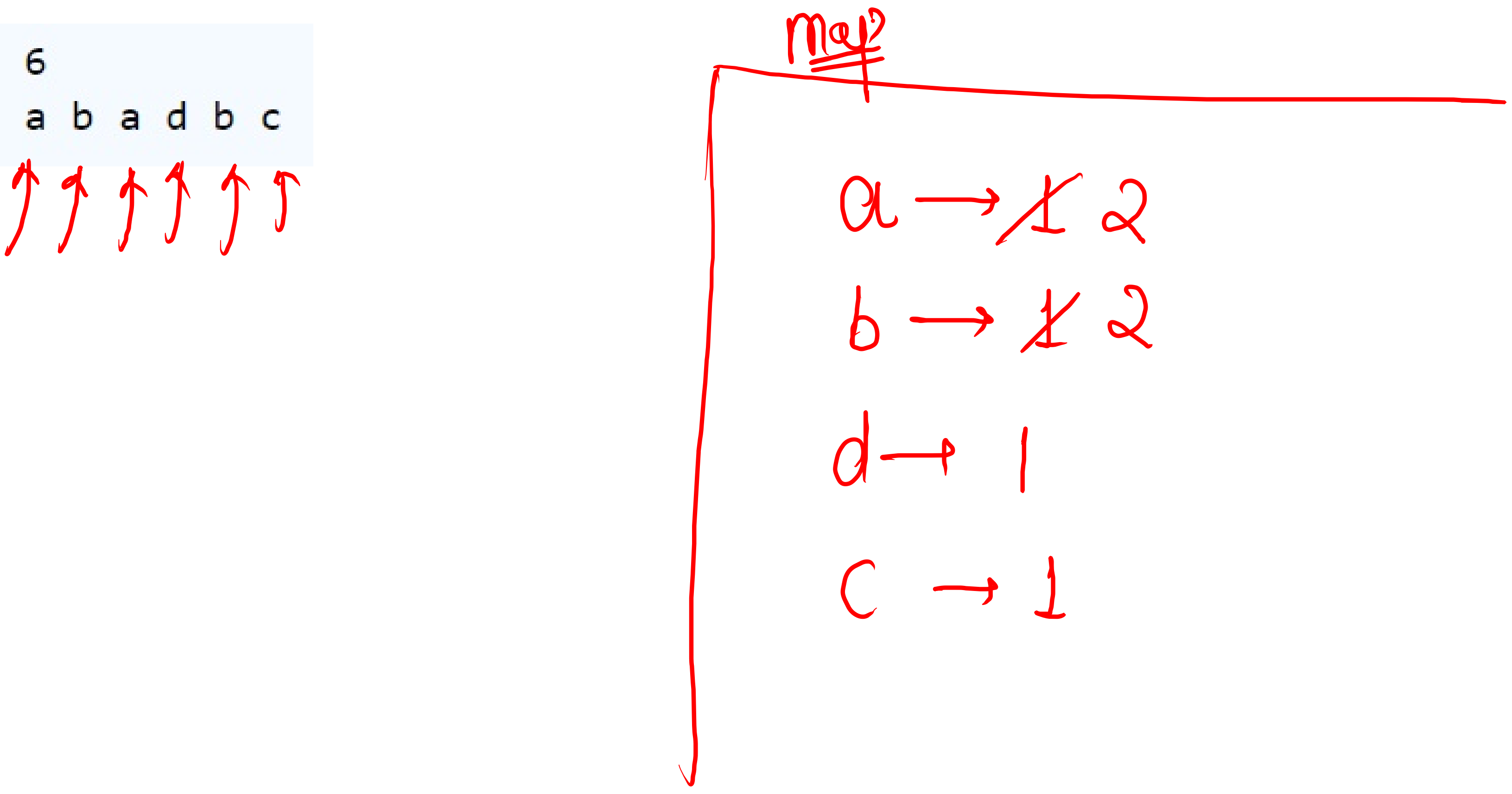
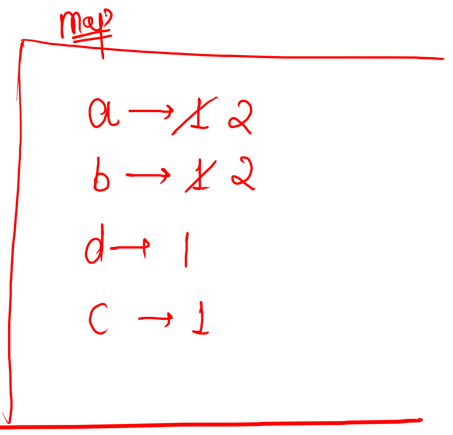
Character and it's Frequency

```
6
a b a d b c
```



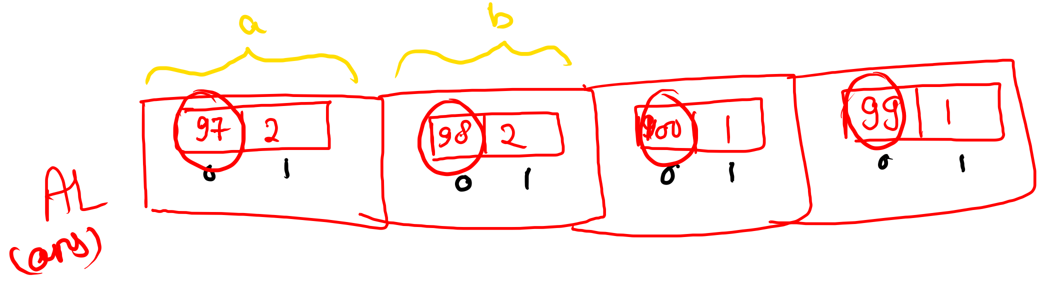
```java
public static void countFreq(ArrayList<Character> arr) {
    HashMap<Character, Integer> map = new HashMap<>();
    for (int i = 0; i < arr.size(); i++) {
        if ( !map.containsKey( arr.get(i) ) ) {
            map.put( arr.get(i), 1 );
        } else {
            int freq = map.get( arr.get(i) );
            map.put( arr.get(i), freq + 1 );
        }
    }

    ArrayList<int[]> ans = new ArrayList<>();
    for (Map.Entry<Character,Integer> entry : map.entrySet()) {
        char key = entry.getKey();
        int val = entry.getValue();
        int[] ar = new int[]{key, val};
        ans.add(ar);
    }

    Collections.sort(ans, (a, b) -> {
        return a[0] - b[0];
    });

    for (int i = 0; i < ans.size(); i++) {
        int[] rem = ans.get(i);
        System.out.println( (char)rem[0] + " " + rem[1] );
    }
}
```

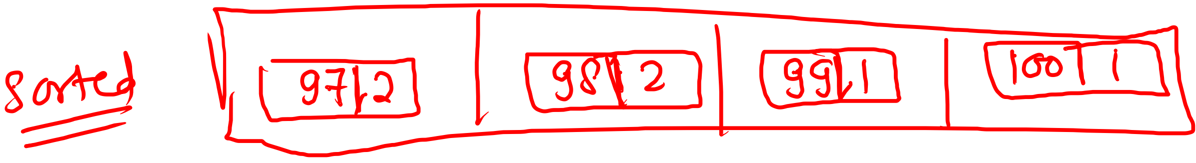
```
6
a b a d b c
```



key = ~~a~~ ~~b~~ ~~d~~ c

val = ~~2~~ ~~2~~ ~~1~~ 1





a, 2 b, 2 c, 1 d, 1

code

```java
public static void countFreq(ArrayList<Character> arr) {
    HashMap<Character, Integer> map = new HashMap<>();
    for (int i = 0; i < arr.size(); i++) {
        if ( !map.containsKey( arr.get(i) ) ) {
            map.put( arr.get(i), 1 );
        } else {
            int freq = map.get( arr.get(i) );
            map.put( arr.get(i), freq + 1 );
        }
    }

    ArrayList<Character> ans = new ArrayList<>();
    for (Map.Entry<Character,Integer> entry : map.entrySet()) {
        char key = entry.getKey();
        ans.add(key);
    }

→   Collections.sort(ans);

    for (int i = 0; i < ans.size(); i++) {
        System.out.println( ans.get(i) + " " + map.get( ans.get(i) ) );
    }
}
```

key (sorted)

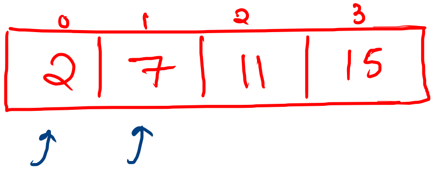
get value from HM

Two Sum 14

```
4 9
2 7 11 15
```

$val1 + val2 == tar$

$n = 4$, $tar = 9$ $(2+7)$



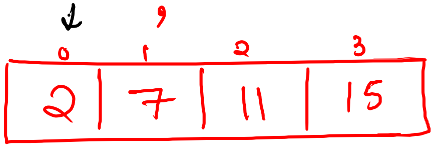
approch

(num) (index)

map (Integer, Integer)

$2 \longrightarrow 0$	
$7 \longrightarrow 1$	
$11 \longrightarrow 2$	
$15 \longrightarrow 3$	
$num \rightarrow idx$	

dry run



✓ $\begin{cases} val1 = 2 \\ val2 = 9 - 2 = 7 \end{cases}$

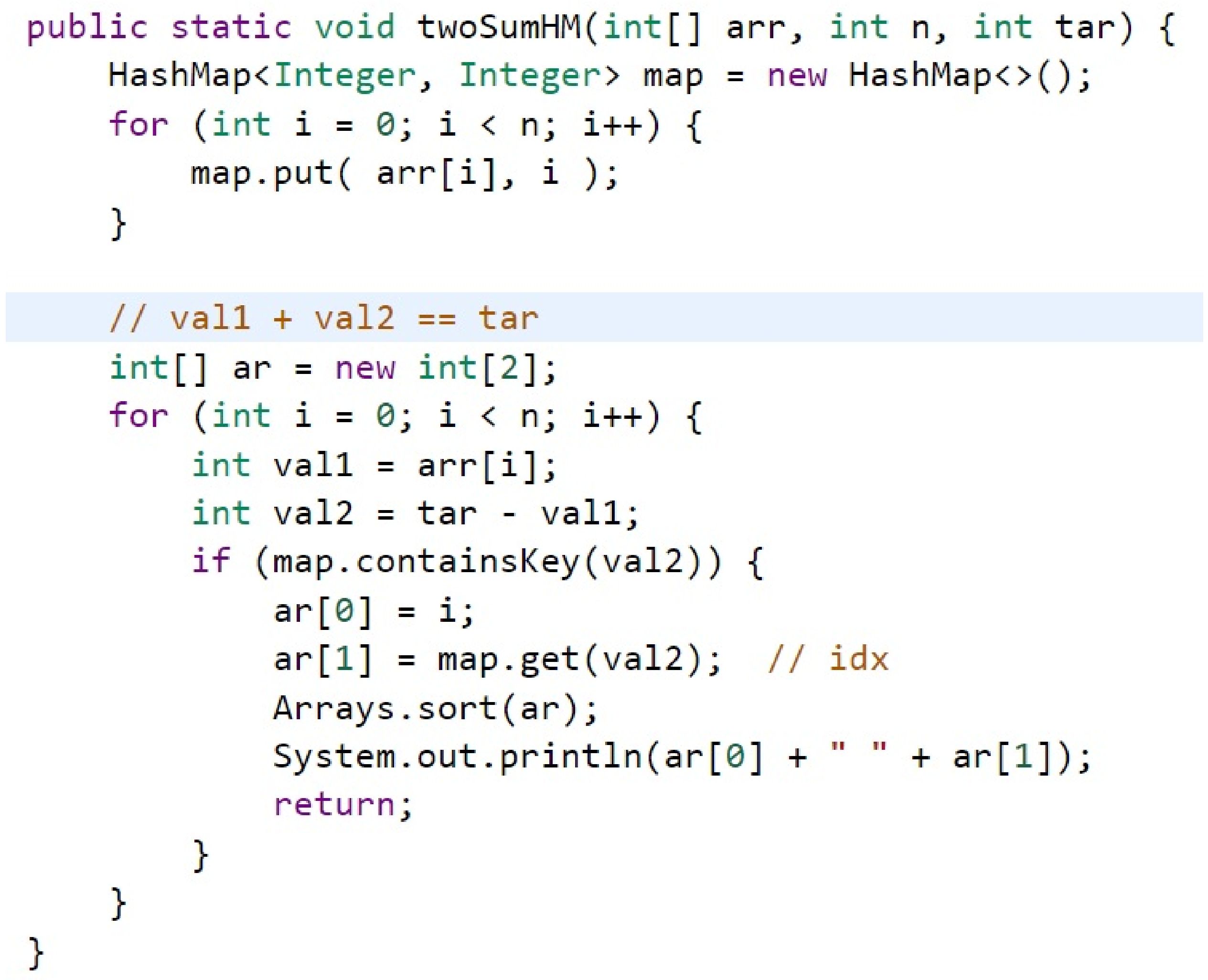
$= tar - val1$

$\begin{cases} val1 = 7 \\ val2 = 9 - 7 = 2 \end{cases}$

✗ $\begin{cases} val1 = 11 \\ val2 = 9 - 11 = -2 \end{cases}$

$\begin{cases} val1 = 15 \\ val2 = 9 - 15 = -6 \end{cases}$

Code



```java
public static void twoSumHM(int[] arr, int n, int tar) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        map.put( arr[i], i );
    }

    // val1 + val2 == tar
    int[] ar = new int[2];
    for (int i = 0; i < n; i++) {
        int val1 = arr[i];
        int val2 = tar - val1;
        if (map.containsKey(val2)) {
            ar[0] = i;
            ar[1] = map.get(val2);  // idx
            Arrays.sort(ar);
            System.out.println(ar[0] + " " + ar[1]);
            return;
        }
    }
}
```

⇒ HashSet (valdataType)

adv

↳ it always store values in sorted order

↳ all f^n have complexity of O(1)

↳ it will also override already present value

Syntex :-

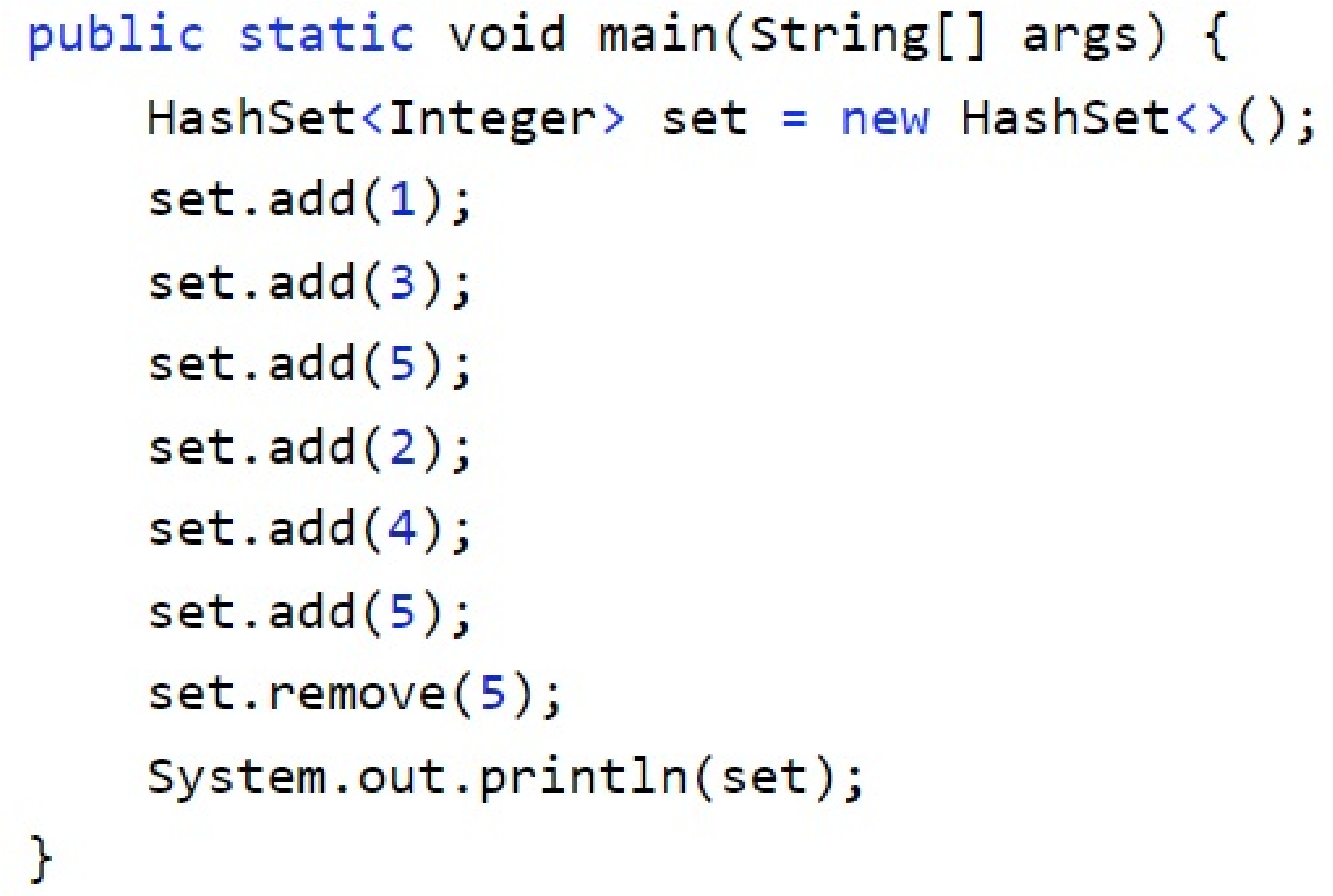
```
HashSet < KeyDataType> set = new HashSet<>();
```

→ **HashSet fⁿ**

- set.add(val); → to add
- set.contains(val); → to check
  return boolean DT
- Set.size();
- set.isEmpty();
- set.remove(val); → to delete.

set

→ add and remove function



```java
public static void main(String[] args) {
    HashSet<Integer> set = new HashSet<>();
    set.add(1);
    set.add(3);
    set.add(5);
    set.add(2);
    set.add(4);
    set.add(5);
    set.remove(5);
    System.out.println(set);
}
```

Unique Number of Occurrences

arr = [1, 1, 1, 1, 2, 2, 3, 3, 3, 3]

Dry run

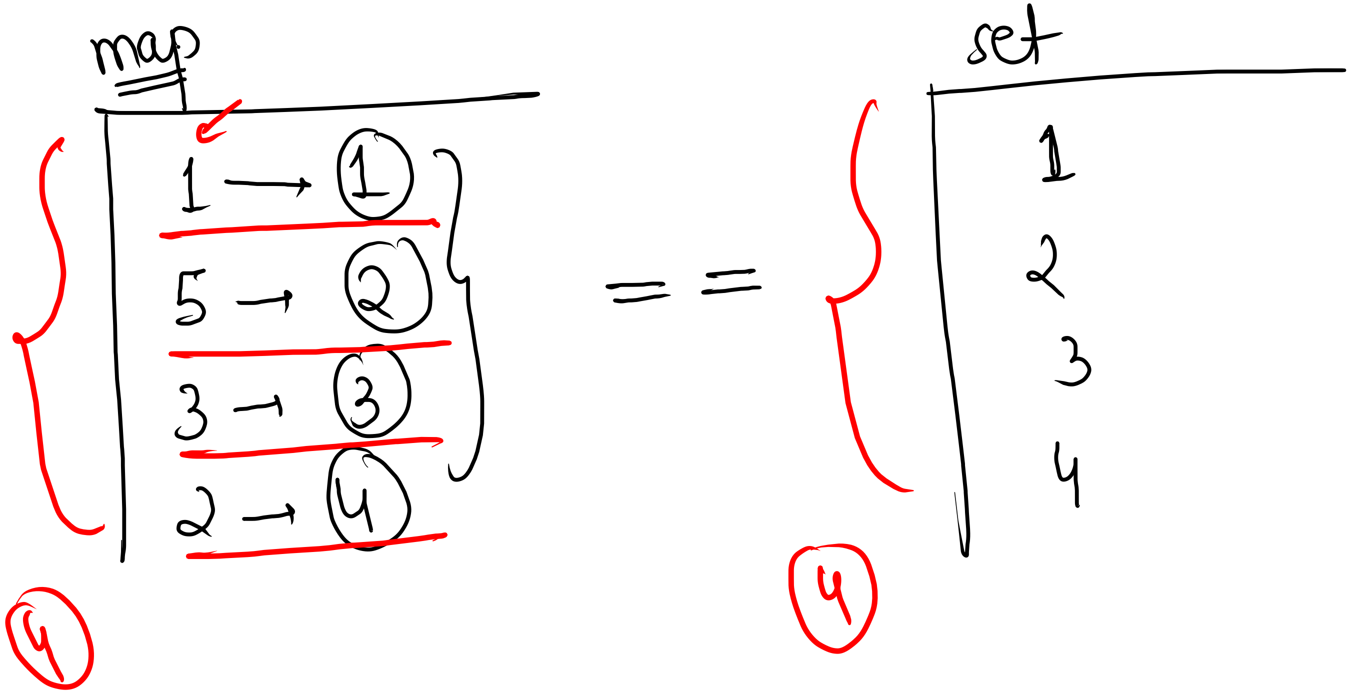
$1 \longrightarrow 4$ ✓

$2 \longrightarrow 2$

$3 \longrightarrow 4$ ✓

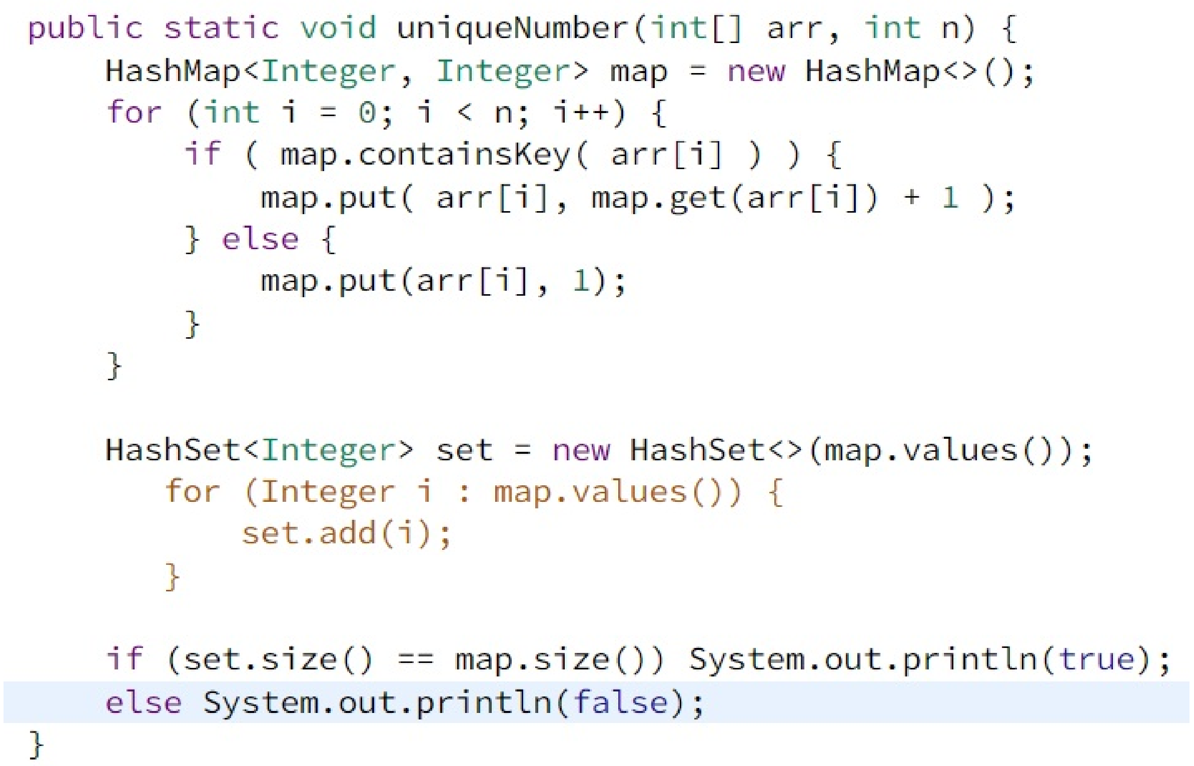
False

arr = [1, 5, 5, 3, 3, 3, 2, 2, 2, 2]



Unique Number of Occurrences

code



```java
public static void uniqueNumber(int[] arr, int n) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        if ( map.containsKey( arr[i] ) ) {
            map.put( arr[i], map.get(arr[i]) + 1 );
        } else {
            map.put(arr[i], 1);
        }
    }

    HashSet<Integer> set = new HashSet<>(map.values());
       for (Integer i : map.values()) {
           set.add(i);
       }

    if (set.size() == map.size()) System.out.println(true);
    else System.out.println(false);
}
```