Map, HashMap LinkedHashMap, TreeMap

1. Map:

Map là container giúp lưu các phần tử theo cặp key, value (khóa - giá trị). Mỗi giá trị của key sẽ ánh xạ sang một value tương ứng. So với Set thì **Map** thậm chí còn mạnh mẽ và giải quyết được nhiều vấn đề hơn.

Tính chất:

- **Các key trong map** là những giá trị riêng biệt, không có 2 key nào có giá trị giống nhau, value thì có thể trùng nhau.
- **Map** có thể tìm kiếm nhanh.
- **Map** có thể dùng key làm index để truy cập vào value tương ứng.
- **Map** trong java chỉ lưu các phần tử là object.

1. Map:

Áp dụng map:

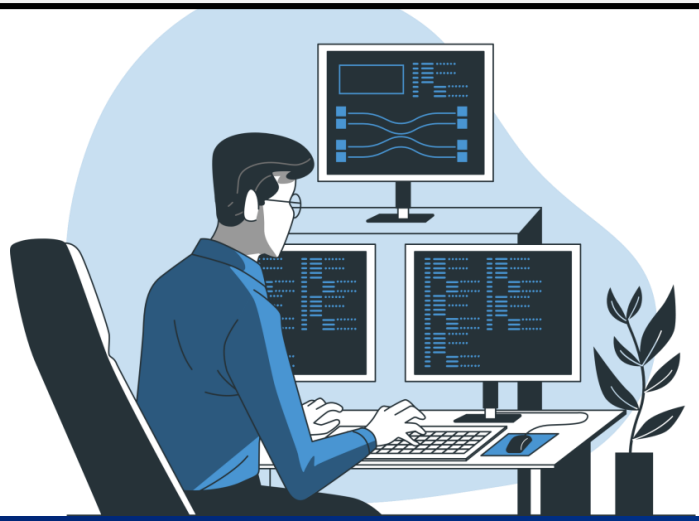
- Các bài toán liên quan tới tần suất của các phần tử.
- Các bài toán cần tìm kiếm, thêm, xóa phần tử một cách nhanh chóng.
- Dùng map thay cho các bài toán sử dụng mảng đánh dấu khi dữ liệu không đẹp.

2. HashMap:

Hash map được cài đặt bằng hash table, Hashmap không có thứ tự nào cả, các bạn cần chú ý điều đó.

Khai báo HashMap:

```
Map<keyDataType, valueDataType> map = new HashMap<>();
HashMap<keyDataType, valueDataType> map = new HashMap<>();
```



2. HashMap:

Các hàm phổ biến của Hashmap:

Hàm	Chức năng
put(key, val)	Thêm cặp key, value vào map
size()	Trả về kích thước của map
isEmpty()	Kiểm tra map rỗng
clear()	Xóa toàn bộ các phần tử trong map
containsKey(x)	Kiểm tra sự tồn tại của phần tử x trong tập key
containsValue(x)	Kiểm tra sự tồn tại của phần tử x trong tập value
get(x)	Trả về value tương ứng với key x
remove(x)	Xóa cặp phần tử có key là x khỏi map
replace(x, y)	Thay thế cặp phần tử (x, y)

2. HashMap:

Duyệt Map:

EXAMPLE

```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashMap<Integer, Integer> map = new HashMap<>();
        map.put(1, 2);
        map.put(1, 4);
        map.put(3, 4);
        map.put(2, 5);
        map.put(2, 3);
        Set<Map.Entry<Integer, Integer>> entrySet = map.entrySet();
        for(Map.Entry<Integer, Integer> entry : entrySet){
            System.out.println(entry.getKey() + " " + entry.getValue());
        }
}
```

OUTPUT

```
1 4
2 3
3 4
```

2. HashMap:

Đếm giá trị khác nhau trong map:

EXAMPLE

```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashMap<Integer, Integer> map = new HashMap<>();
        int[] a = {1, 1, 2, 1, 2, 3, 1, 2};
        for(int x : a){
            map.put(x, 1);
        }
        System.out.println(map.size());
}
```

OUTPUT

3

2. HashMap:

Kiểm tra sự tồn tại của nhiều phần tử trong tập hợp:

EXAMPLE

```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashMap<Integer, Integer> map = new HashMap<>();
        int[] a = {1, 1, 2, 1, 2, 3, 1, 2};
        for(int x : a)
            map.put(x, 1);
        int q = sc.nextInt();
        while(q-- > 0){
            int x = sc.nextInt();
            if(map.containsKey(x))
                System.out.println("FOUND");
            else
                System.out.println("NOT FOUND");
        }
}
```

OUTPUT

FOUND

2. HashMap:

Đếm tần suất của các giá trị trong mảng:

EXAMPLE

```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashMap<Integer, Integer> map = new HashMap<>();
        int[] a = {1, 1, 2, 1, 2, 3, 1, 2};
        for (int x : a) {
            if (map.containsKey(x)) {
                int tanSuat = map.get(x);
                map.put(x, tanSuat + 1);
            } else {
                map.put(x, 1);
            }
        }
        Set<Map.Entry<Integer, Integer>> entrySet = map.entrySet();
        for (Map.Entry<Integer, Integer> entry : entrySet) {
            System.out.println(entry.getKey() + " " + entry.getValue());
        }
}
```

OUTPUT

```
1 4
2 3
3 1
```

3. LinkedHashMap:

- **LinkedHash** được cài đặt bằng bảng băm kết hợp với danh sách liên kết, tương tự như map có điều khác biệt là trong **LinkedHashMap** có thứ tự, key sẽ được sắp xếp theo thứ tự bạn thêm nó vào trong map.
- Các hàm của LinkedHashMap không có gì khác so với HashMap.

3. LinkedHashMap:

Duyệt LinkedHashMap:

EXAMPLE

```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedHashMap<Integer, Integer> map = new LinkedHashMap<>();
        int[] a = {4, 1, 1, 2, 1, 2, 3, 1, 2};
        for (int x : a) {
            if (map.containsKey(x)) {
                int tanSuat = map.get(x);
                map.put(x, tanSuat + 1);
            } else {
                map.put(x, 1);
            }
        }
        Set<Map.Entry<Integer, Integer>> entrySet = map.entrySet();
        for (Map.Entry<Integer, Integer> entry : entrySet) {
            System.out.println(entry.getKey() + " " + entry.getValue());
        }
    }
```

OUTPUT
4 1
1 4
2 3
3 1

4. TreeMap:

TreeMap được cài đặt bằng cây đỏ đen, key trong **TreeMap** được sắp xếp theo thứ tự tăng dần.


```
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        TreeMap<Integer, Integer> map = new TreeMap<>();
        int[] a = {4, 1, 1, 2, 1, 2, 3, 1, 2};
        for (int x : a) {
            if (map.containsKey(x)) {
                int tanSuat = map.get(x);
                map.put(x, tanSuat + 1);
            } else {
                map.put(x, 1);
            }
        }
        Set<Map.Entry<Integer, Integer>> entrySet = map.entrySet();
        for (Map.Entry<Integer, Integer> entry : entrySet)
            System.out.println(entry.getKey() + " " + entry.getValue());
}
```

OUTPUT
1 4
2 3
3 1
4 1

4. TreeMap:

TreeMap có có thêm các hàm sau:

Hàm	Chức năng
firstKey()	Trả về key đầu tiên trong map
lastKey()	Trả về key cuối cùng trong map
firstEntry()	Trả về cặp phần tử đầu tiên trong map
lastEntry()	Trả về cặp phần tử cuối cùng trong map
floorKey(x)	Trả về key lớn nhất nhỏ hơn hoặc bằng x
floorEntry(x)	Trả về cặp phần tử có key lớn nhất nhỏ hơn hoặc bằng x
ceilingKey(x)	Trả về key nhỏ nhất lớn hơn hoặc bằng x
ceilingEntry(x)	Trả về cặp phần tử có key nhỏ nhất lơn hơn hoặc bằng x