Android

1. Android là hệ điều hành mã nguồn mở dựa trên nền tảng linux và dung để phát triển ứng dụng trên các thiết bị thông minh(điện thoại, tivi,..)(version 1.0-8.0 giờ là 10)
2. Sdk: software development kit là một tập hợp công cụ hỗ trợ cho việc phát triển phần mềm thông qua một nền tảng nào đó
3. Các thành phần cơ bản : có 4 component chính :
4. Activity : là giao diện có thể nhận tương tác của người dung, là thành phần quan trong của android.
5. Service : là thành phần giúp ứng dụng có thể chạy ngầm mà không cần giao diện hiển thị với người dung
6. Content provider : quản lý các tập hợp đc chia sẻ bởi ứng dụng
7. BroadcastReceiver: cho phép nhận và gửi thông điệp trong ứng dụng, giữa các ứng dụng với nhau hoặc giữa các ứng dụng với hệ thống
8. Activity : là màn hình cho phép người dung có thể tương tác được, Foreground Task là màn hình đang sử dụng, BackgroundTask
9. Các chế độ launmode trong activity: Khi activity khai báo trong manifest bạn có thể chỉ định sẽ liên kết với các task như thế nào bằng chế độ launchmode(cũng có thể dung intent flags để khai báo intent.addFlags(Intent.FRAG\_ACTIVITY\_SINGLE\_TOP)
10. STANDARD(mặc định) : là việc xếp chồng các activity vào stack trong cùng 1 tak bất kể activity đã tồn tại trước đó hay chưa (khi run có 2 task, 1 task là màn hình)
11. SINGLE\_TOP : khá giống standard tuy nhiên nếu tồn tại 1 instance của activity ở trên đỉnh nó sẽ không tạo activity mới , hệ thống sẽ đưa intent tới instance qua lời gọi phương thức onNewIntent()
12. SINGLE\_TASK : cần thêm android:taskAffinity="" thì mới hoạt động được.

Nếu chưa tồn tại instance(phiên bản) nào của activity thì nó sẽ tạo ra 1 task mới để chứa nó. Trong task mới khi được gọi đến thì nó sẽ xóa tất cả các activity trước đó. 2 activity cùng khai báo singletask thì sẽ hoạt đồng trong cùng task mới đó

1. SINGLE\_INSTANCE: giống như single task nhưng activity có cùng chế độ nhưng sẽ tạo ra task khác nhau để chứa nó.

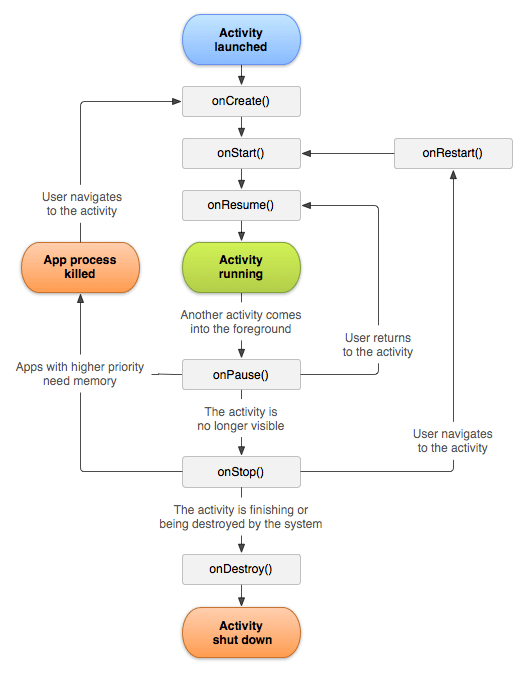
Activity

* Activities là gì? Có những cách nào truyền dữ liệu giữa các activity.
* Activity là màn hình cho phép người dùng có thể tương tác được.
* Có rất nhiều cách để chia sẻ dữ liệu giữa các activity ví dụ như:

Intent, ApplicationContext, interface,shared preference...vv

* note: muốn truyền đối tượng thì đối tượng đó phải implement serializable hoặc parcelable
* Vòng đời của activities như thế nào? Giải thích

Có 7 method chính sẽ được gọi từ khi activity bắt đầu đến kết thúc:

* OnCreate: được gọi đầu tiên, đây là nơi khởi tạo các thành phần và dữ liệu activity
* Onstart: phương thức này được gọi khi activity đang hiển thị tới người dùng và chưa thể tương tác được
* OnResume:Được gọi khi activity bắt đầu nhân tương tác với người dùng.
* OnPause: phương thức này được gọi khi system mở lại activity trước đó. Phương thức này cũng lưu dữ liệu chưa thay đổi, stop animation
* OnStop:được gọi khi activity không hiển thị cho người dùng thấy trong 1 thời gian dài.
* OnDestroy: Call khi hệ thống hủy activity.
* OnRestart:được gọi khi activity được hiển thị lại từ trạng thái stop.

Có 3 vòng lặp chính mà bạn cần nắm rõ:

**Entire lifetime**: xảy ra giữa onCreate() và onDestroy(). 1 activity sẽ cài đặt các trạng thái trong onCreate() và giải phóng toàn bộ tài nguyên trong onDestroy(). Ví dụ có 1 luồng chạy ngầm tải dữ liệu từ trên mạng, ta có thể tạo luồng tại onCreate() và kết thúc luồng tại onDestroy().

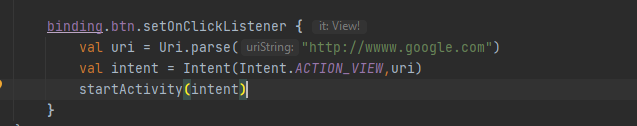
**Visible lifetime**: xảy ra giữa onStart() và onStop(). Trong giai đoạn này, người dùng có thể nhìn thấy activity trên màn hình, tuy nhiên nó không ở trên đầu ngăn xếp và không thể tương tác với người dùng. Giữa 2 phương thức này người dùng có thể lưu trữ được tài nguyên cần thiết hiển thị lên activity cho người dùng. Ví dụ ta cần quan sát thay đổi ảnh hưởng tới giao diện, thì ta có thể đặt Broadcast Receiver tại onStart() và loại bỏ tại onStop().

**Foreground lifetime**: xảy ra giữa onResume() và onPause(). Trong giai đoạn này activity ở trên đầu ngăn xếp và có thể tương tác với người dùng.

* Khi ở chế độ background quá lâu nó sẽ bị hệ thống thu hồi để giải phóng tài nguyên, khi quay lại nó sẽ nhảy vào onCreate() mà ko phải onResume() hay onRestart()

Intent

1. Explicit Intent (intent tường minh) : chỉ rõ nơi đến , cũng có dữ liệu truyền theo : startActivity, sendBroadcast,…
2. Implicit Intent(intent không tường minh) : dụng để call những activity hệ thống, có thể truyền action và dữ liệu muốn hướng đến
3. Nếu trường hành động là ACTION\_CALL, trường dữ liệu sẽ là số để gọi
4. ACTION\_VIEW , trường dữ liệu là google thì start sẽ đến trang google



Intent có thể đc dùng để gửi ý định startActivity, startservice, stopservice, sendBroadcast

. Activity : lấy thông tin intent đã nhận

.Service : thông tin intent chuyển đến phương thức onStartCommand

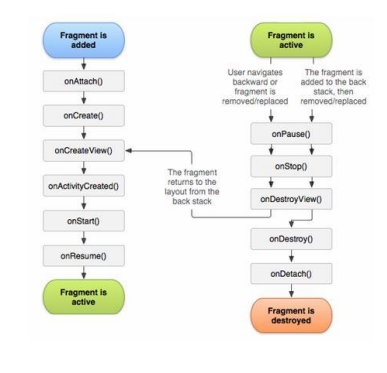
.BroadCast: thông tin intent đã nhận chuyển đến phương thức onReceive của đtg broadcast

1. INTENT FILTER : dùng để lọc các action intent (có thể lọc cả những ý định ngầm) , lọc dựa vào action , category, data
2. PENDING\_INTENT : là 1 token mà bạn đưa đến 1 ứng dụng khác , cho phép ứng dụng này sử dụng permission trong ứng dụng của bạn để thực hiện chức năng nào đó( VD dung trong notification , khi click vào notification sẽ start activity mà ta tham chiếu đến

Fragment

1. FLAGMENT: là 1 phần của giao diện người dung, có thể có nhiều fragment trong activity,vòng đời fragment ảnh hưởng đến activity chứa nó

Cách truyền dữ liệu trong fragment : setArgument, sharepreferren



onAttach(): đc gọi khi fragment đã liên kết với activity

onCreate() : gọi sau phương thức onCreate của activity

onCreateView(): cấu trúc fragment

onActivityCreated() : đc gọi khi phương thức onCreate của hoạt động đã trả về

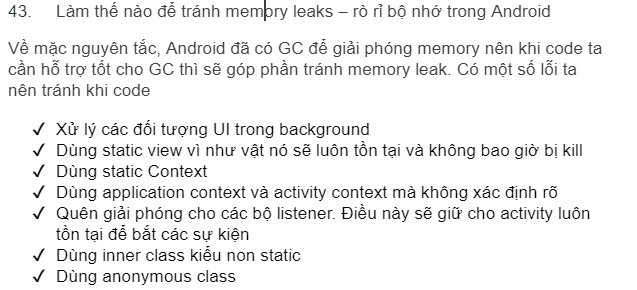
…tương tự activity

onDetach(): gọi khi fragment tách ra khỏi hoạt động

ARN

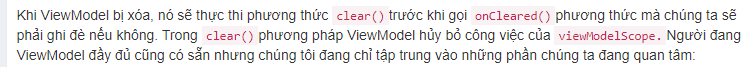
* ANR viết tắt của Application Not Responding. Nếu ứng dụng thực hiện quá nhiều task trên main thread và không phản hồi trong một thời gian dài(>=5s), Android system sẽ hiển thị là dialog này.

Memory leak



Coroutine

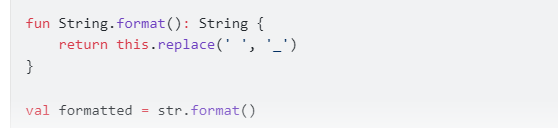
* Viewmodel scope:



* Job : nắm vòng đời của coroutine
* Mỗi khi launch một coroutine nó trả về đối tượng job :
* Job.join : đợi chạy xong task mới chạy tiếp
* Job.cancel : Hủy 1 coroutine
* Async : Async launch coroutine sẽ trả về kiểu Int, String, Unit, .., khai báo async kiểu lazy thì coroutine ko chạy ngay, nó phải chờ code lệnh start
* Dispatcher : Quyết định thread mà mà coroutine chạy trên đó :
* Dispatcher.Main : chạy trên UI Thread, update UI
* Dispatcher.IO : chạy trên background Thread của Thread pool : thường được dung khi đọc viết file, viết file, database, networking (có thể dung thay suspend fun trong database)
* Dispatcher.Default : Thương được dung trong sorting a list, parse json, kiểu như logic , tính toán
* Dispatcher.Unconfined : ko giới hạn trên thread nào
* Suspend fun : chạy đồng thời, chỉ khai báo được trong coroutine
* GlobalScope : Chạy liên tục không giới hạn
* lifecycle Scope, Viewmodel Scope: chạy đến khi nào tạo coroutine tiếp theo
* withContext : get Data bằng dispatcher.IO, nhưng update UI bằng Thread Main

So sánh Kotlin và Java

* Kotlin :
* Tự động ép kiểu khi khai báo biến
* Gọi parameter gán giá trị trực tiếp, ko cần theo thứ tự
* Sử dụng when linh hoạt hơn
* Data Class: Tự động override các phương thức equals(), tostring(),hashCode()
* Extension Function : định nghĩa các phương thức ko cần extent, sử dụng từ khóa this bên trong tương ứng với đối tượng



* Lambda function : fun ko tên
* Infix Function : fun là 1 thành phần của 1 class hoặc 1 extent của class, có 1 parameter duy nhất

Null Safety

* Null safety: kiểm soát lỗi null pointer exception

var str2 : Sting? = null

* ? : cho phép gán giá trị null cho biến
* Str2?.length : nếu str2 ko null thì trả về length còn null thì return về null
* Str2!!length : ném nullpointe exception nếu giá trị null return về hàm đó
* Val length = str2?.length ?: -1 : toán tử elvis nếu str2 ko null trả về str2.length nếu null trả về -1
* Ưu điểm kotlin : ngắn gọn, giảm thiểu lỗi code
* Ưu điểm java:
* dễ đọc , dễ hiểu
* Tài liệu tự học nhiều hơn hăn kotlin
* Có thể build ứng dụng nhanh hơn

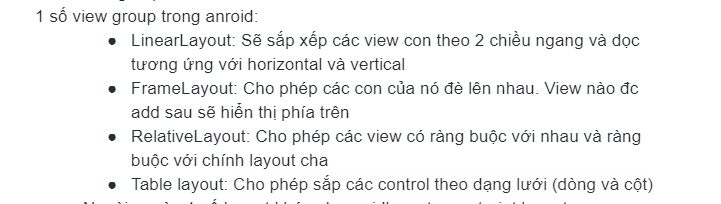
Scope Function

* Apply , run, with : this
* Also, let : it
* Apply, also: chỉ return object
* Let, run, with : return bất cứ điều gì
* Apply, run, also,let : nó là phần nối thêm tham chiếu đến giá trị của đối tượng
* With như 1 phg thức chứa đối tượng
* Dùng let để check null tốt nhất
* Có thể convert it thành tên khác rõ ràng hơn

Kiến trúc android

* Android gồm 4 tầng chính : chia làm 5 phần :
* Nhân Linux: Đây là nhân nền tảng mà hệ điều hành Android dựa vào nó để phát triển. Đâu là lớp chứa tất cả các thiết bị giao tiếp ở mức thấp dùng để điều khiển các phần cứng khác trên thiết bị Android.
* Thư viện: Chứa tất cả các mã cái mà cung cấp cấp những tính năng chính của hệ điều hành Android, đôi với ví dụ này thì SQLite là thư viện cung cấp việc hỗ trợ làm việc với database dùng để chứa dữ liệu. Hoặc Webkit là thư viện cung cấp những tính năng cho trình duyệt Web.
* Android runtime: Là tầng cùng với lớp thư viện Android runtime cung cấp một tập các thư viện cốt lõi để cho phép các lập trình viên phát triển viết ứng dụng bằng việc sử dụng ngôn ngữ lập trình Java. Android Runtime bao gồm máy ảo Dalvik(ở các version < 4.4, hiền tài là phiên bản máy ảo ART được cho là mạnh mẽ hơn trong việc xử lý biên dịch). Là cái để điều khiển mọi hoạt động của ứng dụng Android chạy trên nó(máy ảo Dalvik sẽ biên dịch ứng dụng để nó có thể chạy(thực thi) được , tương tự như các ứng dụng được biên dịch trên máy ảo Java vậy). Ngoài ra máy ảo còn giúp tối ưu năng lượng pin cũng như CPU của thiết bị Android
* Android framework: Là phần thể hiện các khả năng khác nhau của Android(kết nối, thông báo, truy xuất dữ liệu) cho nhà phát triển ứng dụng, chúng có thể được tạo ra để sử dụng trong các ứng dụng của họ.
* Application: Tầng ứng dụng là tầng bạn có thể tìm thấy chuyển các thiết bị Android như Contact, trình duyệt…Và mọi ứng dụng bạn viết đều nằm trên tầng này.

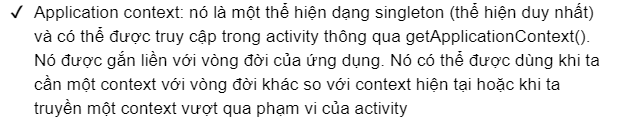
ViewGroup

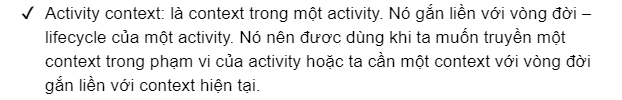


Các View: Textview,…

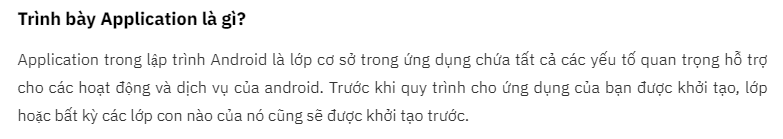
Context

Context là bối cánh của tình trạng hiện tại. thông thường ta gọi nó để lấy thông tin liên quan đến phần khác của ứng dụng(activity và package) có những loại sau :





Application

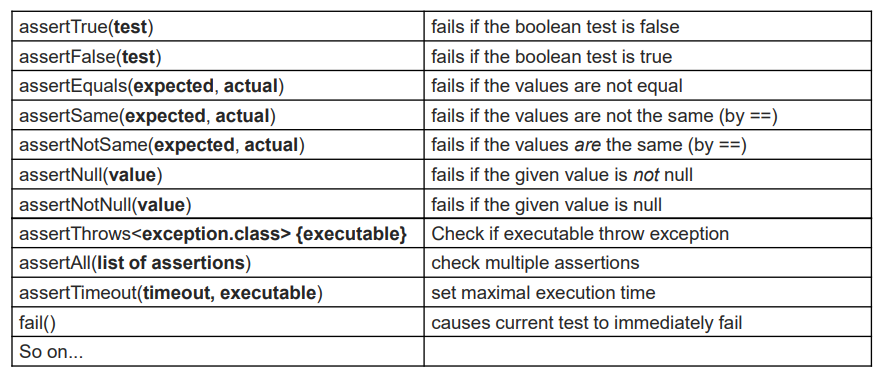


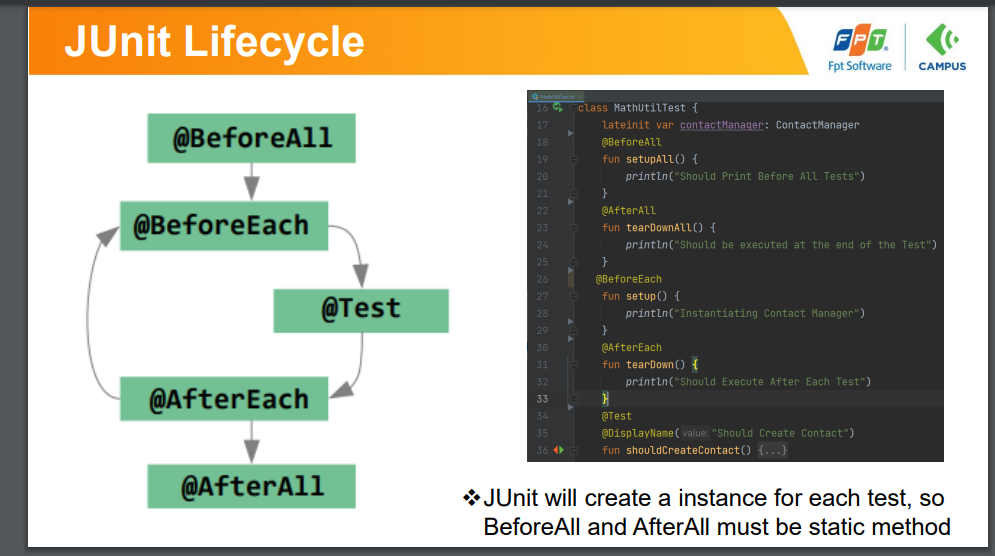
Unit Test

* Local Unit test: Framework junit5

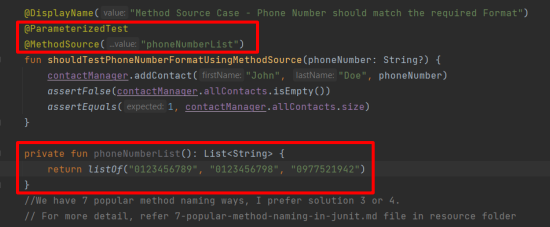
1. Viết và test nhanh
2. có thể test lặp lại
3. có test pass và test fail(hiển thị đỏ và xanh)
4. chạy đồng thời nhiều bài test

* function để test là có trả về
* function test : ko có tham số
* @DisplayName: như kiểu cmt hiện tên trong luk run
* @DisapledonOs(value = [OS.MAC])
* @RepeatTest: Test 5 lần (cho value = 5)
* Phương thức xác nhận :

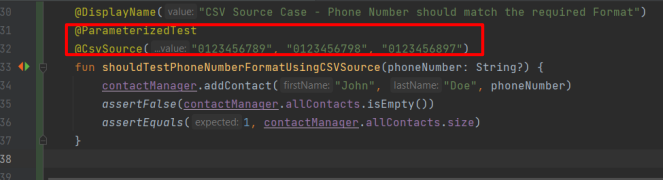




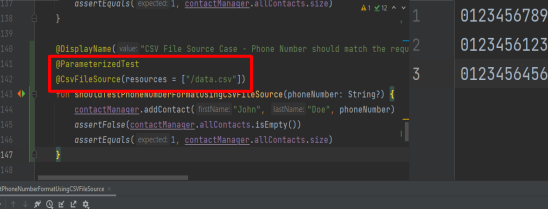
* MothodSource



* Csv Source



* Csv File



Design Patterns

* Nhóm khởi tạo
* Nhóm cấu trúc
* Nhóm hành vi

