Dò bài cũ

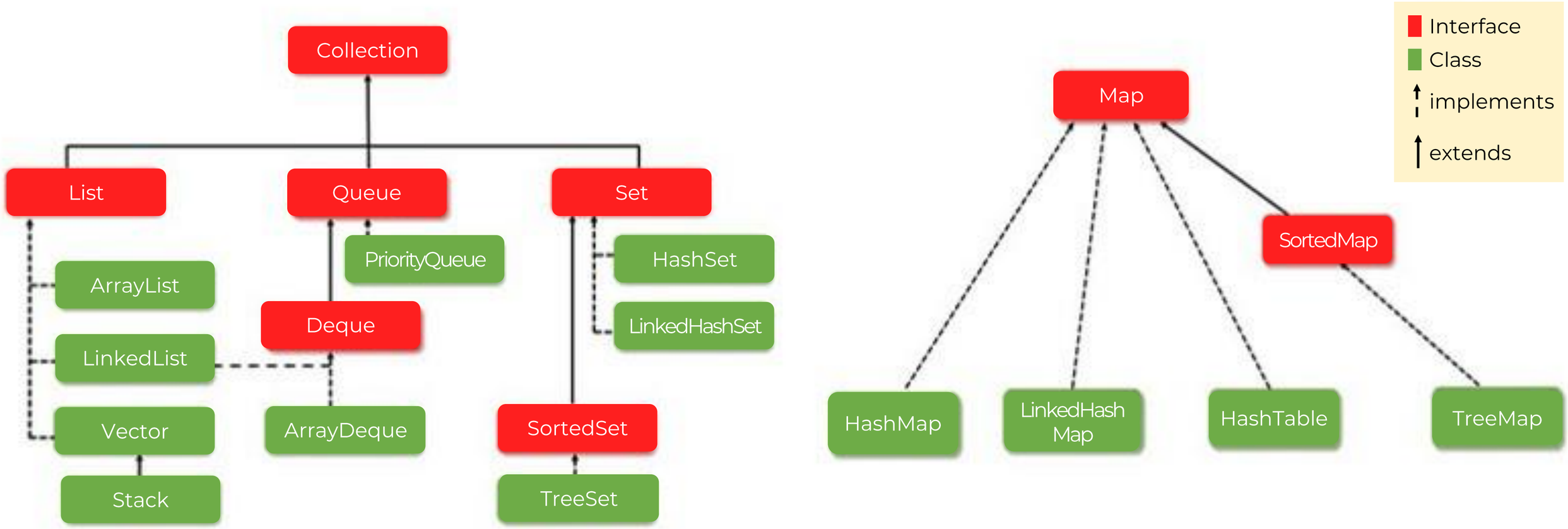
1. **Collection là gì**

2. **So sánh List và Set**

3. **So sánh Array và ArrayList**

4. **So sánh ArrayList và LinkedList**

Java Collection Framework



List vs Set

- **List**
  - Là một **danh sách** có thứ tự của các phần tử.
  - Các phần tử trong List **có thể trùng lặp**

- **Set**
  - Là một **tập hợp** các phần tử **không có thứ tự**
  - **Không** chứa các phần tử **trùng lặp**

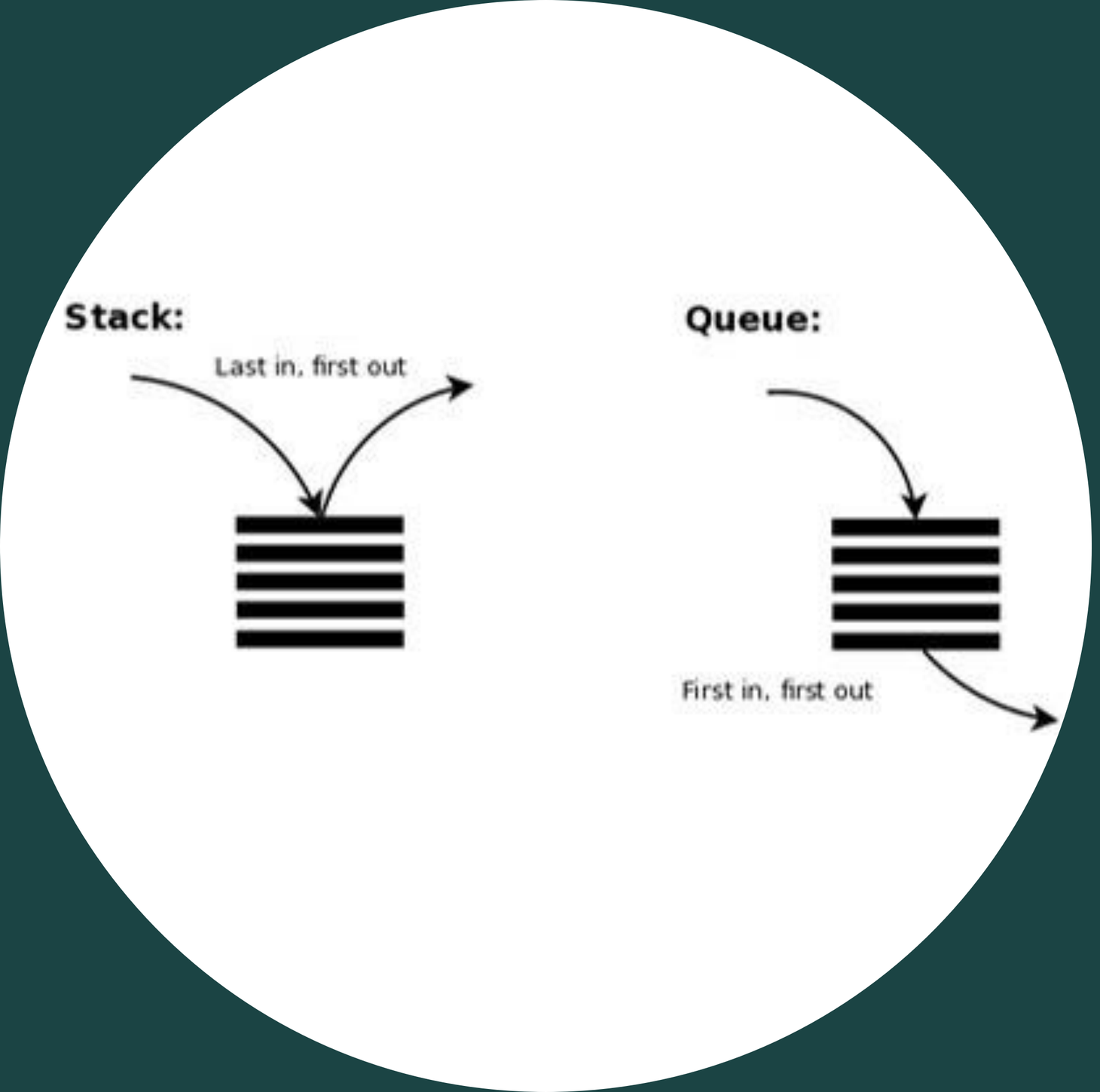
Array vs ArrayList

Tiêu chí	**Array**	**ArrayList**
Kích thước	Cố định, không thay đổi sau khi tạo	Co giãn linh hoạt (tự động mở rộng 50% size)
Kiểu dữ liệu	Hỗ trợ cả kiểu nguyên thủy (int, char)	Chỉ dùng được **object** (Integer, String,...)
Thêm phần tử	Gán qua chỉ số: arr[i] = x;	list.add(x);
Xóa phần tử	Không có, phải tự thực hiện thủ công	list.remove(index);
Tính linh hoạt	Không linh hoạt (**dễ lỗi** khi thiếu/đủ size)	Linh hoạt, tiện dụng hơn nhiều
Thuộc về	Java core	Java **Collections** Framework

ArrayList vs LinkedList

- **ArrayList**
  - Một mảng có thể **co/giản** được vùng nhớ
  - Đặc điểm
    - **Truy cập: Nhanh**
    - **Thêm/xóa: Chậm**

- **LinkedList**
  - Sử dụng một **danh sách liên kết** để lưu trữ
  - Đặc điểm
    - **Truy cập: Chậm**
    - **Thêm/xóa: Nhanh**



Generic Stack Queue

NỘI DUNG CHÍNH

1. Generic, ưu điểm khi sử dụng generic
2. Khái niệm và các method cơ bản của Stack
3. Khái niệm và các method cơ bản của Queue

Generic

- **Generics**
  - Là cơ chế cho phép sử dụng kiểu dữ liệu như là **tham số**
  - Cho phép **tạo** ra các **class**, **method**, **interface** mà **không cần** cố định kiểu dữ liệu **cụ thể** ngay từ đầu.
  - Khi **sử dụng**, chỉ cần **truyền kiểu** cụ thể vào như **String**, **Integer**, **Employee**, ...

- **Ưu điểm**
  - Phát hiện lỗi tại thời điểm biên dịch – **compile time**
  - **Không** cần phải **ép kiểu**
  - Xây dựng các thuật toán tổng quát, **tái sử dụng mã nguồn**

Quy ước đặt tên

- **E**: element
- **T**: type
- **N**: number
- **K**: key
- **V**: value

Stack (ngăn xếp)

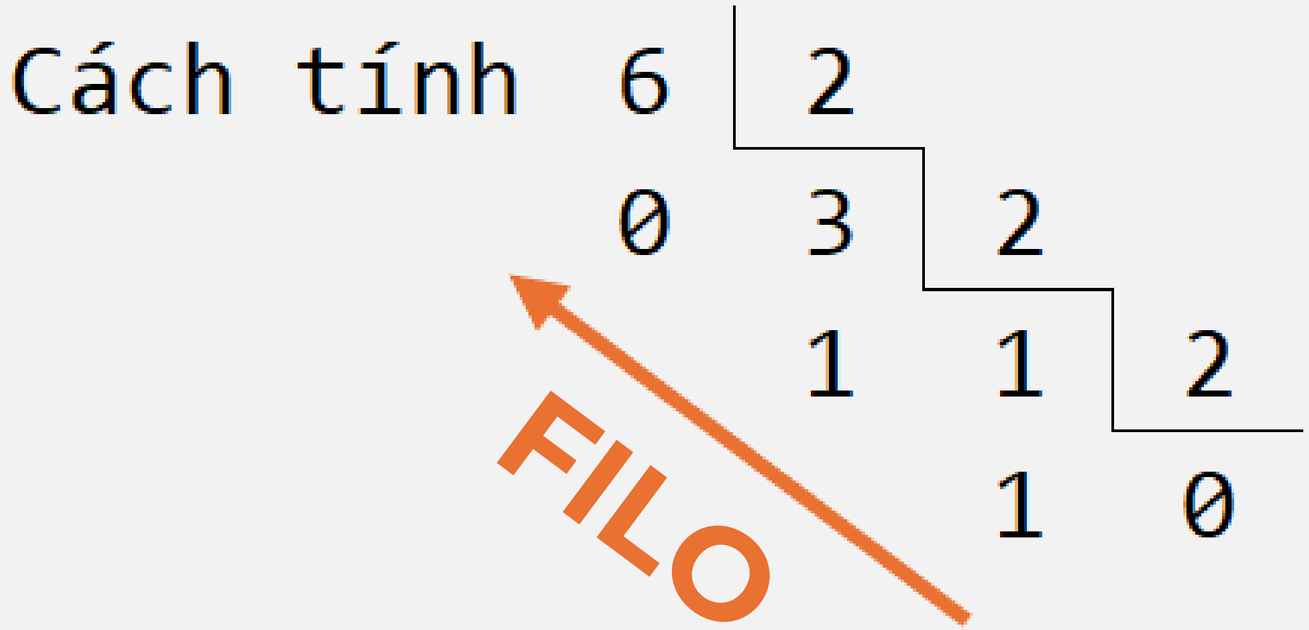
- Là một cấu trúc dữ liệu dạng **danh sách**, thêm và lấy các phần tử theo quy tắc **FILO** (**f**irst-**i**n-**l**ast-**o**ut)

- Các phương thức **CRUD** của Stack
  - **push()**: thêm
  - **peek()**: lấy ra xem không xóa
  - **pop()**: lấy ra xem và xóa
  - **isEmpty()**: kiểm tra rỗng
  - **size()**: trả về số lượng phần tử

Stack (ngăn xếp)

- Bài toán **quy đổi** hệ **thập phân** sang hệ **nhị phân**.
- **Ví dụ:** Chuyển đổi số **6** thành hệ nhị phân
  - **push()**: thêm phần dư vào Stack
  - **pop()**: lấy ra xem và xóa theo **FILO**

Thập phân	Nhị phân
6	110



Queue (hàng đợi)

- Là một cấu trúc dữ liệu dạng **danh sách**, thêm là lấy phần tử theo quy tắc **FIFO** (**f**irst-**i**n-**f**irst-**o**ut)

- Các phương thức **CRUD** của Queue
  - **add()** / **offer()**: thêm
  - **element()** / **peek()**: lấy ra xem
  - **remove()** / **poll()**: lấy ra xem vào xóa
  - **isEmpty()**: kiểm tra rỗng
  - **size()**: trả về số lượng phần tử

DATABASE

THANK YOU