1, So sánh khác nhau giữa ArrayList và Linked list? Khi nào nên dung ArrayList, khi nào dùng LinkedList

Bài làm

|  |  |  |
| --- | --- | --- |
| Khác nhau | ArrayList | LinkedList |
| Cấu trúc dữ liệu | ArrayList sử dụng mảng động để lưu trữ các phần tử  -ArrayList  là một cấu trúc dữ liệu dựa trên chỉ mục (index), trong đó mỗi phần tử (element) được liên kết với một chỉ mục. | Lisked sử dụng danh sách liên kết để lưu trữ các phần tử  -Các phần tử trong LinkedList được gọi là node, mỗi node cần lưu trữ 3 thông tin: tham chiếu phần tử trước nó, giá trị của phần tử và một tham chiếu tới phần tử kế tiếp. |
| Thao tác thêm và xóa | Thao tác thêm và xóa phần tử với ArrayList là chậm bởi vì nó sử dụng nội bộ mảng. Bởi vì sau khi thêm hoặc xóa các phần tử cần sắp xếp lại. | Thao tác thêm và xóa phần tử với LinkedList nhanh hơn ArrayList. Bởi vì nó không cần sắp xếp lại các phần tử sau khi thêm hoặc xóa. Nó chỉ cần cập nhật lại tham chiếu tới phần tử phía trước và sau nó. |
| Truy xuất ngẫu nhiên | ArrayList có thể truy xuất ngẫu nhiên phần tử. | LinkedList không thể truy xuất ngẫu nhiên. Nó phải duyệt qua tất cả các phần tử từ đầu tiên đến phần tử cuối cùng để tìm phần tử. |
|  | Sử dụng bộ nhớ ít hơn | Yêu cầu bộ nhớ nhiều hơn |
| Sử dụng khi | **ArrayList tốt hơn trong việc lưu trữ và truy xuất dữ liệu (get)** | **LinkedList tốt hơn trong việc thao tác dữ liệu (thêm/ xóa).** |
|  |  |  |

2, Mối khối try có bắt buộc phải đi kèm 1 khối catch không?

Đúng! Phần try và phần catch trong khối try/catch là những phần bắt buộc phải có. Một khối try có thể đi cùng nhiều khối catch. Ngoài ra, ta còn có thể lắp một phần có tên finally vào làm phần cuối cùng của khối try/catch.

3, Luồng thực thi chương trình của khối try-catch-finally là gì?

Khi có một exception xảy ra chương trình sẽ duyệt tuần tự các khối catch theo thứ tự được khai báo cho đến khi tìm thấy catch với exception tương ứng thì sẽ tiến hành thực thi khối lệnh bên trong nó và bỏ qua các khối catch bên dưới.

4, Khi nào khối finally không được thực thi?

Khối finally sẽ không được thực thi nếu chương trình bị thoát (bằng cách gọi System.exit() hoặc xảy ra một lỗi không thể tránh khiến chương trình bị chết).

5, Throw và throws dùng để làm gì? So sánh throw throws

- Từ khóa throw trong java được sử dụng để ném ra một ngoại lệ rõ ràng.

-Từ khóa **throws** trong java được sử dụng để khai báo một ngoại lệ. Nó thể hiện thông tin cho lập trình viên rằng có thể xảy ra một ngoại lệ, vì vậy nó là tốt hơn cho các lập trình viên để cung cấp các mã xử lý ngoại lệ để duy trì luồng bình thường của chương trình.

Khác nhau giữa throw và throws

|  |  |
| --- | --- |
| **throw** | **throws** |
| Từ khóa throw trong java được sử dụng để ném ra một ngoại lệ rõ ràng. | Từ khóa throws trong java được sử dụng để khai báo một ngoại lệ. |
| Ngoại lệ checked không được truyền ra nếu chỉ sử dụng từ khóa throw. | Ngoại lệ checked được truyền ra ngay cả khi chỉ sử dụng từ khóa throws. |
| Sau throw là một **instance**. | Sau throws là một hoặc nhiều **class**. |
| Throw được sử dụng trong phương thức có thể quăng ra Exception ở bất kỳ dòng nào trong phương thức (sau đó dùng try-catch để bắt hoặc throws cho thằng khác sử lý) | Throws được khai báo ngay sau dấu đóng ngoặc đơn của phương thức. Khi một phương thức có throw bên trong mà không bắt lại (try – catch) thì phải ném đi (throws) cho thằng khác xử lý. |
| Không thể throw nhiều exceptions. | Có thể khai báo nhiều exceptions, Ví dụ:  public void method() throws IOException, SQLException { } |



