

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

MỤC LỤC

LỜI NÓI ĐẦU.....	5
Nhóm 1: Java Core & OOP (Kiến thức nền tảng).....	6
Nhóm 2: Spring Framework & Spring Boot.....	7
Nhóm 3: Database.....	8
Nhóm 4: Security & Web API.....	9
Nhóm 5: System Design & Performance.....	10
Nhóm 6: DSA (Data Structures & Algorithms).....	11
Nhóm 7: DevOps, Git & Linux.....	12
Nhóm 8: Kinh Nghiệm & Kỹ Năng Mềm.....	13
ĐÁP ÁN THAM KHẢO.....	14
Nhóm 1: Java Core & OOP (Kiến thức nền tảng).....	15
1. 4 Tính chất OOP: Trình bày chi tiết 4 tính chất của lập trình hướng đối tượng.....	17
2. Class vs Object: Phân biệt sự khác nhau giữa Class và Object.....	18
3. Abstract vs Interface: Phân biệt chi tiết Abstract Class và Interface.....	18
4. Java Inheritance: Java có hỗ trợ đa kế thừa không? Tại sao?.....	19
5. Interface Solutions: Cách dùng Interface để giải quyết bài toán đa kế thừa hành vi.....	20
6. Multiple Interfaces: Một Class có thể implement nhiều Interface không? Ứng dụng thực tế?.....	21
7. Platform Independence: Tại sao code Java có tính độc lập nền tảng (JVM)?... 21	
8. JVM, JRE, JDK: Phân biệt các thành phần này và vai trò của IDE.....	21
9. String handling: So sánh String, StringBuilder và StringBuffer.....	22
10. Access Modifiers: Giải thích 4 loại Access Modifier và phạm vi truy cập (Scope).....	22
11. Garbage Collection (GC): Cơ chế tự động dọn rác trong Java hoạt động như thế nào?.....	23
12. Exception Handling: Cách xử lý ngoại lệ trong Java (Try-catch-finally)...	24
13. Concurrency: Khái niệm Thread và cơ chế Synchronize.....	24
14. Virtual Thread: Virtual Thread là gì và nó khác gì so với Platform Thread? (Java 21).....	25
Nhóm 2: Spring Framework & Spring Boot.....	25
15. Spring vs Spring Boot: Phân biệt sự khác nhau cốt lõi và ưu điểm của từng loại.....	25
16. IoC & DI: Định nghĩa Inversion of Control và Dependency Injection.....	25
17. DI Performance: DI/IoC có làm chậm hiệu năng hệ thống không? Cơ chế load Bean?.....	27

18. Bean Annotations: Phân biệt @Component, @Service, @Repository, và @Controller.....	28
19. Bean Scopes: Trình bày các loại Scope của Bean trong Spring.....	28
20. AOP: Lập trình hướng khía cạnh (Aspect-Oriented Programming).....	29
21. Annotation General: Khái niệm Annotation và vai trò của nó.....	30
22. @Transactional: Ý nghĩa và cách quản lý giao dịch trong Spring.....	30
23. Mapstruct: Cách dùng Mapstruct để mapping dữ liệu tự động.....	31
24. Mapstruct Date: Xử lý các trường thời gian khi sử dụng Mapstruct.....	31
25. JWT Verification: Trong Spring, sử dụng hàm nào để decode và verify JWT?.....	31
26. CRUD Architecture: Thiết kế cấu trúc code chuẩn cho module User.....	32
Nhóm 3: Database.....	32
27. SQL Order: Thứ tự thực thi của các mệnh đề trong một query SQL.....	32
28. SQL Duplicates: Cách viết SQL để xóa các bản ghi trùng email (10k users). 33	
29. Database Analysis: Quy trình phân tích và thiết kế DB.....	33
30. Pagination: Kỹ thuật phân trang bằng LIMIT và OFFSET.....	33
31. Index Impact: Hiệu năng bảng có Index và không có Index.....	33
32. Index Types: Phân biệt Single Index và Composite Index.....	33
33. Composite Index Rule: Quy tắc Leftmost Prefix.....	34
34. Explain Plan: Cách kiểm tra một câu query có ăn Index hay không.....	34
35. B-Tree: Tại sao Database dùng cấu trúc B-Tree cho Index?.....	34
36. Database Scaling: Mô hình Master-Slave (Read/Write Splitting).....	34
37. Redis & Webflux: Kinh nghiệm làm việc với Redis và Webflux.....	34
Nhóm 4: Security & Web API.....	34
38. SQL Injection: Khái niệm và kỹ thuật phòng tránh.....	34
39. XSS: Cross-Site Scripting là gì và cách ngăn chặn.....	35
40. Token Flow: Quy trình Access Token và Refresh Token thực tế.....	35
41. JWT Stateless: Tại sao nói JWT là Stateless và ưu điểm của nó.....	35
42. Token Revocation: Cách xử lý thu hồi Token (Redis Blacklist).....	35
43. JWT Payload: Phần Payload chứa những gì?.....	35
44. GET vs POST: So sánh chi tiết. Tại sao POST bảo mật hơn?.....	36
45. POST for Data Retrieval: Có thể dùng POST để lấy dữ liệu không? Case nào nên dùng?.....	36
Nhóm 5: System Design & Performance.....	36
46. API Slowdown: Các bước phân tích và tối ưu khi API bị chậm.....	36
47. Payment Scaling: Thiết kế hệ thống linh hoạt cho nhiều ví điện tử (Strategy Pattern).....	36
48. High Concurrency: Giải pháp cho hệ thống chịu tải 10,000 request/s.....	36

49. Alternative to Rate Limit: Các phương án bảo vệ service nếu không dùng Rate Limit.....	37
50. Kafka Integration: Ứng dụng Kafka trong bài toán xử lý đồng thời cao.....	37
51. Kafka Reliability: Cơ chế Retry và DLT (Dead Letter Topic).....	37
52. Realtime Systems: Kinh nghiệm xử lý bài toán realtime.....	37
53. Design Patterns: Các mẫu thiết kế thường dùng.....	37
54. DDD: 3 câu tóm gọn về Domain-Driven Design (DDD).....	37
55. Clean Code: Các quy chuẩn và công cụ để giữ code luôn sạch.....	38
56. Performance Cases: Các tình huống thực tế bạn từng tối ưu hiệu năng.....	38
Nhóm 6: DSA (Data Structures & Algorithms).....	38
57. Data Structures for Filtering: Cấu trúc dữ liệu lọc trùng String hiệu quả nhất.....	38
58. Sorting Algorithms: Các thuật toán sắp xếp và độ phức tạp.....	38
59. Searching: Thuật toán Binary Search: Nguyên lý và độ phức tạp.....	38
60. Valid Anagram: Giải thuật bài LeetCode 242.....	38
61. Sum 1 to n: Các phương pháp tính tổng (Vòng lặp vs Công thức Toán)....	39
Nhóm 7: DevOps, Git & Linux.....	39
62. Docker Commands: Liệt kê các lệnh Docker hay dùng nhất.....	39
63. Dockerfile: Ý nghĩa các thành phần chính trong Dockerfile.....	39
64. Stop vs Kill: Phân biệt docker stop và docker kill.....	39
65. Linux Components: Các thành phần lõi của Linux.....	39
66. Git Conflict: Quy trình resolve conflict khi dùng Git.....	39
LỜI KẾT.....	40

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 mãi 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. Nhớ 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.

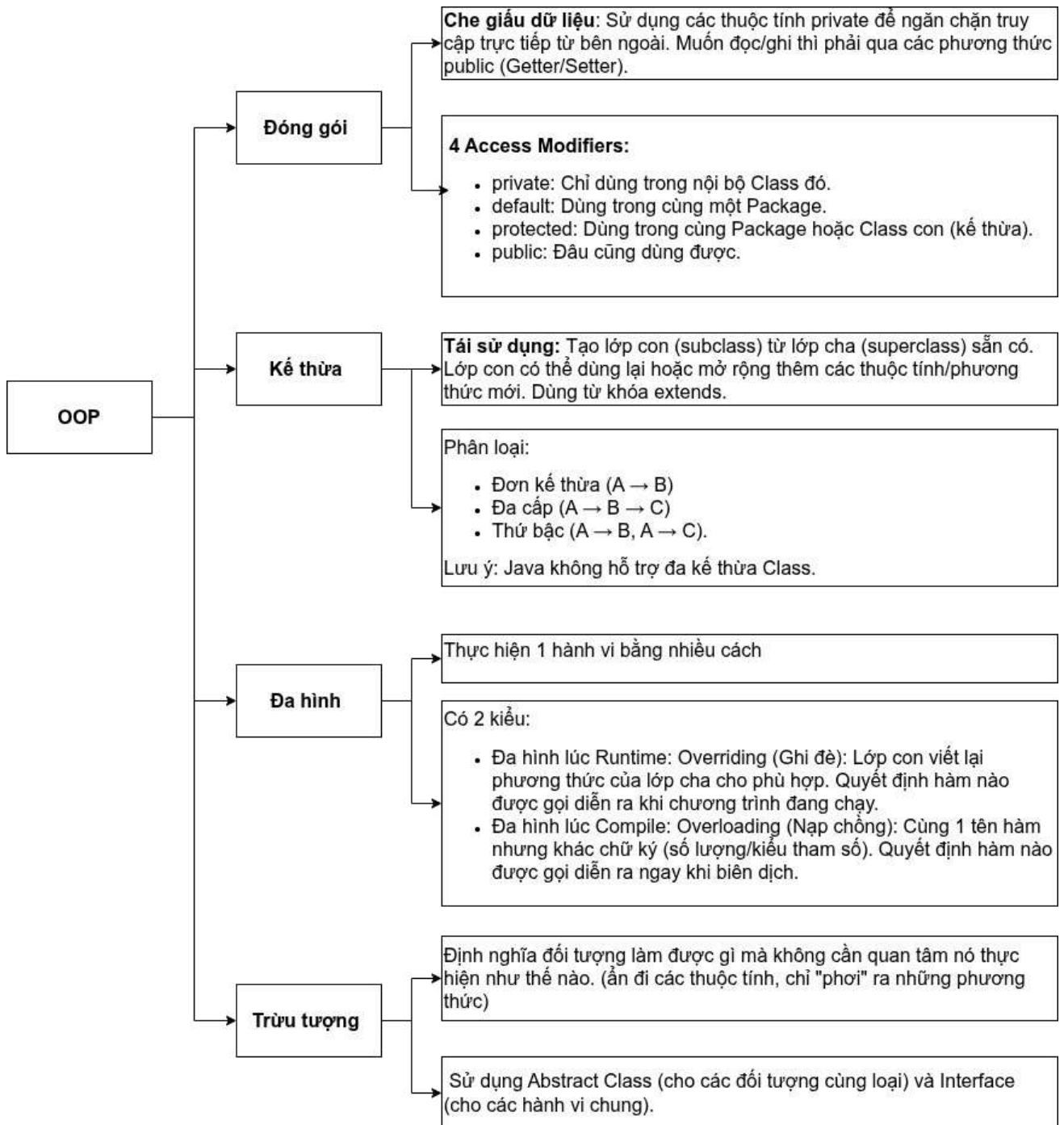
Con đường trở thành một Engineer thực thụ còn dài và nhiều thử thách, nhưng với sự kiên trì và tư duy đúng đắn, mình tin bạn sẽ sớm chạm tay vào Offer mơ ước.

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 góc 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.



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).