**How to Create Custom Immutable Class?**

* There are many immutable classes like String, Boolean, Byte, Short, Integer, Long, Float, Double etc. In short, all the wrapper classes and String class is immutable.
* We can also create immutable class by creating final class that have final data members.
* No setter methods.

**ImmutableDemo.java**

**public** **final** **class** Employee

{

**final** String pancardNumber;

**public** Employee(String pancardNumber)

{

**this**.pancardNumber=pancardNumber;

}

**public** String getPancardNumber(){

**return** pancardNumber;

}

}

**Priority Que Java**

* The **PriorityQueue** class implements the [Queue interface](https://www.educative.io/edpresso/what-is-the-java-queue-interface) in Java
* A PriorityQueue is beneficial when the objects are supposed to be processed based on the priority rather than the First-In-First-Out
* The internal working of the PriorityQueue is based on the Binary Heap.

**HashMap Internal**

**Hashing:-**

* Process of converting Object to into form
* Necessary write to method hashcode for better performance and avoid issues.
* hash code of null will always be 0.

**Hashcode Method**

* hashCode() method is used to get the hash Code of an object.
* hashCode() method of object class returns the memory reference of object in integer form

**Equal Method**

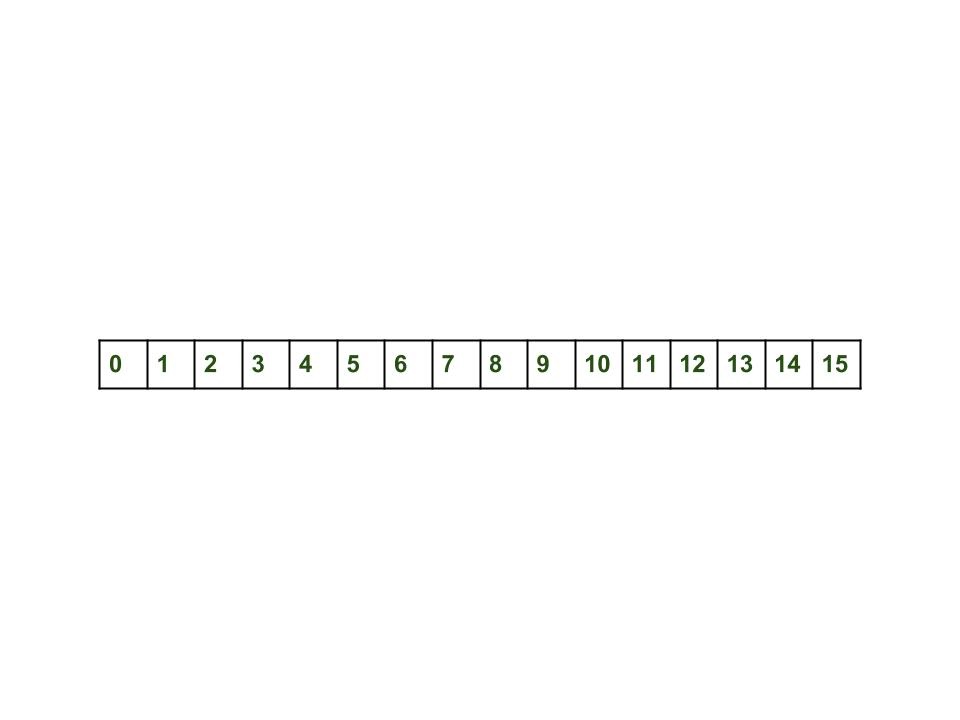
* Equals method is used to check that 2 objects are equal or not.
* This method is provided by Object class.
* You can override this in your class to provide your own implementation.
* HashMap uses equals() to compare the key whether the are equal or not.
* If equals() method return true, they are equal otherwise not equal.

**Buckets**

* A bucket is one element of HashMap array.
* It is used to store nodes.
* Two or more nodes can have the same bucket. In that case link list structure is used to connect the nodes.

**Index**

**index = hashCode(key) & (n-1).**

**Initially Empty hashMap: Here, the hashmap is size is taken as 16.   
 HashMap map = new HashMap();   
**

**Callable and Future in Java**

**Need of Callable**

* There are two ways of creating threads – one by extending the Thread class and other by creating a thread with a Runnable.
* **Limitation** :- Can’t return when threads end
* **Callable:-**
  + the call() method needs to be implemented which returns a result on completion.
  + Note that a thread can’t be created with a Callable, it can only be created with a Runnable.