Может, хватит уже о жизненном цикле?

- Нет, не хватит!

Вячеслав Третьяк

GROCON

GRODNO 2018

Я

- Разработчик–многостаночник
- Программирую под Android с 2009
- 4 игры в маркете (почти ☺)
- Более 15 приложений различной сложности по работе
- Всё о чём буду рассказывать — пережил сам или падаваны

P. S. Вопросы — после доклада.

Капитан очевидность спешит на помощь

- 40% — Официальная документация
  - Технически — всё есть
  - Практически — тонкие нюансы упущены
- 30% — Информация из интернета
  - Всё разжёвано
  - Основа этого доклада
- 20% — Исходные коды Android Framework
  - Чтоб убедиться на 100%
- 10% — Лично пережитая боль

Спасибо за внимание

Нет, давайте всё–таки поговорим

- Уже 2018 год, первый релиз Android был 10 лет назад
- Много подходов, архитектур, библиотек
- Много информации (книги, интернет, курсы)
- Но приложения по прежнему не всегда работают
- Иногда даже крэшатся
- Ошибаются даже очень опытные и высококлассные разработчики
- Проблема *кажется* настолько простой, что никто не считает её за проблему

Этот доклад не про

- Архитектуру
  - Clean architecture, MVC, MVP, MVVM, MVI, VIPER, …
- Android Architecture Components
  - android.arch.lifecycle, LiveData, ViewModel, …
- Внедрение зависимостей
  - Dagger, Koin, …
- Работу с асинхронностью
  - RxJava, Bolts, Event Bus, …
- Kotlin *(хотя примеры кода на нём)*
- Тестирование
- Ничего про модные и хайповые подходы, технологии и библиотеки

А про

- ...

А про

- ...
- ...

А про

- …
- …
- Потяну интригу

А про

- …
- …
- Потяну интригу
- …

А про

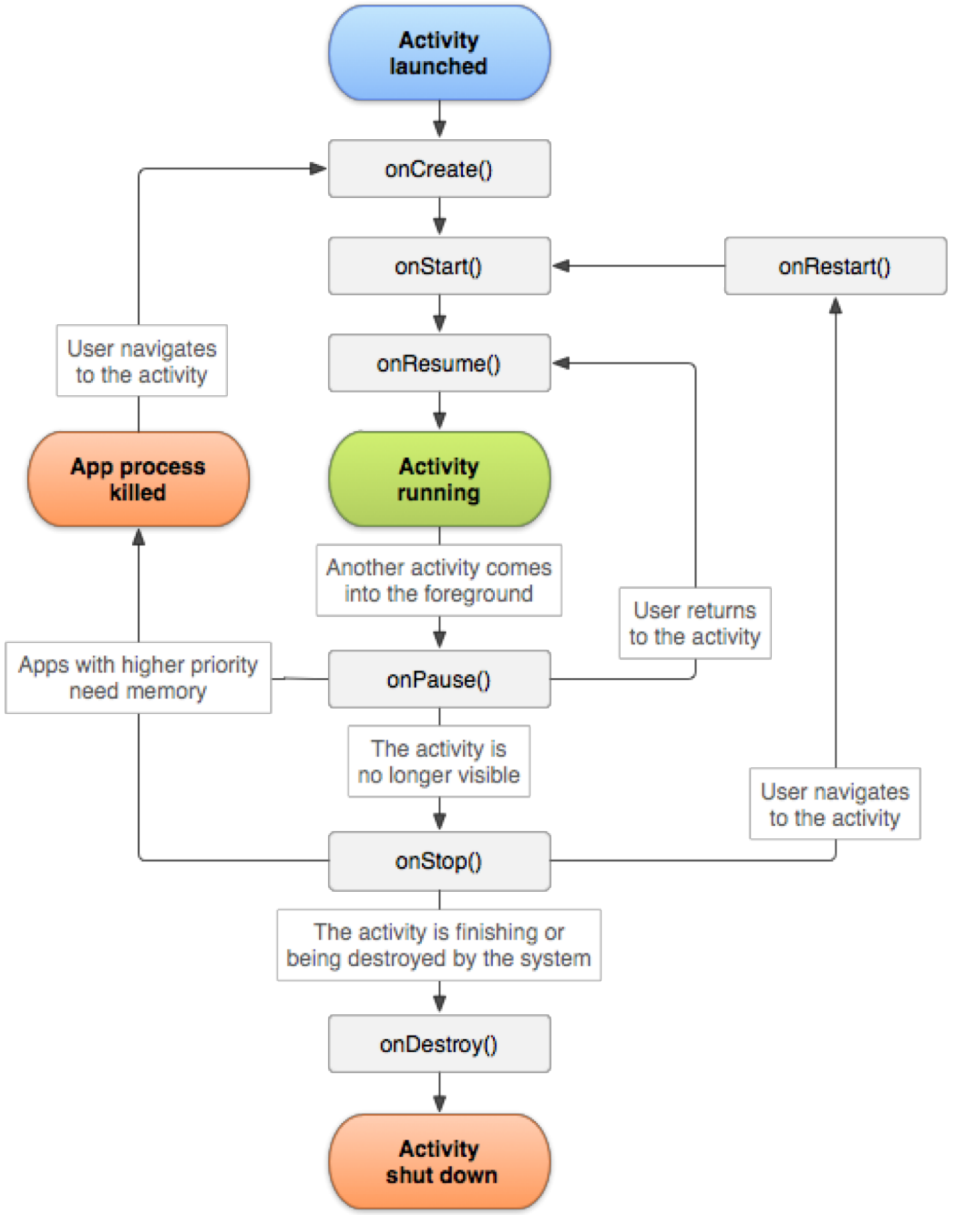
- …
- …
- Потяну интригу
- …
- onCreate

А про

- …
- …
- Потяну интригу
- …
- onCreate
- onStart

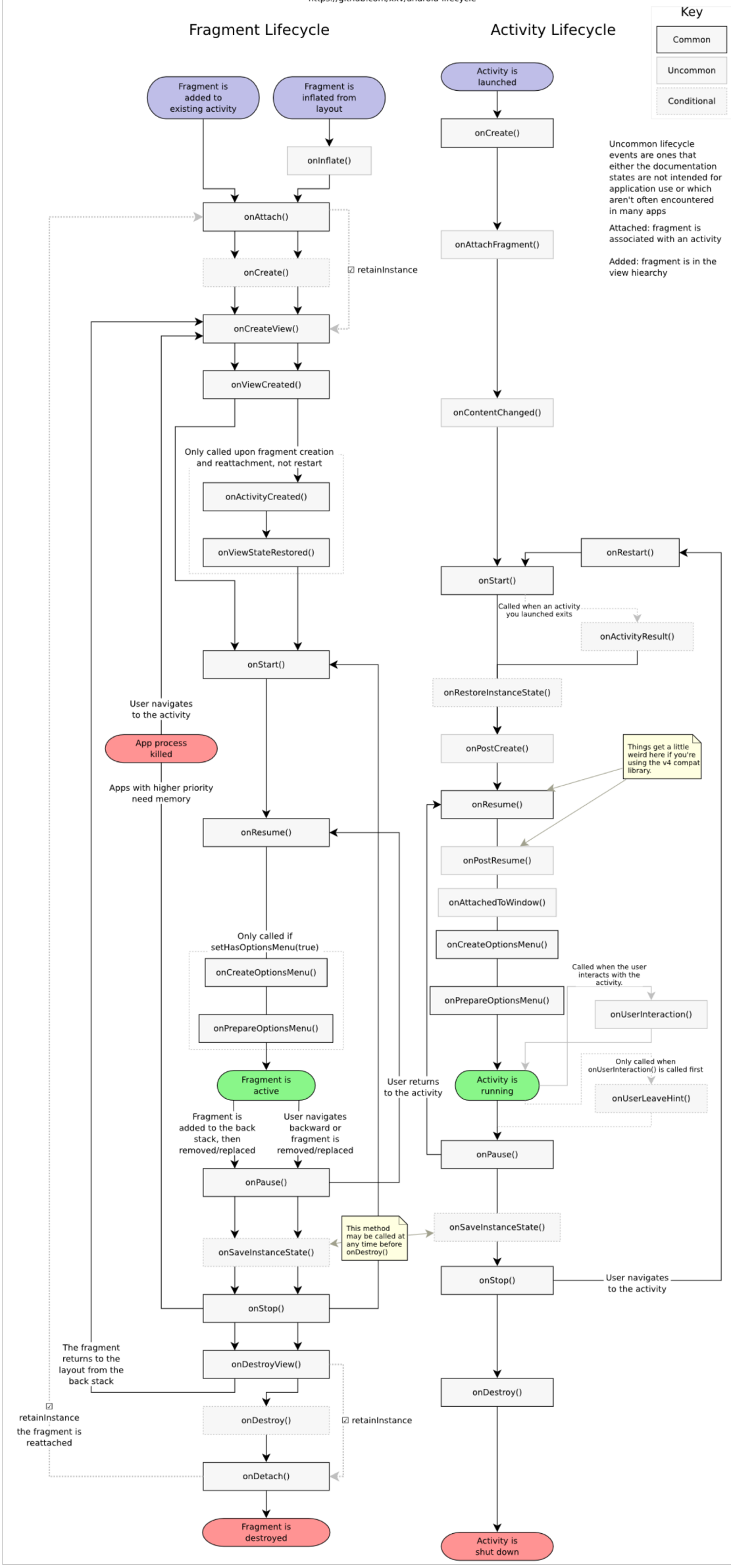
А про

- …
- …
- Потяну интригу
- …
- onCreate
- onStart
- onResume
- …
- Всё то, что начинающий разработчик под Android учит на своём втором дне обучения


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

The Complete Android Activity/Fragment Lifecycle

v0.9.0 2014-04-22 Steve Pomeroy <stevep@thelevelup.com>
CC-BY-SA 4.0
https://github.com/xxv/android-lifecycle



Activity

Activity.onCreate

- НЕТ — listener–ы (setOnClickListener)
- НЕТ — Подписываться на потоки, event bus или broadcast-ы
- НЕТ — Запускать асинхронные задачи
- НЕТ — Получать доступ к ресурсам (камера)
- НЕТ — Обновлять UI и изменять состояние приложения
- НЕТ — Работать с фрагментами
  - Хоть это и практикуется самими гуглерами
  - *Информация детально не проверена, но лучше перестраховаться*

** P. S. Хотя во многих туториалах вот всё это и предлагают делать в onCreate*

Activity.onCreate

- После onCreate — Activity не обязательно доступна пользователю
- Простой способ — finish в onCreate
- Опасаюсь, что есть другие варианты (внешние факторы)
  - *Информация детально не проверена, но лучше перестраховаться*
- Следовать рекомендациям — код более целостный и последовательный
  - Даже если finish гарантированно не вызывается, и внешние факторы не влияют
- Исключение — «Launcher Activity»

Activity.onCreate

- Аналог конструктора для обычных классов
- ДА — Инициализация
- ДА — Внедрение зависимостей

- ДА — setContentView
- ДА — findViewById
- ДА — Брать нужные параметры из extras
- ДА — Восстановление instance state (*но не обновлять UI*)
  - В переменки
  - Т.е. если на старте экрана надо один раз показать welcome–диалог, то **не** нужно сразу показывать его в onCreate

```
override fun onCreate(savedInstanceState : Bundle?) {
    super.onCreate(savedInstanceState)

    App.self.component
        .plus(SomeModule(this)).inject(this)

    setContentView(R.layout.some_layout)
    someView = findViewById(R.id.some_view)

    if (savedInstanceState != null) {
        isDialogAlreadyShown = savedInstanceState
            .getBoolean(KEY_DIALOG_SHOWN)
    }
}
```

Activity.onRestart

- Не нужно
- В 99% случаев

Activity.onStart

- Activity готова ко взаимодействию, и, может, даже видима (мультиоконность)
- Основной «фарш»

- ДА — setOnClickListener (и прочие listener–ы)
- ДА — Обновление UI из последнего состояния
- ДА — Подписки на потоки, event bus и broadcast-ы
- ДА — Запускать асинхронные задачи
- ДА — Изменение состояния приложения

Activity.onStart

- НЕТ — Запуск анимаций
  - В 99% случаев
- НЕТ — Захват эксклюзивных ресурсов (камера)
  - В 90% случаев
- НЕТ — Запуск видео
  - В 75% случаев
- НЕТ — Запуск аудио
  - В 75% случаев

** Подробней — через пару слайдов*

```
override fun onStart() {
    super.onStart()

    someView.setOnClickListener {
        doSomething()
    }

    someView.text = someState.someText
    subscription = someManager.observable.subscribe(...)

    if (!isDialogAlreadyShown) {
        isDialogAlreadyShown = false
        showSomeDialog()
    }
}
```

Activity.onResume

- Activity видима, и с ней можно взаимодействовать
  - *Есть нюанс в виде window focus*

- ДА — Всё, что ещё не сделано на onStart
  - Запуск аудио
  - Запуск видео
  - Захват эксклюзивных ресурсов (камера)
  - Запуск анимаций

```
override fun onResume() {
    super.onResume()

    fotoapparat = Fotoapparat.with(this)
        .into(cameraView)
        .lensPosition(front())
        .build()

    fotoapparat.start()
}
```

onStart VS onResume

- Есть мультиоконность
- Раньше — приложение на экране ⇔ единственное и resumed
- Сейчас — приложение на экране ⇔ started, но не обязательно resumed

- onResume / onPause — приложение «застывает» при переходе к другому
- onStart / onStop — продолжает работать
- Функционал — onStart / onStop, украшательства — onResume / onPause

- Игра — onResume / onPause
- Видеочат — onStart / onStop

Activity.onPause

- Противоположность onResume

- ДА — Остановить всё, что запустил onResume
- ДА — Освободить всё, что «захватил» onResume

- ДА — Последний шанс сохранить пользовательские данные

Пользовательские данные?

- Всё то, что не должно потеряться при следующем переключении на приложение

- Игра — всё, что на экране
- Текстовый редактор — текст *(но, возможно, не положение курсора)*
- Форма — введённые данные *(но, возможно, не текущее поле ввода)*

- Звонок ➔ вернулся в приложение ➔ пустой экран ➔ грусть–печаль ☹
- Не надо так

Последний шанс сохранить?

- Не onStop, не onDestroy?
- Да
- После onPause, Android может «прибить» приложение
- Причём жестко — прибивая linux–процесс
- Ни один метод, ни один финалайзер не будет вызван
- onPause — последний шанс
- А как же onSaveInstanceState?
  - Это отдельная тема, обсудим позже

```
override fun onPause() {
    fotoapparat.stop()
    persistUserData()

    super.onPause()
}
```

Activity.onStop

- Противоположность onStart

- ДА — Остановка асинхронных задач
- ДА — Отписка от потоков, event bus и broadcast-ов
- ДА — Освобождение всего остального, захваченного в onStart

- ДА — Очистка listener–ов, eg. setOnClickListener(null)

setOnClickListener(null)?

- UI–события собираются в очередь
- Распределяются не обращая особого внимания на стадию жизненного цикла
- Можно получить вызов listener–а после onPause или onStop
  - 100% (на основании пережитой боли ☺)
  - API >= 19 — вероятно, уже поправлено
  - API <= 18 — баг точно есть

```
// Простой способ открыть несколько
// новых экранов вместо одного
// для API <= 18

someView.setOnClickListener {
    startActivity(Intent(...))
}
```

setOnClickListener(null)?

- Можно в обработчике делать setEnabled(false)
- Но на случай рестарта (без onCreate) нужно ещё setEnabled(true) в onRestart / onStart
- setOnClickListener(null) выглядит более удобным

- А можно ли получить повторный вызов listener–а до onStop, но уже после onPause?
  - Хороший вопрос ☺
  - Но очищать listener–ы в onStop — первый шаг к лучшему коду

```
override fun onStop() {
    someView.setOnClickListener(null)
    subscription.unsubscribe()

    super.onStop()
}
```

Activity.onDestroy

- Не нужно
  - В 99% случаев
- Всё, что нужно освободить — уже освободили в onPause / onStop

Activity.onSaveInstanceState (1/2)

- Не годится для сохранения важных данных
- Ключевое слово — **instance**
- Состояние конкретного экземпляра этой Activity
  - Позиция скролла, текст в инпутах, открытые диалоги
  - Можно сказать — «кэш»
  - Если он вдруг пропадёт, то пользователь не должен расстроиться

Activity.onSaveInstanceState (1/2)

- Activity запустилось в другом Task–е — другой instance state
  - Почему? — Ошибка в launchMode, в коде или внешние факторы
  - А пользователь всё равно расстроится
- А ещё пользователь — человек
  - Нажал на Home, намереваясь потом продолжить работу с приложением
  - Сделал всё, что надо было
  - Вызывает список последних приложений
  - Случайно всё очищает
  - Открывает приложение снова — а там уже пусто, грусть–печаль ☹
- Поэтому, всё важное — сохранять на onPause

Activity.onSaveInstanceState (2/2)

- If called, this method will occur **after** onStop() for applications targeting platforms starting with Build.VERSION_CODES.P
- For applications targeting earlier platform versions this method will occur **before** onStop() and there are no guarantees about whether it will occur **before or after** onPause()
  - На моей памяти до onPause не было, но до onStop было 100%
- Ну вызывается и вызывается, какая разница–то?

Activity.onSaveInstanceState (2/2)

- **java.lang.IllegalStateException: Can not perform this action after onSaveInstanceState**

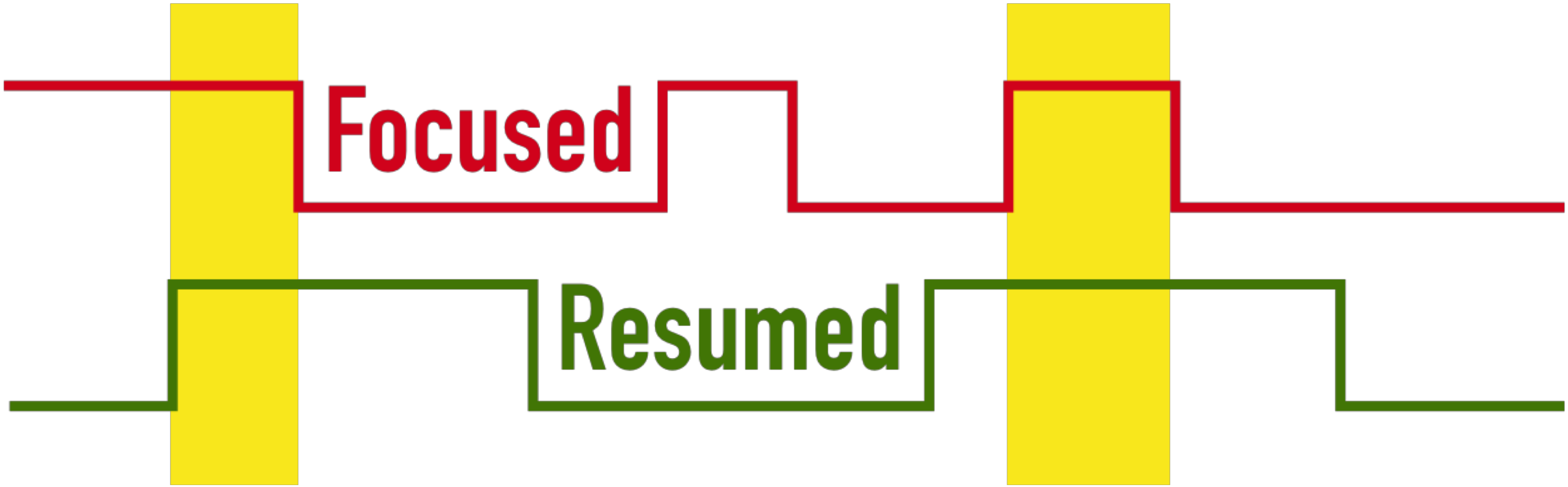
- Её можно получить, даже если делать всё абсолютно правильно
  - onStart — подписываемся на обзёрвабл
  - В обработчике — реплейсим фрагмент
  - onStop — отписываемся
- Но если onSaveInstanceState будет раньше, чем onStop, то может быть крэш

- commitAllowingStateLoss — не выход
- Решение есть (при условии, что у вас есть «состояние» приложения)

```
private var canUseFragments : Boolean = true

// override fun onRestart() { super.onRestart()
//     canUseFragments = true } // ➔ onStart

override fun onStart() { super.onStart()
    canUseFragments = true }

// onResumeFragments()? Гуглеры сами не уверены, но, кажется, не нужно
override fun onResume() { super.onResume()
    canUseFragments = true }

override fun onSaveInstanceState(outState : Bundle) {
    super.onSaveInstanceState(outstate)
    canUseFragments = false
}
```

Activity.onWindowFocusChanged

- Скорее всего, не нужно для обычных приложений
  - Но, например, бывает нужно для игр
- onResume не гарантия того, что Activity «доступна» пользователю
  - Например, может быть частично перекрыта шторкой
- onWindowFocusChanged — гарантия
- Но есть один маленький нюанс
  - onWindowFocusChanged не очень привязан к жизненному циклу


Focused
Resumed

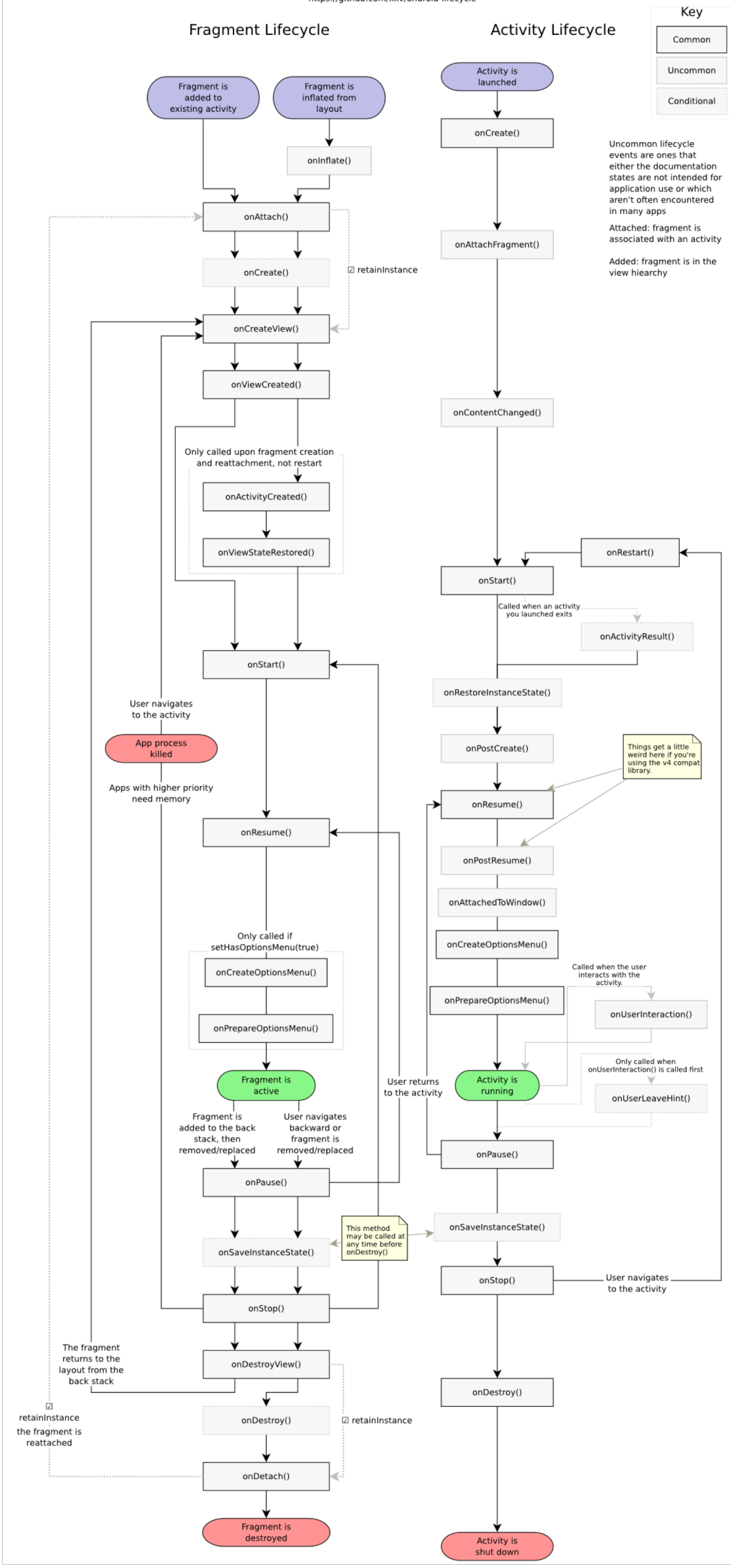
Activity.*

- onRestoreInstanceState
- onPostCreate
- onAttachedToWindow
- onAttachFragment
- onContentChanged
- …
- *И другие*
- *…*
- В 99% случаев не нужно для большинства приложений

Fragment

The Complete Android Activity/Fragment Lifecycle

v0.9.0 2014-04-22 Steve Pomeroy <stevep@thelevelup.com>
CC-BY-SA 4.0
https://github.com/xxv/android-lifecycle



Like Activity

- Жизненный цикл фрагментов — сложно
- Взаимное влияние фрагмента и Activity — очень сложно
- Делать похоже на Activity — больше целостности, меньше боли
- Упрощает написание, понимание и поддержку такого кода

Fragment.<constructor>

- Наверное, не стоит об этом упоминать
  - *Т.к. это должен знать любой разработчик под Android*
- Но всё–таки
  - a) У фрагмента **должен** быть **публичный** конструктор без параметров
  - b) **Не стоит** использовать конструкторы с параметрами

```
// Пишем конструкторы без параметров,
// даже если они ничего не делают.

public class SomeFragment extends Fragment {
    // Тупо – ДА
    // Надёжно – ДА.
    public SomeFragment() {
        super();
    }
}

// Пример на Java, т.к. с Kotlin сложнее
// выстрелить себе в ногу
```

Fragment.onAttach

- Нужен, только если используется setRetainInstance(true)
- Но лучше не использовать setRetainInstance(true)
  - *Для сохранения состояния между изменениями конфигурации есть другие методы (onSaveInstanceState / ViewModel / и т.д.)*

Fragment.onCreate

- НЕТ — Получать доступ к ресурсам (камера)
- НЕТ — Подписываться на потоки, event bus или broadcast-ы
- НЕТ — Запускать асинхронные задачи

- *Очевидно* НЕТ — findViewById, обновлять UI, setOnClickListener
  - Т.к. UI ещё не создан ☺

- *Возможно* НЕТ — Сохранять ссылку на Activity в переменку
  - Т.к. может быть setRetainInstance(true)
  - *Но лучше не использовать setRetainInstance(true)*

Fragment.onCreate

- Аналог конструктора (как и в Activity)
- ДА — Инициализация
- *Возможно* ДА — Внедрение зависимостей
  - *Если не используется setRatainInstance(true)*
- *Возможно* ДА — Сохранить ссылку на Activity (**vs** каждый раз вызывать getActivity())
  - *Если не используется setRatainInstance(true)*
- ДА — Брать нужные параметры из arguments
- ДА — Восстановление instance state
  - Так же как и в Activity — в переменки

** Но лучше не использовать setRetainInstance(true)*

```
override fun onCreate(savedInstanceState : Bundle?) {
    super.onCreate(savedInstanceState)

    AndroidInjection.inject(this)
    owner = requireActivity() as ISomeFragmentOwner

    if (savedInstanceState != null) {
        isDialogAlreadyShown = savedInstanceState
            .getBoolean(KEY_DIALOG_SHOWN)
    }
}
```

Fragment.onCreateView

- *Очевидно* ДА — создать View ☺

- ДА — findViewById
  - Аккуратно, если у фрагмента есть несколько разных layout
  - Закэшированные view от старого layout — за–null–ить
  - Поможет избежать возможных ошибок
  - Даст GC выполнить свою работу

- НЕТ — Восстановление instance state (хоть он и передаётся)

```
override fun onCreateView(inflater : LayoutInflater,
        container : ViewGroup?,
        savedInstanceState : Bundle?) : View {

    View contentView = inflater.inflate(
        R.layout.some_layout,
        container,
        false)

    someView = contentView
        .findViewById(R.id.some_view) as SomeView

    return contentView
}
```

Fragment.onViewCreated

- Склоняюсь к тому, чтоб его не использовать
- findViewById можно сделать и в onCreateView
- onViewCreated не вызывается на API < HONEYCOMB_MR2 (13)

** Если приложение рассчитано только на более новые API, или всегда используются только support–фрагменты, то можно и onViewCreated*

Fragment.*

- onStart
- onResume
- onPause
- onStop

- Полностью аналогично как и для Activity
  - Включая за–null–ивание listener–ов

- Единообразие и целостность — это хорошо ☺

Fragment.onDestroyView

- Не нужно в 99.99% случаев
- Всё что нужно уже должно было быть почищено в onPause и onStop
- С остальным справится GC
- Вьюшки тоже не надо за–null–ивать
  - Либо фрагмент будет удалён, и GC сделает своё дело
  - Либо потом будет вызван onCreateView, и View–шки будут перезаписаны

Fragment.onDestroy

- Аналогично Activity — не нужно
  - Кроме того, не вызывается если setRetainInstance(true)
  - *Но лучше не использовать setRetainInstance(true)*

Fragment.onDetach

- Не нужно
  - Если только не используется setRetainInstance(true)
  - Но…
  - Всё, честно, в последний раз
  - Лучше не использовать setRetainInstance(true) ☺

Fragment.onSaveInstanceState

- Вызывается из Activity.onSaveInstanceState
- Всё, что актуально для Activity — актуально и для фрагментов
- Если child–фрагменты не используются, то всё хорошо
- Если используются — то надо избегать IllegalStateException (аналогично Activity)

onWindowFocusChanged?

- У фрагментов этого метода нет
  - Для обычных приложений не понадобится (99%)
  - Возможно, может быть полезным для игр
- Но если всё-таки понадобилось, то
- a) Прокидывать из Activity
  - – Лишняя связь с Activity
  - + Работает на всех версиях Android
- b) view.getViewTreeObserver().addOnWindowFocusChangeListener
  - + Фрагмент независим
  - – Только для API >= JELLY_BEAN_MR2 (18)

Fragment.*

- onInflate
- onActivityCreated
- onViewStateRestored

- В 99% случаев не нужно для большинства приложений

Заключение

TL;DR

- Пользуйтесь рекомендациями из этого доклада
- Аккуратно выбирайте что запускать на onStart, а что на onResume
- Сохраняйте всё важное на onPause
- Осторожно с onSaveInstanceState
- Fragment — максимально похоже на Activity

- И всё будет хорошо
  - *В 99% случаев* ☺

Источники вдохновения

- https://developer.android.com/
  - Как же без него ☺
- https://github.com/xxv/android-lifecycle
  - Всё, что нужно знать о жизненном цикле
- https://www.techyourchance.com
  - Местами спорно, но есть что почитать даже для синьёра–рассиньёра
- https://github.com/aosp-mirror
  - Или https://android.googlesource.com/
  - Так же можно скачать локально через sdk manager (но только java-часть)

Минутка рекламы

HolyCycle

- Абстракция над жизненными циклами Activity и Fragment–а
- Унификация + «workaround» для onSaveInstanceState
- Можно подключить к существующему проекту
- Можно внедрять постепенно
- Можно использовать, даже если нельзя менять базовый класс Activity или Fragment–а
- Нет rocket science, всегда можно заглянуть в исходники
- Покрыта тестами (unit + ui)

НЕ является заменой / конкурентом android.arch., могут использоваться вместе*

HolyCycle — ViewController

```
↓ onControllerCreate(Bundle extras)
? onControllerRestoreInstanceState(Bundle savedInstanceState)
    ↓ onControllerGetContentLayoutId() // + getView() for Activity
    ↓ onControllerContentViewCreated() // + findViewById() for Fragments
        ↓ onControllerStart()
            ↓ onControllerResume()
                ↓ onControllerFocus()
                ↓ onControllerBlur()
            ↓ onControllerPause()
            ↓ onControllerPersistUserData()
        ↓ onControllerStop()
        ? onControllerSaveInstanceState(Bundle outState)
```

HolyCycle — приглашаю присоединиться

- Пока не могу назвать её production–ready
- Возможно, не самое удачное именование методов
- Возможно, есть баги
- Возможно, нужно будет дополнить функционал

- Приглашаю присоединиться
  - Пробуйте
  - Используйте
  - Сабмитьте Issues
  - Создавайте Pull requests

https://github.com/restorer/holycycle

https://eightsines.com/me.html#talks

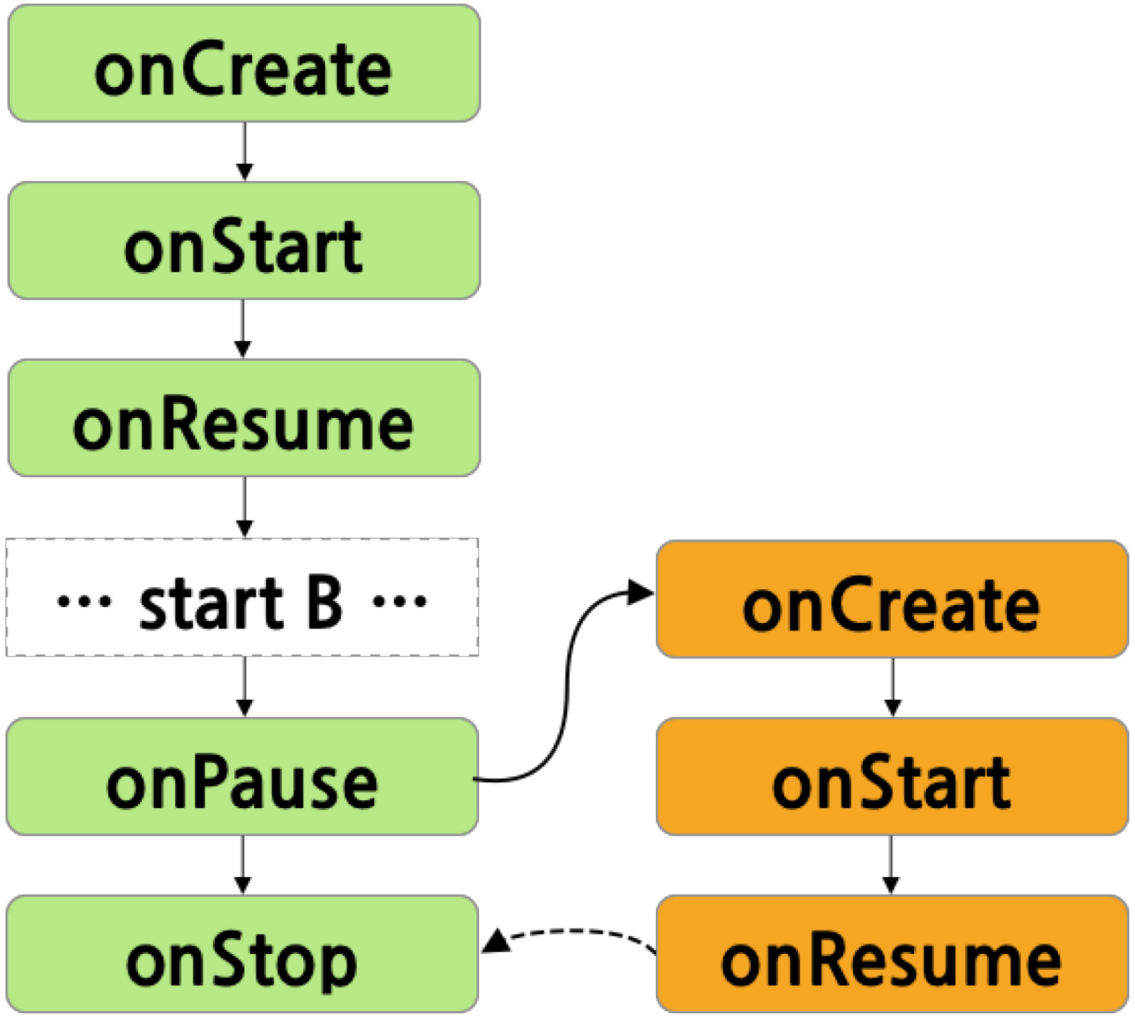
Дополнительно №1

Реплейс фрагмента после onSaveInstanceState

```
private fun onSubscriptionNext() {
    if (canUseFragments) {
        getSupportFragmentManager().replace(...)
    } else {
        shouldShowFragment = true
    }
}

// в onStart()
    if (shouldShowFragment) {
        shouldShowFragment = false
        getSupportFragmentManager.replace(...)
    }

// в onSaveInstanceState - сохранить shouldShowFragment в instance state
// в onCreate - восстановить shouldShowFragment из instance state
```

Дополнительно №2

Запуск Activity из Activity


onCreate
onStart
onResume
··· start B ···
onPause
onStop
onCreate
onStart
onResume

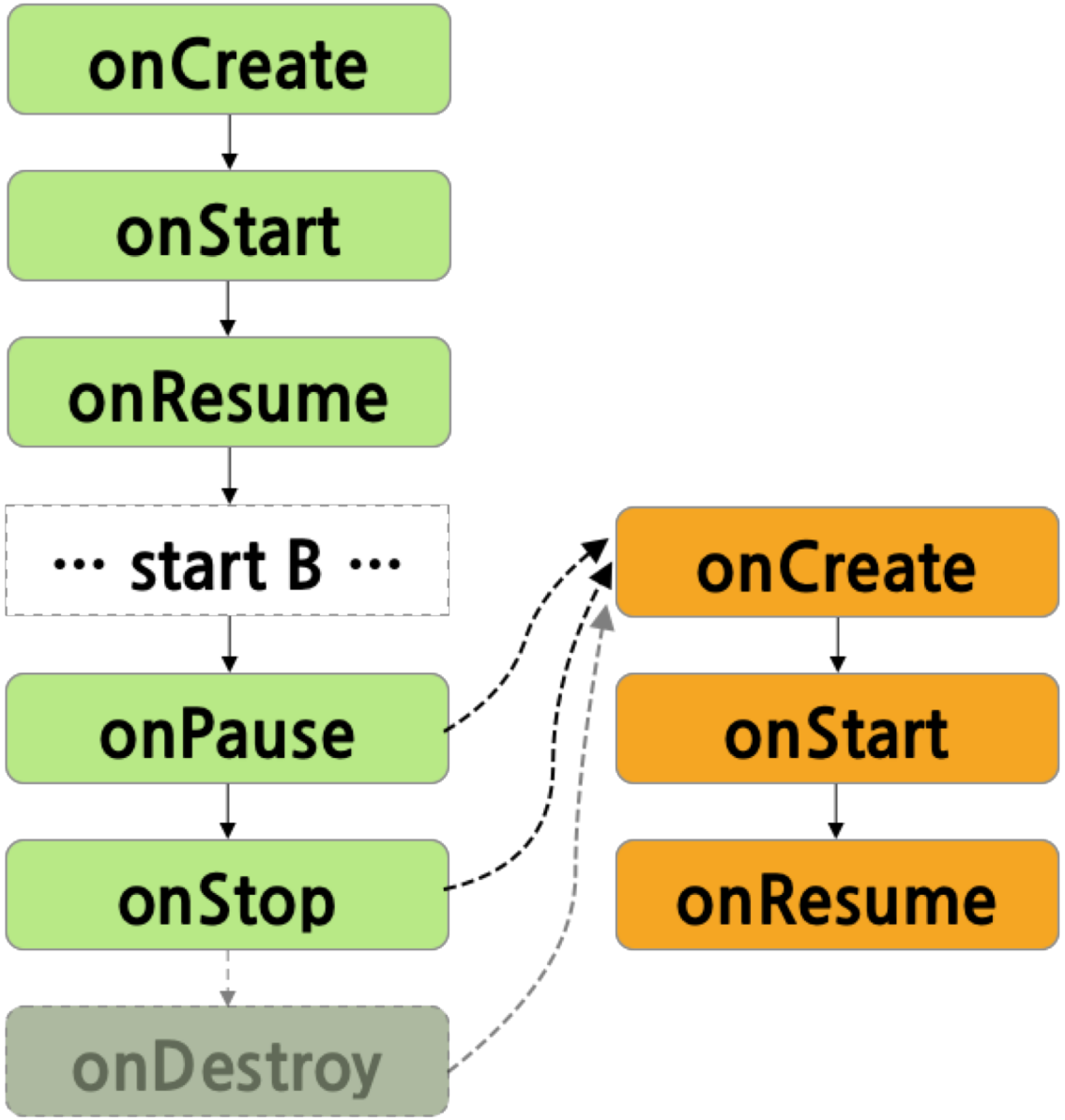
?

Запуск Activity из Activity

- **A.onPause()** / B.onCreate(), B.onStart(), B.onResume() / **A.onStop()** ?

- Так и должно быть
  - Есть даже такие вопросы на собеседованиях
  - Но есть хитрые девайсы из поднебесной… ☺

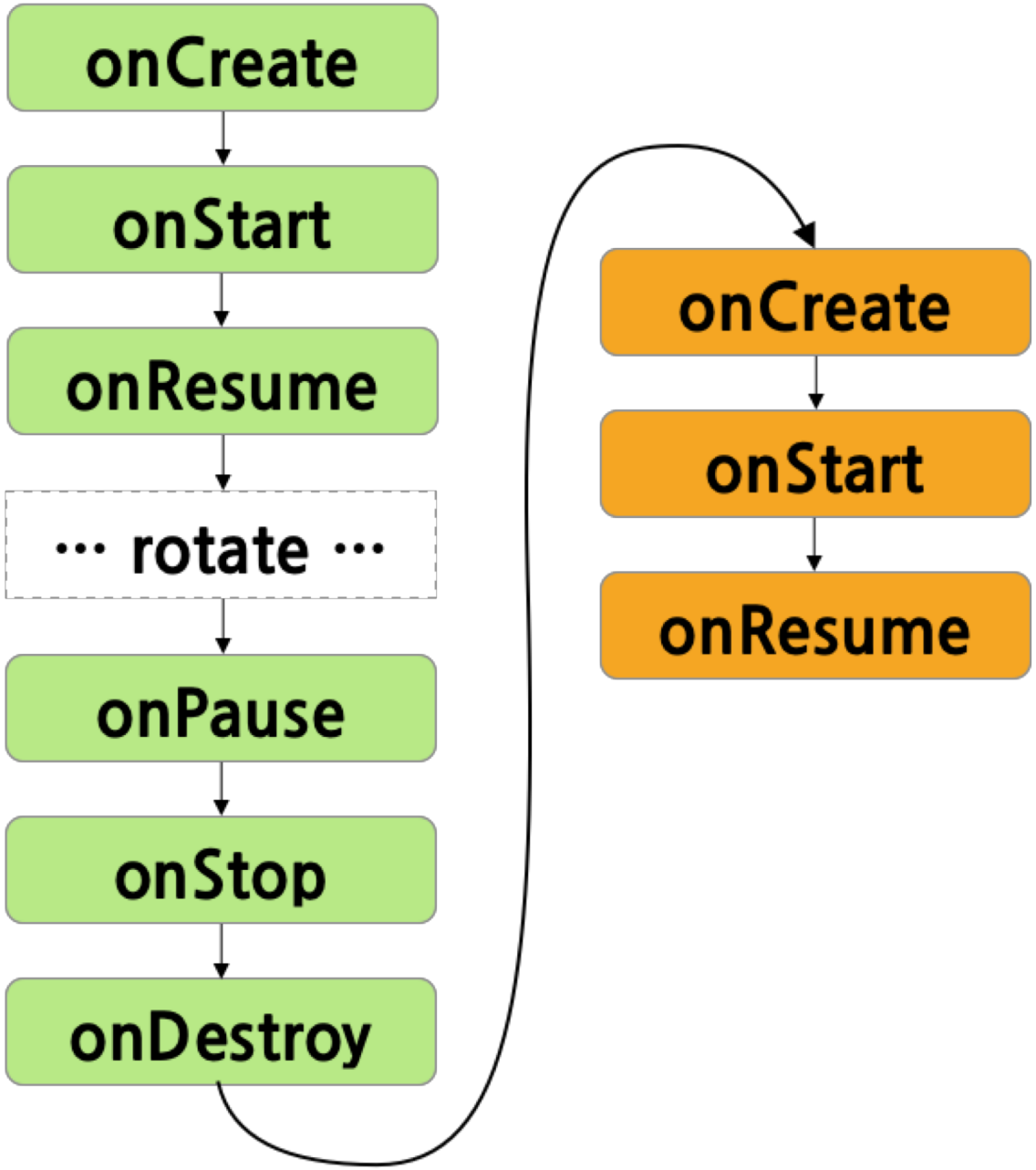
- Безопасней будет ожидать, что новая Activity может запуститься когда угодно



Может и нет,
но надёжней
думать так

Дополнительно №3

Пересоздание Activity после изменения конфигурации


onCreate
onStart
onResume
··· rotate ···
onPause
onStop
onDestroy
onCreate
onStart
onResume

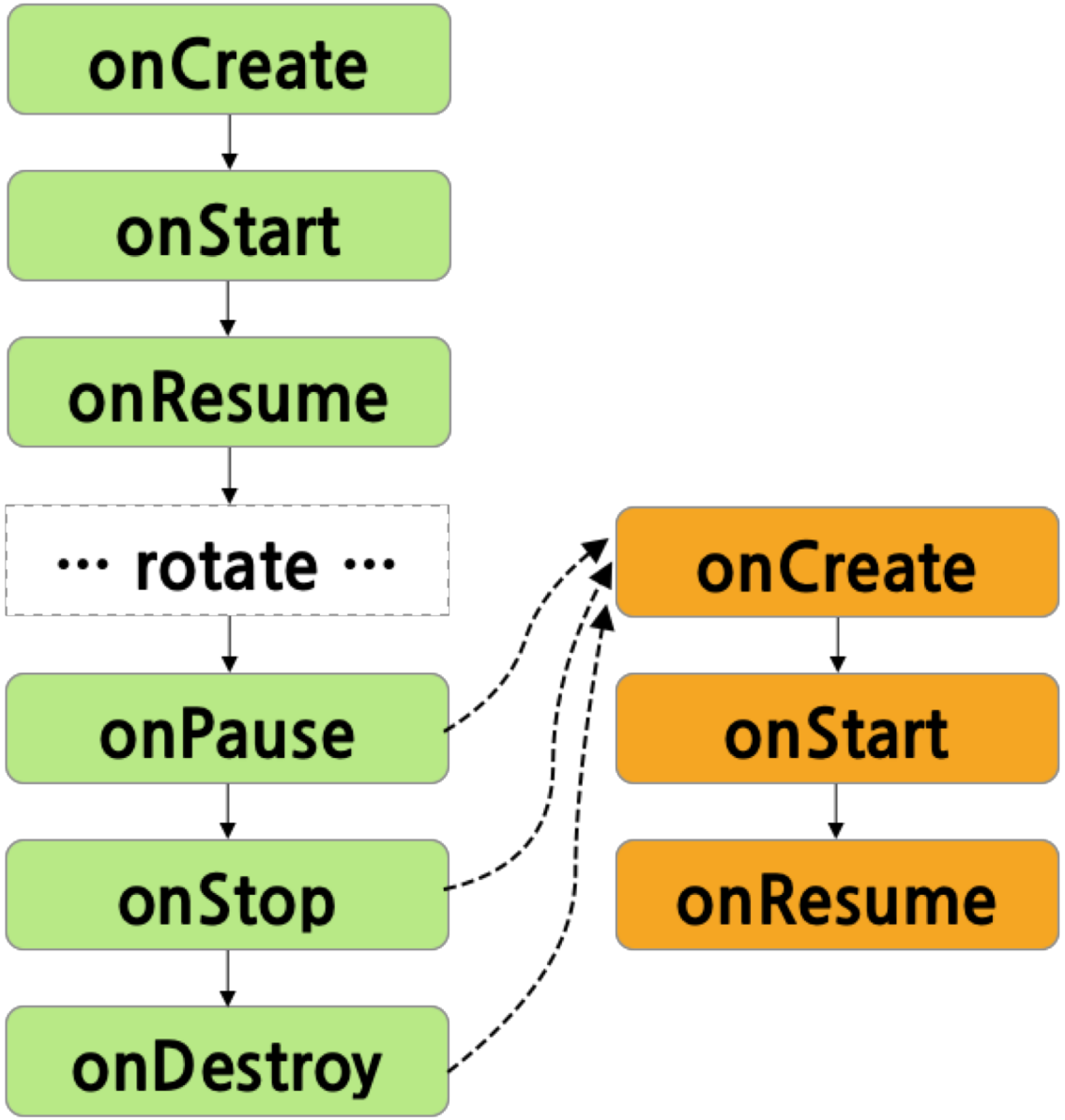
?

Пересоздание Activity после изменения конфигурации

- Старый инстанс завершится (onDestroy), новый создастся (onCreate)?

- Возможно
  - Быстрый поиск по исходникам говорит что это так
  - Но есть хитрые девайсы из поднебесной… ☺

- Безопасней будет ожидать, что новый инстанс может запуститься когда угодно



Может и нет,
но надёжней
думать так

https://github.com/restorer/holycycle

https://eightsines.com/me.html#talks