1) why immutable key should be present in hashmap

we know the process right....first hashcode of that key will get created and then index will get created ,

if I'll make my object as a muttable then anybody can change the content ,

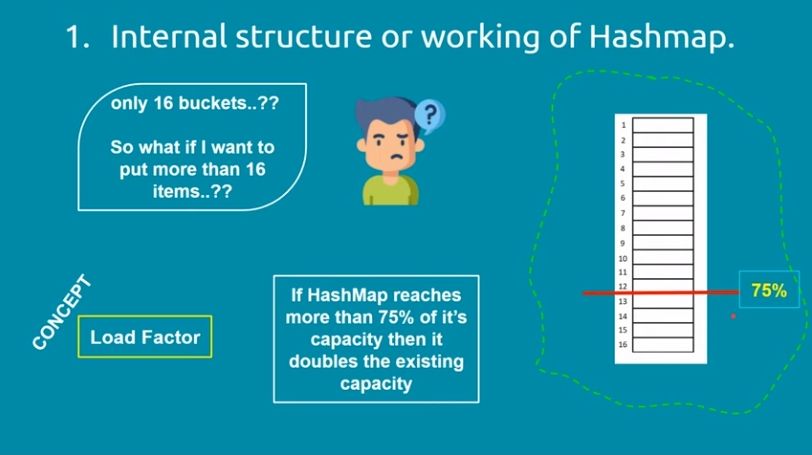
and after changing the content if ill try to find the that object inside the hasmap then hashcode will get again calculated and then index will also get calculated based on the hashcode , and now index will be the different bcz hascode will be different .

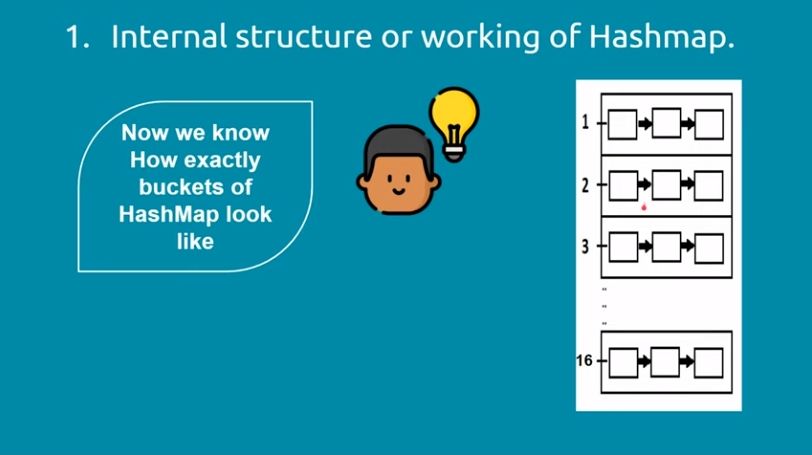
therefore it will give the null as the output ....

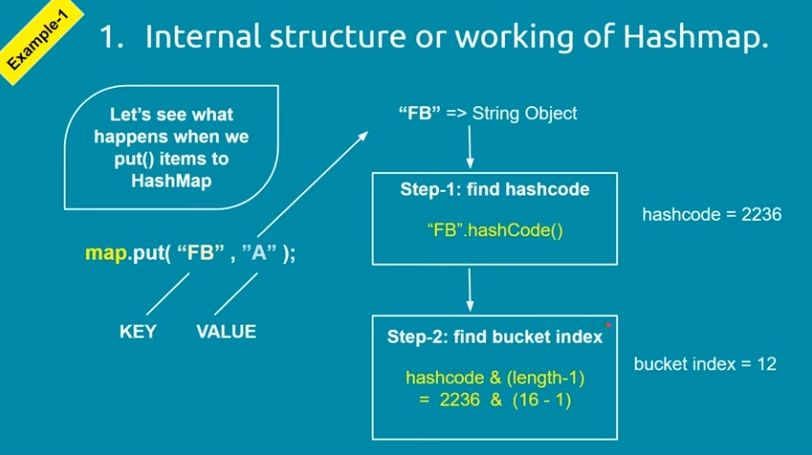
2) if immutable what make it difference in performance

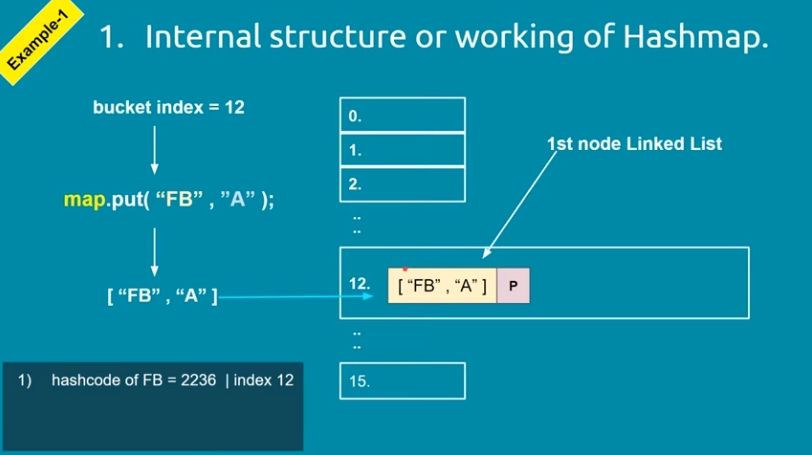
Actually while putting the value in hashmap that hashcode gets cached called as cached hashcode .

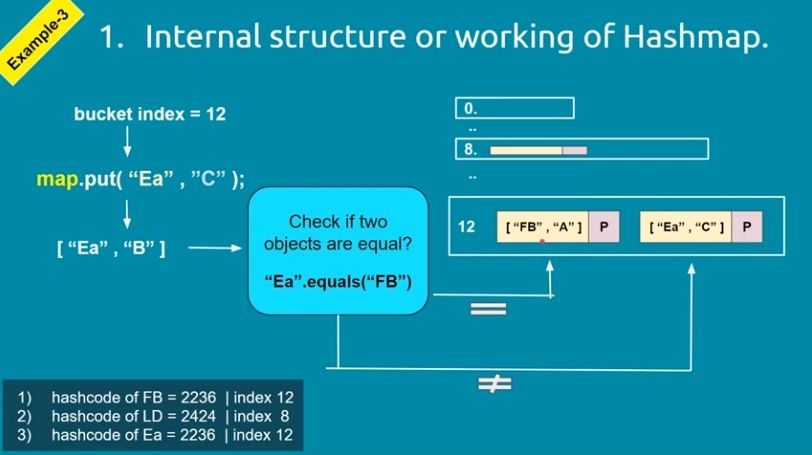
so hashcode will get calculated only once during putting , it wont get calculated while getting the value..



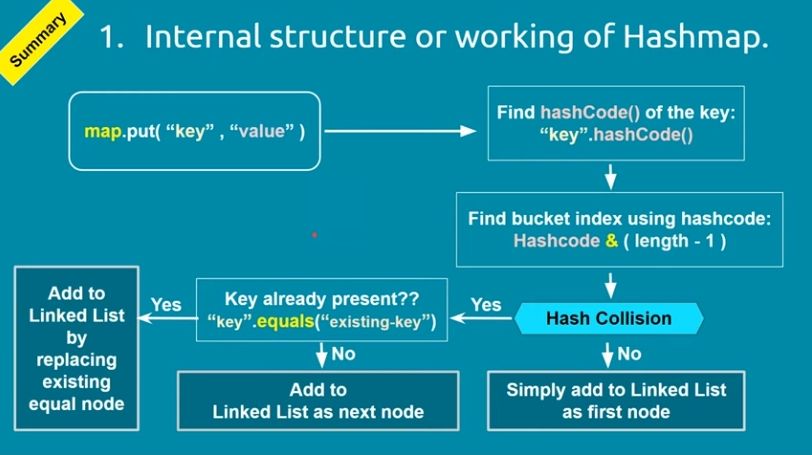






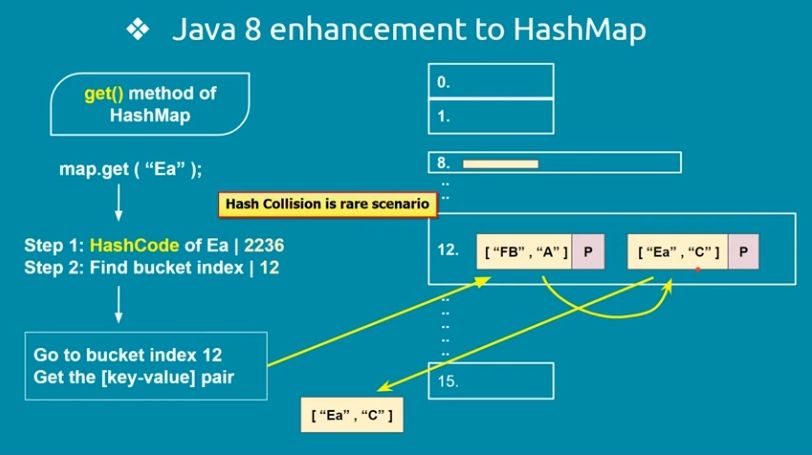


Put method



.get

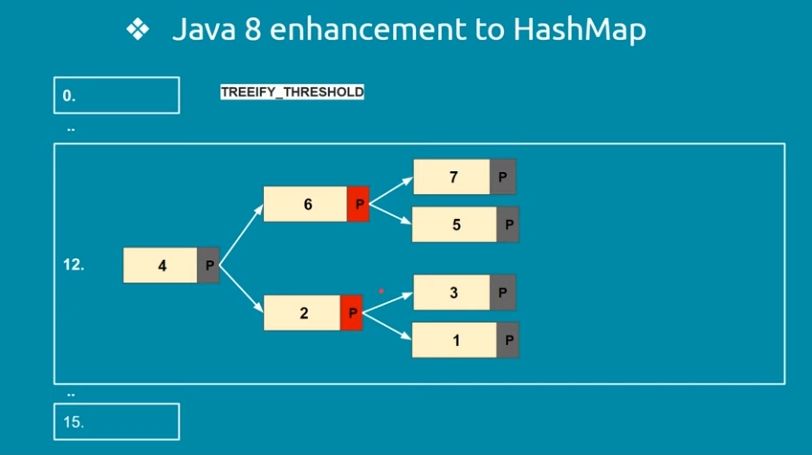
it get the result very quickly , bcz of hashcode and the index of bucket...

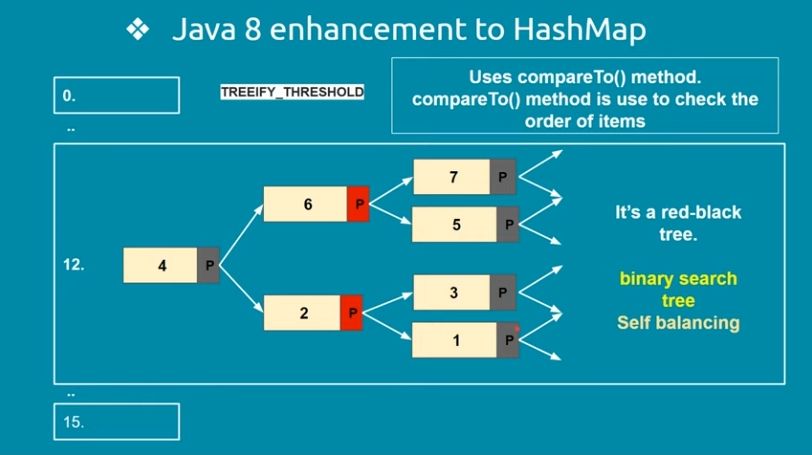


if multiple node will be there then performance will get reduce ....

therefore enhancement in java 8

if number of nodes in bucket will cross certain threshold called as Treefy threshold then lnked list converted into tree





Incase of null key , Hashmap implementation consider it as special case and doesnot call hashCode method instead it stores Entry object to 0 bucket location.