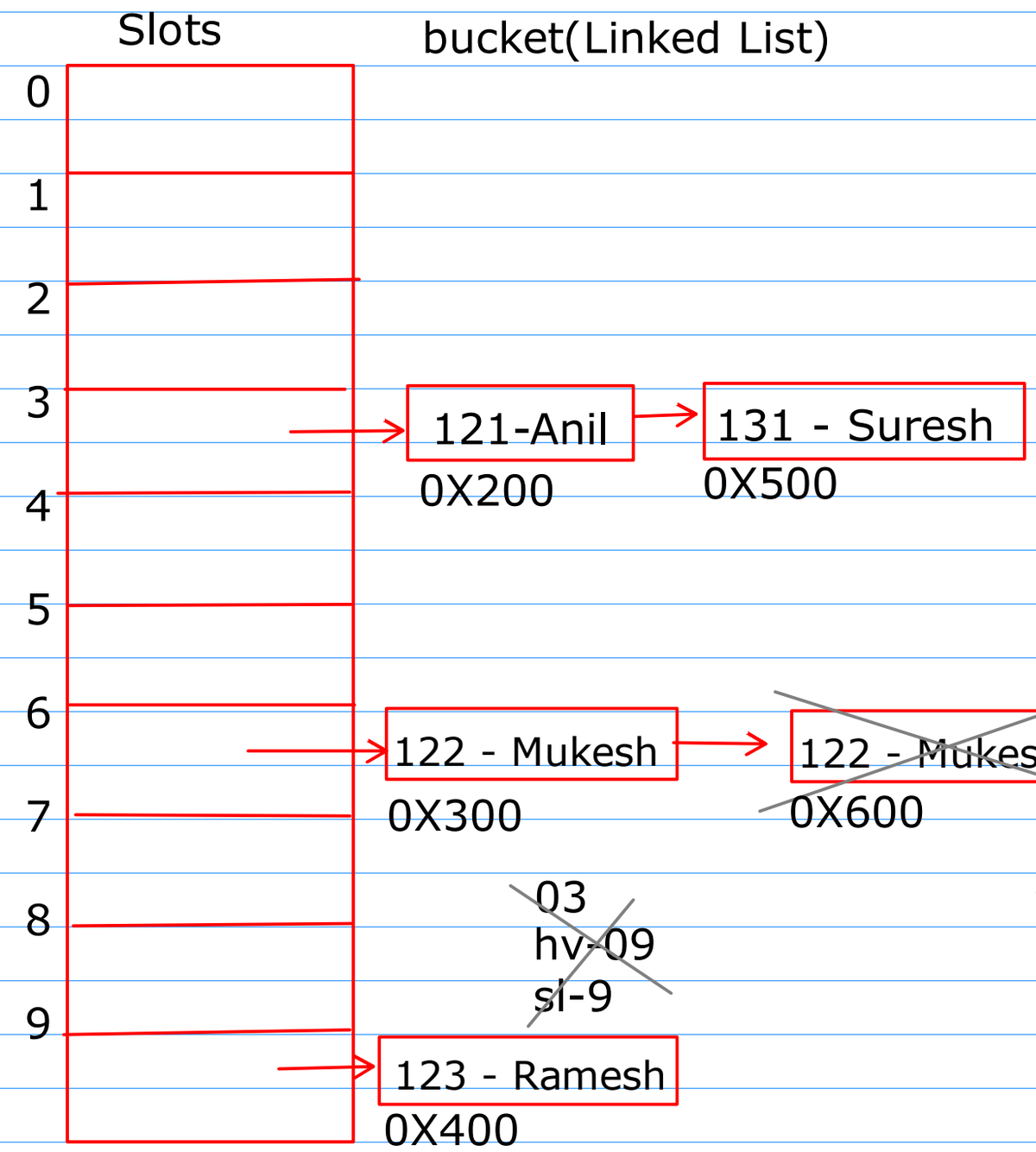
Collection
List,Queue,Set

Hashtable




```
int hashFunction(){
    hashvalue = id * 3
}
```

slotvalue = hashvalue % noofslots

hashvalue = 121*3 = 363
slotvalue = 363%10 = 3

hashcode()
equals()

hashvalue = 122*3 = 366
slotvalue = 366 %10 = 6

hashvalue = 123*3 = 369
slotvalue = 369%10 = 9

hashvalue = 131*3 = 393
slotvalue = 393%10 = 3

Loadfactor = noofelements / noofslots
= 5/10 = 0.5
Loadfactor = 10/10 = 1
Loadfactor = 15/10 = 1.5

1. Open Addressing no of elements <= no of slots
2. Seperate Chaining no of elements > no of slots

LF<=1 -> Open Addressing
LF>1 -> Seperate Chaining

Ready made Data Structure
-> Collection Framework (Encapsulation)
-> For users -> Abstraction

Linked List

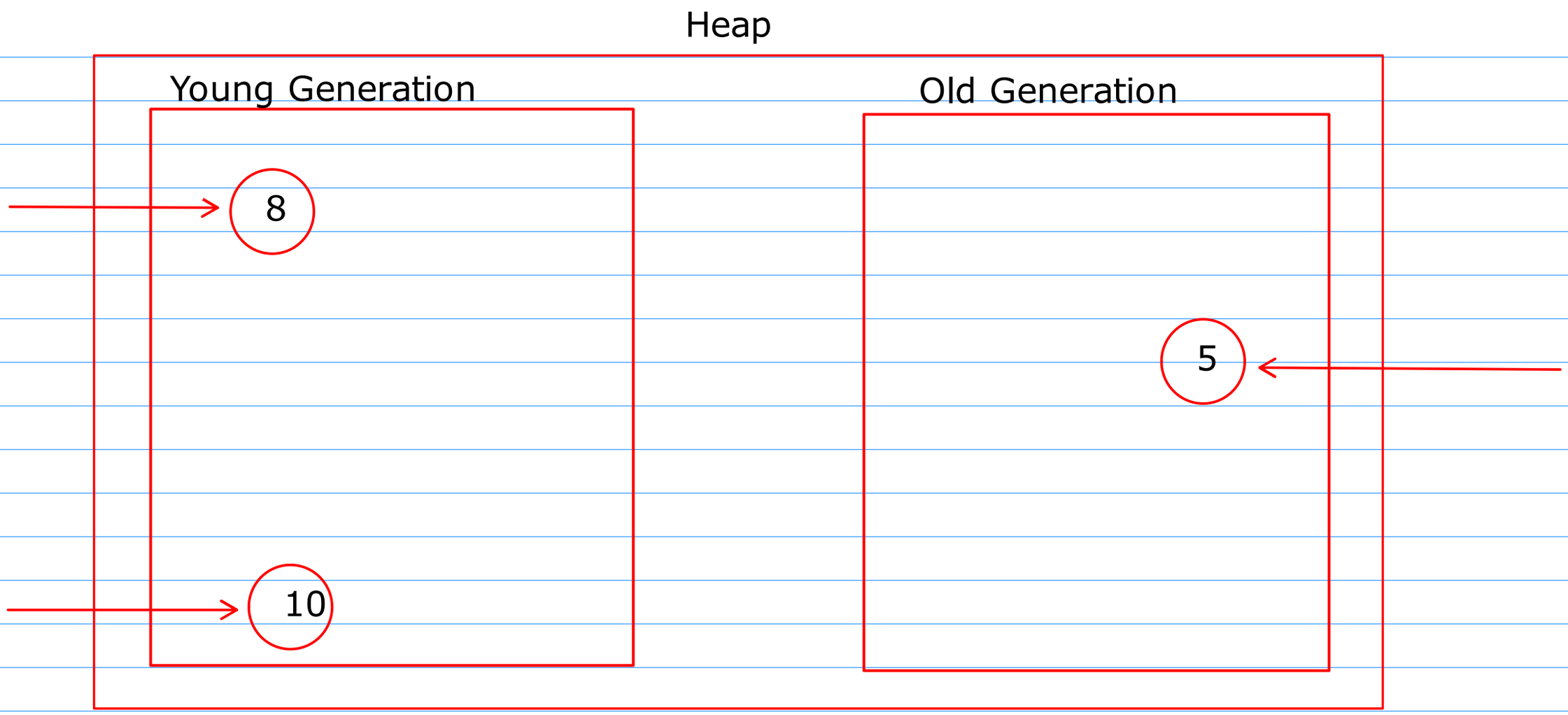
```
add(E e){
statemet1
statemet2
statemet2
while(e.eqlas())
if(statemet)
}
else
add
```

```java
    }

    public Employee(int id, String name) {
        this.id = id;
        this.name = name;
    }

    @Override
    public int hashCode() {
        return id * 13;
    }

//  @Override
//  public boolean equals(Object obj) {
//      if (this == obj)
//          return true;
//      if (!(obj instanceof Employee))
//          return false;
//      Employee other = (Employee) obj;
//      return id == other.id;
//  }

    @Override
```

```
<terminated> Program02 [Java Application] D:\Softwares\sts-4.19.0\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_
Employee [id=121, name=Anil]hashvale - 1573
Employee [id=132, name=Mukesh]hashvale - 1716
Employee [id=143, name=Ramesh]hashvale - 1859
Employee [id=132, name=Mukesh]hashvale - 1716
```

Set<K,null> -> Map<K,V>
HashSet<K,null> -> HashMap<K,V>
LinkedHashSet<K,null> -> LinkedHashMap<K,V>
TreeSet<K,null> -> TreeMap<K,V>

String Tokenizer




```
Test t1 = new Test(); // GC
t1 = null;

Test t1= new Test(); // GC
t1 = new Test();
```

```
void method(){
Test t1 = new Test(); // GC
}
```

GC ->

- it performs the GC in two ways
1. Minor GC (On Young generation)
2. Major GC (On Young + Old Generation)

Mark and Compact Algorithm