**Collections trong Java**

Collections giúp chúng ta lưu trữ và thao tác tới các nhóm đối tượng, ta có thể tìm kiếm phân loại them sửa xóa trong các nhóm đối tượng đó

1, interface collection

**Set:**là một collection không thể chứa 2 giá trị trùng lặp. Set được sử dụng để biểu diễn các bộ

**List:**là một collection có thứ tự (đôi khi còn được gọi là một chuỗi). List có thể chứa các phần tử trùng lặp.

**Map:**là một đối tượng ánh xạ mỗi key tương úng với một giá trị. Map không thể chứa giá trị trùng lặp. Mỗi key có thể ánh xạ đến nhiều nhất một giá trị.

Các phương thức

public boolean add(Object element)

public boolean remove(Object element)

…

Ví dụ về collection

* **Array List implement từ List**

List<String> arrayList = new ArrayList<String>();

        arrayList.add("PHP");

        arrayList.add("Java");

        arrayList.add("Java");

        arrayList.add("C++");

        System.out.println("Các phần tử của ArrayList");

System.out.print("\t" + arrayList + "\n");

* **Hash Set implement từ Set**

Set<String> arrayList = new HashSet<String>();

        arrayList.add("PHP");

        arrayList.add("Python");

        arrayList.add("Java");

        arrayList.add("C++");

        System.out.println("Các phần tử của ArrayList");

System.out.print("\t" + arrayList + "\n");

* **Hash Map implement từ Map**
* Comparable và Comparator

Comparable và Comparator cả hai đều là các giao diện và có thể được sử dụng để sắp xếp các phần tử của collection

Comparable phải implement interface còn compatator thì ko

So sánh tên :

 List<Student> listStudents = new ArrayList<Student>();

        // add students to list

        listStudents.add(new Student(1, "Vinh", 19, "Hanoi"));

        listStudents.add(new Student(2, "Hoa", 24, "Hanoi"));

        listStudents.add(new Student(3, "Phu", 20, "Hanoi"));

        listStudents.add(new Student(4, "Quy", 22, "Hanoi"));

        // sort list student by it's name ASC

        System.out.println("sort list student by it's name ASC: ");

        Collections.sort(listStudents, new Comparator<Student>() {

            @Override

            public int compare(Student o1, Student o2) {

                return o1.getName().compareTo(o2.getName());

            }

        });

So sánh tuổi

Collections.sort(listStudents, new Comparator<Student>() {

            @Override

            public int compare(Student o1, Student o2) {

                return o1.getAge() > o2.getAge() ? 1 : -1;

            }

        });

1454