Tổng hợp 70+ Câu Hỏi Phỏng vấn Java Backend (Intern - Fresher)

MỤC LỤC

LỜI NÓI ĐẦU

Chào bạn,

Bộ tài liệu này là kết quả của hàng chục buổi phỏng vấn thực tế mình đã trải qua và ghi lại. Không lý thuyết suông, không rườm rà - đây là 70+ câu hỏi "xương máu" mà nhà tuyển dụng dùng để sàng lọc ứng viên Intern/Fresher.

Tại sao bạn cần bộ tài liệu này?

Trúng trọng tâm: Thay vì bơi trong biển kiến thức, bạn chỉ cần học đúng những gì thực sự hay được hỏi.
Điểm mấu chốt (Keywords): Giúp bạn biết rõ nhà tuyển dụng đang chờ đợi từ khóa nào để trả lời cho "trúng" và "chuyên nghiệp".
Giá trị thực tế: Chỉ với **15k** - bạn tiết kiệm được hàng trăm giờ tự mò mẫm và tăng **80% sự tự tin** khi bước vào phòng phỏng vấn.
Đây là khoản đầu tư nhỏ nhất cho bước ngoặt sự nghiệp lớn nhất của bạn.

Chúc bạn ôn luyện tốt và sớm có job ngon!

Nhóm 1: Java Core & OOP (Kiến thức nền tảng)

Đừng coi Java Core là những định nghĩa khô khan, hãy coi nó là ngôn ngữ của tư duy thiết kế. Một lập trình viên xuất sắc không chỉ biết Java chạy như thế nào, mà phải hiểu tại sao nó lại được thiết kế như vậy. Java Core là cái cơ bản nhất nhưng lại dễ bị đánh trượt nhất. Nhiều anh em cứ mải mê học Framework mà quên mất bản chất OOP hay cách JVM nó quản lý bộ nhớ. Đi phỏng vấn, người ta chỉ cần hỏi vài câu về đa hình hay xử lý String là biết ngay bạn có "gốc" hay không. Đừng để đến lúc bị hỏi "Tại sao dùng String Pool?" mà lại lúng túng nhé. Nắm chắc phần này thì mới nói chuyện tiếp mấy phần sau được!

1. 4 Tính chất OOP: Trình bày chi tiết 4 tính chất của lập trình hướng đối tượng.
2. Class vs Object: Phân biệt sự khác nhau giữa Class và Object.
3. Abstract vs Interface: Phân biệt chi tiết Abstract Class và Interface.
4. Java Inheritance: Java có hỗ trợ đa kế thừa không? Tại sao?
5. Interface Solutions: Cách dùng Interface để giải quyết bài toán đa kế thừa hành vi.
6. Multiple Interfaces: Một Class có thể implement nhiều Interface không? Ứng dụng thực tế?
7. Platform Independence: Tại sao code Java có tính độc lập nền tảng (JVM)?
8. JVM, JRE, JDK: Phân biệt các thành phần này và vai trò của IDE.
9. String handling: So sánh String, StringBuilder và StringBuffer.
10. Access Modifiers: Giải thích 4 loại Access Modifier và phạm vi truy cập (Scope).
11. Garbage Collection (GC): Cơ chế tự động dọn rác trong Java hoạt động như thế nào?
12. Exception Handling: Cách xử lý ngoại lệ trong Java (Try-catch-finally).
13. Concurrency: Khái niệm Thread và cơ chế Synchronize.
14. Virtual Thread: Virtual Thread là gì và nó khác gì so với Platform Thread? (Java 21).

ĐÁP ÁN THAM KHẢO

Chào bạn, khi bạn mở đến trang này, mình biết bạn là một người thực sự nghiêm túc với sự nghiệp của mình. Toàn bộ đáp án dưới đây không phải là những định nghĩa khô khan trong giáo trình, mà là những "vũ khí" mình đã đúc kết được sau hàng chục "trận chiến" phỏng vấn thực tế — từ những lần thất bại ê chề đến những buổi trò chuyện đầy tâm đắc.

Nhưng hãy nhớ kỹ điều này: **Đáp án mình viết ra là góc nhìn của mình, dựa trên kinh nghiệm và môi trường mình đã kinh qua**. Kỹ thuật luôn có nhiều con đường để đi, và một lập trình viên giỏi không phải là người học thuộc lòng giỏi nhất, mà là người biết vận dụng kiến thức linh hoạt nhất.

Mình khuyên bạn hãy dùng bộ đáp án này như **một nguồn tham khảo chính,** một kim chỉ nam để bạn biết "nhà tuyển dụng đang thực sự chờ đợi điều gì". Từ đó, hãy tự tay code thử, tự mình đặt câu hỏi "Tại sao?" và tự biến những kiến thức này thành trải nghiệm của riêng bạn. Đừng nói lại y hệt những gì mình viết, hãy trình bày nó bằng sự thấu hiểu và ngôn ngữ của chính bạn – đó mới là lúc bạn thực sự chinh phục được người đối diện.

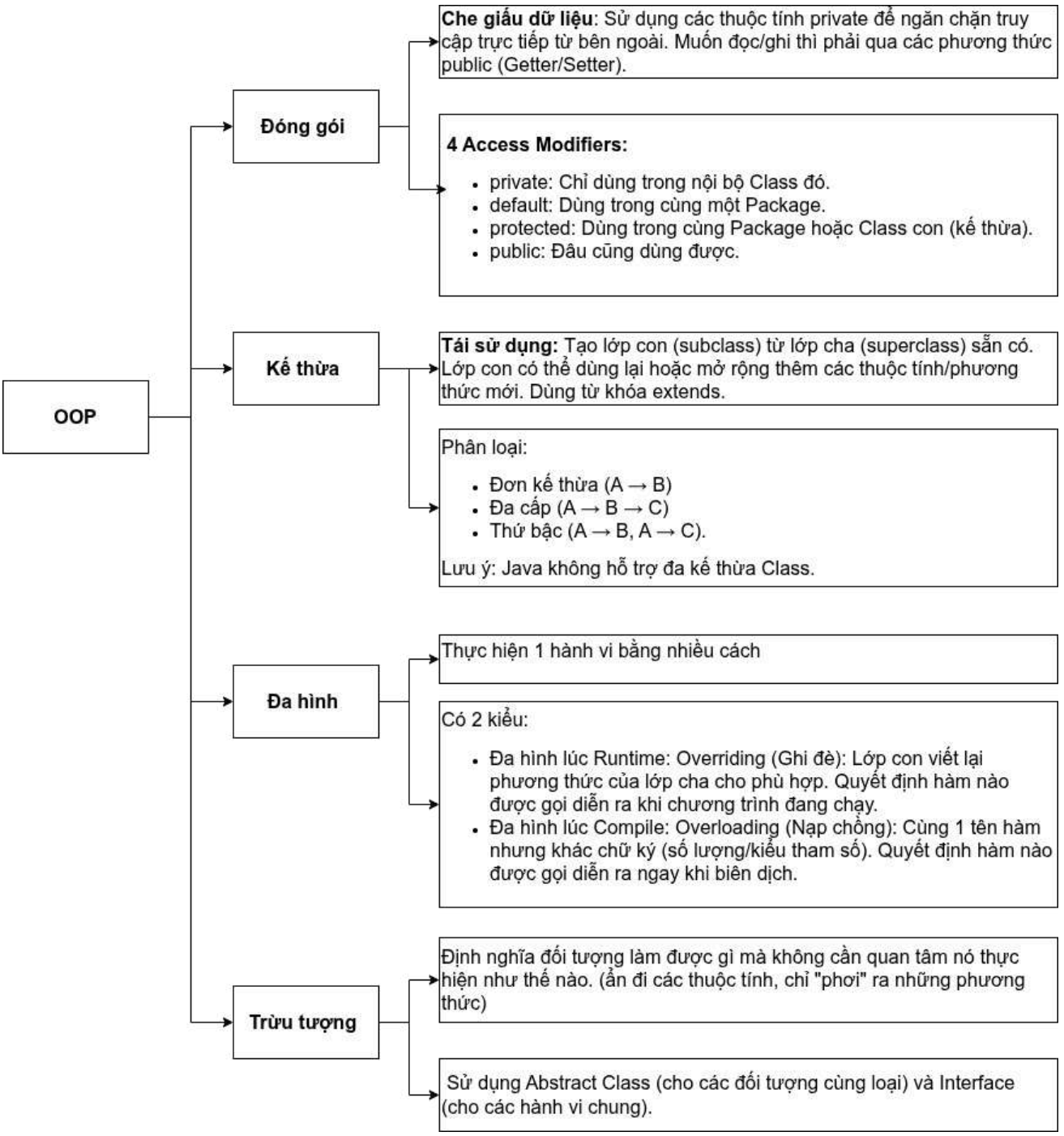
Con đường trở thành một Engineer thực thụ còn dài và nhiều thử thách, nhưng với sự kiên trì và tư duy đúng đắn, mình tin bạn sẽ sớm chạm tay vào Offer mơ ước.

Cứ đi rồi sẽ đến! Đừng bỏ cuộc bạn nhé!

Nhóm 1: Java Core & OOP (Kiến thức nền tảng)

Trước khi chúng ta đi vào từng ngóc ngách của Java Core, mình muốn tặng bạn một chiếc "la bàn" mà mình đã đúc kết và sử dụng xuyên suốt quá trình ôn luyện. Một sai lầm phổ biến của nhiều anh em là nhảy ngay vào học từng định nghĩa rời rạc mà quên mất bức tranh tổng quan.

Sơ đồ tư duy (Mindmap) dưới đây chính là "xương sống" kết nối 4 tính chất cốt lõi của OOP mà bản thân mình tự đúc rút và mô hình hóa lên trong quá trình ôn tập cho các buổi phỏng vấn. Hãy dành ra vài phút để nhìn kỹ nó. Sơ đồ này không chỉ giúp bạn tóm gọn kiến thức mà còn giúp đại não hình thành các "ngăn kéo" thông tin rõ ràng. Khi đứng trước nhà tuyển dụng, nếu lỡ có phút giây lúng túng, hãy nhắm mắt lại và hình dung về 4 nhánh cây này — bạn sẽ tự khắc tìm lại được mạch logic để trả lời một cách trôi chảy nhất.


OOP
Đóng gói
Che giấu dữ liệu: Sử dụng các thuộc tính private để ngăn chặn truy cập trực tiếp từ bên ngoài. Muốn đọc/ghi thì phải qua các phương thức public (Getter/Setter).
4 Access Modifiers:
private: Chỉ dùng trong nội bộ Class đó.
default: Dùng trong cùng một Package.
protected: Dùng trong cùng Package hoặc Class con (kế thừa).
public: Đâu cũng dùng được.
Kế thừa
Tái sử dụng: Tạo lớp con (subclass) từ lớp cha (superclass) sẵn có. Lớp con có thể dùng lại hoặc mở rộng thêm các thuộc tính/phương thức mới. Dùng từ khóa extends.
Phân loại:
Đơn kế thừa (A → B)
Đa cấp (A → B → C)
Thứ bậc (A → B, A → C).
Lưu ý: Java không hỗ trợ đa kế thừa Class.
Đa hình
Thực hiện 1 hành vi bằng nhiều cách
Có 2 kiểu:
Đa hình lúc Runtime: Overriding (Ghi đè): Lớp con viết lại phương thức của lớp cha cho phù hợp. Quyết định hàm nào được gọi diễn ra khi chương trình đang chạy.
Đa hình lúc Compile: Overloading (Nạp chồng): Cùng 1 tên hàm nhưng khác chữ ký (số lượng/kiểu tham số). Quyết định hàm nào được gọi diễn ra ngay khi biên dịch.
Trừu tượng
Định nghĩa đối tượng làm được gì mà không cần quan tâm nó thực hiện như thế nào. (ẩn đi các thuộc tính, chỉ "phơi" ra những phương thức)
Sử dụng Abstract Class (cho các đối tượng cùng loại) và Interface (cho các hành vi chung).

1. 4 Tính chất OOP: Trình bày chi tiết 4 tính chất của lập trình hướng đối tượng.

Như sơ đồ trên, để trả lời gãy gọn nhất, bạn hãy tóm tắt **mỗi tính chất thành 2 luận điểm cốt lõi** sau:

A. Tính Đóng gói (Encapsulation)

- Che giấu dữ liệu: Sử dụng các thuộc tính private để ngăn chặn truy cập trực tiếp từ bên ngoài. Muốn đọc/ghi thì phải qua các phương thức public (Getter/Setter).
- 4 Access Modifiers: private (trong class), default (trong package), protected (package & lớp con), public (mọi nơi).

B. Tính Kế thừa (Inheritance)

- Tái sử dụng: Tạo lớp con (subclass) từ lớp cha (superclass) sẵn có để dùng lại hoặc mở rộng thêm code. Dùng từ khóa extends.
- Phân loại: Đơn kế thừa, Đa cấp, Thứ bậc. (Lưu ý: Java không hỗ trợ đa kế thừa Class).

C. Tính Đa hình (Polymorphism)

- Thực hiện 1 hành vi bằng nhiều cách
- Có 2 kiểu đa hình:
    - Runtime Polymorphism (Overriding): Lớp con viết lại phương thức của cha. Quyết định hàm nào được gọi diễn ra khi đang chạy.
    - Compile-time Polymorphism (Overloading): Cùng tên hàm nhưng khác chữ ký. Quyết định hàm nào được gọi diễn ra ngay khi biên dịch.

D. Tính Trừu tượng (Abstraction)

- Chỉ định nghĩa đối tượng làm được gì mà không cần quan tâm nó thực hiện như thế nào (Che giấu các thuộc tính của đối tượng, chỉ "phơi" ra các phương thức)
- Triển khai: Sử dụng Abstract Class (cho các đối tượng cùng loại) và Interface (cho các hành vi chung).