onCreate()

زمانی که سیستم اولین بار activity رو create میکنه . در این method ترجیحا باید منطق اولیه راه اندازی برنامه که فقط یک بار در طول حیاط یه activity بهش نیاز داریم رو قرار بدید . مثلا یک viewModel رو به activity بخواید provide کنید یا بایند کردن دیتا به لیست یا بایند کردن layout و ممبر هاش به activity . این متود پارامتر savedInstanceState رو دریافت میکنه که یک باندل آبجکت هستش و حاوی استیت قبلی اکتیویتیمون میشه

onStart()

این متود فراخاونی شود اکتیویتی قابل روئیت میشه برای کاربر . برنامه اکتیویتی رو آماده میکنه برای ورود به foreground و تعامل داشتن و بسیار سریع جریان داره این متد و به اتمام میرسه .

onResume()

زمانی که اکتیویتی وارد این متود میشه ینی وارد foreground هم میشه . این کالبک به معنیه اینه که اکتیویتی آماده تعامل با کاربر هستش . برنامه در این استیت میمونه تا زمانی که یه اتفاقی بیفته مثلا تماس تلفنی دریافت کنید یا یوزر به یه اکتیویتی دیگه حرکت کنه یا دیوایس اسکرینت خاموش بشه . اگر اکتیویتی از متود Paused به متد Resumed برگشت سیستم یک بار دیگه onResume() رو صدا میزنه و اجرا میشه

If you initialize something after the ON\_START event, release or terminate it after the ON\_STOP event. If you initialize after the ON\_RESUME event, release after the ON\_PAUSE event.  
  
onPause()

این متو زمانی فراخونی میشه که کاربر اکتیویتی رو ترک کرده که به همیشه این معنی رو نمیده اکتیویتی destroy شده باشه . این نشون میده اکتیویتی توی foreground نیستش (though it may still be visible if the user is in multi-window mode) . زمان multi-window mode اکتیویتی تو متد pause هستش و ویزیبل هم هستش .  
چه زمانی پاوس میشه اکتیویتی

-Some event interrupts app execution, as described in the onResume() section. This is the most common case.

-In Android 7.0 (API level 24) or higher, multiple apps run in multi-window mode. Because only one of the apps (windows) has focus at any time, the system pauses all of the other apps.

-A new, semi-transparent activity (such as a dialog) opens. As long as the activity is still partially visible but not in focus, it remains paused.

اجرای این متد بسیار کوتاه هستش و زمان مناسب برای ذخیره دیتاهای اپلیکیشن و یوزر وجود نداره یا نتوورک کال و یا کار با دیتا بیس .. برای از بین بردن heavy-load باید از onStop استفاده کرد   
این اکتیویتی در این استیت میمونه تا زمانی که اکتیویتی resume بشه یا کاملا invisible بشه .

اگر اکتیویتی از حالت Paused برگرده به حالت resume سیستم نمونه ای که ایجاد شده رو تو مموری نگه میداره و اون رو بازیابی میکنه تو این سنیاریو احتیاجی نیست اون کامپوننتایی که قبلا ایجاد کردید رو که منتهی میشن به resume رو مجدد ایجاد کنید . اگر اکتیویتی کاملا invisible بشه سیستم onStop() رو صدا میزنه

onStop()

زمانی که اکتیوتی مدتی به یوزر نمایش داده نشه . یه اکتیوتی جدید لانچ میشه و تمام اسکرین مارو کاور میکنه بعش سیستم onstop رو صدا میزنه . تو این متود برنامه قابل مشاهده نیست برای کاربر وما باید منابعی که احتیاج نداریم رو از بین ببریم یا تنظیمات لازم رو انجام بدیم مثل انیمیشن . توی این متود شما میتونید کارهای مرتبط با ui رو ادامه بدید و مطمئن باشید که ادامه پیدا میکنه و مشکلی براش پیش نمیاد حتی زمانی که اپلیکیشن رو حالت multi-window هستش و همچنین برای به پایان رسوندن و انجام کار ها CPU-intensive هستش مثلا اگر زمان مناسبی برای سیو کردن دیتا تو دیتابیس ندارید میتونی تو این متود این کارو انجام بدید . زمانی که اکتیویتی وارد این متود بشه اکتیویتی آبجکت در مموری باقی میمونه و تمام استیت ها و اطلاعات اعضا رو نگه داری میکنه . همچنین سیستم مسیر فعلی هرکدوم از استیت های ویو داخل layout رو نگه میداره پس اگر یوزر مقداری رو در ویجت ادیتتکست وارد کنه اون محتوا حفظ میشه و نیازی به ذخیره کردن و بازیابیش نیست .

نکته : سیستم ممکنه proccess رو destroy بکنه حتی زمانی که سیستم این کار رو انجام بده زمانی که اکتیویتی تو onStop هستش همچنان سیستم میتونه استیت هارو بازیابی کنه مثله مقادیر وارد شده در ادیت تکست البته اگر یوزر برگرده به اکتیوتی (اطلاعات بیشتر در ادامه این مقاله)

از onStop یا اکتیویتی برمیگرده برای تعامل با کاربر یا دیستروی میشه و فعالیتش متوقف میشه . اگر اکتیوتی برگرده سیستم onRestart() رو صدا میزنه در غیر اون صورت ondestroy()

CPU-intensive =< رفتار پیچیده ی کاربر که حافظه بیشتری درگیر میشود

onRestart()

بعد از onStop فراخونی میشه زمانی که بخوای اکتیوتی مجدد نمایش داده بشه به یوزر (یوزر برگشته به اکتیویتی) و بعد onStart() و بعدش onResume() صدا زده میشه

onDestroy()

قبل از از بین رفتن کامل اکتیویتی این متود فراخونی میشه و این متون به 2 دلیل فراخونی میشه

1- اکتیویتی به اتمام رسیده (بخاطر اینکه یوزر کاملا dismissing کرده یا finish() رو فراخونی کرده)

2- سیستم موقت اکتیویتی رو destroy بکنه بخاطر اینکه کانفیگ تغییر کرده مثل چرخش اکتیویتی یا multi-window

اینجا باید تمام ریسورس هایی که تو onStop متوقف و پاک سازی نکردی رو پاک سازی کنی .  
  
نکته : بجای اینکه لاجیک برنامه رو توی اکتیویتی بنویسی اون رو توی viewmodel پیاده سازی کنید تا از هندل کردن بی مورد دیتا و نوشتن کد اضافه جلوگیری کنید اگر اکتیوی مجدد create بشه بخاطر تغییر کانفیگ برنامه در اون صورت viewModel مجبور به انجام کاره اضافه نیست تا نموه ای از دیتا رو به اکتیوی بعدی پاس بده اگر اکتیویتی مجدد create نشه viewmodel متود oncleared() رو داره که ثبل از destroy شدن میتونه دیتارو پاک کنه

Activity state and ejection from memory

زمانی که سیستم بخواد رم رو آزاد کنه proccess رو از بین میبره . احتمال از بین از بین بردن proccess بستگی به وضعیت اکتیویتی و وضعیت proccess داره احتمال از بین بردن کمو زیاد میکنه .  
  
در onCreate , onStart , onResume وضعیت proccess در حالت foreground(هستش یا داره اماده میشه برای این کار ) و احتمال ازبین رفتن proccess کمه  
در onPause Background (lost focus) احتمالش زیاد   
در onStop() Background (not visible) خیلی زیاده  
Destroyed Empty

سیستم هرگز یک اکتیویتی رو مسقیم از بین نمیبره بلکه فقط پراسس رو از بین میبره و تاثیر میزاره رو اکتیویتی و خوده یوزر هم میتونه با استفاده از تنظیمات اپلیکیشن منیجر یک proccess رو از بین ببره .

Saving and restoring transient UI state

یوزر توقع داره اکتیوتی استیت های UI رو یادش بمونه وقتی تغییر کانفیگ اتفاق میفته مثل چرخش تصویر یا رفتن به حالت multi-window . سیستم destroy میکنه اکتیویتی رو زمانی که کانفیگ عوض میشه و تمام استیت های UI از بین میره داخل اون instance . شما برای ذخیره سازی استیت UI از لوکال استورج یا onSaveInstanceState یا viewModel استفاده کنید . اگر دیتای UI سبک هستش مثل primitive data type یا آبجکت ساده مثل string شما میتونید از onSaveInstanceState() استفاده کنید البته این مورد فراموش نکنید هزینه ی serialization/deserialization رو زیاد میکنه چون مدل کارکردش اینه و بهتره از روش های دیگه هم استفاده بشه .

البته بعضی از سناریو ها هستش که نیازی به هندل کردن نداره چون خوده یوزر با اختیار خودش داره اکتیویتی رو destroy میکنه و متود finish رو کال میکنه تا اکتیوتی بره و نیازی بهش نداره تو این موارد نیازی به هندل کردن این قضیه ندارید .

به صورت پیش فرض سیستم از Bundle برای ذخیره کردن استیت در View استفاده میکنه یعنی اگر اکتیویتی destroy و recreat شد بدون نوشتن کد از طرف شما استیت ذخیره شده هستن .

یک باندل آبجکت اماه نشده برای ذخیره کردن چندین دیتای ناچیز چون احتیاج داره روی مین ترد serialization انجام بده و باعث مصرف system-process memory میشه برای دیتا های بزرگ تر نیاز به استفاده از storage, the onSaveInstanceState() method, and the ViewModel class دارید .

onSaveInstanceState صدا زده نمیشه وقتی که یوزر به صراحت اکتیویتی رو از بین میبره یعنی دکمه بک رو میزنه و finish روفراخونی میکنه

شما میتونی از onRestoreInstanceState استفاده کنید بجای onSaveInstanceState چون بعد از callback onstart این متود فراخونی میشه و تنها زمانی فراخونی میشه که دیتا برای بازیابی داشته باشه .