1. So sánh HashMap với List/Array và danh sách liên kết
   1. Tổng quan về cấu trúc

|  |  |
| --- | --- |
| Cấu trúc dữ liệu | Mô tả |
| HashMap | Lưu trữ dữ liệu theo cặp key/value, dùng hàm băm để truy xuất dữ liệu rất nhanh theo key |
| List/Array | Lưu trữ phần tử theo chỉ số trong mảng, cho phép truy xuất ngẫu nhiên nhanh |
| Danh sách liên kết | Cấu trúc danh sách liên kết, mỗi phần tử là 1 node có con trỏ đến phần tử tiếp theo |

* 1. So sánh chi tiết

|  |  |  |  |
| --- | --- | --- | --- |
| Tiêu chí | HashMap | ArrayList/Array | LinkedList |
| **Kiểu lưu trữ** | Cặp key-value | Mảng tuyến tính | Node liên kết |
| **Truy xuất phần tử** | Nhanh qua key | Rất nhanh qua chỉ số | Chậm |
| **Tìm kiếm giá trị** | Chậm | Chậm | Chậm |
| **Chèn cuối danh sách** | Rất nhanh | Rất nhanh | Rất nhanh |
| **Chèn/xoá giữa** | Không tối ưu | Chậm (dịch mảng) | Nhanh(nếu biết node trước) |
| **Duyệt tuần tự** | Qua entrySet hoặc iterator | Qua chỉ số hoặc for-each | Dùng iterator |
| **Bảo toàn thứ tự** | Không đảm bảo | Có | Có |
| **Tối ưu bộ nhớ** | Không (tốn thêm vì lưu key + hash) | Tốt | Tốn do con trỏ |
| **Ứng dụng phổ biến** | Ánh xạ dữ liệu, tìm kiếm theo key | Danh sách dữ liệu, truy xuất theo vị trí | Hàng đợi, danh sách thay đổi thường xuyên |

* 1. Trường hợp sử dụng
     1. HashMap

HashMap hỗ trợ truy xuất theo key rất nhanh. Mỗi phần tử có thể được truy cập trực tiếp bằng khóa mà không cần duyệt toàn bộ danh sách

Ví dụ: Tra cứu thông tin sinh viên theo mã số sinh viên (MSSV), kiểm tra số dư tài khoản theo số tài khoản.

* + 1. List/Array

Truy cập theo chỉ số (index) trong ArrayList hoặc mảng rất nhanh. Dữ liệu có thể được sắp xếp cố định và thao tác dễ dàng qua vòng lặp.

Tiêu tốn ít bộ nhớ và không cần xử lý phức tạp nếu danh sách là cố định và không thay đổi.

Ví dụ: Danh sách điểm thi, danh sách sản phẩm trên web theo trang (paging), quản lý học sinh theo thứ tự nhập vào.

* + 1. Danh sách liên kết

Trong danh sách liên kết, mỗi phần tử là một node có liên kết tới node kế tiếp. Khi chèn hoặc xoá, không cần dịch chuyển các phần tử còn lại như List/Array, chỉ cần thay đổi liên kết.

Danh sách liên kết hỗ trợ tốt việc thêm/xoá ở đầu/cuối danh sách – thích hợp để cài đặt Queue (FIFO) và Stack (LIFO).

Ví dụ: Trình xử lý tác vụ (task queue), danh sách phát nhạc (playlist), undo/redo trong phần mềm chỉnh sửa.

* 1. Ví dụ minh họa



Hinh 1: So sánh tốc độ của HashMap với List/Array

Giải thích chi tiết:

*HashMap<String, Integer> map = new HashMap<>();*

*map.put("A", 1);*

*map.put("B", 2);*

*long startTime = System.nanoTime();*

*map.get("A");*

*long mapTime = System.nanoTime() - startTime;*

- Mục tiêu: Đo thời gian truy cập phần tử có key là "A" trong HashMap.

-System.nanoTime() trả về thời gian chính xác đến nanosecond, được dùng để đo thời gian thực thi.

*List<String> list = new ArrayList<>();*

*list.add("A");*

*list.add("B");*

*startTime = System.nanoTime();*

*list.contains("A");*

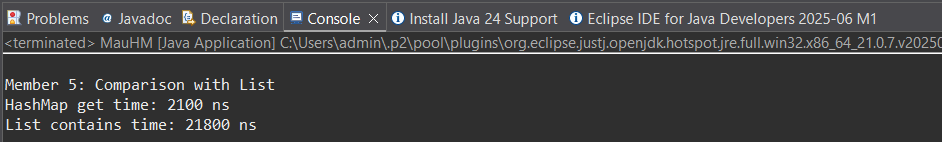
*long listTime = System.nanoTime() - startTime;*

- Mục tiêu: Đo thời gian kiểm tra xem "A" có tồn tại trong ArrayList hay không.

- Sau đó in kết quả ra màn hình

*System.out.println("HashMap get time: " + mapTime + " ns");*

*System.out.println("List contains time: " + listTime + " ns");*



Hinh 2: Kết quả

HashMap mất 2100ns để truy cập phần tử theo key

List/Array mất 21800ns để tìm phần tử theo giá trị

→ Kết luận: HashMap truy xuất nhanh hơn nhờ vào có chế hàm băm

Chương trình trên minh họa rằng:

- HashMap phù hợp cho tìm kiếm theo key

- List/Array cần duyệt toàn bộ danh sách nên chậm hơn đáng kể khi gặp danh sách lớn