**Концепция**

**Класс Activity** (активностей/действий) является важным компонентом приложения для Android, а способ запуска и объединения активностей (Activities) является фундаментальной частью модели приложений данной платформы.

Пользователи выполняют определенные действия в приложении через активности: делают фото/видео, отправляют письмо, отвечают на звонок и т.д. Таким образом, активность служит точкой входа для взаимодействия приложения с пользователем.

**Класс Activity** предназначен для облегчения работы мобильных приложений, где взаимодействие пользователя с приложением не всегда начинается со стартового экрана. Например, если пользователь запустит приложение почты, то увидит список писем. Если же он пользуется приложениями социальных сетей, которые могут запустить почту, то пользователь может попасть непосредственно на экран отправки письма.

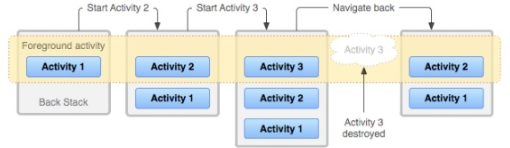
Когда одно приложение вызывает другое, вызывающее приложение запускает активность другого приложения, а не само приложение.

Большинство приложений содержат несколько экранов, что означает, что они содержат несколько активностей. Обычно в качестве главного экрана указывается одна активность в приложении, которая появляется первой при запуске приложения пользователем. Каждая  активность может затем начать другую активность для выполнения различных действий.

**Стек**

Все активности работают в стеке. Это означает:

Каждая стартуемая активность помещается на вершину стека. Предыдущая активность остается в стеке, но её выполнение приостанавливается, и система «сохраняет» её текущее состояние (интерфейс). При нажатии кнопки «Назад», текущая активность удаляется из вершины стека (активность уничтожается) и возобновляется работа предыдущей активности с восстановлением предыдущего состояния её пользовательского интерфейса. Активности в стеке никогда не переупорядочиваются; выполняется только добавление активностей в стек и удаление из него. Т.е. при запуске новой активности, она добавляются в стек, и удаляется из стека, когда пользователь выходит из неё. Таким образом, стек переходов работает по принципу «последним пришел — первым вышел».



**Жизненный цикл активности**

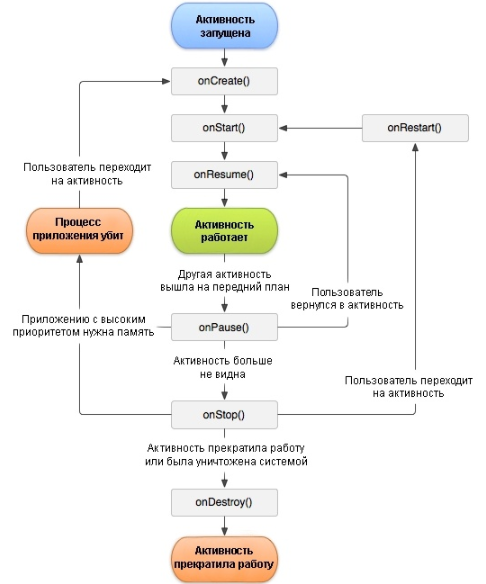
Для того, чтобы полностью понимать принцип работы активности, важно знать, что у каждой активности есть свой жизненный цикл: то есть она может находиться в одном из нескольких разных состояний, в зависимости от того, что происходит в приложении, или от действий пользователя. По мере того, как пользователь перемещается по приложению, активности в приложении проходят через разные состояния в своём жизненном цикле. Класс Activity предоставляет ряд **callback-методов**, которые сообщают активности о том, что состояние изменилось.

В **callback-методах** можно описать, как должна вести себя активность на конкретных этапах жизненного цикла. Например, при создании потокового видеоплеера, можно приостановить загрузку видео, когда пользователь переключается на другое приложение, и возобновлять, когда пользователь возвращается в приложение. Иными словами, каждый **коллбэк** позволяет выполнять определённую работу, соответствующую заданному состоянию. Правильное управление переходом между состояниями делает работу приложения более надёжной и эффективной.

Для навигации между этапами жизненного цикла активности класс Activity предоставляет базовый набор из шести **коллбэков**:

* **onCreate()**
* **onStart()**
* **onResume()**
* **onPause()**
* **onStop()**
* **onDestroy()**

Система вызывает каждый из этих **коллбэков** как только активность переходит в новое состояние.



#### ****Сallback-м****етоды ЖЦ активности:

* **onCreate()** - Этот метод срабатывает, когда система создаёт активность. Его наличие обязательно, поскольку здесь выполняется первоначальная настройка активности. В методе выполняется основная логика запуска, которая должна выполниться только один раз. После того, как onCreate() завершит выполнение, активность переходит в состояние **Started** и система следом вызывает**onStart()** и **onResume()**.
* **onStart() -**Когда активность переходит в состояние **Started**, система вызывает этот метод. Вызов **onStart()**делает активность видимой для пользователя, так как приложение готовится к переходу активности на передний план и становится интерактивной. Метод onStart() завершается очень быстро и, как и с onCreate(), активность не остаётся в состоянии Started, а переходит в состояние **Resumed**, после чего система вызывает метод **onResume()**
* **onResume() -**Когда активность переходит в состояние Resumed, она выходит на передний план, а затем система вызывает метод onResume(). Это состояние, в котором приложение взаимодействует с пользователем. Приложение остается в этом состоянии, пока не произойдёт что-то, что переключит фокус с приложения (к примеру, поступает входящий вызов или блокировка экрана). Когда происходит событие, прерывающее текущее состояние, активность переходит в состояние **Paused** и система вызывает метод **onPause()**

Метод onPause() можно также использовать для освобождения системных ресурсов, регулирования сенсоров или любых других ресурсов, которые могут влиять на расход батареи, пока активность приостановлена. Однако, активность в состоянии Paused может быть ещё частично видимой, поэтому лучше всего использовать для освобождения ресурсов onStop() вместо onPause().

* **onStop()** - Когда активность больше не видна пользователю, она переходит в состояние Stopped, и система вызывает метод onStop(). Это может произойти, например, когда вновь запущенная активность охватывает весь экран. Система также может вызвать onStop(), когда активность завершила свою работу и вот-вот будет уничтожена. В состоянии **Stopped** активность либо возвращается для взаимодействия с пользователем, либо полностью завершается. Если активность возвращается, система вызывает onRestart(). Если активность завершается, система вызывает метод **onDestroy()**.

В методе onStop() приложение должно освобождать или регулировать ресурсы, которые не нужны, пока приложение не отображается пользователю, поэтому данный метод используется для выполнения относительно затратных в плане расхода CPU операций Например, приложение может приостановить анимацию или переключиться с более детального на менее детальное обновление местоположения.

* **onDestroy() -**Метод onDestroy() вызывается до того, как активность будет уничтожена. Система вызывает этот метод по следующим причинам:
  + Активность завершает свою работу поскольку пользователь закрывает активность либо в приложении вызывается метод **finish()**. В данном случае onDestroy() — это последний **коллбэк** жизненного цикла активности.
  + Система временно уничтожает активность из-за изменения конфигурации (например, поворот устройства или использование многооконного режима). После вызова onDestroy() в данном случае система немедленно создаёт новый экземпляр активности и затем вызывает onCreate() в новом экземпляре

Метод onDestroy() освобождает все ресурсы, которые ещё не были освобождены в методах ранее, таких как **onStop()**.

Система убивает процессы, когда ей нужно освободить оперативную память. Вероятность того, что система убьет определенный процесс, зависит от состояния процесса в данный момент.

В зависимости от сложности активности, не всегда обязательно должны быть реализованы все методы жизненного цикла. Однако важно чтобы были реализованы те, которые обеспечивают правильную работу приложения.

Хорошая реализация **коллбэков** может помочь избежать таких проблем, как:

* Сбой приложения, если пользователю приходит звонок или он переключается на другое приложение.
* Потребление системных ресурсов, когда пользователь не использует приложение.
* Потеря прогресса в приложении, если пользователь вышел из него и вернулся позже.
* Сбой или потеря прогресса при повороте экрана.