



LẬP TRÌNH JAVA 1

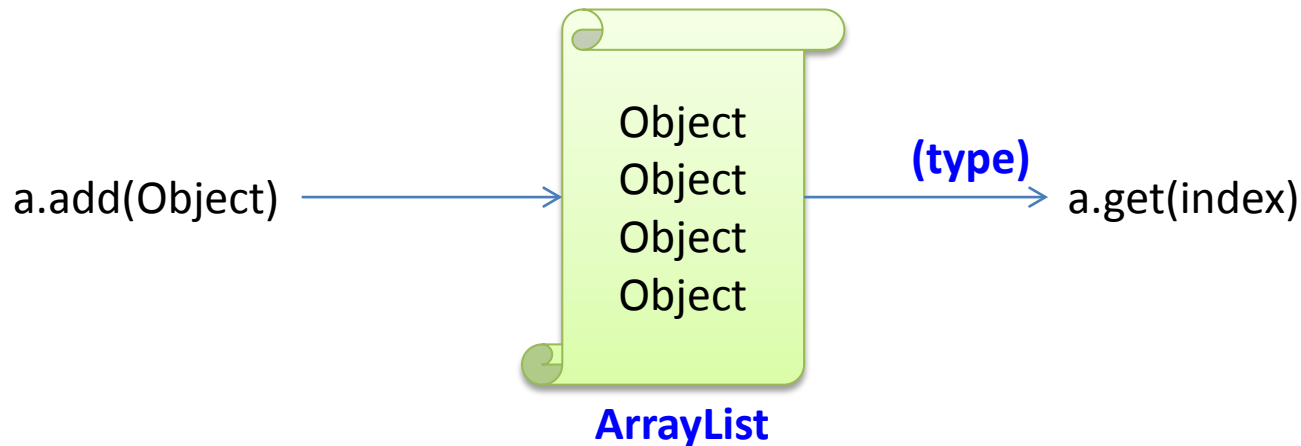
BÀI 5: ARRAYLIST

- ❑ Kết thúc bài học này bạn có khả năng
 - ❖ Hiểu và ứng dụng ArrayList
 - ❖ Hiểu và ứng dụng các hàm tiện ích của Collections

- ❑ Mảng có số phần tử cố định. Vì vậy có các nhược điểm sau:
 - ❖ Không thể bổ sung thêm hoặc xóa bớt các phần tử.
 - ❖ Lãng phí bộ nhớ
 - Nếu khai báo mảng với kích thước lớn để nắm giữ một vài phần tử.
 - Khai báo mảng với kích thước nhỏ thì không đủ chứa
- ❑ ArrayList giúp khắc phục nhược điểm nêu trên của mảng.
 - ❖ ArrayList có thể được xem như mảng động, có thể thêm bớt các phần tử một cách mềm dẻo.
- ❑ ArrayList còn cho phép thực hiện các phép toán tập hợp như hợp, giao, hiệu...

```
ArrayList a = new ArrayList();  
a.add("Cường");  
a.add(true);  
a.add(1);  
a.add(2.5)  
Integer x = (Integer)a.get(2);
```

+ Khi add thêm số nguyên thủy thì tự động chuyển sang đối tượng kiểu **wrapper**
+ Khi truy xuất các phần tử, cần **ép về kiểu gốc** của phần tử để xử lý



ArrayList

ArrayList (Không định kiểu)

ArrayList có thể chứa các phần tử bất kể loại dữ liệu gì.

- + Các phần tử trong ArrayList được đối xử như một tập các đối tượng (kiểu **Object**)
- + Khi truy xuất các phần tử, cần **ép về kiểu gốc** của phần tử để xử lý

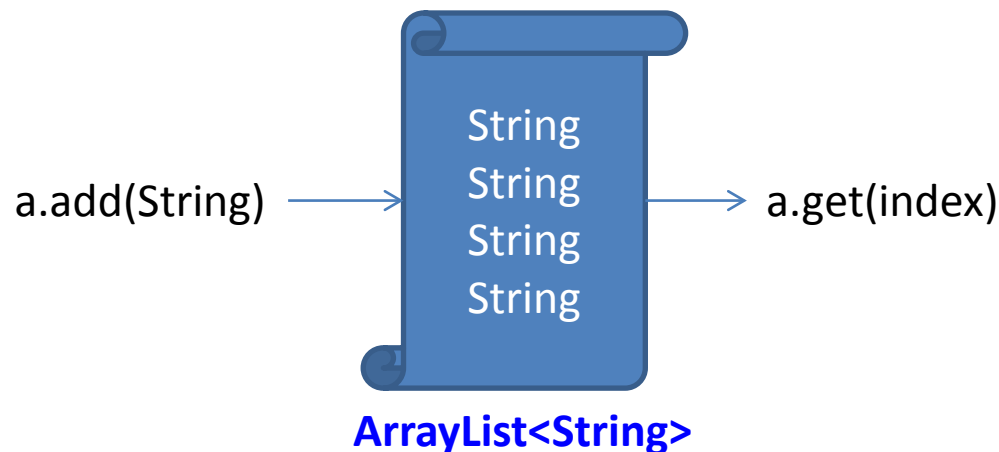
ArrayList<Type> (Có định kiểu)

ArrayList chỉ chứa các phần tử có kiểu đã chỉ định.

- + Khi truy xuất các phần tử **không cần ép** về kiểu gốc của phần tử để xử lý
- + Chặt chẽ, tránh rủi ro lập trình nhầm dữ liệu
- + Hiệu suất xử lý nhanh hơn

```
ArrayList<String> a = new ArrayList<String>();  
a.add("Cường");  
a.add("Tuấn");  
a.add("Phương");  
a.add("Hạnh");  
String s = a.get(2);
```

+ Khi truy xuất các phần tử **không cần ép** về kiểu gốc của phần tử để xử lý



Chú ý: <Type> là kiểu dữ liệu không phải kiểu nguyên thủy (phải sử dụng wrapper)

| PHƯƠNG THỨC | MÔ TẢ |
|------------------------------------|--------------------------------|
| boolean add(Object) | Thêm vào cuối |
| void add(int index, Object elem) | Chèn thêm phần tử vào vị trí |
| boolean remove(Object) | Xóa phần tử |
| Object remove(int index) | Xóa và nhận phần tử tại vị trí |
| void clear() | Xóa sạch |
| Object set(int index, Object elem) | Thay đổi phần tử tại vị trí |
| Object get(int index) | Truy xuất phần tử tại vị trí |
| int size() | Số phần tử |
| boolean contains(Object) | Kiểm tra sự tồn tại |
| boolean isEmpty() | Kiểm tra rỗng |
| int indexOf(Object elem) | Tìm vị trí phần tử |

```
ArrayList<String> a = new ArrayList<String>();  
a.add("Cường");           ← [Cường]  
a.add("Tuấn");            ← [Cường, Tuấn]  
a.add("Phương");          ← [Cường, Tuấn, Phương]  
a.add("Hồng");             ← [Cường, Tuấn, Phương, Hồng]  
a.add(1, "Hạnh");          ← [Cường, Hạnh, Tuấn, Phương, Hồng]  
a.set(0, "Tèo");           ← [Tèo, Hạnh, Tuấn, Phương, Hồng]  
a.remove(3)                ← [Tèo, Hạnh, Tuấn, Hồng]
```



```
ArrayList<String> a = new ArrayList<String>();  
a.add("Cường");  
a.add("Tuấn");  
a.add("Phương");  
a.add("Hồng");  
a.add(1, "Hạnh");  
a.set(0, "Tèo");  
a.remove(3);  
a.remove("Phương");  
int x = a.size() - a.indexOf("Hồng");
```

1. Biến x có giá trị bằng bao nhiêu?

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

2. Nếu thay ~~a.indexOf("Hồng")~~ bằng **a.indexOf("Phương")** thì kết quả x có giá trị là bao nhiêu

- ❑ Duyệt theo **chỉ số** với for hoặc sử dụng **for-each**.
Với ArrayList for-each thường được sử dụng hơn

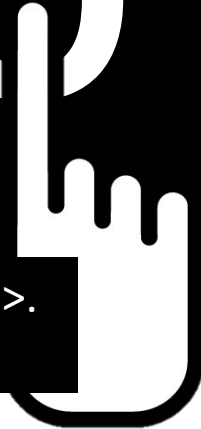
```
ArrayList<Integer> a = new ArrayList<Integer>();  
a.add(5);  
a.add(9);  
a.add(4);  
a.add(8)
```

```
for(int i=0;i<a.size();i++){  
    Integer x = a.get(i);  
    <<xử lý x>>  
}
```

```
for(Integer x : a){  
    <<xử lý x>>  
}
```



DEMO



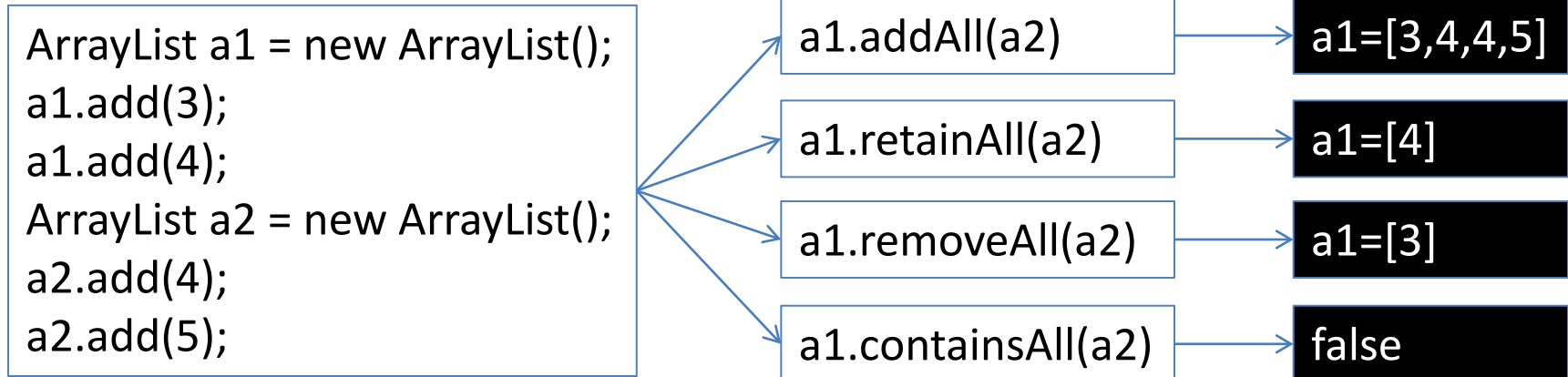
Nhập vào danh sách số thực `ArrayList<Double>`.
Tính tổng và xuất ra màn hình

❑ Sử dụng **ArrayList<SVPoly>** để nắm giữ danh sách sinh viên. Thông tin mỗi sinh viên gồm họ tên và điểm trung bình. Viết chương trình thực hiện việc quản lý như menu sau:

1. Nhập danh sách sinh viên
2. Xuất danh sách sinh viên đã nhập
3. Xuất danh sách sinh viên theo khoảng điểm
4. Tìm sinh viên theo họ tên
5. Tìm và sửa sinh viên theo họ tên
6. Tìm và xóa theo họ tên
7. Kết thúc

```
public class SVPoly{  
    public String hoTen;  
    public Double diemTB;  
}
```

| PHƯƠNG THỨC | MÔ TẢ |
|--|----------------------|
| <code>addAll(Collection)</code> | Hợp 2 tập hợp |
| <code>removeAll(Collection)</code> | Hiệu 2 tập hợp |
| <code>retainAll(Collection)</code> | Giao 2 tập hợp |
| <code>boolean containsAll(Collection)</code> | Kiểm tra sự tồn tại |
| <code>toArray(T[])</code> | Chuyển đổi sang mảng |



- ❑ Lớp tiện ích **Collections** cung cấp các hàm tiện ích hỗ trợ việc xử lý ArrayList

| PHƯƠNG THỨC | MÔ TẢ |
|--|-----------------------------------|
| int binarySearch (List list, Object key) | Tìm kiếm theo thuật toán chia đôi |
| void fill (List list, Object value) | Gán giá trị cho tất cả phần tử |
| void shuffle (List list) | Hoán vị ngẫu nhiên |
| void sort (List list) | Sắp xếp tăng dần |
| void reverse (List list) | Đảo ngược |
| void rotate (List list, int distance) | Xoay vòng |
| void swap(List list, int i, int j) | Tráo đổi |

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

```
a.add(3);
```

```
a.add(9);
```

```
a.add(8);
```

```
a.add(2);
```

← [3, 9, 8, 2]

```
Collections.swap(a, 0, 2);
```

← [8, 9, 3, 2]

```
Collections.shuffle(a);
```

← [X, X, X, X]

```
Collections.sort(a);
```

← [2, 3, 8, 9]

```
Collections.reverse(a);
```

← [9, 8, 3, 2]



DEMO

Nhập danh sách 5 câu hỏi. Tráo ngẫu
nhiên và xuất danh sách câu hỏi đã tráo



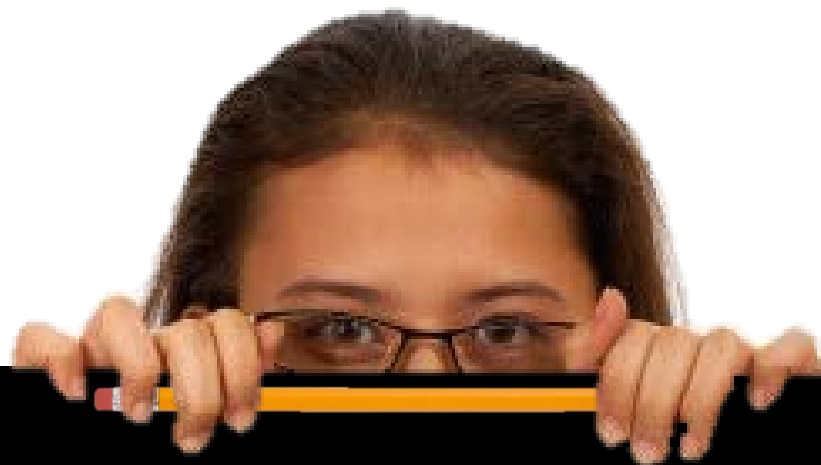
- ❑ Có 2 cách sử dụng `Collections.sort()` để sắp xếp `ArrayList<Object>`
- ❑ Cách 1: `Collections.sort(ArrayList)` đối với các phần tử có khả năng so sánh (Integer, Double, String...)
- ❑ Cách 2: `Collections.sort(ArrayList, Comparator)` bổ sung tiêu chí so sánh cho các phần tử. Cách này thường áp dụng cho các lớp do người dùng định nghĩa (NhanVien, SinhVienPoly...)

- ❑ Tiêu chí so sánh được chỉ ra để thực hiện việc sắp xếp. Trong bài này tiêu chí so sánh 2 SVPoly là so sánh theo điểm.

```
ArrayList<SVPoly> list = new ArrayList<SVPoly>();  
Comparator<SVPoly> comp = new Comparator<SVPoly>() {  
    @Override  
    public int compare(SVPoly o1, SVPoly o2) {  
        return o1.diemTB.compareTo(o2.diemTB);  
    }  
};  
Collections.sort(list, comp);
```

Kết quả của compare() được sử dụng để sắp xếp o1 và o2. Có 3 trường hợp xảy ra:

- ✓ = 0: o1 = o2
- ✓ > 0: o1 > o2
- ✓ < 0: o1 < o2



DEMO

Bổ sung vào đề mô QL SVPoly

8. Sắp xếp theo điểm

9. Sắp xếp theo họ và tên



- ☐ Giới thiệu ArrayList
- ☐ ArrayList có định kiểu
- ☐ Thao tác ArrayList
- ☐ Lớp tiện ích Collections

