Buổi 04
Bộ sưu tập

(http://docs.oracle.com/javase/tutorial/collections/index.html)

Mục tiêu

- Khung Bộ sưu tập (gói java.util):
  - Danh sách: ArrayList, Vector → Các bản sao được đồng ý
  - Bộ: HashSet, TreeSet → Các bản sao không được đồng ý
  - Bản đồ: HashMap, TreeMap
  - Hàng đợi: LinkedList, PriorityQueue
  - Deque: LinkedList, ArrayDeque

Khung Bộ sưu tập

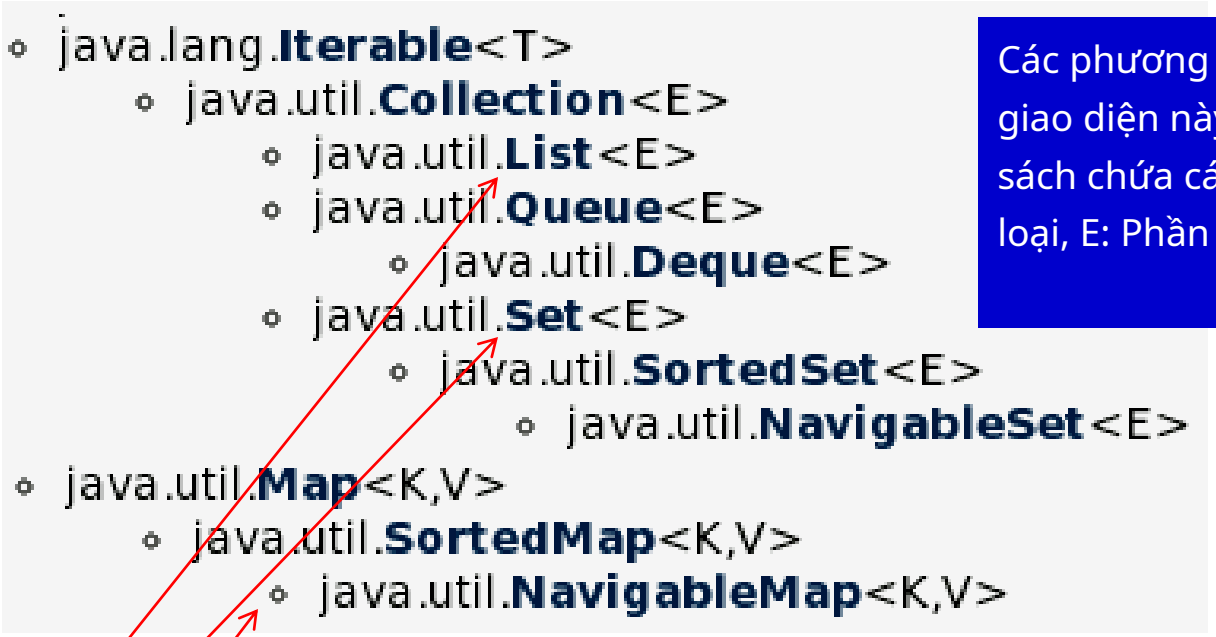
- Nền tảng Java 2 bao gồm một*khung bộ sưu tập*.
- MỘT*thu thập*là đối tượng đại diện cho một nhóm đối tượng.
- Khung Bộ sưu tập là một kiến trúc thống nhất để đại diện và thao tác các bộ sưu tập.
- Khuôn khổ bộ sưu tập nói chung không phải là chuỗi an toàn.

Khung Bộ sưu tập...

- **Giảm nỗ lực lập trình**bằng cách cung cấp các cấu trúc dữ liệu và thuật toán hữu ích để bạn không phải tự viết chúng.
- **Tăng hiệu suất**bằng cách cung cấp các triển khai hiệu suất cao của các cấu trúc dữ liệu và thuật toán hữu ích.

- **Cung cấp khả năng tương tác giữa các API không liên quan** bằng cách thiết lập một ngôn ngữ chung để chuyển các bộ sưu tập qua lại.
- **Giảm nỗ lực cần thiết để tìm hiểu các API**bằng cách loại bỏ nhu cầu tìm hiểu nhiều API thu thập đặc biệt.
- **Giảm nỗ lực cần thiết để thiết kế và triển khai các API** bằng cách loại bỏ sự cần thiết phải tạo ra các API bộ sưu tập đặc biệt.

Giao diện bộ sưu tập



Các phương thức được khai báo trong các giao diện này có thể hoạt động trên danh sách chứa các phần tử thuộc kiểu tùy ý. T: loại, E: Phần tử, K: Khóa, V: Giá trị

Chi tiết về điều này sẽ được giới thiệu trong chủ đề Chung

3 loại nhóm:

Danh sách có thể chứa các yếu tố trùng lặp **Đặt** chỉ có thể chứa các phần tử riêng biệt

Bản đồ có thể chứa các cặp <khóa, giá trị>. Chìa khóa của yếu tố là dữ liệu để tìm kiếm nhanh

Hàng đợi, Deque chứa các phương pháp của danh sách hạn chế.

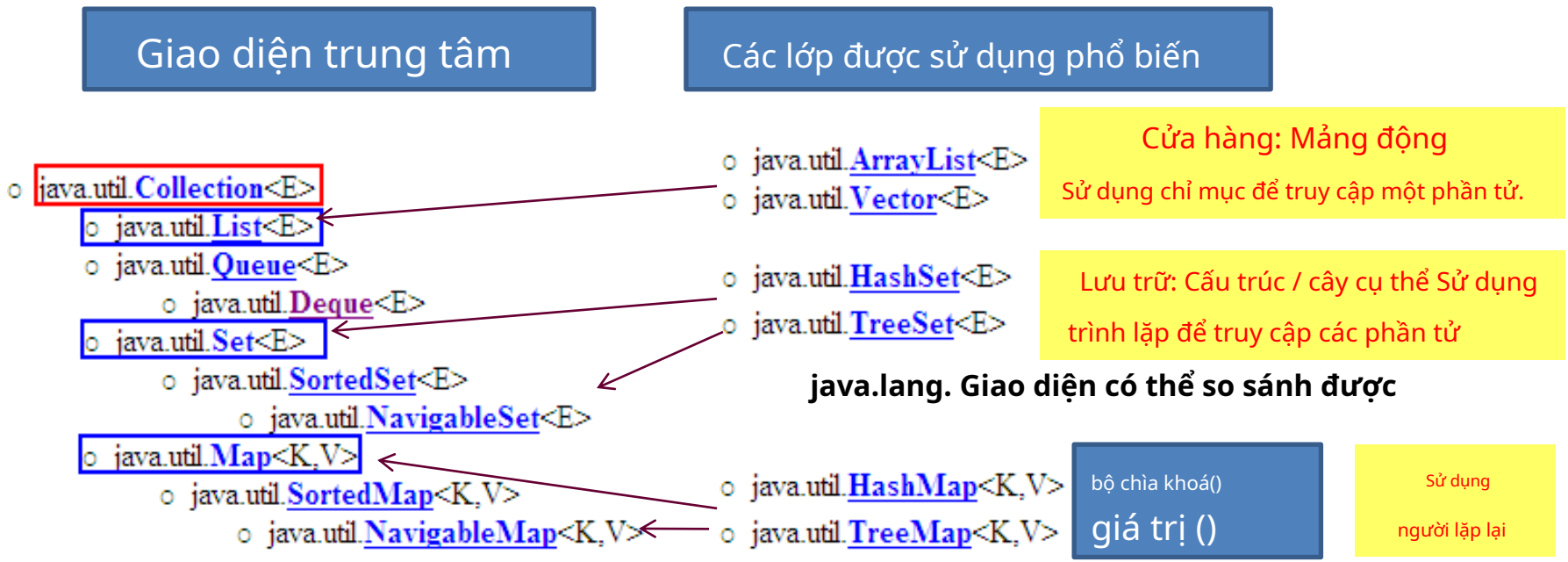
Các phương pháp phổ biến trên nhóm là: Thêm, Xóa, Tìm kiếm, Xóa,...

Các phương thức chung của giao diện Bộ sưu tập

Method	Description
`add(Object x)`	Adds x to this collection
`addAll(Collection c)`	Adds every element of c to this collection
`clear()`	Removes every element from this collection
`contains(Object x)`	Returns true if this collection contains x
`containsAll(Collection c)`	Returns true if this collection contains every element of c
`isEmpty()`	Returns true if this collection contains no elements
`iterator()`	Returns an Iterator over this collection (see below)
`remove(Object x)`	Removes x from this collection
`removeAll(Collection c)`	Removes every element in c from this collection
`retainAll(Collection c)`	Removes from this collection every element that is not in c
`size()`	Returns the number of elements in this collection
`toArray()`	Returns an array containing the elements in this collection

Các yếu tố có thể là được lưu trữ bằng cách sử dụng một số các cách như mảng, cây, bảng băm. Đôi khi, chúng tôi muốn duyệt các phần tử dưới dạng danh sách→Chúng tôi cần một danh sách các tài liệu tham khảo→Trình lặp lại

Khung Bộ sưu tập...



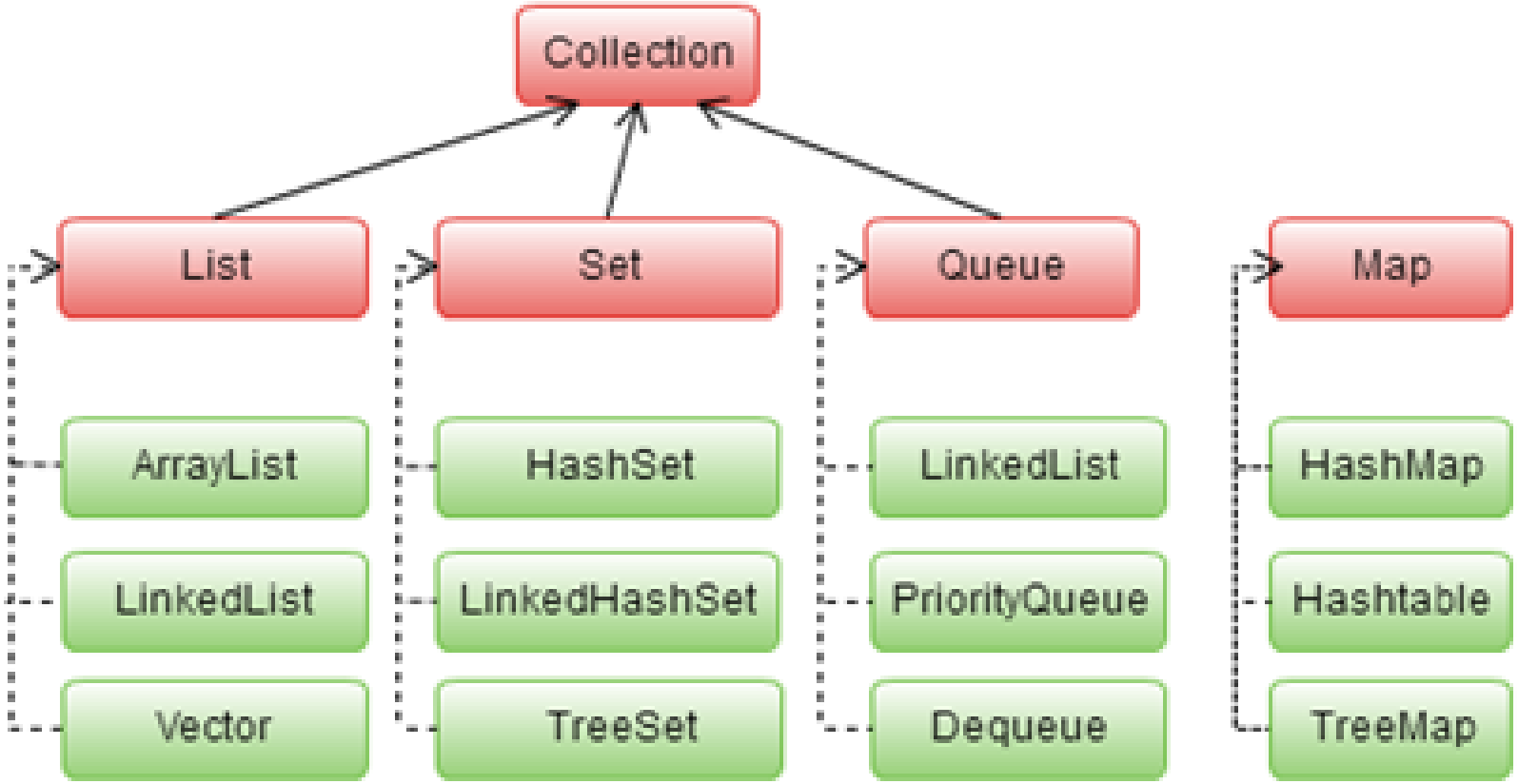
Một TreeSet sẽ lưu trữ các phần tử theo thứ tự tăng dần. Thứ tự tự nhiên được áp dụng cho số và thứ tự từ vựng (từ điển) được áp dụng cho chuỗi.

Nếu bạn muốn một TreeSet chứa các đối tượng của riêng mình, bạn phải triển khai phương thức CompareTo (Đối tượng), được khai báo trong giao diện So sánh.

Mảng so với Bộ sưu tập

Mảng	Bộ sưu tập
1. Kích thước được cố định	1. Kích thước không cố định (kích thước động), kích thước tăng trưởng được.
2. đối với mảng bộ nhớ không tốt để sử dụng, nhưng để hoạt động tốt hơn sử dụng mảng	2. bộ nhớ Bộ sưu tập tốt hơn để sử dụng, nhưng bộ sưu tập hiệu suất thì không tốt để sử dụng.
3. Có thể chứa cả các kiểu nguyên thủy (byte, sort, int, long... vv) và các kiểu đối tượng.	3. Chỉ có thể chứa các loại đối tượng
4. Không có cấu trúc dữ liệu cơ bản trong mảng. Bản thân mảng được sử dụng làm cấu trúc dữ liệu trong java.	4. Mỗi lớp Collection có cấu trúc dữ liệu cơ bản.
5. Không có phương thức tiện ích nào trong mảng	5. Mỗi Bộ sưu tập đều cung cấp các phương thức tiện ích (sắp xếp, tìm kiếm, truy xuất, v.v.). Nó sẽ làm giảm thời gian viết mã.

Giao diện bộ sưu tập



Danh sách

- Danh sách giữ cho nó các phần tử trongđơn hàng trong đó chúng đã được thêm vào.
- Mỗi phần tử của Danh sách có một chỉ mục, bắt đầu từ 0.
- Các phương pháp phổ biến:
  - **void add (int index, Object x)**
  - **Đối tượng get (int index)**
  - **int indexOf (Đối tượng x)**
  - **Loại bỏ đối tượng (int index)**

Các lớp Triển khai danh sách giao diện

- AbstractList
- Lập danh sách
- Vector (giống như ArrayList nhưng nó làđồng bộ)
- Danh sách liên kết:*danh sách được liên kết có thể được sử dụng như một ngăn xếp, hàng đợi hoặc hàng đợi kết thúc kép (deque)*

ArrayList
Lập danh sách

- ArrayList là một lớp thực hiện giao diện Danh sách.
- Ưu điểm của ArrayList so với Array nói chung là ArrayList là động và kích thước của ArrayList có thể lớn lên hoặc thu nhỏ lại.
- ArrayList lưu trữ các phần tử theo thứ tự chèn.
- ArrayList có thể có các phần tử trùng lặp. ArrayList không được đồng bộ hóa.

ArrayList
Lập danh sách

Sr.No.	Constructor & Description
1	**ArrayList()** This constructor is used to create an empty list with an initial capacity sufficient to hold 10 elements.
2	**ArrayList(Collection c)** This constructor is used to create a list containing the elements of the specified collection.
3	**ArrayList(int initialCapacity)** This constructor is used to create an empty list with an initial capacity.

Phương pháp

Phương pháp	Sự miêu tả
void add (int index, Phần tử đối tượng)	Chèn phần tử được chỉ định vào vị trí được chỉ định trong danh sách này.
boolean thêm (Đối tượng o)	Thêm phần tử được chỉ định vào cuối danh sách này.
boolean addAll (Bộ sưu tập C)	Nối tất cả các phần tử trong bộ sưu tập được chỉ định vào cuối danh sách này, theo thứ tự chúng được trả về bởi Iterator của bộ sưu tập được chỉ định
void clear ()	Xóa tất cả các phần tử khỏi danh sách này.
boolean chứa (Đối tượng o)	trả về true nếu danh sách này chứa phần tử được chỉ định.
Đối tượng get (int mục lục)	trả về phần tử ở vị trí được chỉ định trong danh sách này.

Phương pháp	Sự miêu tả
khoảng trống được bảo vệ **removeRange (int fromIndex, int Chỉ mục)**	loại bỏ khỏi danh sách này tất cả các phần tử có chỉ mục nằm giữa fromIndex (bao gồm) và toIndex (độc quyền).
Tập đối tượng (int index, Phần tử đối tượng)	thay thế phần tử ở vị trí đã chỉ định trong danh sách này bằng phần tử được chỉ định.
int size ()	trả về số phần tử trong danh sách này.
int indexOf (Đối tượng o) **int lastIndexOf (Đối tượng o)**	trả về chỉ số của lần xuất hiện đầu tiên (cuối cùng) của phần tử được chỉ định trong danh sách này hoặc -1 nếu danh sách này không chứa phần tử.
Đối tượng loại bỏ (int mục lục)	loại bỏ phần tử ở vị trí được chỉ định trong danh sách này.
boolean loại bỏ (Đối tượng o)	loại bỏ sự xuất hiện đầu tiên của phần tử được chỉ định khỏi danh sách này, nếu nó có mặt.

Ví dụ

```
public class ArrayListDemo {
    public static void main (String args []) {

        ArrayList al = new ArrayList ();
        System.out.println (“Kích thước lúc đầu:"
    + al.size ());
        // thêm các phần tử
        al.add ("C");                    al.add ("A");
        al.add ("E");                    al.add ("B");
        al.add ("D");                    al.add ("F");
        al.add (1, "A2");
```

```
System.out.println (“Kích thước sau:"
    + al.size ());
     System.out.println (“Nội dung:" + al);

     al.remove ("F"); al.remove (2);
     System.out.println (“Kích thước sau khi xóa:"
    + al.size ());
     System.out.println (“Nội dung:" + al);

}
}
```

Đầu ra

Kích thước lúc đầu: 0 Kích thước

sau: 7

Nội dung: [C, A2, A, E, B, D, F] Kích thước

sau khi xóa: 5 Nội dung: [C, A2, E, B, D]

Chung trong java

- Các**Java Generics**lập trình được giới thiệu trong Java SE 5 để đối phó với các đối tượng an toàn kiểu.
- Generics, buộc lập trình viên java phải lưu trữ loại đối tượng cụ thể.
- Chủ yếu có 3 ưu điểm của thuốc generic:
  - **Loại-an toàn:**Chúng tôi chỉ có thể giữ một loại đối tượng duy nhất trong generic. Nó không cho phép lưu trữ các đối tượng khác.
  - **Loại đúc không bắt buộc:**Không cần phải đánh máy đối tượng.

chúng ta cần nhập cast.

```
Danh sách list = new ArrayList ();
list.add ("xin chào");
String s = (String) list.get (0); // typecasting
```

chúng ta không cần đánh máy đối tượng.

List <Chuỗi> list = new ArrayList <Chuỗi> (); list.add ("xin chào");

Chuỗi s = list.get (0);

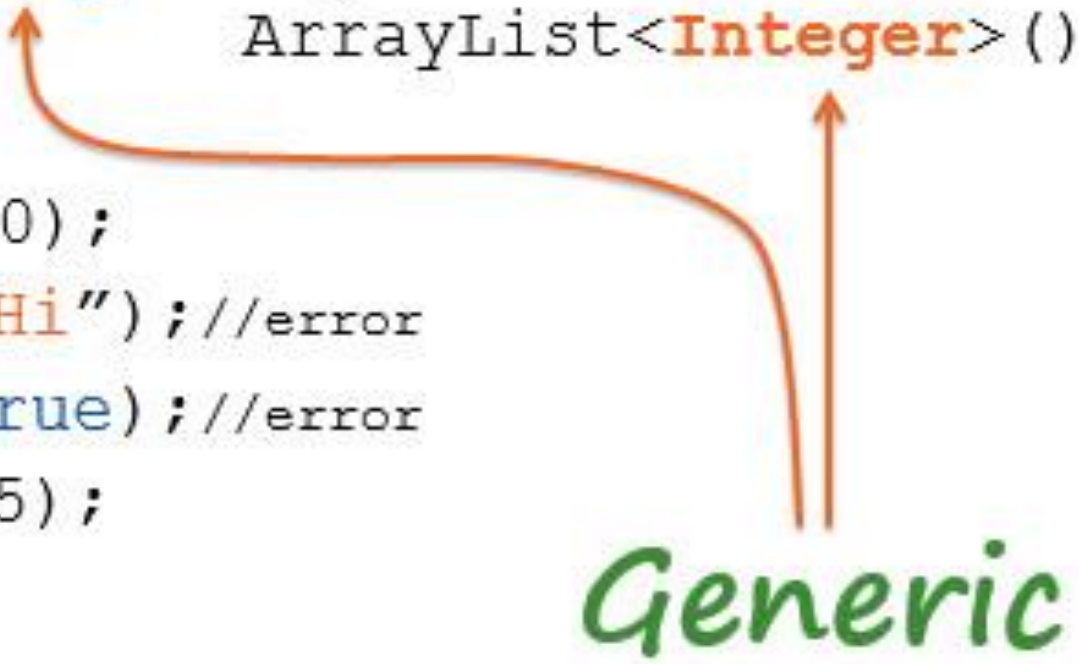
- **Kiểm tra thời gian biên dịch:**Nó được kiểm tra tại thời điểm biên dịch nên sự cố sẽ không xảy ra trong thời gian chạy.

List <Chuỗi> list = new ArrayList <Chuỗi> (); list.add ("xin chào");
list.add (32); // Lỗi thời gian biên dịch

Ví dụ

```
ArrayList<Integer> mylist = new
                        ArrayList<Integer>();

mylist.add(10);
mylist.add("Hi");//error
mylist.add(true);//error
mylist.add(15);
```



Các lớp triển khai ArrayList

```
 ArrayList al = new ArrayList (); for (int i = 101;
i <= 110; i ++) {
                    al.cộng(tôi);
}
 for (int i = 0; i <al.kích cỡ(); i ++)
{System.out.println (al.get (i));
}
// hoặc sử dụngTrình lặp lại
          Iterator iter = al.iterator ();
          while (iter.hasNext ())
{System.out.println (iter.next ());
          }
```

Ví dụ: ArrayList of Car, Motor, Truck

giai cấp công cộng**ListVehicle**{

Danh sách ArrayList <Vehicle>;

công cộng**ListVehicle**() {

list = new ArrayList <Vehicle> ();

}

phương tiện cá nhân**đầu vào**() {

Phương tiện v;

Nhà sản xuất dây;

int năm; chi phí gấp đôi;

Màu chuỗi;

Máy quét trong = Máy quét mới (System.in);

```
        trong khi (đúng) {
System.out.println ("nhà sản xuất (Mazda201
    7): ");
                nhà sản xuất = in.nextLine ();
if (productionr.matches ("^ [A-Za-
    z] \\ d {4} $ "))
                        nghỉ; }
```

```
trong khi (đúng) {
System.out.println ("nhà sản xuất (Mazda2017):
    ");
        nhà sản xuất = in.nextLine (); if (productionr.matches
("^ [A-Za-z] \\ d {4} $"))
                    nghỉ; }
System.out.print ("Năm sản xuất:");
    năm = Integer.parseInt (in.nextLine ());
    System.out.print ("Chi phí:");
    cost = Double.parseDouble (in.nextLine ());
     System.out.print ("Màu:");
     color = in.nextLine ();
     v = Xe mới (nhà sản xuất, năm, chi phí, màu sắc);

                trả lại v; }
```

```
boolean công khaiinputCar() {
    String typeofEngine;
    ghế int;
    Xe v = input ();
    Máy quét trong = Máy quét mới (System.in);
    System.out.print ("Loại động cơ:"); typeofEngine
    = in.nextLine ();
    System.out.print ("Số ghế:"); ghế = in.nextInt ();

    danh sách trả lại.cộng(Xe mới (v.getManosystem
   (), v.getYear (), v.getCost (), v.getColor (),
   typeofEngine, chỗ ngồi));

        }
```

```
boolean công khaiinputMotor() {
        năng lượng kép;
        Xe v = input ();
        Máy quét trong = Máy quét mới (System.in);
        System.out.print ("Nguồn:");
        power = in.nextDouble ();
        trở về
danh sách.cộng(Mới
    Động cơ (v.getManosystemurer (), v.getYear (),
    v.getCost (), v.getColor (), power));
          }
```

```
boolean công khaiinputTruck() {
        tải kép;
        Xe v = input ();
        Máy quét trong = mới
    Máy quét (System.in);
        System.out.print ("Đang tải:"); load
        = in.nextDouble ();
        return list.add (mới
    Xe tải (v.getManosystemurer (), v.getYea r (),
    v.getCost (), v.getColor (), tải));

        }
```

```
khoảng trống công cộngđầu ra() {
    System.out.println ("Danh sách Xe");

System.out.println ("Loại Màu Giá Năm Của
Nhà Sản XuấtofEngine NumberofSeats");

    for (Xe v: danh sách) {
        if (v instanceof Car)
System.out.println (v.toString ());
            }
    System.out.println ("----------
- - - - - - - - ");
```

```
System.out.println ("Danh sách Động cơ");
    System.out.println ("Công suất Màu Chi
    phí Năm của Nhà sản xuất");
        for (Xe v: danh sách) {
            if (v instanceof Motor)
    System.out.println (v.toString ());
                }
        System.out.println ("----------
    - - - - - - - - - ");
```

```
System.out.println ("Danh sách Xe tải");
    System.out.println ("Tải màu chi phí theo năm của nhà sản xuất");
            for (Xe v: danh sách) {
                if (v instanceof Truck)
System.out.println (v.toString ());
            }
    }
```

```
khoảng trống công cộng getByMan Producurer(Sợi dây
    nhà chế tạo){
        System.out.println ("===== Danh sách Xe
        ==========");
        System.out.println ("Màu Chi phí Năm Sản xuất");

        vì (Xe v: danh sách) if (v.getMan bệ sản xuất ().
    bằng (nhà sản xuất urer))

    System.out.println (v.toString ());
    System.out.println
    ("==================================================");
        }
```

```
khoảng trống công cộng getByMan ProducurerA(Sợi dây
    nhà chế tạo){
                System.out.println ("===== Danh
    sách Xe ========="); System.out.println
    ("Màu Chi phí Năm Sản xuất");

                for (Xe v: danh sách) if
    (v.getManosystemurer (). indexOf (manuf
    acturer)> = 0)
    System.out.println (v.toString ());
    System.out.println
    ("================================");
        }
```

```
khoảng trống công cộng getByCostGreater(gấp đôi
    Giá cả){
        System.out.println ("Danh sách Xe");

        System.out.println ("Màu Chi phí Sản          Năm
        xuất");
        cho (Xe v: danh sách)
            if (v.getCost ()> = cost)
    System.out.println (v.toString ());
    System.out.println ("================");
         }
```

khoảng trống công cộng**getByCostFrom**(nhân đôi từ,
nhân đôi thành) {

```
System.out.println ("Danh sách Xe");

System.out.println ("Màu Chi phí Năm Sản xuất");

cho (Xe v: danh sách)
    if ((v.getCost ()> = from) &&
(v.getCost () <= to))
System.out.println (v.toString ());
System.out.println ("================");
    }
```

Lớp Bộ sưu tập

- Một lớp hỗ trợ có chứa các phương thức tĩnh chấp nhậnbộ sưu tập dưới dạng tham số của chúng.

Demo bộ sưu tập

```
import java.util.ArrayList;
import java.util.Vector;
import java.util.Collections;
import java.util.Random;
public class CollectionsDemo {
    public static void main(String[] args){
        ArrayList ar= new ArrayList();
        Vector v = new Vector();
        Random rd= new Random();
        for (int i=1; i<=10; i++){
            ar.add(rd.nextInt(30));
            v.add(rd.nextInt(30));
        }
        System.out.println("ar=" + ar);
        System.out.println("v=" + v);
        boolean dis= Collections.disjoint(ar, v);
        System.out.println("ar and v is disjunct: " + dis);
        Collections.addAll(v, ar.toArray());
        System.out.println("After adding, v=" + v);
        int minVal= (int)Collections.min(v);
        int maxVal= (int) Collections.max(v);
```

```
            System.out.println("min= " + minVal + ", max= "+ maxVal);
            int fre= Collections.frequency(v, 8);
            System.out.println("Occurences of 8: " + fre);
            Collections.sort(v);
            System.out.println("After sorting, v=" + v);
            int pos = Collections.binarySearch(v, 8);
            System.out.println("Position of 8: " + pos);
            Collections.shuffle(v);
            System.out.println("After shuffling, v=" + v);

        }
    }
```

```
run:
ar=[16, 22, 13, 29, 12, 8, 23, 8, 17, 10]
v=[3, 2, 24, 13, 24, 18, 22, 8, 3, 1]
ar and v is disjunct: false
After adding, v=[3, 2, 24, 13, 24, 18, 22, 8, 3, 1, 16, 22, 13, 29, 12, 8, 23, 8, 17, 10]
min= 1, max= 29
Occurences of 8: 3
After sorting, v=[1, 2, 3, 3, 8, 8, 8, 10, 12, 13, 13, 16, 17, 18, 22, 22, 23, 24, 24, 29]
Position of 8: 4
After shuffling, v=[3, 3, 17, 8, 23, 8, 12, 24, 13, 18, 2, 24, 1, 29, 22, 16, 22, 13, 10, 8]
```

Sắp xếp với So sánh và So sánh

trong Java

Comparable	Comparator
1) Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides **multiple sorting sequences**. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.
2) Comparable **affects the original class**, i.e., the actual class is modified.	Comparator **doesn't affect the original class**, i.e., the actual class is not modified.
3) Comparable provides **compareTo() method** to sort elements.	Comparator provides **compare() method** to sort elements.
4) Comparable is present in **java.lang** package.	A Comparator is present in the **java.util** package.
5) We can sort the list elements of Comparable type by **Collections.sort(List)** method.	We can sort the list elements of Comparator type by **Collections.sort(List, Comparator)** method.

Ví dụ có thể so sánh trong Java

```
lớpHọc sinhdụng cụCó thể so sánh được <Stude nt>
{.............. ..
int công cộngCompareTo (Sinh viên st) { nếu
(age == st.age)
trở về0;
khác nếu(age> st.age)
trở về1;
khác
trở về-1;
}
}
```

Sắp xếp

```
khoảng trống tĩnh công cộngmain (Chuỗi args []) {
ArrayList <Học sinh>        al =MớiArrayList <Stu
vết lõm> ();
al.add (Mới      Học sinh(101,"Vijay",23)); Học
al.add (Mới      sinh(106,"Ajay",27)); Học sinh(105
al.add (Mới      ,"Jai",21));
Collections.sort (al);
}
```

Giao diện so sánh

lớpNameComparator**dụng cụ**Bộ so sánh <Student> { **int công cộng**so sánh (Sinh viên s1, Sinh viên s2) { **trở về** s1.name.compareTo (s2.name); }}

lớpAgeComparator**dụng cụ**Bộ so sánh <Student> { **int công cộng**so sánh (Sinh viên s1, Sinh viên s2) { **nếu**(s1.age == s2.age)
trở về0;
khác nếu(s1.age> s2.age) **trở về**1;
khác
trở về-1;
}}

Giao diện so sánh - Demo

khoảng trống tĩnh công cộngmain (String args []) {ArrayList

<Student> al =**Mới**ArrayList <Học viên> ();

al.add (**Mới**Học sinh(101,"Vijay",23)); al.add (**Mới**Học sinh(106, "Ajay",27)); al.add (**Mới**Học sinh(105,"Jai",21));

System.out.println ("Sắp xếp theo tên");

Collections.sort (al,**Mới**NameComparator ()); **vì**

(Sinh viên st: al) {

System.out.println (st.rollno +""+ st.name +""+ st.age); }

System.out.println ("sắp xếp theo độ tuổi");

Collections.sort (al,**Mới**Bộ so sánh tuổi ()); **vì**

(Sinh viên st: al) {

System.out.println (st.rollno +""+ st.name +""+

st.age); }}

Thao tác dữ liệu thường xuyên

- Lớp Collections cung cấp năm thuật toán để thực hiện thao tác dữ liệu thông thường trên các đối tượng Danh sách, bao gồm:
  - đảo ngược()
  - lấp đầy()
  - copy ()
  - tráo đổi()
  - addAll ()

Đang tìm kiếm

- Điều kiện: Danh sách theo thứ tự tăng dần
- Thuật toán tìm kiếm nhị phân tìm kiếm một phần tử được chỉ định trong Danh sách được sắp xếp.
  - Trả về vị trí> = 0→Hiện nay
  - Trả về vị trí <0→Vắng mặt

Thành phần

- tần số -đếm số lần phần tử được chỉ định xuất hiện trong tập hợp được chỉ định.

- rời rạc -xác định xem hai Bộ sưu tập có rời rạc hay không; nghĩa là chúng không chứa phần tử nào chung.

```
public void sortByManosystemurer () {
    Collections.sort(danh sách, mới
Bộ so sánh <Vehicle> (){
    @Ghi đè
    public int so sánh (Xe v1, Xe v2) {

        trở về
((v1.getManosystemurer (). CompareToIgno
reCase (v2.getManosystemurer ())));
                    }
            });
    }
```

```
public void sortByCost () {
    Collections.sort(danh sách, Bộ so sánh mới <Vehicle> (){
    public int so sánh (Xe v1, Xe v2) {

        return (int) v1.getCost () - (int) v2.getCost ();
                    }
    });
    }
```

```
public static void main(String[] args) {
    ListVehicle a=new ListVehicle();
    Scanner in=new Scanner(System.in);
    while(true){
        System.out.print("\n 1. input a Car");
        System.out.print("\n 2. input a Motor");
        System.out.print("\n 3. input a Truck");
        System.out.print("\n 4. output a list of Vehicles");
        System.out.print("\n 5. Search by manufacturer");
        System.out.print("\n 6. Search by manufacturer ()");
        System.out.print("\n 7. Search by cost (greater than)");
        System.out.print("\n 8. Search by cost (from to)");
        System.out.print("\n 9. Sort by manufacturer");
        System.out.print("\n 10. Sort by type of engine");
        System.out.print("\n 0. Exit");
        System.out.print("\n Your choice(0->10): ");
        int choice;
```

Ví dụ: Tài liệu, Sách, Tạp chí, Báo

```
public static void main(String[] args) {
    ListDocument d=new ListDocument();
    Scanner in=new Scanner(System.in);
    while(true){
        System.out.print("\n 1. input a Book");
        System.out.print("\n 2. input a Magazine");
        System.out.print("\n 3. input a Nespaper");
        System.out.print("\n 4. output a list of Documents");
        System.out.print("\n 5. Delete a Document");
        System.out.print("\n 6. Edit a Document");

        System.out.print("\n 0. Exit");
        System.out.print("\n Your choice(0->6): ");
        int choice;
        choice=in.nextInt();
```

Chuyển đổi ArrayList sang Array

- **public Object [] toArray ()**

```
Danh sách<String> list = newLập danh sách();
        // thêm lượt 4 phần tử. list.add (“Tạp
        chí "); list.add ("Khoa học");

        list.add ("Và công nghệ"); list.add
        ("về Thông tin và Truyền thông ”);

        // chuyển đổi listString thành mảng. Sợi
dây[]mảng = danh sách.toArray(Mới
    Sợi dây[list.size ()]);
        System.out.println ("\ n
    "+ Arrays.toString (mảng));
```

Chuyển đổi Array thành ArrayList

- Arrays.asList (T... a)

Danh sách<String> listNames =
Mảng.**asList**("John", "Peter", "Tom",
"Mary");
Danh sách<Integer> listNumbers =
Mảng.**asList**(1, 3, 5, 7, 9, 2, 4, 6, 8);

System.out.println (listName);
System.out.println (listNumbers);

Lớp vectơ

- Vector thực hiện một mảng động. Nó tương tự như ArrayList, nhưng có hai điểm khác biệt
  - Vector được đồng bộ hóa.
  - Vector chứa nhiều phương thức kế thừa không thuộc khuôn khổ tập hợp.
- Người xây dựng:
  - **Vectơ ():**Hàm tạo này tạo một vectơ mặc định, có dung lượng ban đầu là 10.
  - **Vectơ (kích thước int):**Hàm tạo này chấp nhận một đối số bằng với kích thước được yêu cầu