**Câu hỏi phỏng vấn Java và cơ sở dữ liệu**

1. **Hãy giải thích khái niệm lớp (class) và đối tượng (object) trong Java.**

* Class là mẫu thiết kế để tạo ra các đối tượng, định nghĩa các phương thức và thuộc tính của đối tượng
* Object là thể hiện của 1 class, có hành vi và trạng thái riêng được định nghĩa bởi class tương ứng

1. **Java có hỗ trợ kế thừa không? Nếu có, hãy giải thích cách thực hiện kế thừa trong Java.**

Có. Kế thừa trong Java là 1 subclass có thể kế thừa lại những phương thức, thuộc tính và hành vi là superclass sở hữu đồng thời subclass có thể tự định nghĩa những phương thức, thuộc tính và hành vi của riêng nó

1. **Sự khác biệt giữa biến cục bộ (local variable) và biến thành viên (instance variable) trong Java là gì?**

* Biến cục bộ (local variable) là biến được khai báo bên trong phương thức, chỉ được sử dụng bên trong phương thức.
* Biến thành viên (instance variable) là biến được khai báo bên trong class và bên ngoài phương thức, được sử dụng cho toàn bộ class và có thể sử dụng trong các phương thức khác của class
* Biến cục bộ được lưu trữ stack, biến thành viên lưu trữ trong heap

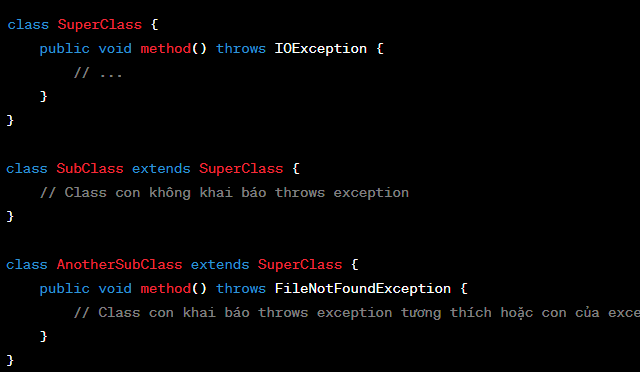
1. **Giải thích khái niệm exception và cách xử lý ngoại lệ (exception handling) trong Java.**

* Exception trong Java là những ngoại lệ chương trình xảy ra trong quá trình thực thi chương trình, có thể là lỗi runtime hoặc lỗi compile
* Cách xử lý ngoại lệ: Sử dụng try catch để bắt exception, sử dụng throw có thể tự định nghĩa exception hoặc sử dụng exception có sẵn , throws để ném ra exception cụ thể khi khai báo phương thức

1. **Class con khi kế thừ lại class cha có throws exceptions thì class con có cần throws exceptions hay k?**

Không cần, nếu class con không muốn chuyển tiếp exception đến người gọi thì k cần throws lại exception của class cha.

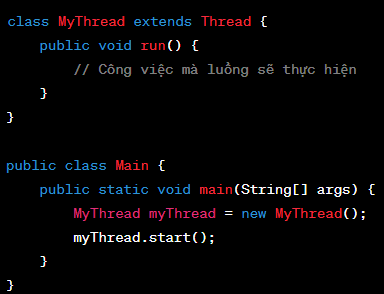
Tuy nhiên nếu class con muốn throws lại exception của class cha thì phải throws exception tương đương hoặc là con của exception trong class cha



1. **Làm thế nào để tạo một luồng (thread) trong Java? Hãy giải thích cách thức quản lý đồng thời (concurrency) trong Java.**

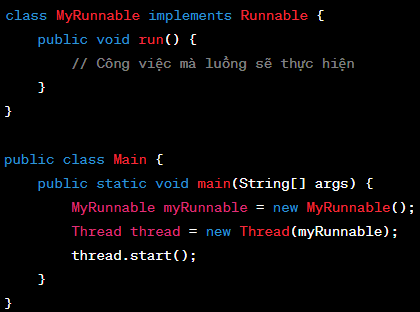
* Có 2 cách tại luồng (thread) trong Java:
* Sử dụng lớp Thread

Class con kế thừa lớp Thread -> Ghi đè run() -> Tạo đối tượng và gọi start() để thực thi



* Sử dụng interface Runnable

Tạo class thực thi interface Runnable -> Ghi đè run() -> tạo đối tượng mới r cho vào constructor của lớp Thread -> gọi start() để thực thi



1. **Hãy giải thích cách sử dụng các phương thức equals() và hashCode() trong Java.**
2. **Java có hỗ trợ garbage collection không? Hãy giải thích nguyên tắc hoạt động của garbage collection trong Java.**

* JVM của Java cho phép quản lý bộ nhớ bằng cơ chế thu gom rác (Garbage collection)
* Garebage collection sẽ loại bỏ những object không còn được sử dụng để cấp phát bộ nhớ mới cho những object khác

1. **Giải thích khái niệm interface và lớp abstract trong Java. Sự khác biệt giữa hai khái niệm này là gì?**

* Interface là tập hợp những phương thức trừu tượng , chứa các hành vi các lớp khác có thể triển khai
* Abstract class là tập hợp những phương thức có thể là trừu tượng hoặc không , chứa các thuộc tính và hành vi các lớp khác có thể triển khai

**Sự khác biệt giữa interface và abstract class là:**

* Interface chỉ chứa các phương thức abstract trong khi đó abstract class có thể chứa phương thức abstract và phương thức non-abstract
* Interface hỗ trợ đa kế thừa trong khi abstract class thì không
* Interface thể hiện tính trừu tượng nhiều hơn là abstract class
* Interface chỉ có thể chứa các thuộc tính static final trong khi abstract class có thể chứa thuộc tính final, non-final, static, non-static

1. **Làm thế nào để đọc và ghi dữ liệu từ tệp tin trong Java?**
2. **Hãy giải thích khái niệm RESTful API và cách bạn đã sử dụng nó trong các dự án trước đây.**

* Kiến trúc REST (Representationnal State Transfer) là 1 kiểu kiến trúc phần mềm dựa trên giao thức HTTP để thiết kế và triển khai các dịch vụ web. Nó tập trung vào việc sử dụng các phương thức HTTP như GET, POST, PUT, DELETE để truyền tải và thao tác dữ liệu trên resources
* RESTful API hoạt động theo nguyên tắc và quy ước được định nghĩa trong kiến trúc REST. RESTful API có những đặc điểm sau:
* **Stateless (Không lưu trạng thái):** Mỗi yêu cầu tới API cần chứa đủ thông tin để được xử lý mà không cần dựa vào trạng thái trước đó của hệ thống.
* **Tài nguyên được xác định qua URL**: Mỗi tài nguyên (resource) được đại diện bởi một địa chỉ URL duy nhất.
* **Sử dụng các phương thức HTTP**: RESTful API sử dụng các phương thức chuẩn của giao thức HTTP như GET, POST, PUT, DELETE để thao tác với tài nguyên.
* **Truyền tải dữ liệu qua định dạng chuẩn**: Dữ liệu được truyền tải qua định dạng như JSON, XML để đảm bảo tính tương thích và giao tiếp giữa các hệ thống.
* **Độc lập với ngôn ngữ**: RESTful API không phụ thuộc vào ngôn ngữ lập trình, cho phép giao tiếp giữa các ứng dụng được viết bằng các ngôn ngữ khác nhau.
* RESTful API thường được sử dụng để xây dựng các dịch vụ web, ứng dụng di động, và hệ thống phân tán. Nó cung cấp một cách tiếp cận đơn giản và linh hoạt để tương tác với các tài nguyên trên mạng.

1. **Giải thích cách kết nối và làm việc với cơ sở dữ liệu trong Java backend.**
2. **Hãy cho biết về Spring Framework và các thành phần chính của nó mà bạn đã sử dụng.**

* Spring framework là bộ khung phát triển ứng dụng Java mạnh mẽ và phổ biến. Cung cấp 1 cách tiếp cận linh hoạt cho việc xây dựng các ứng dụng Java theo kiến trúc hướng đối tượng và phát triển uwsngdujng dễ dàng
* Cách thành phần chính của Spring framework:
* **Dependency Injection (DI)** : Là design pattern giúp giảm thiểu sự phụ thuộc giữa các class, làm cho code trở nên linh hoạt và trực quan hơn, dễ dàng cho việc bảo trì và nâng cấp code
* **Inversion of Control (IoC):** Là trình đảo ngược quyền điều khiển các đối tượng và phụ thuộc.
* **Spring MVC**: Là một module trong Spring Framework phát triển các ứng dụng theo mô hình MVC (Model-View-Controller). Nó cung cấp các thành phần xử lý yêu cầu HTTP, xác định các tầng điều hướng và hiển thị dữ liệu
* **Spring Security**: Là một module trong Spring Framework giúp xây dựng tính năng bảo mật cho ứng dụng (ví dụ để xác thực người dùng hoặc là phân quyền truy cập người dùng vào trang web)
* **Spring Data** : Spring Framework cung cấp Spring Data để làm việc với cơ sở dữ liệu (SQL và NoSQL)
* **Spring Boot**: Là phần mở rộng của Spring Framework đơn giản hóa quá trình cấu hình và triển khai ứng dụng

1. **Giải thích khái niệm ORM (Object-Relational Mapping) và ví dụ về việc sử dụng ORM trong Java backend.**

* ORM (Object-Relational Mapping) là 1 kỹ thuật lập trình phần mềm cho phép ánh xạ dữ liệu giữa các đối tượng (object) trong ngôn ngữ lập trình với các bảng (table) trong cơ sở dữ liệu quan hệ (relational database). ORM cho phép lập trình viên làm việc với dữ liệu dưới dạng đối tượng (object) thay vì dưới dạng câu lệnh SQL thông thường
* Giảm thiểu sự phức tạp khi làm việc với CSDL
* Ví dụ: Hibernate, JPA (Java Persistance API) và MyBatis

1. **Hãy mô tả cách bạn quản lý phiên làm việc (session) trong ứng dụng Java backend**
2. **Hãy nói về việc xử lý bảo mật trong ứng dụng Java backend, bao gồm xác thực (authentication) và xác autorisation (authorization).**

* Trong ứng dụng Java Backend, việc xử lý bảo mật là rất quan trọng để đảm bảo rằng chỉ những người dùng có quyền truy cập thích hợp mới có thể truy cập và thao tác với tài nguyên và chức năng của hệ thống. Điều này thường bao gồm hai khía cạnh chính: xác thực (authentication) và xác autorisation (authorization).
* **Xác thực (Authentication):**

Xác thực là quá trình xác định danh tính của người dùng hoặc hệ thống. Mục đích là đảm bảo rằng người dùng đã đăng nhập hoặc hệ thống đã được xác định chính xác. Các phương pháp xác thực phổ biến trong ứng dụng Java Backend đã được đề cập ở trên, bao gồm Basic Authentication, Token-based Authentication, OAuth, LDAP và SAML. Quá trình xác thực này thường sử dụng thông tin xác thực như tên người dùng, mật khẩu, token hoặc chứng chỉ số để xác định danh tính và quyền truy cập.

* **Xác autorisation (Authorization):**

Xác autorisation là quá trình kiểm tra và quyết định xem người dùng hoặc hệ thống có quyền truy cập và thực hiện các hoạt động cụ thể trong ứng dụng hay không. Nó dựa trên quyền truy cập và vai trò được xác định trước. Một số cách để xác định và quản lý quyền truy cập là sử dụng roles (vai trò) và permissions (quyền) được gán cho từng người dùng hoặc nhóm người dùng. Framework như Spring Security cung cấp các cơ chế để xác định và kiểm tra quyền truy cập dựa trên các annotation, cấu hình hoặc phân quyền tùy chỉnh.

Thông qua việc kết hợp xác thực và xác autorisation, ứng dụng Java Backend có thể đảm bảo rằng chỉ những người dùng được xác thực và có quyền truy cập tương ứng mới có thể thao tác với các tài nguyên và chức năng quan trọng, đồng thời bảo vệ dữ liệu và ngăn chặn các hành động trái phép hoặc đe dọa đến bảo mật của hệ thống.

1. **Làm thế nào để tối ưu hóa hiệu suất (performance) của ứng dụng Java backend?**
2. **Hãy mô tả quá trình triển khai (deployment) ứng dụng Java backend của bạn và công cụ đã sử dụng.**
3. **Hãy giải thích sự khác biệt giữa Servlet và JSP trong Java backend.**
4. **Làm thế nào để xử lý các request AJAX trong ứng dụng Java backend?**
5. **Giải thích khái niệm dependency injection và cách bạn đã áp dụng nó trong Java backend.**
6. **Hãy cho biết về việc kiểm thử (testing) trong Java backend và công cụ kiểm thử mà bạn đã sử dụng.**
7. **Làm thế nào để xử lý và đảm bảo tính nhất quán của dữ liệu trong các giao tác (transactions) trong Java backend?**
8. **Giải thích khái niệm caching và cách bạn đã sử dụng caching trong ứng dụng Java backend.**
9. **Hãy nói về việc xử lý lỗi và ghi nhật ký (logging) trong Java backend.**
10. **Làm thế nào để bảo mật thông tin nhạy cảm như mật khẩu trong quá trình truyền dữ liệu trong ứng dụng Java backend?**
11. **Giải thích sự khác biệt giữa các phiên bản Java (Java SE, Java EE và Java ME) và cách bạn đã sử dụng chúng trong các dự án trước đây.**
12. **Hãy cho biết về quy trình phát triển phần mềm (software development lifecycle) mà bạn đã tham gia và vai trò của bạn trong quy trình đó.**
13. **Hãy giải thích khái niệm Hibernate và cách bạn đã sử dụng nó trong ứng dụng Java backend.**
14. **Làm thế nào để xử lý và xây dựng API Web Service (SOAP hoặc RESTful) trong Java backend?**
15. **Hãy mô tả quá trình xử lý JSON trong Java backend và các thư viện bạn đã sử dụng.**
16. **Giải thích sự khác biệt giữa JDBC và JPA trong Java backend và cách bạn đã sử dụng chúng.**
17. **Làm thế nào để xây dựng và quản lý các phiên bản API trong Java backend?**
18. **Hãy cho biết về việc sử dụng cache để cải thiện hiệu suất trong Java backend và công cụ caching bạn đã sử dụng.**
19. **Hãy mô tả quá trình xử lý và gửi email từ ứng dụng Java backend của bạn.**
20. **Giải thích cách bạn xử lý tương tác với cơ sở dữ liệu NoSQL trong Java backend.**
21. **Hãy cho biết về các nguyên tắc cơ bản của kiến trúc phần mềm (software architecture) trong Java backend và cách bạn đã áp dụng chúng.**
22. **Hãy giải thích khái niệm Spring Boot và tại sao nó được sử dụng phổ biến trong phát triển ứng dụng Java backend.**
23. **Làm thế nào để xử lý và gửi các yêu cầu bất đồng bộ trong Java backend?**
24. **Giải thích cách bạn sử dụng Git trong quy trình phát triển ứng dụng Java backend của bạn.**
25. **Hãy nói về việc sử dụng Docker trong việc triển khai và quản lý ứng dụng Java backend.**
26. **Làm thế nào để xử lý và bảo mật thông tin nhạy cảm như mật khẩu trong cơ sở dữ liệu của ứng dụng Java backend?**
27. **Giải thích khái niệm Dependency Management và cách bạn quản lý các phụ thuộc (dependencies) trong ứng dụng Java backend của bạn.**
28. **Làm thế nào để xử lý và đảm bảo tính nhất quán dữ liệu khi có nhiều phiên bản ứng dụng Java backend đang chạy?**
29. **Hãy mô tả quy trình kiểm tra (testing process) mà bạn đã sử dụng trong việc kiểm thử ứng dụng Java backend của bạn.**
30. **Làm thế nào để đảm bảo bảo mật trong việc truyền dữ liệu giữa ứng dụng Java backend và ứng dụng di động hoặc frontend?**
31. **Giải thích khái niệm Spring Boot và lợi ích của việc sử dụng nó trong phát triển ứng dụng Java.**
32. **Làm thế nào để tạo một ứng dụng Spring Boot?**