Android Studio Debugging: Базовые понятия и возможности

Часть первая, основы

**Что такое отладка?**

Я уверен что 95% читателей этой публикации прекрасно знают что такое отладка приложения, и, говоря по простонародному, уже не раз дебажили свои приложения в поисках ошибки или проверки работы своей программы.

И все же краткое вступление с теорией будет. Поэтому если вы знаете что такое отладка, то пропустите этот раздел и листайте вниз, или сразу перейдите к [второй части](https://medium.com/@artem_shevchenko/android-studio-debugging-%D0%BF%D1%80%D0%BE%D0%B4%D0%B2%D0%B8%D0%BD%D1%83%D1%82%D1%8B%D0%B9-%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C-e05dac22439f).

**Отладка или debugging приложения** — это процесс поиска и исправление ошибок или неправильного поведения кода.

Например: при получении ответа с сервера и его обработке внутри вашего приложения происходит непонятная ошибка и на экране выводится устаревшая информация вместо новой. Чтоб разобраться что происходит и найти ошибку мы можем при помощи отладки поставить выполнение программы на паузу в нужном нам месте и посмотреть содержимое наших объектов и переменных.

Именно для этого и нужна отладка: приостанавливать выполнение кода в нужном месте и изучении состояния приложения.

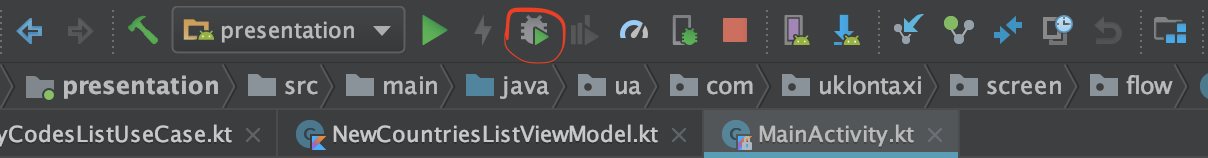
**Основы отладки**

В этой статье я расскажу базовые вещи и возможности отладчика в Android Studio. В [следующей статье](https://medium.com/@artem_shevchenko/android-studio-debugging-%D0%BF%D1%80%D0%BE%D0%B4%D0%B2%D0%B8%D0%BD%D1%83%D1%82%D1%8B%D0%B9-%D1%83%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C-e05dac22439f) я постараюсь рассказать о некоторых тонкостях и продвинутых возможностях работы с отладкой.

**Как запустить отладку — Android Debugger**

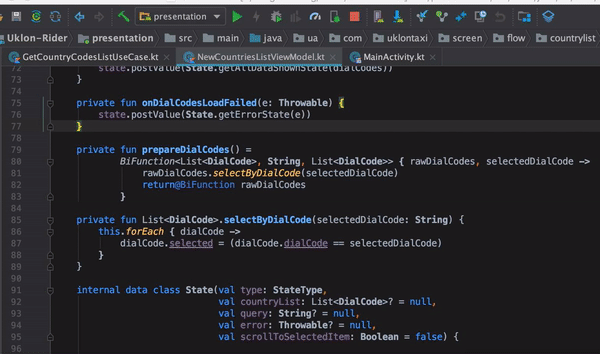
Запустить отладчик в Android Studio можно двумя способами:

**Первый способ**: Запустить приложение в режиме отладки. Стоит использовать если нужно отладить именно запуск приложения. Для отладки конкретного действия-экрана лучше использовать второй способ, так быстрее.



Нажать на иконку с жучком или клавиши (⌃+D) и выбрать в диалоге устройство на котором запустить отладку приложения.

**Второй способ**: Подключить Android Debugger к уже запущенному процессу. Удобно, потому что бывает нужно продебажить конкретный экран или нажатие на кнопку, а запускать каждый раз приложение в режиме отладки долго и неудобно.



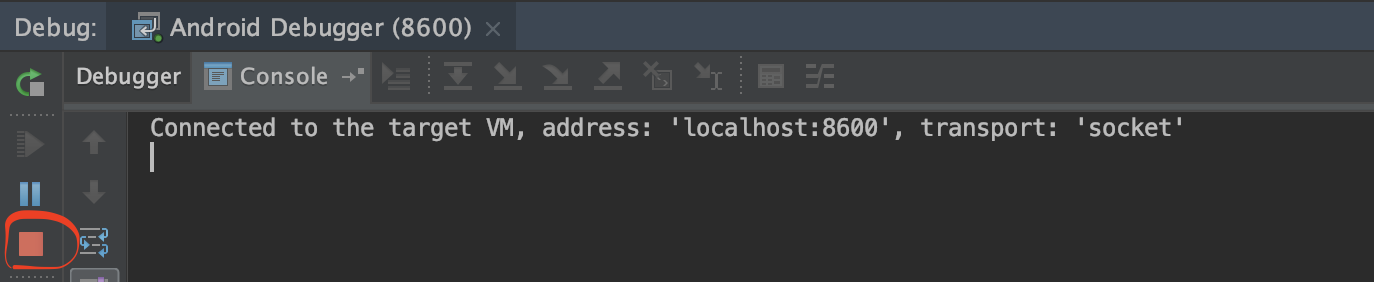
**Как остановить отладку**

Остановить отладку можно здесь:



Верхняя панель действий, (⌘+F2)

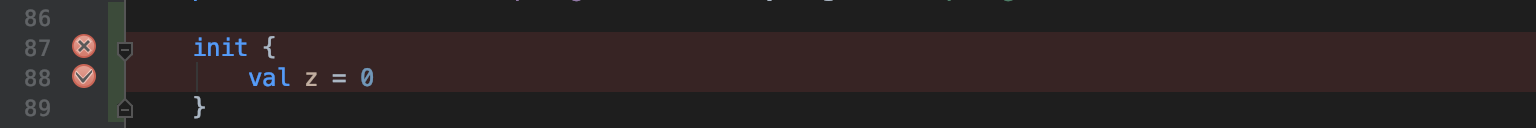
Или здесь:



Вкладка Debug, обычно находится снизу Android Studio, (⌘+F2)

**Breakpoins или точки остановки**

**Breakpoint или точка остановки** — специальный флаг для Android Debugger-a который означает “Останови выполнение программы в этом месте”.



Две точки остановки во время отладки приложения

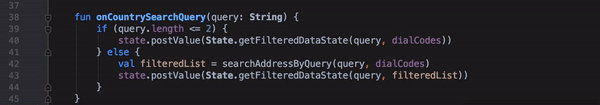
На скриншоте выше видно две точки остановки. Интересно, что они сообщают некую информацию о себе: одна из них с галочкой а другая с крестиком.

**Галочка** значит что эта точки остановки активная и кода выполнение кода подойдет к этой строке выполнение программы будет остановлено.

**Крестик** обозначает что выполнение кода на этой строке невозможно (например у пустой строки или, как в примере выше, у ключевого слова)

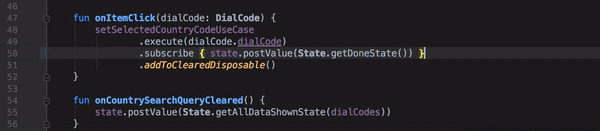
**Как сделать точку остановки — breakpoint**

Для добавления точки остановки нужно кликнуть на серой области напротив строки с кодом или нажать ⌘+F8.



**Выбор точки остановки — set breakpoint**

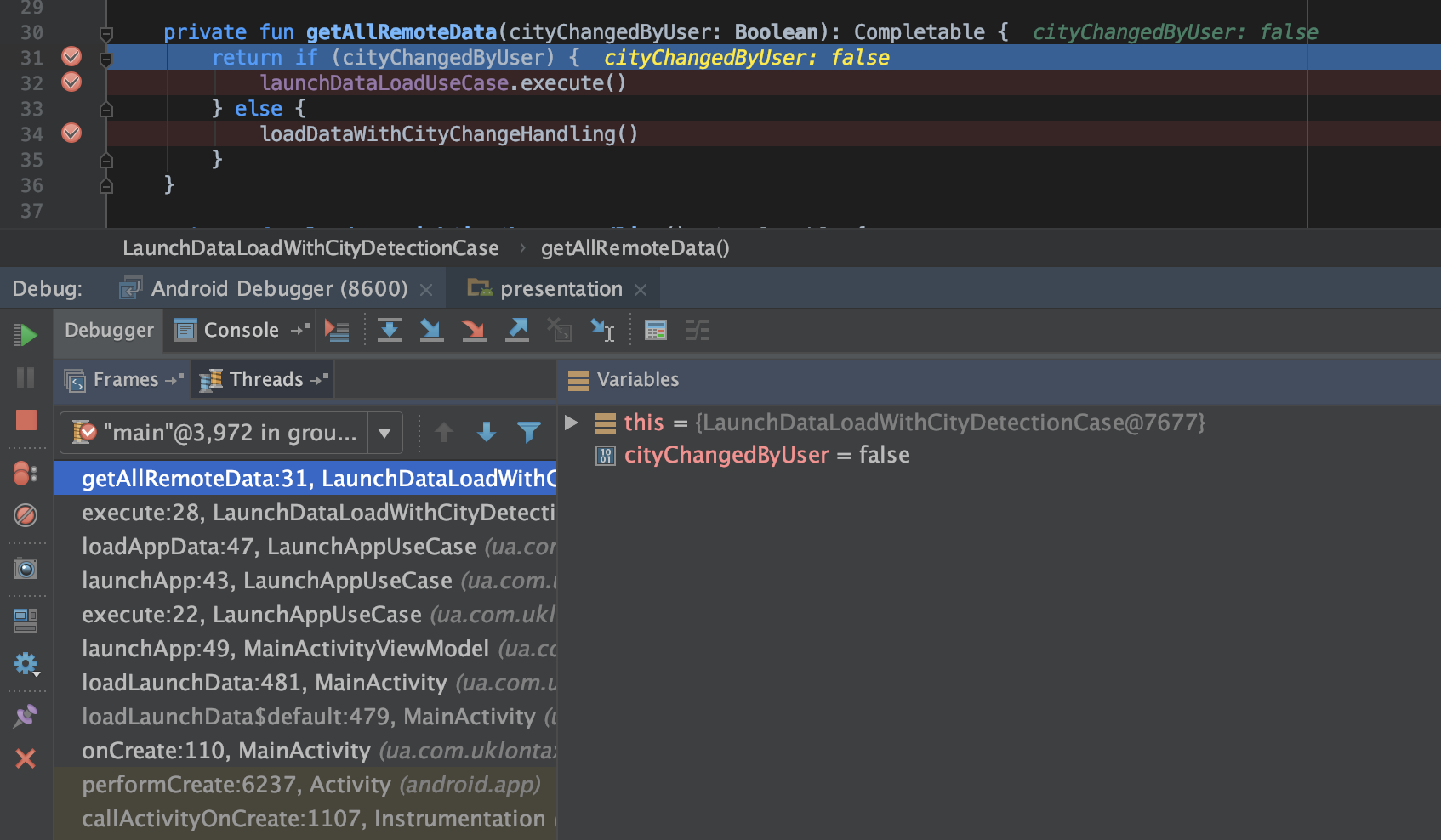
Практически все современные Android приложения используют Kotlin и RxJava. И при типичном использовании этой связки языка и библиотеки при добавлении точки остановки иногда появляется такой выбор:



Android Studio предлагает нам выбрать где именно мы хотим добавить точку остановки — момент выполнения метода subscribe в Rx-цепочке или выполнение кода внутри лямбды, которую мы передаем в subscribe. А можно поставить точку остановки сразу для двух действий (All).

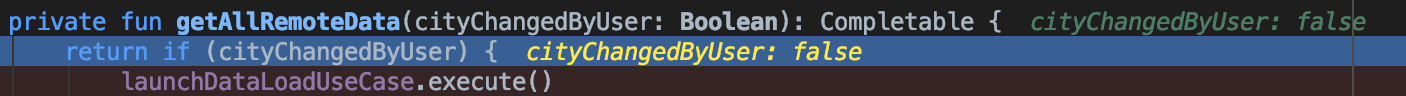
***Примечание для новичков:*** Если вы хотите отладить действие, которое вы планируете выполнить внутри метода subscribe, и не поняли в примере выше где именно вам поставить точку остановки — вам нужно поставить ее внутри лямбда выражения.

**Просмотр информации во время отладки приложения**



Остановка выполнения приложения при достижении breakpoint-a. Синим цветом Android Studio подсвечивает на какой именно точке выполнение программы было остановлено.

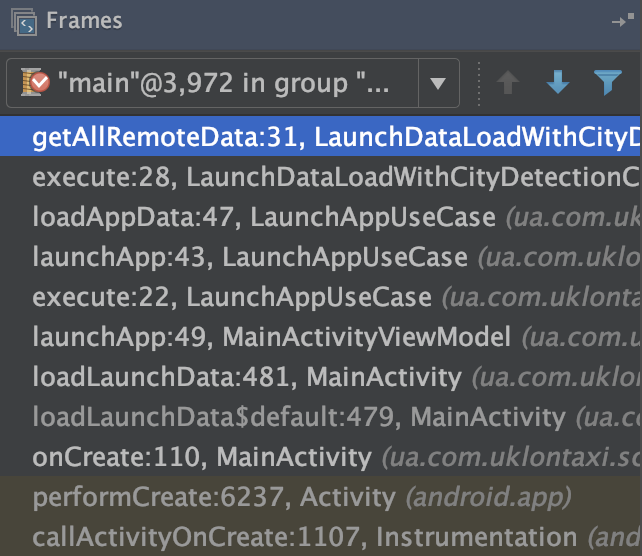
**Подсказки от Android Studio**



Подсказки от Android Studio

Подсказки от Android Studio во время отладки — приятный бонус. В них отображается полезная информация, например значение локальных переменных, аргументов и объектов (для них выводится информация из метода toString() соответствующего объекта).

**Стек вызова методов — Frames**



Стек. Синим цветом подсвечивается текущий метод.

В вкладке Debugger во время остановки выполнения приложения выводится стек вызова методов.

Благодаря этому можно понять — откуда мы попали в текущий метод.

Желтым цветом подсвечиваются классы Android SDK или библиотек — там, где у нас нет доступа к исходному коду.

https://miro.medium.com/max/802/1*MAQDHh156g0D4DkhJEfIOw.png

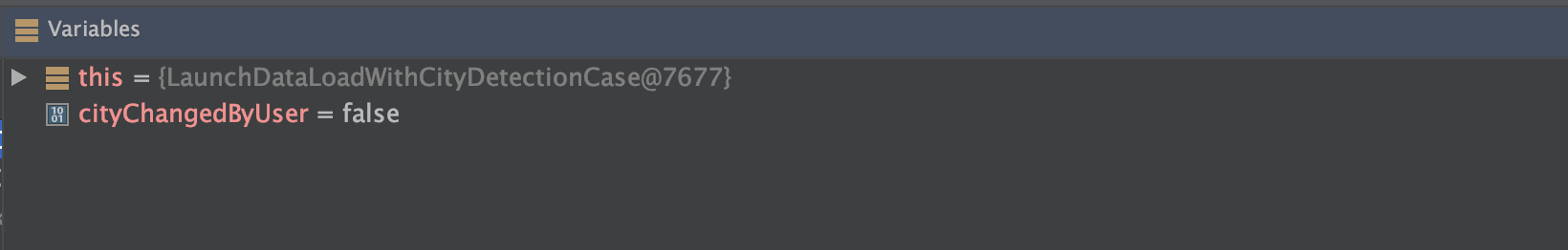
Вывод информации о текущем потоке

Формат вывода информации о методе такой:

https://miro.medium.com/max/968/1*Sl6jBPPAapI-jTgrIaAz6A.png

* Название метода — methodThree
* Номер строки с точкой остановки— 22
* Название класса — DebugStateDemoObject
* package класса

**Просмотр информации состояния объекта и переменных — Variables**



Вывод состояния во время остановки

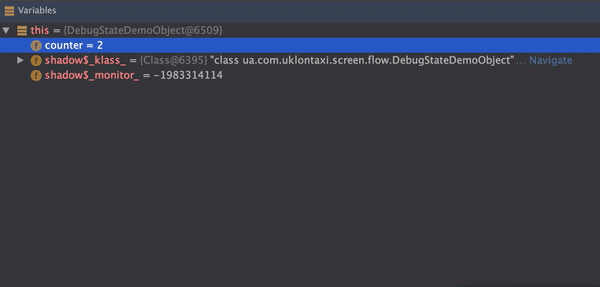
Помимо панели со стеком вызова методов в вкладе Debug также находится и панель с выводом текущего состояния и значениями локальных переменных.

На первой строке выводится информация об объекте, чей метод был вызван и остановлен для отладки. Интересно, что по двойному клику на это строчку можно будет увидеть дополнительную информацию о состоянии текущего класса.

На второй строке выводится информация о локальной переменой, в данном случае это еще и аргумент переданный в этот метод.

**Изменение значения переменой “на лету”**

Значение локальной переменной можно изменить нажав правой клавишей мышки -> Set value… или по нажатию на F2.



Для применения изменений нажмите Enter

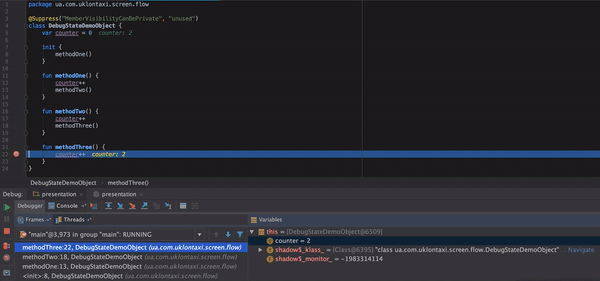
**Стек методов + состояние объекта**

Стоит запомнить, что состояние объекта при переходе по стеку методов не меняется. То есть, при просмотре методов из стека в разделе с информацией о состоянии объекта и локальных переменных, он же Variables, будет показано самое последнее состояние объекта не зависимо от просматриваемого метода.

Для иллюстрации этого давайте проведем эксперимент. Итак у нас есть следующий код:

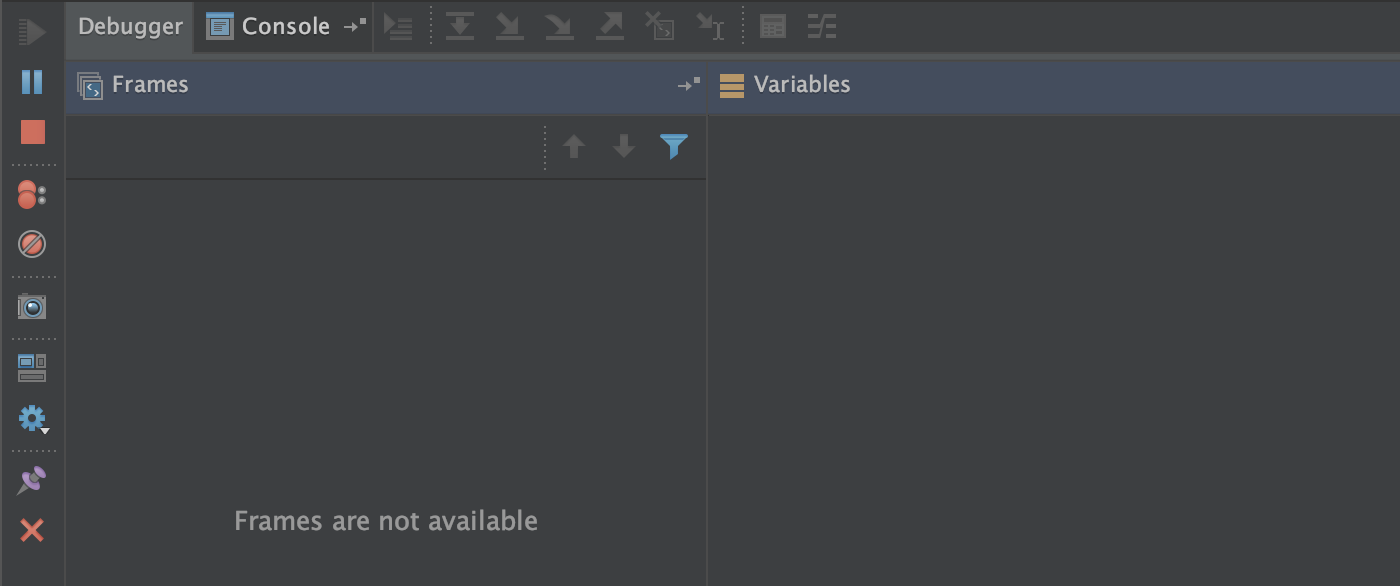
Давайте добавим точку остановки на 19 строку внутри метода ***methodThree()*** и запустим этот код в режиме отладки.

Но перед этим поясню что мы хотим уточнить: если значения объектов сохраняются в стек, то при переходе в стеке вызова методов внутри вкладки ***Debug*** значение поля counter будет меняться. Если состояние объекта не сохраняется, то counter всегда будет с его актуальным (последним) значением.



Обратите внимание на значение counter в вкладке Variables

**Управление процессом отладки**



Интересующая нас панель управления — ряд кнопок слева

Во время отладки Android Debugger в Android Studio позволяет нам управлять процессом отладки.

Без знания этих инструментов процесс отладки будет сложным и запутанным, так что давайте познакомимся с этими возможностями поближе.

**Продолжить выполнение программы**

https://miro.medium.com/max/60/1*Gfbh5qGqm2B2XR4cwBsDBg.png?q=20

https://miro.medium.com/max/62/1*Gfbh5qGqm2B2XR4cwBsDBg.png

Когда во время отладки выполнение кода программы подходит к строке на которой установлен Breakpoint (точка остановки), выполнение программы приостанавливается. По нажатию на эту кнопку выполнение программы будет выполнятся дальше.

**Приостановить выполнение программы**

https://miro.medium.com/max/60/1*P_OxX0gyGaEgn47qJcVUbw.png?q=20

https://miro.medium.com/max/62/1*P_OxX0gyGaEgn47qJcVUbw.png

Есть возможность и просто приостановить приложение во время отладки поставив его на паузу. Сам этот режим, его цель и как он работает я, если честно, не очень понял. Я сделал эксперимент с бесконечным циклом, внутри которого инкрементяться 3 переменных типа Long, и в случайный момент делал паузу: выполнение кода останавливалось на строке с объявлением *while(true)*, я даже видел значения переменных, но на этом все.

**Остановка отладки**

https://miro.medium.com/max/60/1*HEdaIIMMA-rNyN51IHIRvQ.png?q=20

https://miro.medium.com/max/64/1*HEdaIIMMA-rNyN51IHIRvQ.png

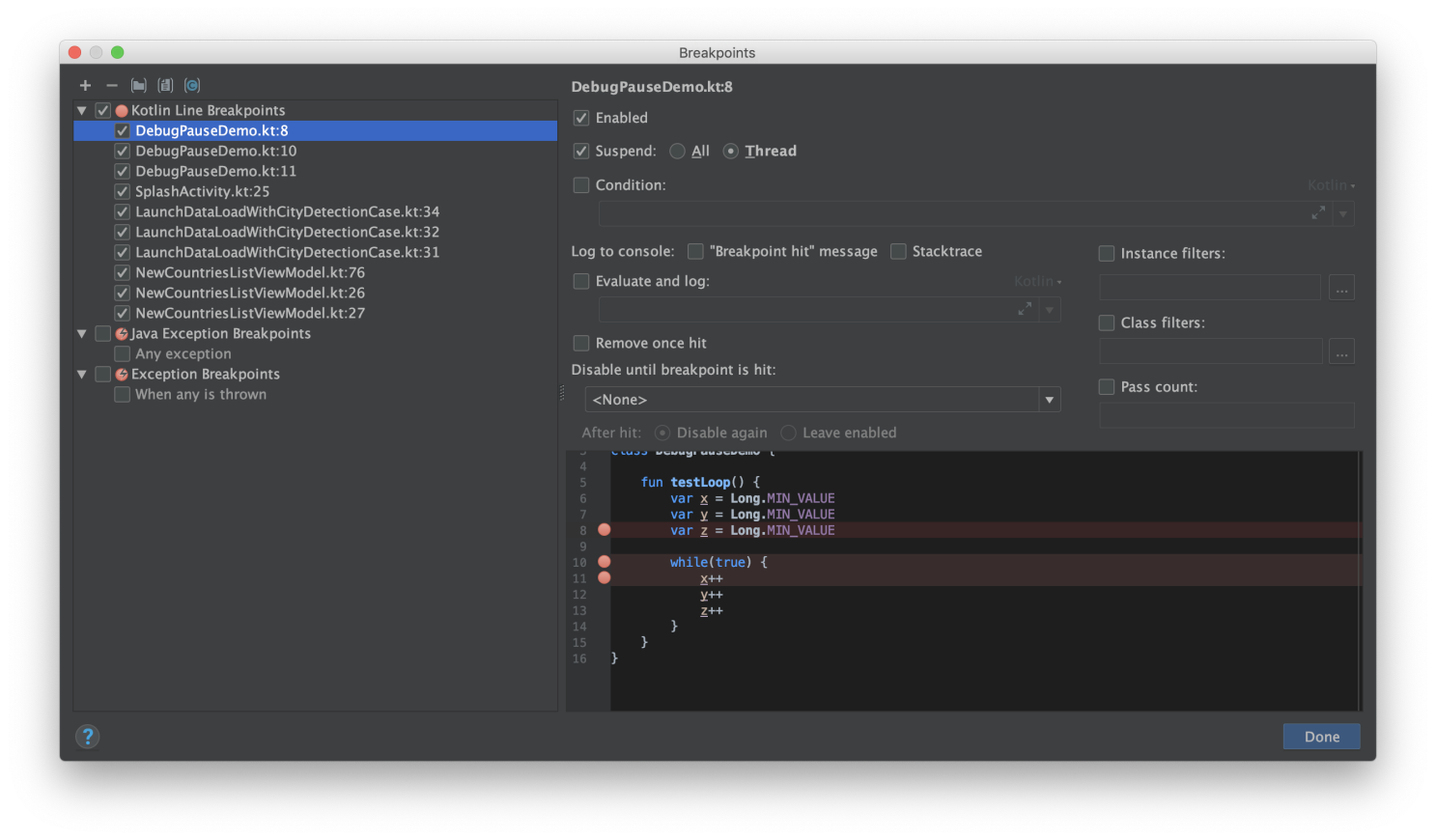
Саму отладку можно остановить по нажатию на кнопку стоп. В последнее время останавливается не только отладка но и приложение; грустно.

**Просмотр и настройка breakpoints (точек остановок) в проекте**

https://miro.medium.com/max/60/1*LOunSpxXbN13O8PcXzpVwg.png?q=20

https://miro.medium.com/max/66/1*LOunSpxXbN13O8PcXzpVwg.png

Эта возможность архи полезная: во первых здесь можно просмотреть все breakpoints в проекте. Во-вторых здесь можно отключить/включить выборочно нужные нам breakpoints. Мы еще вернемся к возможностям этой функции позже, в следующей статье.



Так выглядит окно с просмотром всех breakpoints на проекте

**Приглушить все breakpoints**

https://miro.medium.com/max/60/1*iyjeT9Dz4mGNdWQwZIatcQ.png?q=20

https://miro.medium.com/max/62/1*iyjeT9Dz4mGNdWQwZIatcQ.png

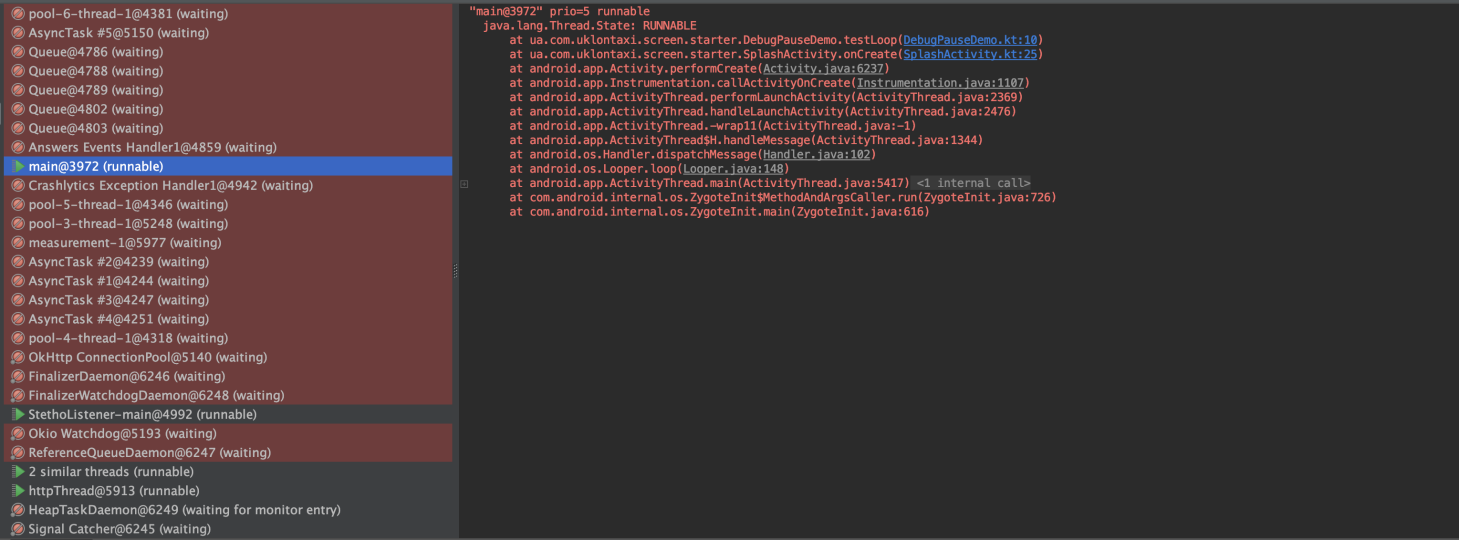
А еще можно приглушить (Mute) все наши точки остановки по нажатию на эту кнопку. Интересно что во время отладки по нажатию на эту кнопку точки остановки действительно не отрабатывают, но у меня после отключения этой возможности точки оставались “приглушенными” и дальше. Может быть баг Android Studio а может и эмулятора — я этого не понял и выглядит это странно. Если кто-то мне подскажет что происходит я буду очень рад!

**Получение dump-а потоков приложения**

https://miro.medium.com/max/60/1*FWxLWOy4vwl-uPi365QBVw.png?q=20

https://miro.medium.com/max/62/1*FWxLWOy4vwl-uPi365QBVw.png

Есть возможность получить dump с состоянием всех потоков в процессе и даже вывод стектрейса с местом выполнения кода. Выглядит это так:



Этот режим будет полезен при отладке сложных, многопоточных приложений.

**Настройка отображения данных и UI вкладки Debug**

https://miro.medium.com/max/16/1*EyeZv09nQqjcetcIRHZfrw.png?q=20

https://miro.medium.com/max/60/1*EyeZv09nQqjcetcIRHZfrw.png

Первая иконка может восстановить разметку вкладки Debug в первоначальное состояние (если чото сдвинули мышкой неудачно)

Вторая кнопка откроет меню в котором можно настроить отображение информации.

Третья кнопка предлагает закрепить текущую вкладку в разделе Debug а четвертая кнопка, та что крестик, закроет вкладку соотвественно.

**Навигация по коду**

https://miro.medium.com/max/60/1*fNZdd95qVeBhabdQOFoAgw.png?q=20

https://miro.medium.com/max/370/1*fNZdd95qVeBhabdQOFoAgw.png

В момент срабатывания точки остановки выполнение кода приостанавливается и мы можем узнать состояние наших объектов и переменных. Но иногда одной точки остановки может быть недостаточно, и для основательного исследования работы нашего кода нам понадобится шаг за шагом выполнять какое-то действие и смотреть что к чему.

Для этого мы можем воспользоваться возможностью двигаться по коду, выполняя нужный нам фрагмент за фрагментом.

Давайте разберемся с этим!

**Перейти к текущей точке остановки**

https://miro.medium.com/max/60/1*aqVir2cnk-5VIbnFrdyrQQ.png?q=20

https://miro.medium.com/max/62/1*aqVir2cnk-5VIbnFrdyrQQ.png

По нажатию на эту кнопку курсор и фокус будет перемещены на текущую точку остановки. Удобно если потерялся.

**Переход к следующей строке**

https://miro.medium.com/max/60/1*qtNEr0UrKYZmi0NL39T7-Q.png?q=20

https://miro.medium.com/max/58/1*qtNEr0UrKYZmi0NL39T7-Q.png

Переход к следующей строке кода внутри текущего метода, не заходя в вызываемый метод (если он вызывается).

**Переход дальше + принудительно**

https://miro.medium.com/max/60/1*CQdB9WnxZl5nJXoF_bIN0A.png?q=20

https://miro.medium.com/max/102/1*CQdB9WnxZl5nJXoF_bIN0A.png

Переход в вызываемый метод к его первой строке.

Красная стрелка — тоже самое, но делает это принудительное, если по какой то причине просто перейти нельзя.

**Выход из текущего метода**

https://miro.medium.com/max/60/1*ArMzSVDIGBXE59ij2I2hYQ.png?q=20

https://miro.medium.com/max/56/1*ArMzSVDIGBXE59ij2I2hYQ.png

Выход из текущего метода на следующую строку кода

**Выполнить до курсора**

https://miro.medium.com/max/60/1*xAmdiABJsdKjqd6IwyqTsg.png?q=20

https://miro.medium.com/max/60/1*xAmdiABJsdKjqd6IwyqTsg.png

А еще можно выполнить код до той строки, на которой находится курсор. Курсор перед этим можно перемещать до куда нам нужно. Очен здоровская возможность!