🟢 PHẦN 1 — CÂU HỎI DỄ (Level Fresher – Warm Up)

---

1. LinearLayout là gì?

Đáp án: Layout sắp xếp view theo 1 chiều duy nhất (vertical hoặc horizontal).

---

2. RelativeLayout là gì?

Đáp án: Layout cho phép đặt view tương đối với parent hoặc view khác (trên, dưới, trái, phải).

---

3. ConstraintLayout là gì?

Đáp án: Layout mạnh nhất hiện nay, dùng "constraint" để định vị view. Giúp giảm nested layout và tối ưu hiệu năng.

---

4. Hiệu năng: LinearLayout vs ConstraintLayout?

Đáp án: ConstraintLayout hiệu năng tốt hơn vì giảm độ sâu view hierarchy.

---

5. RecyclerView dùng để làm gì?

Đáp án: Hiển thị danh sách lớn với hiệu năng cao nhờ tái sử dụng ViewHolder.

---

6. ViewHolder là gì?

Đáp án: Class giữ reference view để tái sử dụng, tránh gọi findViewById nhiều lần.

---

7. Adapter trong RecyclerView làm gì?

Đáp án: Tạo ViewHolder, bind dữ liệu và quản lý số lượng item.

---

8. ViewBinding là gì?

Đáp án: Cơ chế truy cập view type-safe, không cần findViewById.

---

9. DataBinding là gì?

Đáp án: Cho phép binding data trực tiếp trong XML bằng expression (@{}), hỗ trợ MVVM.

---

10. LayoutInflater dùng để làm gì?

Đáp án: Dùng để "inflate" XML thành View object.

---

11. ItemDecoration trong RecyclerView là gì?

Đáp án: Dùng để vẽ divider, khoảng cách giữa các item.

---

12. DiffUtil dùng để làm gì?

Đáp án: Tính sự khác nhau giữa 2 list để update RecyclerView hiệu quả.

---

13. ListAdapter là gì?

Đáp án: Adapter có tích hợp DiffUtil tự động → update list mượt, hiện đại.

---

14. onCreateViewHolder() dùng khi nào?

Đáp án: Khi RecyclerView cần tạo ViewHolder mới.

---

15. onBindViewHolder() dùng khi nào?

Đáp án: Khi RecyclerView muốn bind data vào ViewHolder tương ứng.

---

16. GridLayoutManager là gì?

Đáp án: LayoutManager hiển thị item dạng lưới (grid).

---

17. Khi nào cần custom view?

Đáp án: Khi UI cần vẽ bằng Canvas hoặc logic layout đặc biệt.

---

18. onMeasure() làm gì?

Đáp án: Xác định kích thước view (width, height).

---

19. onDraw() làm gì?

Đáp án: Vẽ UI lên canvas.

---

20. Tại sao dùng RecyclerView thay vì ListView?

Đáp án: Tối ưu hơn, hỗ trợ ViewHolder, animation, DiffUtil, layout manager linh hoạt.

🟠 PHẦN 2 — MỨC TRUNG BÌNH (Fresher → Junior)

21. Ưu điểm của ConstraintLayout so với RelativeLayout?

Đáp án:

- Giảm nested layout
- Hỗ trợ chain, barrier, guideline
- Hiệu năng tốt hơn

---

22. Chain trong ConstraintLayout là gì?

Đáp án: Một nhóm view được liên kết theo trục ngang/dọc, quản lý spacing giữa chúng.

---

23. Guideline dùng làm gì?

Đáp án: Đường tham chiếu cố định để căn UI linh hoạt giữa các màn hình.

---

24. Barrier dùng để làm gì?

Đáp án: Tự động dời view dựa trên kích thước của view khác → UI tự co giãn đẹp.

---

25. RecyclerView có lưu state khi scroll không?

Đáp án: Có, thông qua LayoutManager.

---

26. Sự khác nhau giữa ViewBinding và DataBinding?

Đáp án:

- ViewBinding: chỉ bind view → nhanh, nhẹ
- DataBinding: bind logic XML → mạnh cho MVVM, nhưng build chậm

---

27. Khi nào nên dùng DataBinding?

Đáp án:

- Dự án dùng MVVM
- Cần binding trực tiếp từ ViewModel vào XML

---

28. setHasStableIds(true) có tác dụng gì?

Đáp án: Giúp RecyclerView xác định item theo ID → animation mượt hơn, tối ưu update.

---

29. Sự khác nhau giữa notifyDataSetChanged() và DiffUtil?

Đáp án:

- notifyDataSetChanged(): reload toàn bộ → kém hiệu năng
- DiffUtil: tính toán sự thay đổi → chỉ update phần cần thiết

---

30. Làm sao tối ưu hiệu năng RecyclerView?

Đáp án:

- DiffUtil / ListAdapter
- Tránh nested layout
- Dùng ConstraintLayout
- setHasFixedSize(true)
- Pagination

---

31. LayoutManager có những loại nào?

Đáp án:

- LinearLayoutManager
- GridLayoutManager
- StaggeredGridLayoutManager

---

32. Tại sao không nên đặt RecyclerView trong ScrollView?

Đáp án: Vì RecyclerView đã tự handle scroll → lồng nhau gây lag, đo đạc sai.

---

33. Hai cách register click của item trong RecyclerView?

Đáp án:

- Lắng nghe click trong ViewHolder
- Callback về Adapter

---

34. Two-way binding trong DataBinding là gì?

Đáp án: Binding 2 chiều với `@={}`, UI và data cập nhật lẫn nhau.

---

35. Custom attribute trong DataBinding là gì?

Đáp án: BindingAdapter: tạo attribute mới cho XML.

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

36. Constraint bias hoạt động thế nào?

Đáp án: Bias thay đổi vị trí view theo tỉ lệ giữa 2 constraint (0.0 → 1.0).

---

37. So sánh Weight (LinearLayout) và Bias (ConstraintLayout)?

Đáp án:

- Weight: chia không gian khi orientation linear
- Bias: dịch chuyển giữa 2 constraint nhưng không thay đổi kích thước

---

38. ConstraintLayout khi không đặt đủ constraint sẽ ra sao?

Đáp án: Layout bị lệch, collapse hoặc crash trong runtime (Ambiguous constraints).

---

39. Tại sao ListAdapter là chuẩn hiện đại?

Đáp án: Vì có DiffUtil tích hợp → update list chính xác và nhanh.

---

40. Điều gì xảy ra nếu onBindViewHolder tạo listener mới mỗi lần bind?

Đáp án:

- Tốn memory
- Có thể gây duplicate event
- ViewHolder reuse → listener bị chồng chéo

---

41. Sự khác nhau giữa viewType và itemViewType?

Đáp án: Cho phép RecyclerView hiển thị nhiều loại item khác nhau.

---

42. ViewType dùng trong trường hợp nào?

Đáp án: Chat app (left/right bubble), list header/footer, multiple sections.

---

43. Giải thích cơ chế RecyclerView pool?

Đáp án: RecyclerView lưu cache ViewHolder để tái sử dụng giữa các màn hình.

---

44. Tại sao onMeasure quan trọng trong custom view?

Đáp án: Vì nếu tính sai size → UI bị méo, crash hoặc không hiển thị.

---

45. Tại sao override onDraw có thể ảnh hưởng hiệu năng?

Đáp án: Vì vẽ nhiều path/bitmap mỗi frame → gây lag → phải tối ưu canvas & paint.

---

46. invalidate() khác postInvalidate()?

Đáp án:

- invalidate(): gọi từ main thread
- postInvalidate(): gọi từ background thread

---

47. RecyclerView DiffUtil không hoạt động đúng, nguyên nhân?

Đáp án:

- areItemsTheSame / areContentsTheSame viết sai
- ID item trùng
- List cũ/new giống reference (không copy list)

---

48. Tại sao nên dùng ViewHolder pattern trong ListView?

Đáp án: Vì ListView không có ViewHolder built-in như RecyclerView → tránh lag khi scroll.

---

49. Vì sao DataBinding bị build chậm?

Đáp án: Vì compiler phải xử lý binding expression và generate code cho các XML.

---

50. CustomView cần tối ưu gì khi vẽ bitmap?

Đáp án:

- Cache bitmap
- Dùng BitmapShader
- Tránh decode mỗi lần onDraw

● PHẦN 4 — HARD QUESTIONS (trả lời tốt = được đánh giá rất cao)

---

51. ConstraintLayout mất performance khi nào?

Đáp án:

- Quá nhiều constraint phức tạp
- Dùng chain + guideline + barrier cùng lúc
- deep nested group

---

52. Khi nào nên dùng StaggeredGridLayoutManager?

Đáp án: Khi item có chiều cao không đồng đều (Pinterest layout).

---

53. Tại sao RecyclerView cần LayoutManager?

Đáp án: Separation of concerns: LayoutManager quyết định hình dạng list, Adapter chỉ lo data.

---

54. Giải thích cơ chế DispatchDraw trong ViewGroup?

Đáp án: DispatchDraw quyết định vẽ các child view → quan trọng trong custom ViewGroup.

---

55. Vì sao DataBinding sinh lỗi runtime khó debug?

Đáp án: Vì lỗi xảy ra trong XML expression → khó trace stacktrace.

---

56. ViewBinding có thể dùng chung với DataBinding không?

Đáp án: Có, nhưng không cho cùng XML.

---

57. Khác nhau giữa invalidate() và requestLayout()?

Đáp án:

- invalidate(): redraw
- requestLayout(): đo lại size + vẽ lại (tốn hơn)

---

58. CustomViewGroup cần override phương thức nào?

Đáp án:

- onMeasure()
- onLayout()
- (có thể) onDraw()

---

59. Làm sao render 60 FPS trong CustomView?

Đáp án:

- Cache paint
- Tránh allocation object trong onDraw
- Dùng Canvas clip
- Giảm overdraw

---

60. Làm sao debug layout performance?

Đáp án:

- Layout Inspector
- Profile GPU Rendering
- Enable “Debug GPU Overdraw”
- Systrace
