**HashMap in Java**

**Assignment**

Q1. Implement a Map in java which takes the input and print the list in sorted order based on key.

Input: 5- Rahul, 7 Lakshman, 1 Ram, 4 Krrish, 2 Lakshay,

Output: {1=Ram, 2=Lakshay, 4=Krrish, 5=Rahul, 7=lakshman}

Solution:

**package** Assignment\_HashMap\_And\_Stack;

**import** java.util.Scanner;

**import** java.util.TreeMap;

**public** **class** HashMap\_Problem1

{

**public** **static** **void** main(String[] args)

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the number of elements of Key-pair: ");

**int** n = sc.nextInt();

TreeMap<Integer, String> tm = **new** TreeMap<>();

System.***out***.print("Enter the elements in Key and value : ");

**for**(**int** i=0;i<n;i++)

{

**int** key = sc.nextInt();

String value = sc.next();

tm.put(key, value);

}

System.***out***.println("The value of Map is: " + tm);

}

}

/\*

Enter the number of elements of Key-pair: 5

Enter the elements in Key and value : 5 Rahul

7 Lakshman

1 Ram

4 Krish

2 Lakshay

The value of Map is: {1=Ram, 2=Lakshay, 4=Krish, 5=Rahul, 7=Lakshman}

\*/

Q2. Implement a Map in java which takes the input and print the list in sorted order based on value

Input: 5- Rahul, 7 Lakshman, 1 Ram, 4 Krrish, 2 Lakshay,

Output: {Rahul=5, krrish=4, lakshay=7, lakshman=2, ram=1}

Solution:

**package** Assignment\_HashMap\_And\_Stack;

**import** java.util.\*;

**public** **class** HashMap\_Problem2 {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the number of elements of Key-pair: ");

**int** n = sc.nextInt();

TreeMap<String, Integer> tm = **new** TreeMap<>();

System.***out***.print("Enter the elements in Key and value : ");

**for**(**int** i=0;i<n;i++){

**int** key = sc.nextInt();

String value = sc.next();

tm.put(value, key);

}

System.***out***.println("The value of Map is: " + tm);

}

}

/\*

Enter the number of elements of Key-pair: 5

Enter the elements in Key and value : 5 Rahul

7 Lakshman

1 Ram

4 Krish

2 Lakshay

The value of Map is: {Krish=4, Lakshay=2, Lakshman=7, Rahul=5, Ram=1}

\*/

Q3.Detect if an Array contains a duplicate element. At Most 1 duplicate would be there.

input: 1,2,3,4

Output:No

Input: 1,2,3,4,1

Output:Yes

Solution:

Approach :-

We would start traversing the array

As me move ahead, we would keep adding the element in map

If we found any element is already added in the map that means we have found our duplicate

If no element is found then there is no duplicate.

**package** Assignment\_HashMap\_And\_Stack;

**import** java.util.\*;

**public** **class** HashMap\_Problem3 {

**public** **static** **void** main(String[] args) {

@SuppressWarnings("resource")

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the number of elements of array: ");

**int** n = sc.nextInt();

**int**[] arr = **new** **int**[n];

System.***out***.print("Enter the elements of array: ");

**for** (**int** i = 0; i < n; i++) {

arr[i] = sc.nextInt();

}

HashMap<Integer, Integer> tm = **new** HashMap<>();

**int** answer = 0;

**for** (**int** i = 0; i < n; i++) {

**if** (tm.containsKey(arr[i])) {

System.***out***.println("Yes");

answer = 1;

**break**;

}

tm.put(arr[i], 1);

}

**if** (answer == 0)

System.***out***.println("No");

}

}

/\*

Enter the number of elements of array: 4

Enter the elements of array: 1 2 3 4

No

\*/

/\*

Enter the number of elements of array: 5

Enter the elements of array: 1 2 3 4 1

Yes

\*/

Q4. Given an array nums of size n, return the majority element.

Input: 4,2,7,1,9

Output:9

Solution:

**Approach**

We would start traversing the array and will store each element into the TreeMap.

We would simply return the last key of map.

**package** Assignment\_HashMap\_And\_Stack;

**import** java.util.\*;

**public** **class** HashMap\_Problem4 {

**public** **static** **void** main(String[] args) {

@SuppressWarnings("resource")

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the number of elements of array: ");

**int** n = sc.nextInt();

**int**[] arr = **new** **int**[n];

System.***out***.print("Enter the elements of array: ");

**for** (**int** i = 0; i < n; i++) {

arr[i] = sc.nextInt();

}

TreeMap<Integer, Integer> tm = **new** TreeMap<>();

**for** (**int** i = 0; i < n; i++) {

tm.put(arr[i], 1);

}

System.***out***.println("Largest Element of Map is : " + tm.lastEntry().getKey());

}

}

/\*

Enter the number of elements of array: 5

Enter the elements of array: 4 2 7 1 9

Largest Element of Map is : 9

\*/

Q5. Given two strings ransomNote and magazine, return true if ransomNote can be constructed

by using the letters from magazine and false otherwise.

Each letter in magazine can only be used once in ransomNote.

Input: ransomNote = "a", magazine = "b"

Output: false

Input: ransomNote = "aa", magazine = "ab"

Output: false

Input: ransomNote = "aa", magazine =”aab”

Output:true

**Solution:**

Approach:

We would store both words in two different maps one by one and update each character

Frequency.

Then we would start iterating over ransomNote map and would check that for each key in

ransomNote map, the same key should be present in magazineMap and value > ransomNote

map’s value.

If we find any element not following this condition we would return false.

If we iterate the map completely the answer is Yes.

**package** Assignment\_HashMap\_And\_Stack;

**import** java.util.\*;

**public** **class** HashMap\_Problem5 {

**public** **static** **void** main(String[] args) {

@SuppressWarnings("resource")

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the RansomNote String: ");

String s = sc.nextLine();

System.***out***.print("Enter the Magazine String: ");

String r = sc.nextLine();

HashMap<Character, Integer> hm1 = **new** HashMap<>();

**for** (**int** i = 0; i < s.length(); i++) {

**if** (hm1.containsKey(s.charAt(i))) {

hm1.put(s.charAt(i), hm1.get(s.charAt(i)) + 1);

} **else**

hm1.put(s.charAt(i), 1);

}

HashMap<Character, Integer> hm2 = **new** HashMap<>();

**for** (**int** i = 0; i < r.length(); i++) {

**if** (hm2.containsKey(r.charAt(i))) {

hm2.put(r.charAt(i), hm2.get(r.charAt(i)) + 1);

} **else**

hm2.put(r.charAt(i), 1);

}

Integer answer = -1;

**for** (Map.Entry<Character, Integer> e : hm1.entrySet()) {

**if** (e.getValue() > hm2.get(e.getKey())) {

System.***out***.println("false");

answer = 1;

**break**;

}

}

**if** (answer == -1)

System.***out***.println("True");

}

}

/\*

Enter the RansomNote String: aa

Enter the Magazine String: aab

True

\*/

/\*

Enter the RansomNote String: aa

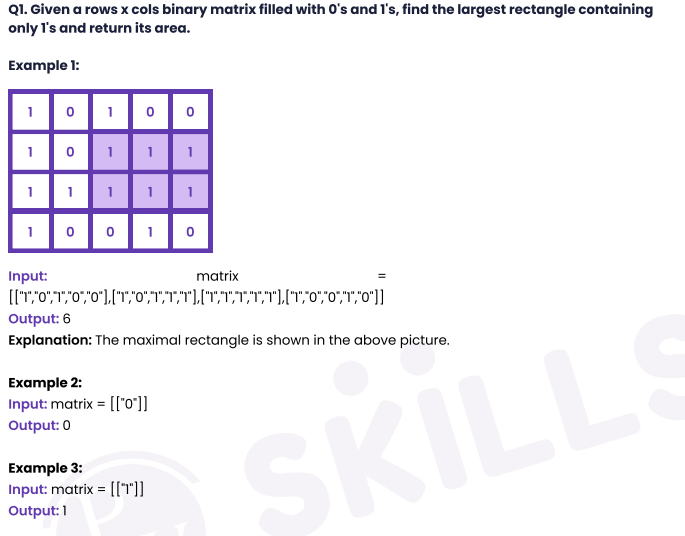
Enter the Magazine String: ab

false

\*/

**Stacks in Java**

**Assignment**



Solution:

Approach :

* + Pick one row
  + Do summation of each index till that row

1. if any index value is 0 then put 0 else previous summation + 1
   * Pass this array to get max area (Largest Rectangle in Histogram that we discussed in lecture.
   * Update max area.

Largest Rectangle in Histogram-:

* + Max area will always have at least one full bar height on any index.
  + Find the largest rectangle including each bar one by one.

1. For each bar, we have to find it's left limit & right limit (to know the maximum width)

b) Find it's left limit (where we find any index's value is smaller than current index in left side array of curr index.

c) Find it's right limit (where we find any index's value is smaller than current index in right side array of curr index.

* + Take the maximum of all the max areas found by each bar
  + Calculate area = width \* height where width = right limit - left limit + 1 and height = curr index's valu
  + Update max area & return it.