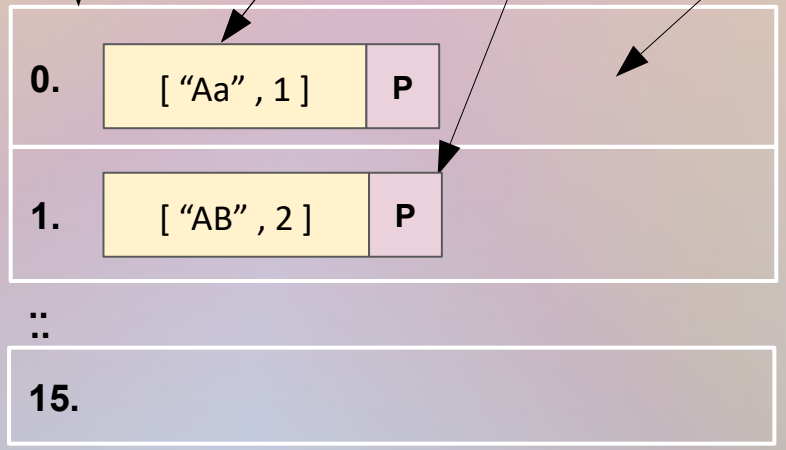
bucket index = 0

[“BB” , 3]

Already available nodes in different buckets

Hash Collision

Trying to put [“BB”, 3] node in 0 bucket



hashCode of Aa = 2112 & index 0
hashCode of AB = 2081 & index 1
hashCode of BB = 2112 & index 0

Use of equals(Object) in HashMap

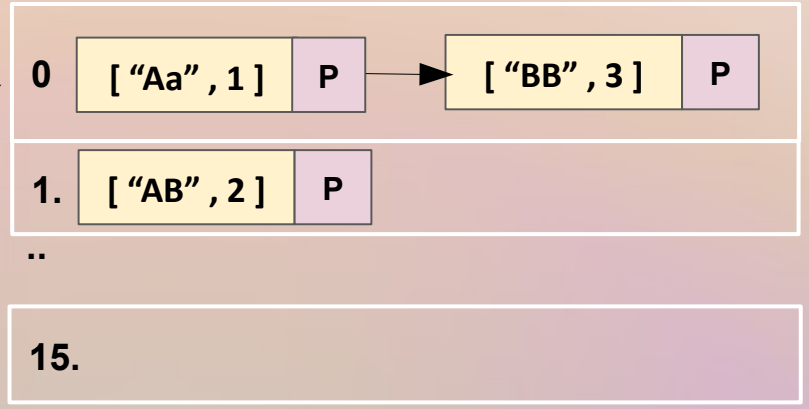
map.put(“BB” , 3);

bucket index = 0

Check if two objects are equal?

Object.equals(Object)

“Aa”.equals(“BB”)



hashCode of Aa = 2112 & index 0
hashCode of AB = 2081 & index 1
hashCode of BB = 2112 & index 0

Internal working of HashMap.

map.put(“key” , “value”)

Find hashCode() of the key:
“key”.hashCode()

Find bucket index using hashcode:
Hashcode & (length - 1)

Hash Collision

Yes

Key already present??
“key”.equals(“existing-key”)

Yes

Replace the
value
of existing node
of Linked List

No

Add to
Linked List as next node

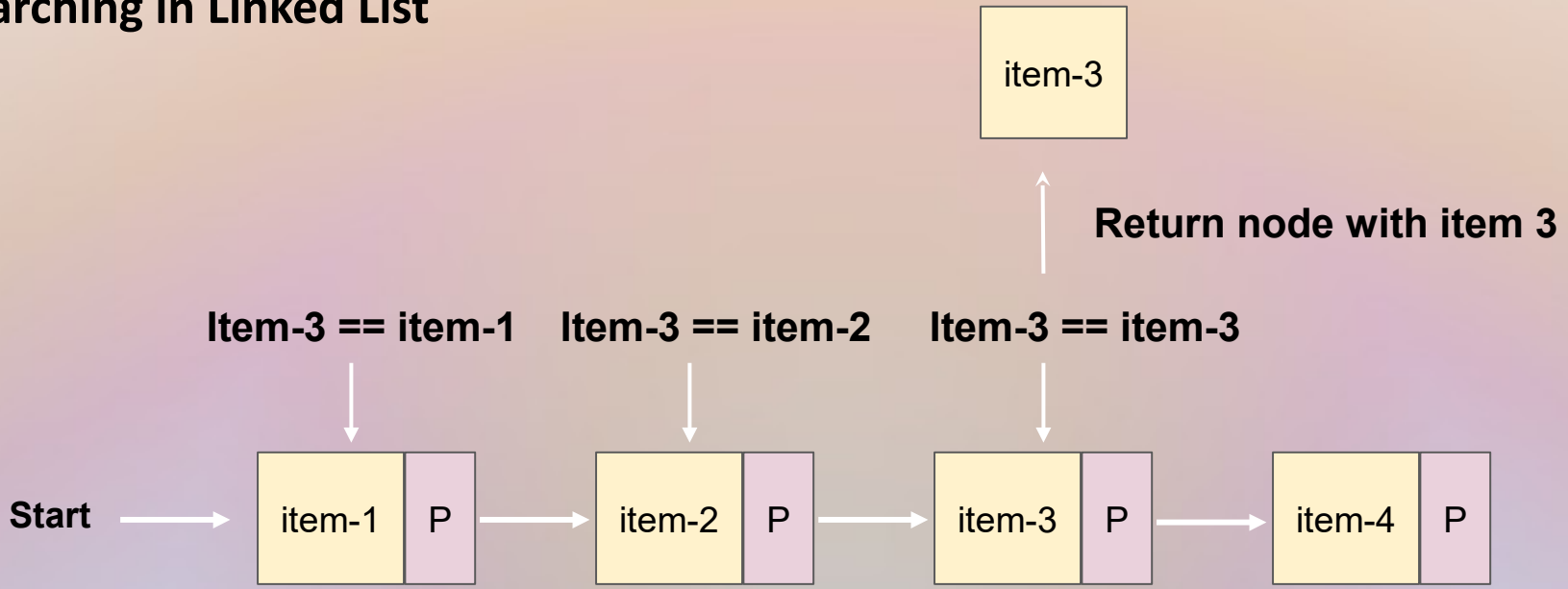
No

Simply add to Linked List as first node

Working of get(Object) method

How to find node with item-3 ?

Searching in Linked List



Working of compareTo(Object) method

- CompareTo(-) method is used, when we are putting huge number of similar entries (elements) to the HashMap.
- It is used to compare the objects and arrange them in proper order based on binary tree data structure.
- Binary tree data structure is implemented in Java 1.8
- By overriding the compareTo(Object) method, we can get the best performance from HashMap
- It is recommended to override compareTo(Object) method in every class, that is using as a key to HashMap.

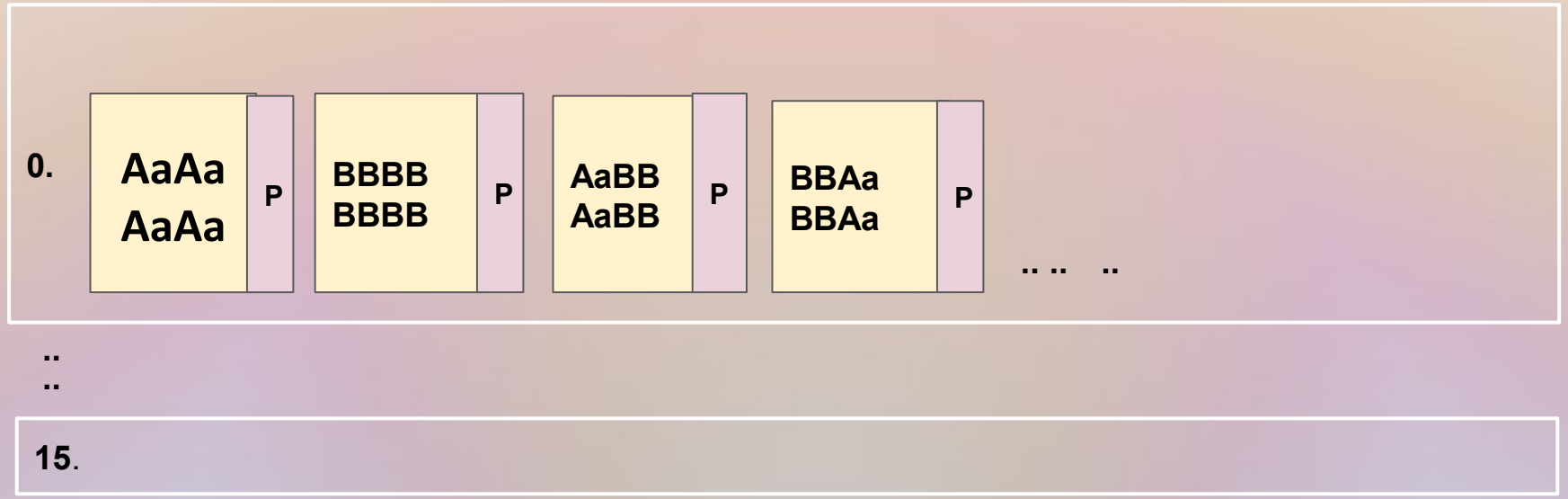
Java 1.8 Enhancement to HashMap

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

```
map.put("AaAaAaAa", 1);   →   -540425953
map.put("BBBBBBBB", 2);   →   -540425953
map.put("AaBBAaBB", 3);   →   -540425953
map.put("BBAaBBAa", 4);   →   -540425953
map.put("AaAaBBBB", 5);   →   -540425953
map.put("BBBBAaAa", 6);   →   -540425953
map.put("AaAaAaBB", 7);   →   -540425953
map.put("BBBBBBAa", 8);   →   -540425953
map.put("AaBBAaAa", 9);   →   -540425953
map.put("BBAaBBBB", 10);  →   -540425953
```

Java 1.8 Enhancement to HashMap

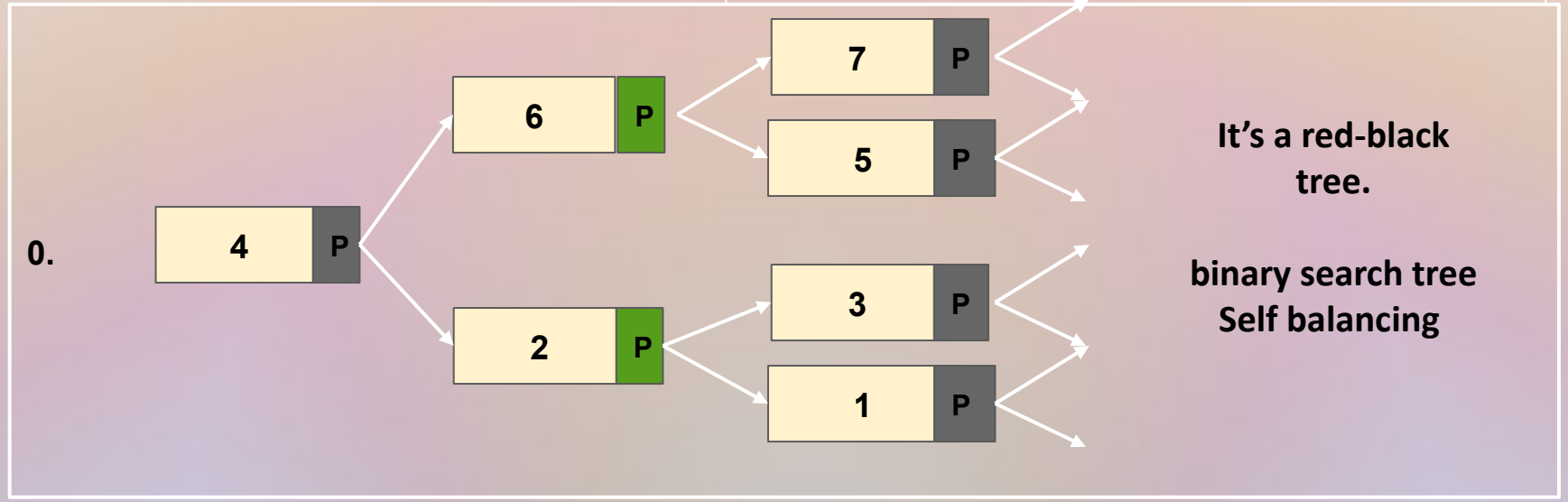
Performance Degradation



Java 1.8 Enhancement to HashMap

TREEIFY_THRESHOLD

Uses compareTo() method.
compareTo() method is use to check the order of items



..

15.

Download the source code from GIT

Please download the source code from below GIT repository URL :

Enter this URL on browser - https://github.com/ChandraKodam5/java
Click on **Code** Drop-down → Click on **Download ZIP** → Open eclipse IDE

Goto **File** → **Import** → Existing projects into workspace → **Next** → click on **checkbox** (**select archive file**) → click on **Browse** → choose the downloaded **HashMapProject** ZIP file.

OR

Type the below command in git bash
$ git clone https://github.com/ChandraKodam5/java.git
Goto **File** → **Import** → Existing projects into workspace → **Next** → click on **checkbox** (**select archive file**) → click on **Browse** → choose the downloaded **HashMapProject** ZIP file.

Thank You

GIT Repository URL

https://github.com/ChandraKodam5/java

References

https://www.youtube.com/watch?v=uyHfT5vSvco