**Câu 1:**

Có 4 loại thành phần cốt lõi:

**Activity**

Hiểu một cách đơn giản thì Activity là nền của 1 ứng dụng. Khi khởi động 1 ứng dụng Android nào đó thì bao giờ cũng có 1 main Activity được gọi, hiển thị màn hình giao diện của ứng dụng cho phép người dùng tương tác.

– Ví dụ : Một ứng dụng email có thể có 1 activity hiển thị 1 danh sách các email mới, 1 activity khác để soạn email, và 1 activity để đọc email. Mặc dù các activity này làm việc cùng nhau trong 1 ứng dụng email, nhưng mỗi chúng lại độc lập với các cái khác. Như vậy, một ứng dụng khác có thể start bất kỳ một trong các activities này ( nếu ứng dụng email cho phép điều đó ). Chẳng hạn, một ứng dụng camera có thể start activity soạn mail mới trong ứng dụng email để cho phép người dùng có thể chia sẻ một bức ảnh

**Service**

– Một service là một thành phần chạy ẩn ( runs in the background) để thực hiện các thao tác chạy lâu dài hoặc điều khiển các tiến trình. Service thì không có giao diện người dùng.

– Ví dụ : Một service phát nhạc ( không cần giao diện ) trong khi người dùng đang sử dụng một ứng dụng khác, hoặc nó trao đổi dữ liệu qua mạng mà không làm ngăn cản sự tương tác của người dùng với một activity. Một thành phần khác, chẳng hạn như một activity, có thể start service và tương tác với nó.

– Một service được implement như là một subclass của lớp Service

**Broadcast Receiver**

 Một broadcast receiver là một thành phần phản ứng với các thông báo từ bên ngoài gửi tới hệ thống. Nhiều broadcasts có nguồn gốc từ hệ thống – chẳng hạn, một broadcast thông báo các màn hình đã tắt, pin yếu, hoặc chụp hình. Các ứng dụng cũng có thể khởi tạo các broadcast

**Content Provider**

Một content provider quản lý một tập hợp dữ liệu chia sẻ của ứng dụng. Bạn có thể lưu trữ dữ liệu trong file hệ thống, một cơ sở dữ liệu SQLite, trên web hay bất kỳ nơi lưu trữ nào khác mà ứng dụng của bạn có thể truy cập. Thông qua content providers, các ứng dụng khác có thể truy vấn hoặc thậm chí sửa đổi dữ liệu ( nếu content provider cho phép điều đó ).

– Ví dụ, hệ thống Android cung cấp một content provider quản lý các thông tin liên hệ của người dùng. Như vậy, bất kỳ ứng dụng nào được cấp quyền truy vấn đến một phần của content provider ( chẳng hạn như ContactsContract.Data) có thể đọc và ghi thông tin về một người cụ thể.

**Câu 3:**

* Vì trong onPause(), việc thực thi execution của nó rất nhanh, và không đủ thời gian cần thiết để thực hiện các hành động save. Cũng vì lí do này mà ta không nên sử dụng onPause để lưu dữ liệu của người dùng, kết nối mạng hoặc thực thi các transaction đến database. Những công việc đó có thể không thể hoàn thành trước khi phương thức kết thúc (app terminated)
* Vì thế các công việc nặng hoặc lưu trữ dữ liệu, trạng thái nên được thực hiện trong onSaveInstanceState
* Bên cạnh đó, onSaveInstanceState() sử dụng Bundle là một parameter – nó giống như một dictionary với các cặp key value pairs (gần giống Map trong Java), nhờ đó việc truy vấn khi hồi phục dữ liệu sẽ đơn giản hơn và đảm bảo hơn (có serialize dữ liệu). Thường được dùng để lưu trữ các trạng thái UI

999

* Ngoài ra, nếu mà mở một activity từ Intent, thì hàm onCreate luôn luôn được gọi với tham số là gói Bundle. Khi ứng dụng được khôi phục, ta có thể restore lại trạng thái như lúc khởi tạo activity
* Tham số saveInstanceState sẽ null nếu là activity đó là màn hình đầu tiên của ứng dụng (entry point) hoặc nếu ứng dụng khởi tạo lại một instance mới của activity.
* Tham số saveInstanceState khác null nếu activity đã được khởi tạo và bị destroy khi orientation của thiết bị thay đổi hoặc nếu có 1 activity trước đó gọi activity khác thông qua intent và có lưu dữ liệu trong bundle

**Câu 4:**

Cursor chỉ hỗ trợ thực hiện trên 1 dòng tại 1 thời điểm ở 1 table. Mặc dù đối với các DBMS khác, cursor có thể được sử dụng để update dữ liệu nhưng trong SQLite thì cursor chỉ có thể đọc read-only

**Câu 5:**

**Handler Message**

* Sử dụng MessageQueue để quản lý các tiến trình main thread và background threads.
* Handler sẽ enqueue các messages và các runnables vào trong MessageQueue của cha nó. Những request này sẽ được thực hiện theo thứ tự và sẽ được dequeue khỏi MessageQueue
* Schedule messages và runnables sẽ được thực hiện tại một thời điểm nào đó trong tương lai
* Handler.sendMessage sẽ có thể nhẹ hơn đôi chút vì handler.post sẽ tạo ra 1 object Message mới. Khi sử dụng handler.sendMessage ta có thể add thêm dữ liệu hoặc runnables vào
* Có nhận tham số đầu vào

**Handler Post**

* Enqueue những actions sẽ được thực hiện trên một thread khác với thread hiện tại
* Handler.post tiện dụng hơn handler.sendMessage

Không nhận tham số đầu vào

**AsyncTask**

* cho phép bạn thực hiện công việc không đồng bộ trên giao diện người dùng của mình. Nó thực hiện các thao tác chặn trong một luồng trình thực hiện rồi phát hành kết quả trên luồng UI mà không yêu cầu bạn tự xử lý các luồng và/hoặc trình xử lý.

**Ưu điểm AsyncTask**

* Đơn giản, dễ sử dụng, vì là 1 class generic nên khi extend AsyncTask sẽ có 3 tham số generic đầu vào (Params, Progress, Result)
* Cung cấp 1 generic solution cho tất cả các việc thực hiện các gọi request
* Có hàm hỗ trợ update progress lên UI trong khi đang executing
* Có thể chạy bất đồng bộ
* Dễ sử dụng, dễ đọc và maintain

**Câu 5:**

**Tính năng gây tốn pin:**

* Các kết nối dữ liệu: Wifi, 3G, 4G, LTE
* Các kết nối không dây: NFC, Bluetooth
* Độ sáng màn hình cao
* Quay phim độ phân giải cao
* Các ứng dụng push notification nhiều (Email, …)

**Kỹ thuật bắt tình trạng pin yếu:**

Sử dụng Broadcast Receiver để lắng nghe Battery service từ đó có thể lấy được thông tin tình trạng pin

**Bắt được tình trạng pin yếu thì:**

* Giải phóng bộ nhớ, xóa các ứng dụng đang chạy ngầm, các ứng dụng push notification, …
* Giảm độ sáng màn hình xuống mức thấp, bật chế độ tiết kiệm điện năng, darkmode
* Tắt các kết nối không cần thiết (Wifi, Bluetooth, 3G, NFC, …)
* Nếu người dùng đang sử dụng Wifi để lướt web thì hiển thị dialog thông báo cho người dùng, nếu người dùng đồng ý thì sẽ tắt các kết nối mạng

**Start ứng dụng không cần GUI, ngay cả khi restart máy:**

* Khai báo permission RECEIVE\_BOOT\_COMPLETED trong file Manifest.xml
* Tạo một broadcast receiver để lắng nghe sự kiện đó, và chạy ứng dụng khi máy vừa khởi động
* Sử dụng Service để có thể chạy ứng dụng mà không cần hiển thị GUI