🟢 PHẦN 1 — CÂU HỎI DỄ (Fresher – level bắt buộc biết)

---

1. Kotlin Coroutines là gì?

Đáp án: Cơ chế xử lý bất đồng bộ nhẹ, không block thread, tối ưu hơn Thread truyền thống.

---

2. launch dùng để làm gì?

Đáp án: Chạy coroutine **không trả kết quả**, dùng cho tác vụ không yêu cầu return.

---

3. async khác launch như thế nào?

Đáp án:

- `async` → trả về Deferred, có `await()` để lấy kết quả
- `launch` → không trả kết quả

---

4. withContext() dùng khi nào?

Đáp án: Để **chuyển thread** trong coroutine, ví dụ IO → Main.

---

5. Dispatcher.IO dùng để làm gì?

Đáp án: Dùng cho tác vụ I/O: API, DB, đọc/ghi file.

---

6. Một suspend function là gì?

Đáp án: Hàm có thể tạm dừng mà không chặn thread, chỉ chạy trong coroutine.

---

7. Data class là gì?

Đáp án: Class chứa dữ liệu, tự sinh equals(), hashCode(), toString(), copy().

---

8. Extension function là gì?

Đáp án: Hàm mở rộng thêm cho class mà không cần kế thừa hoặc sửa code gốc.

---

9. Sealed class là gì?

Đáp án: Class giới hạn số lượng subclass, dùng mô tả trạng thái hữu hạn.

---

10. Null safety trong Kotlin là gì?

Đáp án: Cơ chế giúp tránh NullPointerException bằng cách phân loại `T` và `T?`.

---

11. Dấu ?. có nghĩa là gì?

Đáp án: Safe-call operator, gọi hàm chỉ khi object != null.

---

12. !! dùng để làm gì?

Đáp án: Ép giá trị là non-null → nếu null thì crash.

🟢 PHẦN 2 — CÂU HỎI TRUNG BÌNH (Fresher – Junior)

---

13. Flow là gì?

Đáp án: Luồng dữ liệu bất đồng bộ kiểu "cold stream" – chỉ chạy khi được collect.

---

14. StateFlow khác Flow như thế nào?

Đáp án: StateFlow là **hot stream** và luôn có **giá trị hiện tại**, phù hợp UI.

---

15. SharedFlow dùng để làm gì?

Đáp án: Dùng cho event như toast, navigation, không giữ state.

---

16. LiveData khác StateFlow thế nào?

Đáp án:

- LiveData: lifecycle-aware
- StateFlow: không lifecycle-aware, thuộc Kotlin Flow
- StateFlow modern hơn, dùng nhiều trong MVVM hiện đại

---

17. Sự khác nhau giữa Thread và Coroutine?

Đáp án:

- Thread: nặng, cost cao
- Coroutine: nhẹ, có thể chạy hàng ngàn coroutine không tốn tài nguyên

---

18. CoroutineScope là gì?

Đáp án: Nơi coroutine sống. Khi scope bị cancel → toàn bộ coroutine bị cancel.

---

19. ViewModelScope là gì?

Đáp án: Scope chạy coroutine thuộc vòng đời ViewModel → tránh leak.

---

20. GlobalScope là gì và có nên dùng không?

Đáp án: GlobalScope sống cùng app, dễ leak → **không nên dùng** trong Android.

---

21. Tại sao Flow là "cold"?

Đáp án: Vì chỉ chạy khi có người collect.

---

22. Hai cách collect StateFlow trong Activity?

Đáp án:

- lifecycleScope.launchWhenStarted { ... }
- repeatOnLifecycle(Lifecycle.State.STARTED)

---

23. Tại sao StateFlow không cho phép null?

Đáp án: Vì luôn phải có state hiện tại → null làm state không xác định.

---

24. sealed class dùng trong UI state như thế nào?

Ví dụ:

```
sealed class Result {
  object Loading : Result()
  data class Success(val data: List<User>) : Result()
  data class Error(val msg: String) : Result()
}
```

---

25. Khi nào dùng data class thay class thường?

Đáp án: Khi chỉ cần chứa dữ liệu và muốn tự động generate: equals, hashCode,...

---

26. Sự khác nhau giữa lateinit và lazy?

Đáp án:

- lateinit: var, khởi tạo sau
- lazy: val, khởi tạo khi sử dụng

---

27. lateint dùng sai thì gây lỗi gì?

Đáp án: lateinit property not initialized.

---

28. Một nullable type khác non-nullable type như thế nào?

Đáp án: `String` không chứa null `String?` chứa null → phải kiểm tra trước khi sử dụng

---

29. safe-call (?.) khác let như thế nào?

Đáp án: `?.` dùng để gọi hàm nếu không null. `?.let { }` chạy 1 block nếu không null.

---

30. Elvis operator (?😃 dùng khi nào?

Đáp án: Dùng để gán giá trị mặc định nếu bên trái null.

🟢 PHẦN 3 — CÂU HỎI KHÓ (Junior – Mid-level Kotlin)

---

31. Bạn giải thích cơ chế cooperative cancellation trong Coroutine?

Đáp án: Coroutine tự kiểm tra cancellation qua:

- delay
- yield
- isActive

Không phải Thread-level kill — tránh crash.

---

32. Difference giữa async(start = LAZY) và launch(start = LAZY)?

Đáp án: Cả hai delay execution, nhưng:

- async(LAZY): chỉ chạy khi await()
- launch(LAZY): chỉ chạy khi start()

---

33. Backpressure trong Flow là gì?

Đáp án: Cơ chế quản lý tốc độ producer/consumer để tránh overflow.

---

34. SharedFlow replay = 0 dùng cho trường hợp nào?

Đáp án: Event như toast → không cần ghi nhớ giá trị cũ.

---

35. MutableStateFlow vs MutableSharedFlow?

Đáp án:

StateFlow	SharedFlow
giữ state hiện tại	không lưu state
phù hợp UI state	phù hợp UI event
giống LiveData	giống SingleLiveEvent

---

36. callbackFlow dùng làm gì?

Đáp án: Chuyển callback API → thành Flow (dùng nhiều với Sensor, Location).

---

37. mapLatest trong Flow là gì?

Đáp án: Cancel tác vụ cũ nếu có tác vụ mới đến → debounce.

---

38. Tại sao extension function không override được?

Đáp án: Vì nó không thay đổi class thật, chỉ "gắn thêm" lúc compile.

---

39. Sự khác nhau giữa abstract class và sealed class?

Đáp án:

- abstract: subclass không giới hạn
- sealed: subclass giới hạn trong file → dễ quản lý state

---

40. Vì sao Kotlin hạn chế null?

Đáp án: Để loại bỏ lỗi kinh điển NullPointerException từ Java → code an toàn hơn.
