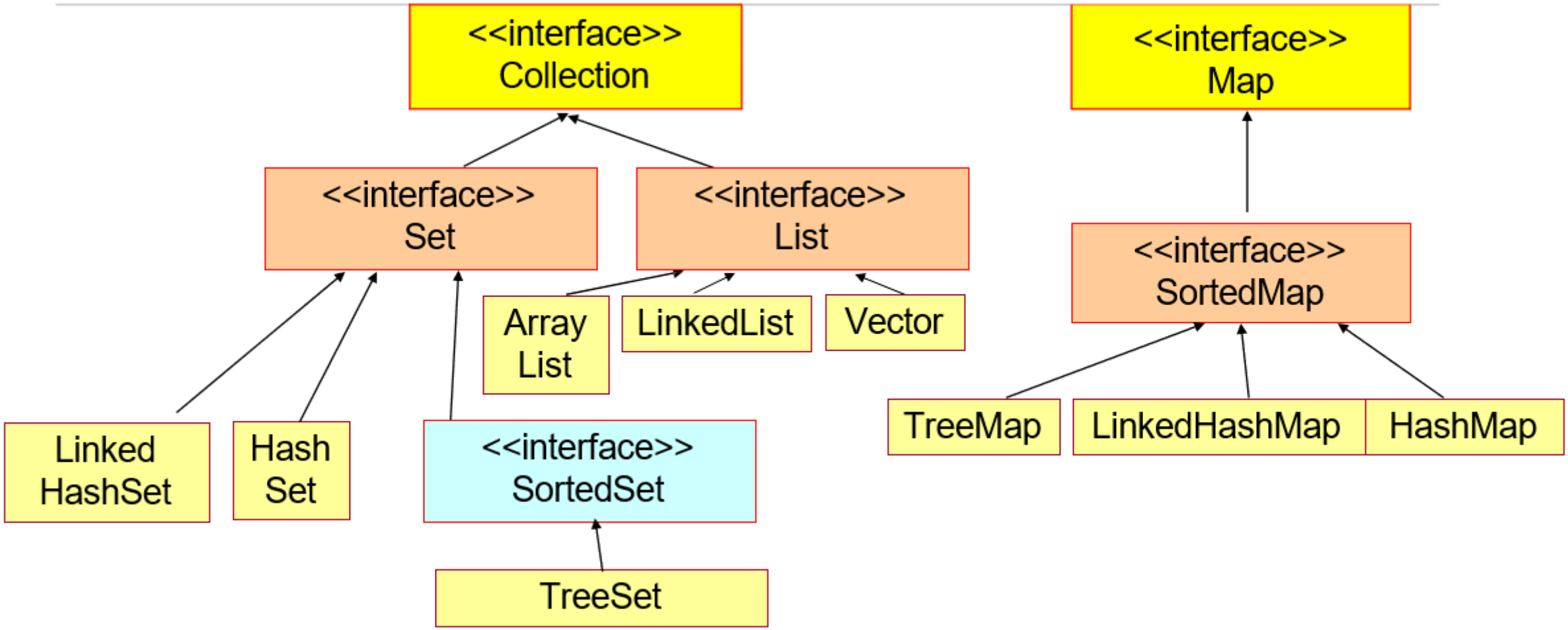
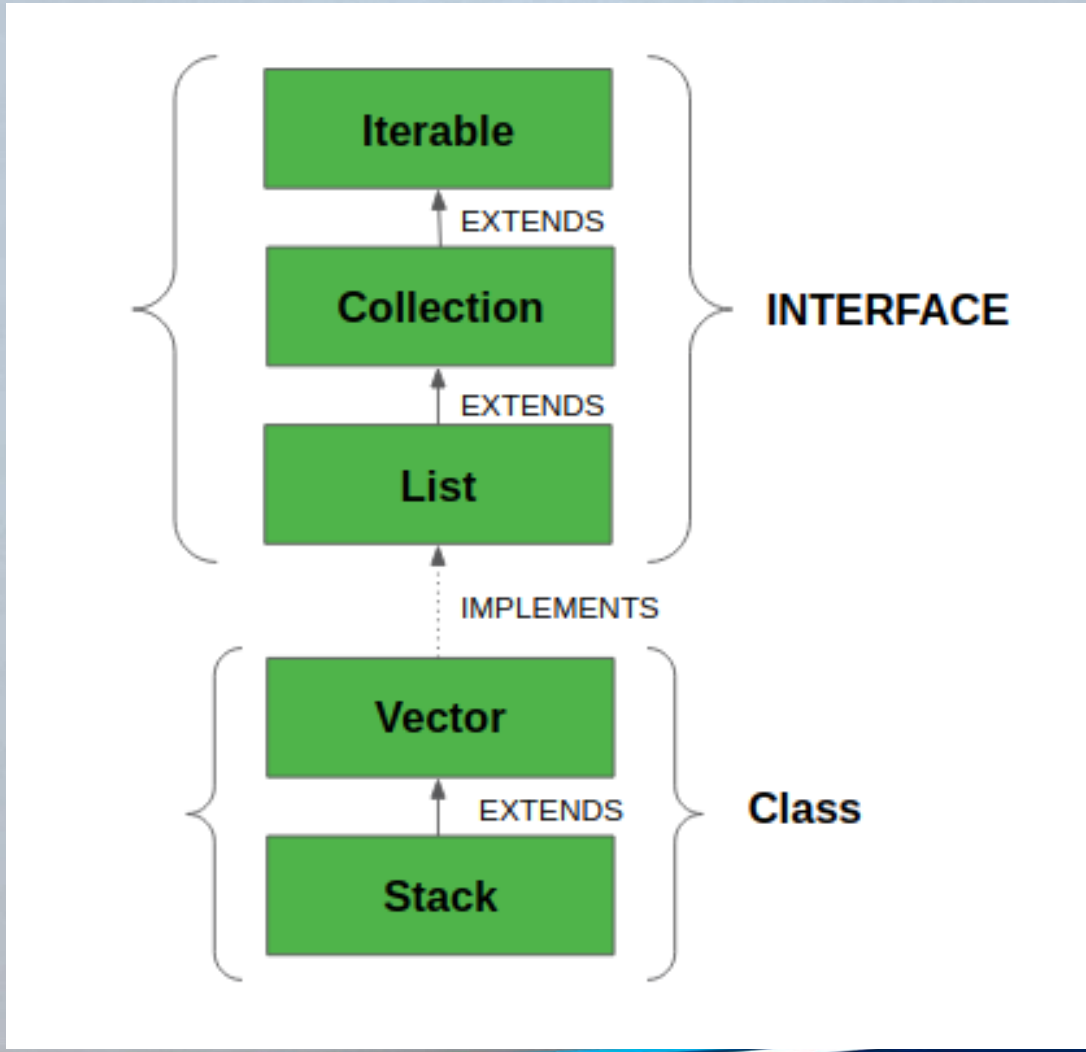
COLLECTIONS FRAMEWORK

Tìm hiểu về Vector

1. SƠ ĐỒ - COLLECTIONS FRAMEWORK(LEGACY)



VECTOR TRONG SƠ ĐỒ COLLECTIONS FRAMEWORK



3

2. VECTOR IMPLEMENTATIONS

- Vector:
  - Implements một mảng động
  - Duy trì thứ tự và vị trí, chỉ số của các giá trị được thêm vào Vector (indexed value)
  - Vector có tính đồng bộ, hỗ trợ đa luồng
  - Vector là lớp không thuộc trong collection framework nhưng được bổ sung thêm các phương thức để phù hợp và thống nhất với các lớp trong Collection Framework
  - Vector class nằm trong package java.util
  - Vector có thể chứa các giá trị trùng lặp.

4

3. KHAI BÁO, CÀI ĐẶT VECTOR

- Vector(): Tạo một vector mặc định có giá trị kích thước là 10.
- Vector v = new Vector();
- Khai báo tường minh:
- Vector<T> v = new Vector<T>();
- Vector(int size): Tạo một vector có giá trị kích thước khởi tạo được định nghĩa bởi giá trị size.
- Vector<T> v = new Vector<T>(int size);
- Vector(int size, int incr): Tạo một vector có kích thước khởi tạo được định nghĩa bởi giá trị size và giá trị kích thước tang theo là incr. Giá trị incr là số phần tử được mở rộng cho vector mỗi khi vector thực hiện tang kích thước bản thân nó.
- Vector<T> v = new Vector<T>(int size, int incr);
- Vector(Collection c): Tạo một vector chứa các giá trị trong tập hợp c.
- Vector<E> v = new Vector<E>(Collection c);

5

3. KHAI BÁO, CÀI ĐẶT VECTOR (TT)

- Vector là mảng động nên có thể được khởi tạo mà không cần định nghĩa kích thước ban đầu.
- Vector nếu không chỉ định rõ <T> thì tập hợp này có lưu trữ các phần từ bất kỳ (Object).
- Ngay cả khi kích thước của vector được định nghĩa, vector có thể tang kích thước nếu vượt quá khai báo ban đầu.

6

3. KHAI BÁO, CÀI ĐẶT VECTOR(TT)

```
import java.util.Arrays;
import java.util.List;
import java.util.Vector;
public class VectorSample {
    public static void main(String[] args) {
        // initialize
        List<Integer> vector = new Vector<>();
        Integer[] arr = {1, 2, 3, 4, 5};
        // gen items
        vector.addAll(Arrays.asList(arr));
```

```
// print all
        System.out.println(vector);
        // remove item at index 1
        vector.remove(1);
        // print one by one
        for (int index : vector) {
            System.out.print(index + "\t");
        }
        System.out.println("");
    }
}
```


7

3. KHAI BÁO, CÀI ĐẶT VECTOR(TT)

- Vector không thể xử lý các kiểu dữ liệu nguyên thủy: int, float, long, char, double
- Các kiểu dữ liệu nguyên thủy có thể được chuyển đổi sang thành các đối tượng bằng cách sử dụng các WRAPPER classes nằm trong package java.lang

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

8

3.KHAI BÁO, CÀI ĐẶT VECTOR(TT)

- Sử dụng chuyển đổi các kiểu dữ liệu nguyên thủy thành đối tượng để Vector có thể xử lý, ví dụ:
  - Integer object = new Integer(i): chuyển kiểu số nguyên sang đối tượng thuộc lớp Integer
  - Float object = new Float(f): chuyển kiểu số thực sang đối tượng thuộc lớp Float
  - Boolean bool = new Boolean(false): chuyển kiểu giá trị boolean sang đối tượng thuộc lớp Boolean
- Chuyển đổi đối tượng về các giá trị có kiểu nguyên thủy trong quá trình xử lý.
  - int in = object.intValue(): converts Integer object to primitive integer
  - float fl = object.floatValue(): converts Float object to primitive float
  - boolean b = object.boolValue(): converts Boolean object to primitive boolean

9

4. MỘT SỐ PHƯƠNG THỨC CỦA VECTOR

- Một số phương thức thông dụng

add(Object element)	Thêm 1 item vào cuối Vector.
add(int index, Object element)	Thêm một giá trị vào vị trí có chỉ số chỉ định trong Vector.
set(int index, Object element)	Lấy và cập nhật giá trị cho phần tử nằm tại vị trí có chỉ số cần được thay đổi giá trị tại đó.
remove(Object element)	Loại bỏ một đối tượng khỏi Vector. Nếu có nhiều đối tượng giống nhau thì Vector sẽ loại bỏ phần tử đầu tiên nó tìm thấy.
remove(int index)	Loại bỏ đối tượng tại vị trí chỉ định. Sau khi loại bỏ đối tượng này, các đối tượng còn lại sẽ được chuyển sang trái để lấp đầy vị trí bị xóa bỏ.
Các phương thức khác được kế thừa từ interface List	

10

4. MỘT SỐ PHƯƠNG THỨC CỦA VECTOR (TT)

get(int index)	Trả về đối tượng tại vị trí chỉ định trong Vector.
addElement(int index)	Thêm một đối tượng vào cuối vector, tang kích thước của nó lên một.
capacity()	Trả về kích thước của vector.
contains(Object o)	Trả về true nếu Vector có chứa đối tượng được truyền vào làm tham số.
indexOf(Object o)	Trả về chỉ số của đối tượng nếu nó thuộc Vector, nếu không sẽ trả về giá trị -1
toArray()	Trả về dãy chứa tất cả các đối tượng thuộc Vector và được sắp xếp đúng thứ tự như Vector.
isEmpty()	Kiểm tra xem Vector có phần tử nào hay không.

11

5. VECTOR VÀ VÍ DỤ ĐIỂN HÌNH

- Minh họa vector và dữ liệu nguyên thủy

Tình huống giả thuyết	Các phương thức sử dụng
➢ Tạo một Vector lưu trữ các số nguyên kèm theo các điều kiện sau: • Các item không được xuất hiện quá 1 lần • Xóa một phần tử tại vị trí chỉ định • Tìm giá trị nhỏ nhất trong Vector • Xuất ra các giá trị trong Vector	➢ add(Object item) ➢ remove(int index) ➢ contains(Object item) ➢ Collections.min() ➢ iterator() ➢ size()

12

5 (TT) MỘT SỐ CÀI ĐẶT CHÚ Ý

- Kiểm tra sự tồn tại của phần tử trong vector sử dụng phương thức contains(object).

```
1. private static void addNotDuplicate(int item, Vector<Integer> vector) {
2.         if (!vector.contains(item)) {
3.             vector.add(item);
4.         } else {
5.             System.err.println(“phần tử đã tồn tại!");
6.         }
```


13

5. (TT) MỘT SỐ CÀI ĐẶT CHÚ Ý

- Kiểm tra vị trí bất kỳ nằm trong khoảng vị trí của vector.
  - private static boolean deleteAtSpecifiedIndex(int index, Vector<Integer> vector) {
  - boolean result = true;
  - if (index < 0 || index > vector.size()) {
  - result = false;
  - }
  - if (result) {
  - vector.remove(index);
  - }
  - return result;
  - }

14

5. (TT) MỘT SỐ CÀI ĐẶT CHÚ Ý

- Sử dụng lớp tiện tích Collections và Vector.
- Tìm min.
  - private static int findMinInteger(Vector<Integer> vector) {
  - return Collections.min(vector);
  - }
- Tìm max
  - private static void sortVector(Vector<Integer> vector) {
  - Collections.sort(vector);
  - }

15

5. (TT) MỘT SỐ CÀI ĐẶT CHÚ Ý

- In nội dung tập hợp dung vòng for;

```
private static void printAllItem(Vector vector) {
        for (Object item : vector) {
System.out.print(item + "\t");
        }
}
```


16

6. ỨNG DỤNG CỦA VECTOR

- Các phương thức cài đặt trong lớp Vector có tính đồng bộ.
- Vector sử dụng trong các chương trình ưu tiên tính đồng bộ và đảm bảo tính an toàn khi thực thi trong cùng luồn, đa luồng.
- Cần một giải pháp lưu trữ linh động và mềm dẻo
- Không quá quan tâm vào tốc độ: tính đồng bộ làm cho Vector chậm hơn so với các lớp kế thừa khác từ interface List.
- Vector là một lớp kế thừa (legacy class) được giới thiệu từ lâu, trước khi ArrayList được xuất hiện từ bản JDK 1.2, với các chương trình sử dụng các bản JDK trước đó có thể sử dụng Vector làm một giải pháp.

17

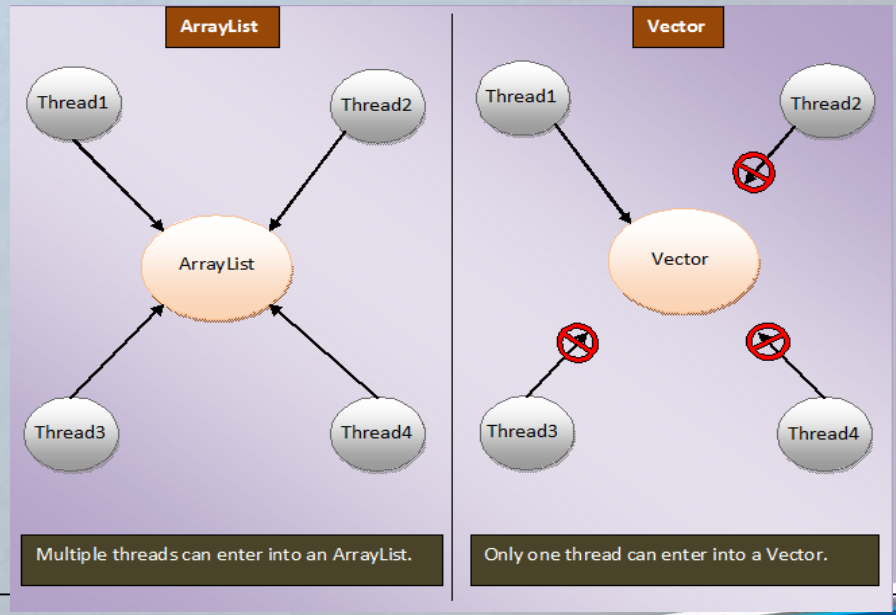
6. SO SÁNH VECTOR VÀ ARRAYLIST

Giống nhau

- Đều cài đặt interface List: đều sử dụng mảng động
- Đều duy trì thứ tự trèn của các phần tử
- Đều có thể chứa các giá trị trùng lặp

Khác nhau

- tính đồng bộ và khả năng hỗ trợ đa luồng (multi-threading)



18

6. SO SÁNH VECTOR VÀ ARRAYLIST(TT)

- Điểm khác nhau

Tiêu chí	ArrayList	Vector
Tính đồng bộ	ArrayList không hỗ trợ tính năng đồng bộ (**non-synchronized**).	Vector là lớp hỗ trợ tính năng đồng bộ (**synchronized)**.
Tốc độ xử lý	ArrayList là nhanh hơn không mất thời gian đồng bộ.	Vector là chậm hơn ví cần đồng bộ nội dung dữ liệu. Tức là, trong môi trường đa luồng, các luồng (thread) giữ vector ở trong trạng thái runnable hoặc non-runnable cho đến khi thread hiện tại giải phóng đối tượng đó.
Phương thức duyệt	ArrayList được duyệt bởi Iterator.	Vector được duyệt bởi Enumeration và Iterator.
Khả năng thay đổi kích thước	Không thể chủ động thay đổi kích thước hiện tại của ArrayList. Kích thước ArrayList chỉ được thay đổi khi thêm hoặc xóa phần tử.	Có thể chủ động thay đổi kích thướng của Vector bằng phương thức **setSize().**

6. SO SÁNH VECTOR VÀ ARRAYLIST(TT)

- Điểm khác nhau

Tiêu chí	ArrayList	Vector
Khả năng thay đổi kích thước	ArrayList tăng 50% kích thước hiện tại nếu số phần tử vượt quá khả năng chứa của nó.	Vector tăng 100% nghĩa là tăng gấp đôi kích thước hiện tại nếu số phần tử vượt quá khả năng chứa của nó.
Cấu hình đồng bộ	Chúng ta có thể làm cho ArrayList đồng bộ bằng cách gọi phương thức: Collections.synchronizedList();	Vector được đồng bộ nội bộ và không thể hủy đồng bộ hóa.
Ứng dụng	ArrayList được ưa thích trong các ứng dụng đơn luồng (single-thread). Nếu bạn muốn sử dụng HashMap trong ứng dụng đa luồng (mulit-thread), có thể thực hiện bằng cách sử dụng phương thức Collections. synchronizedList().	Mặc dù Vector có để sử dụng trong các ứng dụng đa luồng (multi-thread), nhưng ngày nay nó ít được sử dụng. Bởi vì, Collections. synchronizedList() là lựa chọn tốt hơn Vector.

20

TỔNG KẾT

- Vector sử dụng một mảng động như đa số các lớp được implement từ interface List.
- Vector có tính đồng bộ hóa, thích hợp trong trường hợp xây dựng ứng dụng xử lý theo cấu trúc đa luồng.
- Vector bao gồm cả một số phương thức kế thưa (legacy methods) không thuộc về Collections framework.
- Hiệu suất xử lý của Vector không cao và nhanh như một số lớp khác thuộc Collections framework.
- Vector sẽ là sự lựa chọn hoàn hảo nếu người lập trình không biết kích thước của mảng trước đó hoặc chỉ đơn giản là cần một tập hợp có thể linh động thay đổi kích thước của nó trong quá trình chạy ứng dụng.

21

TÀI LIỆU THAM KHẢO

- Bài trình bày của Ngô Việt Trung- Nguyễn Duyên Thái; ĐH CNTT2 K13.
- https://www.tutorialspoint.com/differences-between-arraylist-and-vector-in-java
- https://www.geeksforgeeks.org/java-util-vector-class-java/
- https://www.geeksforgeeks.org/vector-vs-arraylist-java/
- https://www.slideshare.net/abhilash128/vectors-11947188
- https://www.free-powerpoint-templates-design.com/teamwork-business-people-powerpoint-templates/
- https://docs.oracle.com/javase/8/docs/api/java/util/Vector.html
- https://gpcoder.com/2737-so-sanh-arraylist-va-vector-trong-java/

- Quick samples: https://github.com/duyenthaind/vectorSample
- UDF samples: https://github.com/jvjspy/VectorDem

22