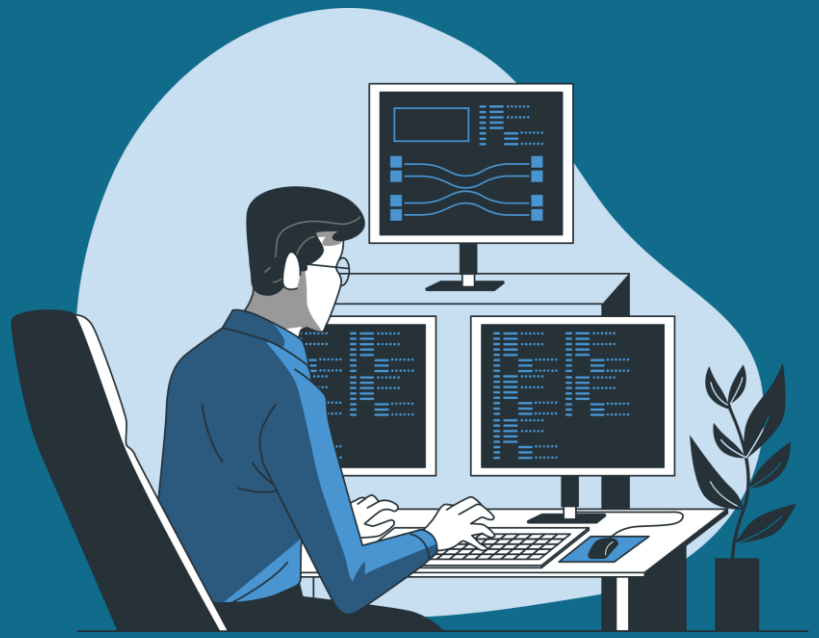


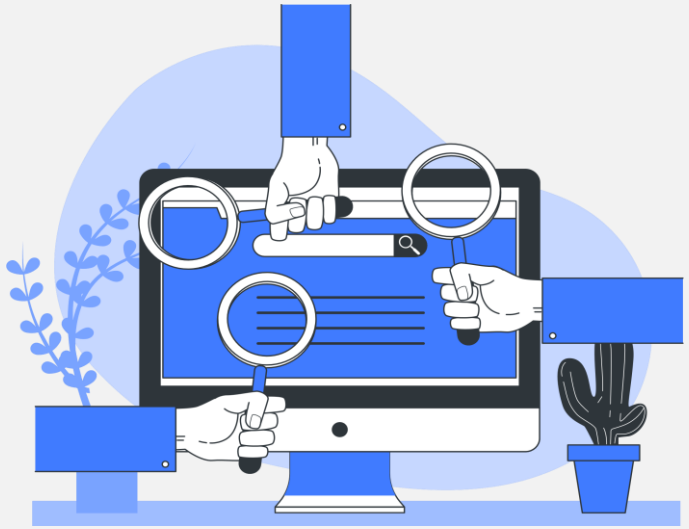
Set, HashSet LinkedHashSet, TreeSet

1. Set:

Set là một container cực kì mạnh mẽ của ngôn ngữ lập trình Java, sử dụng thành thạo **Set** là một kỹ năng cơ bản mà bạn cần đạt được. **Set** sẽ giúp code của các bạn trở nên tối ưu và ngắn gọn hơn rất nhiều.

Java cung cấp 3 lớp:

- **HashSet**
- **LinkedHashSet**
- **TreeSet**



1. Set:

Set là một container mà mỗi phần tử trong đó là duy nhất, tức là sẽ không có 2 phần tử có giá trị giống nhau tồn tại trong set.

Set có tốc độ tìm kiếm phần tử cực kì nhanh nhờ được cài đặt bằng bảng băm (Hash table).

Áp dụng set:

1. Các bài toán liên quan tới việc xóa, thêm, tìm kiếm một phần tử nào đó được thực hiện đi thực hiện lại nhiều lần.
2. Các bài toán liên quan tới các giá trị khác nhau của mảng.

2. HashSet:

Tính chất:

- **HashSet** chỉ lưu được các phần tử khác nhau.
- **HashSet** được cài đặt bằng bảng băm, vì thế tốc độ tìm kiếm phần tử đạt được là O(1).
- **HashSet** không có thứ tự, tức là các phần tử trong HashSet có thể xuất hiện nhiều thứ tự bất kì.
- **HashSet** chỉ lưu được các phần tử kiểu đối tượng.

Khai báo HashSet:

```
Set<dataType> set1 = new HashSet<>();
HashSet<dataType> set2 = new HashSet<>();
```

2. HashSet:

Một số hàm thông dụng trong HashSet:

Hàm	Chức năng
add()	Thêm phần tử vào set
remove()	Xóa phần tử khỏi set
size()	Trả về số lượng phần tử trong set
contains()	Kiểm tra sự tồn tại của 1 phần tử trong set
clear()	Xóa mọi phần tử trong set
isEmpty()	Trả về true nếu set rỗng, ngược lại trả về false

2. HashSet:

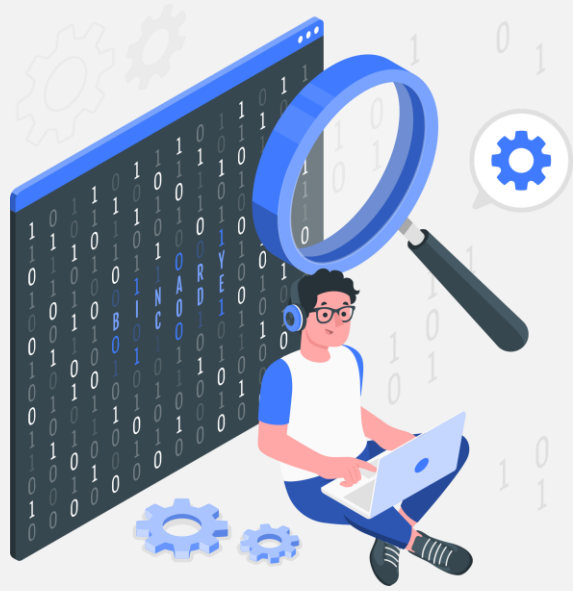
Tìm số lượng giá trị khác nhau trong mảng số nguyên:

EXAMPLE

```
public static void main(String[] args) {
        Set<Integer> set1 = new HashSet<>();
        int[] a = {1, 1, 2, 1, 3, 1, 4, 4, 2};
        for(int x : a){
            set1.add(x);
        }
        System.out.println(set1.size());
}
```

OUTPUT

4



2. HashSet:

Duyệt set:

EXAMPLE

```
public static void main(String[] args) {
        Set<Integer> set1 = new HashSet<>();
        int[] a = {1, 1, 2, 1, 3, 1, 4, 4, 2};
        for(int x : a){
            set1.add(x);
        }
        for(int x : set1){
            System.out.print(x + " ");
        }
}
```

OUTPUT

1 2 3 4

2. HashSet:

Kiểm tra sự tồn tại của 1 giá trị trong set:

EXAMPLE

```
public static void main(String[] args) {
        Set<Integer> set1 = new HashSet<>();
        int[] a = {1, 1, 2, 1, 3, 1, 4, 4, 2};
        for(int x : a){
            set1.add(x);
        }
        if(set1.contains(1)){
            System.out.println("FOUND");
        }
        else{
            System.out.println("NOT FOUND");
        }
}
```

OUTPUT

FOUND

3. LinkedHashSet:

Tính chất:

- **LinkedHashSet** chỉ lưu được các phần tử khác nhau.
- **LinkedHashSet** được cài đặt bằng bảng băm và danh sách liên kết, vì thế tốc độ tìm kiếm phần tử đạt được là O(1) .
- **LinkedHashSet** có thứ tự là thứ tự bạn thêm các phần tử vào set.
- **LinkedHashSet** chỉ lưu được các phần tử kiểu đối tượng.

Khai báo LinkedHashSet:

```
Set<dataType> set1 = new LinkedHashSet<>();
HashSet<dataType> set2 = new LinkedHashSet<>();
```

3. LinkedHashSet:

Các hàm và ví dụ của **LinkedHashSet** tương tự như **HashSet**, trừ việc duyệt các phần tử trong **LinkedHashSet** có thứ tự.


```
public static void main(String[] args) {
        HashSet<Integer> set = new LinkedHashSet<>();
        int[] a = {1, 4, 4, 4, 2, 1, 3, 3};
        for(int x : a){
            set.add(x);
        }
        for(int x : set){
            System.out.print(x + " ");
        }
    }
```

OUTPUT

1 4 2 3

3. TreeSet:

Tính chất:

- **TreeSet** chỉ lưu được các phần tử khác nhau.
- **TreeSet** được cài đặt bằng cây đỏ đen - (Red-black tree), các hàm tìm kiếm, xóa của TreeSet là LogN.
- **TreeSet** có thứ tự là thứ tự tăng dần.
- **TreeSet** chỉ lưu được các phần tử kiểu đối tượng.

Khai báo TreeSet:

```
TreeSet<dataType> set2 = new TreeSet<>();
```

3. LinkedHashSet:

TreeSet có thự tự:

EXAMPLE

```
public static void main(String[] args) {
        TreeSet<Integer> set = new TreeSet<>();
        int[] a = {1, 4, 4, 4, 2, 1, 3, 3};
        for(int x : a){
            set.add(x);
        }
        for(int x : set){
            System.out.print(x + " ");
        }
}
```

OUTPUT

1 2 3 4

3. LinkedHashSet:

Một số hàm thông dụng trong HashSet:

Hàm	Chức năng
floor(x)	Trả về phần tử lớn nhất <= x hoặc trả về null nếu không tồn tại
ceiling(x)	Trả về phần tử nhỏ nhất >=x hoặc trả về null nếu không tồn tại
first()	Trả về phần tử đầu tiên trong set
last()	Trả về phần tử cuối cùng trong set
lower(x)	Trả về phần tử lớn nhất nhỏ hơn x hoặc trả về null nếu không tồn tại
higher(x)	Trả về phần tử nhỏ nhất lớn hơn x hoặc trả về null nếu không tồn tại