План лекции

1. Структура приложения под платформу Андроид:
   - Архитектура;
   - Компоненты;
   - Процессы;
   - Активация компонент;
   - Намерения

Структура приложения под платформу Андроид

Приложение на Android

1) Специальный файл-архив с расширением *.apk*;
2) Представляет собой набор компонент;
3) Приложение не имеет точки входа как обычное приложение на java, т.е. нет метода *main( )*;
4) Окружение приложения представлено через контекст;
5) Каждое приложение выполняется в отдельном процессе под отдельным пользователем в многопользовательской среде с минимальными привилегиями.

Структура приложения под платформу Андроид

Особенности

1) Одно приложение может использовать компоненты других приложений (если эти приложения разрешают их использовать);
2) Приложение не содержит код других приложений и ссылки на них;
3) Система должна запустить процесс для приложения, в котором находится требуемый элемент, и инициализировать нужные ему объекты.

Структура приложения под платформу Андроид

Архитектура Android-приложения является фреймворк-ориентированной (framework-based), т.е. сводится к расширению неких классов или реализации интерфейсов, предоставленных фрейморком.

Такое приложение не может быть запущено вне фреймворка или без него.

I) Уникальная система управления памятью:

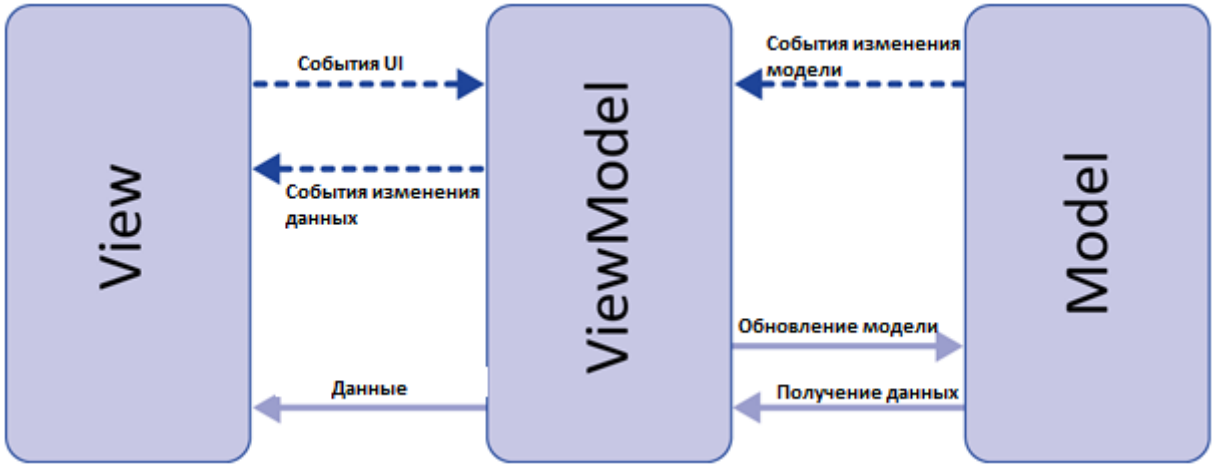
- сборщик мусора может уничтожить объекты, с которыми в текущий момент времени нет взаимодействия, если ОС решит, что ОП мало;
- ОС может уничтожить процесс, который в данный момент времени не показывает пользователю никакого графического интерфейса.

Структура приложения под платформу Андроид

II) Обработка скрытия пользовательских интерфейсов и существование кнопки «назад» на Android-устройствах приводит к необходимости наличия *стека пользовательских интерфейсов*, в котором текущий видимый интерфейс помещается на вершину, а все остальные сдвигаются вниз.

III) Обработка взаимодействия между пользовательским интерфейсом и его логикой следует архитектурному шаблону «Model-View-ViewModel» (**MVVM**, Модель-Представление-Модель представления).

Структура приложения под платформу Андроид



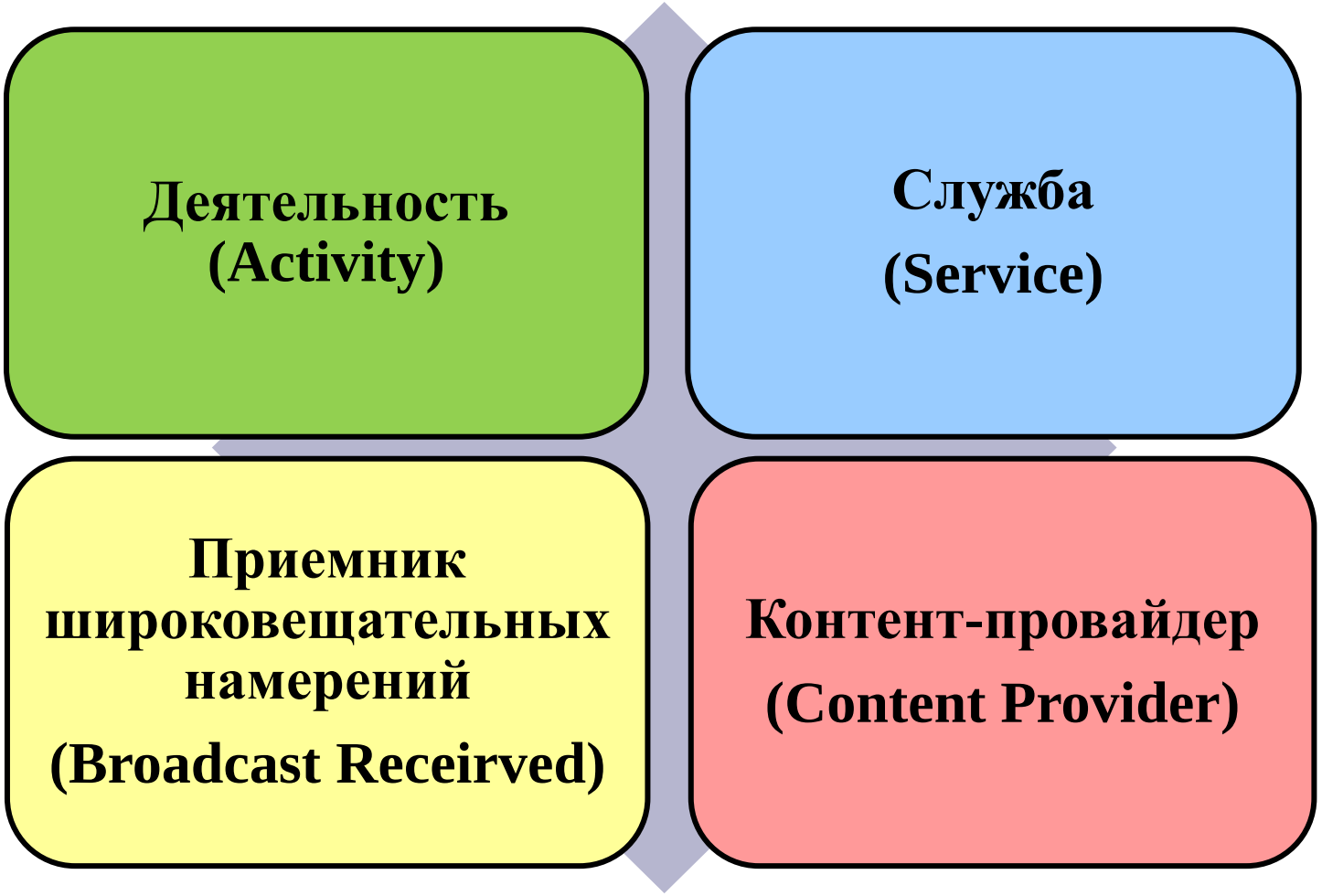
Пользовательский интерфейс реализуется гипертекстовой разметкой (XML).

Логика пользовательского интерфейса реализуется разработчиком как компонент ViewModel.

Функциональные связи между пользовательским интерфейсом и ViewModel реализуются через биндинги (bindings), которые являются правилами типа «если кнопка А была нажата, должен быть вызван метод onButtonAClick() из ViewModel».

Структура приложения под платформу Андроид

Компоненты Android-приложения



Структура приложения под платформу Андроид

Компонент «ДЕЯТЕЛЬНОСТЬ»

Представляет собой визуальный интерфейс (окно, экран) для одного действия, которое пользователь может совершить.

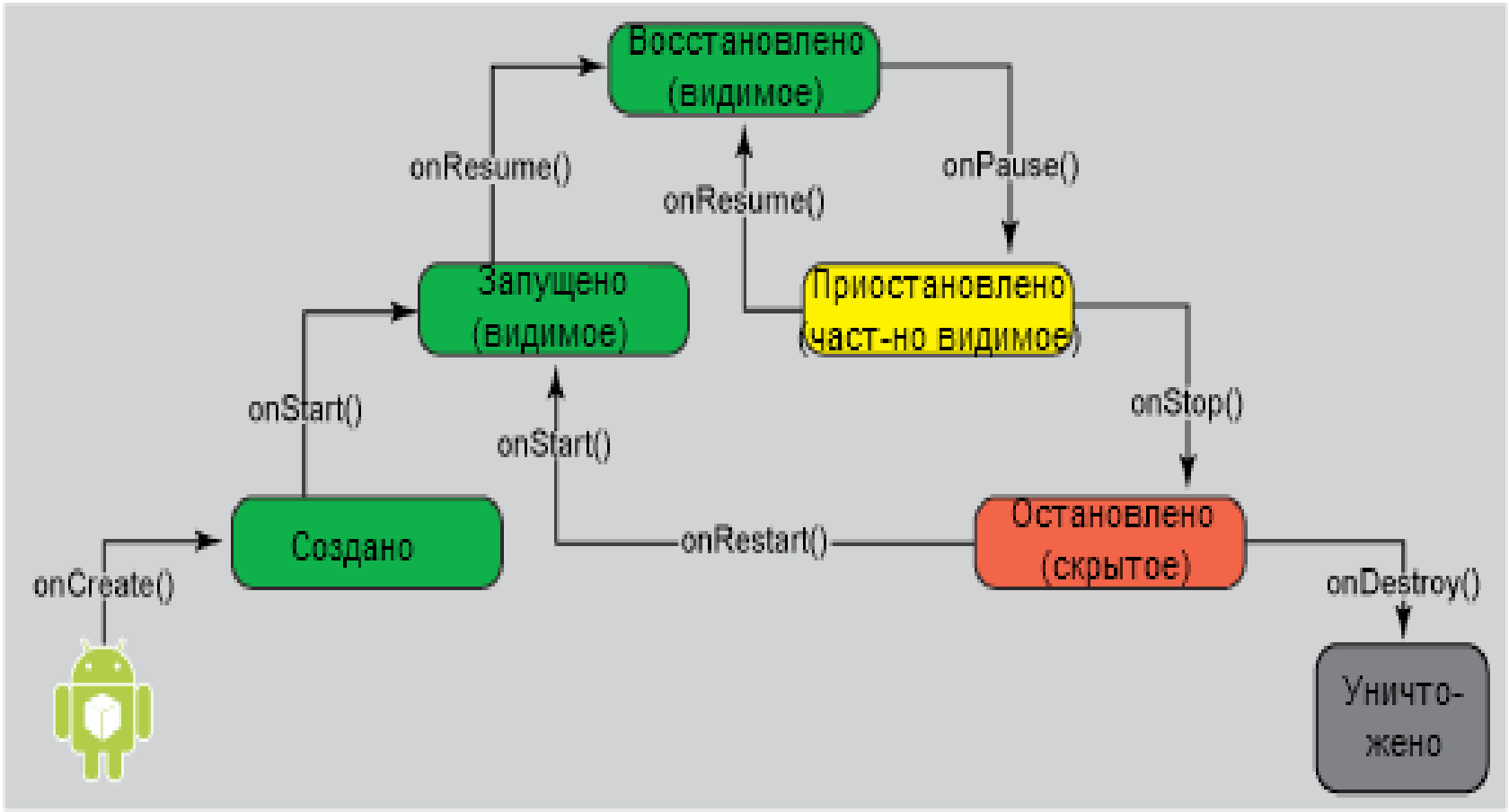
Приложение может состоять из нескольких *Деятельностей.*

(Например, у приложения обмена сообщениями мог бы быть один экран, который показывает список контактов, второй экран, чтобы написать сообщение выбранному контакту, и другие экраны, чтобы делать обзор старых сообщений или изменить настройку).

Каждая *Деятельность* может использовать дополнительные окна (например, диалоговое окно).

Структура приложения под платформу Андроид

Жизненный цикл Деятельности



Структура приложения под платформу Андроид

Деятельность может находиться в одном из трех состояний:

- **Восстановлена** (resumed) — находится на переднем плане и имеет фокус для взаимодействия с пользователем;

- **Приостановлена** (paused) — потеряло фокус, но всё ещё видно пользователю (поверху находится другая *деятельность*, которая или прозрачна или закрывает частично. Приостановленная *деятельность* полностью «живая» (состояние сохранено и оно привязано к оконному менеджеру), но может быть уничтожена системой в случае нехватки памяти);

- **Остановлена** (stopped) — полностью перекрыто другой *деятельностью* (больше не видна пользователю и будет уничтожена системой, когда понадобится память).

Структура приложения под платформу Андроид

Метод *onCreate()* вызывается один раз при создании *Деятельности* (производится первоначальная настройка интерфейса, связывание данных, настройка служб и потоков и т.п.).

Метод *onStart()* вызывается для перехода *Деятельности* к взаимодействию с пользователем (становится видимой).

Метод *onResume()* передает весь фокус ввода *Деятельности* (для инициализации компонентов, регистрации любых широковещательных приемников или других процессов).

Метод *onPause()* вызывается когда пользователь решает перейти к работе с новым окном (остановить анимацию и другие действия, которые загружают процессор; зафиксировать несохранённые данные; освободить системные ресурсы, например, обработку данных от GPS).

Структура приложения под платформу Андроид

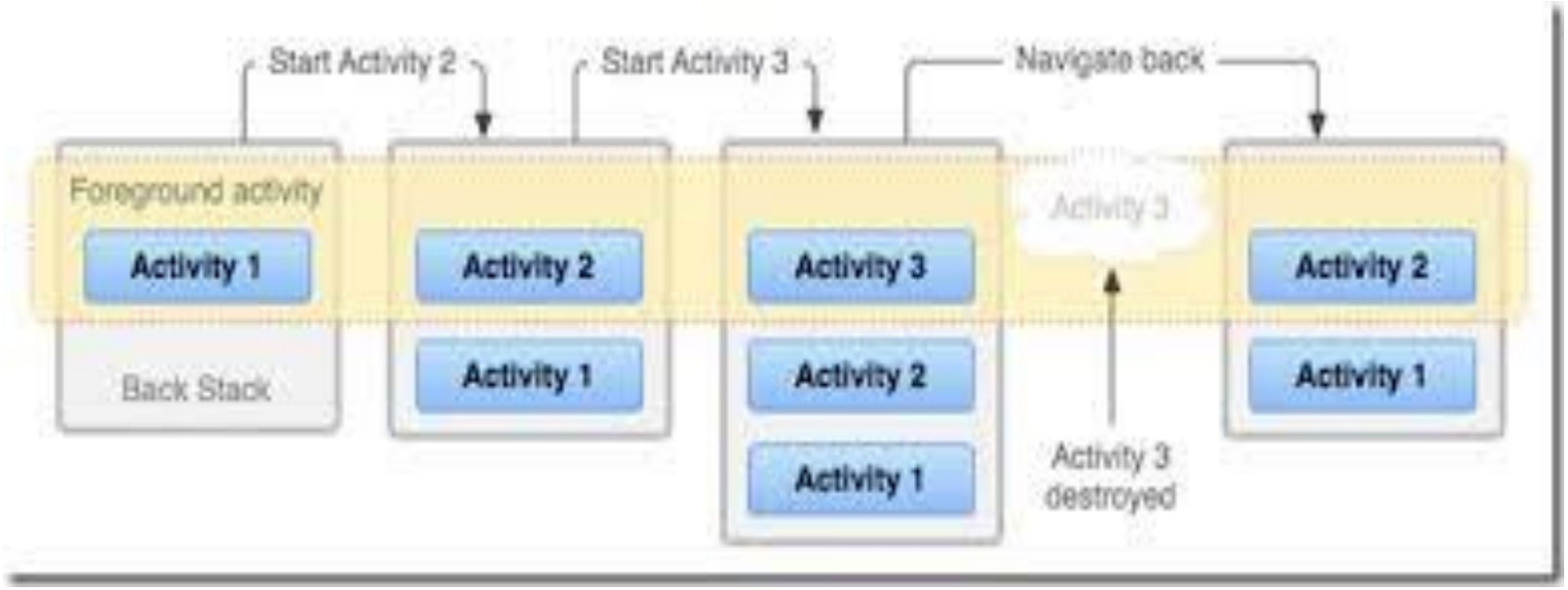
Метод *onStop()* вызывается, когда окно становится невидимым для пользователя (если была запущена другая *Деятельность*, перекрывшая окно текущей активности; можно сохранять данные: для приостановки сложной анимации, потоков, отслеживания показаний датчиков, запросов к GPS, таймеров, *Служб* или других процессов, которые нужны исключительно для обновления пользовательского интерфейса).

Метод *onRestart()* вызывается после того, как *Деятельность* снова была запущена пользователем (для специальных действий, которые должны выполняться только при повторном запуске).

Метод *onDestroy()* вызывается по окончании работы *Деятельность* (удаляет все статические данные активности. Отдаёт все используемые ресурсы).

Стек Деятельностей

Когда новая *Деятельность* запускается, то она помещается в вершину стека, выделенного приложению.



Структура приложения под платформу Андроид

Компонент «СЛУЖБА»

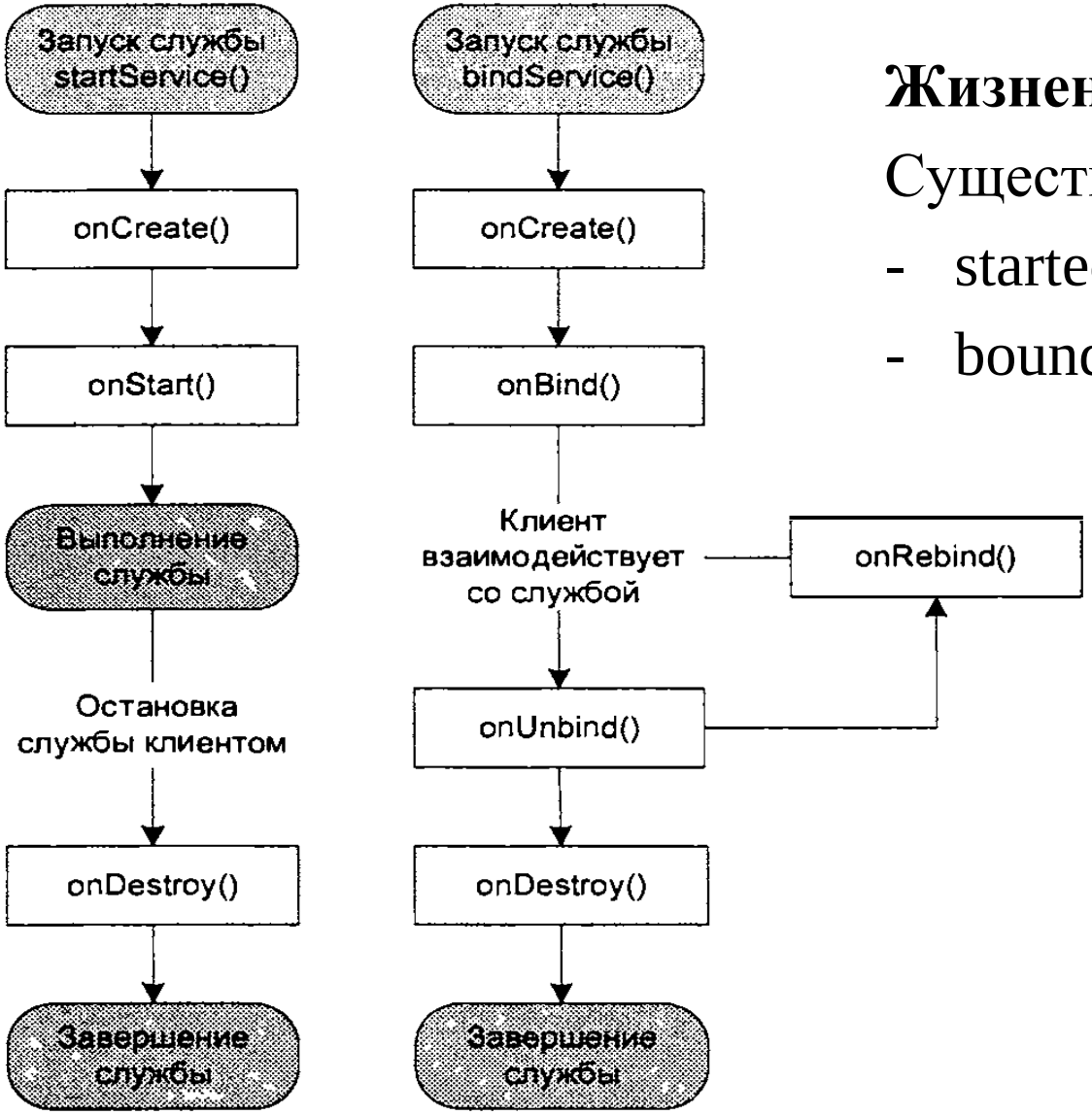
Не имеет визуального представления (пользовательского интерфейса).

Выполняется в фоновом режиме в течении неопределенного времени (например, универсальный проигрыватель, запускающий песни из плейлиста).

Приложения могут подключаться к *Службе* или запускать (если она уже не работает), а также останавливать запущенные.

Когда есть подключение к *Службе*, то обращение к ее функциям осуществляется через интерфейс, предоставленный этой службой.

Структура приложения под платформу Андроид



Жизненный цикл Служб

Существует 2 формы:

- started (запущена);
- bound (привязана).

Структура приложения под платформу Андроид

Если компонент запускает *Службу* вызовом метода *startService()* (который в результате приведет к вызову метода *onStartCommand()*), то *Служба* будет работать, пока не остановит сама себя методом *stopSelf()*, или пока другой компонент не остановит её вызовом метода *stopService()*.

Если компонент вызвал метод *bindService()* для создания *Службы* (и метод *onStartCommand()* не вызвался), то *Служба* работает, пока хотя бы один компонент остался привязанным к ней (как только убраны все привязки клиентов к *Службе*, система ее уничтожит).

Процессы Android

Как только один из компонентов будет затребован, система запускает процесс, состоящий из одного потока выполнения.

Все компоненты инициализируются в основном потоке.

Система может завершить выполнение потока в случае нехватки памяти или если память затребована другим, более важным процессом.

При выборе процесса для уничтожения оценивается его важность с точки зрения пользователя (т.е. в первую очередь завершится процесс с невидимыми *Деятельностями*).

Структура приложения под платформу Андроид

Типы процессов и их приоритеты

Структура приложения под платформу Андроид

Активный процесс (процесс переднего плана) – это процесс приложения:

- с которым пользователь взаимодействует в данный момент;
- выполняющий *Службу*, связанную с *Деятельность*, с которой взаимодействует пользователь;
- содержащий *Службу* и выполняющий метод обратного вызова, который определен для этой службы;
- содержащий *Приемник широковещательных намерений* и выполняющий его метод обратного вызова для приема намерения.

Структура приложения под платформу Андроид

Видимый процесс — это процесс, который:

- имеет *Деятельность*, видимую конечному пользователю в данный момент времени (деятельность потеряла фокус ввода, но еще видна пользователю);
- имеет *Службу*, связанную в данный момент с *Деятельностью*, находящейся на переднем плане (или частично перекрытую).

Сервисный (служебный) процесс – это процесс, содержащий выполняемую на данный момент *Службу*, не относящуюся к предыдущим типам.

Структура приложения под платформу Андроид

Фоновый процесс – это процесс, содержащий *Деятельность*, которая не видна пользователю (существует множество фоновых процессов, работа которых завершается по принципу "последний запущенный закрывается последним").

Пустой процесс – это процесс, не содержащий никаких активных компонентов приложения и используется как «кэш» для уменьшения затрат при вызове компонента.

Структура приложения под платформу Андроид

Особенности

1) Если в одном процессе выполняются несколько компонент, то система оценивает приоритет процесса по компоненте с самым высоким приоритетом;

2) Процесс, который обслуживает другой процесс, имеет приоритет не ниже обслуживаемого процесса;

3) Если для выполнения фоновой деятельности требуется длительное время, то ее лучше запустить как отдельную *Службе*, а не породить в потоке с этой деятельностью (сервисный поток имеет более высокий приоритет, чем фоновый).

Компонент «КОНТЕНТ-ПРОВАЙДЕР»

Данные приложения могут храниться:

- в файлах,
- базе данных SQLite,
- любом другом виде.

Контент-провайдер обеспечивает доступ к данным приложения из других приложений (т.е. можно конфигурировать собственные контент-провайдеры, чтобы разрешить доступ к своим данным из других приложений; использовать контент-провайдеры других приложений для доступа к их хранилищам данных).

Компонент «ПРИЕМНИК ШИРОКОВЕЩАТЕЛЬНЫХ НАМЕРЕНИЙ»

Используется для получения внешних событий и реакции на них (т.е. приложение может иметь любой число приемников, чтобы ответить на любые уведомления, которые считает важными).

Например, об изменениях в состоянии сетевого подключения или в уровне заряда батареи.

Приемники широковещательных намерений не имеют пользовательского интерфейса, однако могут запускать *Деятельность* в ответ на полученное уведомление или показать уведомление для информирования пользователя.

Активация компонентов

Поскольку система запускает каждое приложение в отдельном процессе с правами доступа к файлам, которые ограничивают доступ к ним другим приложениям, *приложения не могут непосредственно* активировать **компонент** другого приложения.

Однако система Android может. Поэтому, чтобы использовать **компонент** другого приложения, необходимо сообщить системе, что есть **Намерение** (**Intent**) запустить **компонент** какого-либо приложения, и система запустит этот компонент.

Структура приложения под платформу Андроид

Компоненты – *Деятельность*, *Служба* и *Приемник широковещательных намерений* – активируются через асинхронные сообщения – **Намерения**.

Намерения связывают различные компоненты друг с другом в реальном времени (безотносительно того, являются ли это компоненты частью одного приложения или разных).

Компонент – **Контент-провайдер** активируется не *Намерением*, запросом от класса **ContentResolver**.

Структура приложения под платформу Андроид

Существуют:

- Явные намерения;
- Неявные намерения.

Явные намерения – определяют адресата по имени, имеют набор значений и используются в основном для сообщений внутри одного приложения.

Неявные намерения – не определяют адресата и используются в основном для активации компонент в другом приложении.

При отсутствии адреса система Android просматривает **Фильтры-Намерений** всех приложений и находит тот компонент, *фильтр-намерений* которого наиболее подходит для его выполнения.

Структура приложения под платформу Андроид

Объект **Intent (Намерение)** содержит информацию, представляющую интерес:

1) *для компонента*, который получает **намерение** и данные, которые передаются этому компоненту;

2) *для системы Android* — имя компонента, который должен обработать **намерение** и набор параметров запуска этого компонента.

Структура приложения под платформу Андроид

Поля объекта **Intent**

- *Имя компонента*, который должен обработать **намерение**. (комбинация полного имени класса целевого компонента (например, "MainActivity") и набора имен пакета в файле манифеста приложения, где компонент постоянно находится (например, "com.samples.yourproject")).

Если оно установлено, то **намерение** поставляется экземпляру определяемого класса.

Если имя не установлено, то система использует другую информацию в объекте **Intent**, чтобы определить местонахождение подходящего адресата.

Структура приложения под платформу Андроид

- *Действие* определяет операцию, которая будет выполнена (класс **Intent** содержит множество констант действия: MAIN, VIEW, PICK, EDIT и т.д.; можно определять собственные действия для активизации *Деятельности* - тогда к действию добавляется имя пакета приложения в качестве префикса). Например

  com.samples.yourproject.CUSTOM_ACTION;

- *Данные* - это URI данных и тип MIME для этих данных (разные *Деятельности* соединены с разными видами спецификаций данных);

- *Категория* содержит дополнительную информацию о виде компонента, который должен обработать намерение (класс **Intent** определяет несколько констант CATEGORY, например, CATEGORY_BROWSABLE);

Структура приложения под платформу Андроид

- *Дополнения* - это пары «ключ/значение» для дополнительной информации, которая нужна компоненту (например, действие ACTION_TIMEZONE_CHANGED имеет дополнение *time-zone*, которое идентифицирует новый часовой пояс и т.д.);

- *Флаги* указывают системе, как запускать *Деятельность* (например, какому заданию должна принадлежать активность) и как обработать ее после того, как *Деятельность* запустили (например, принадлежит ли она списку недавних активностей).

Структура приложения под платформу Андроид

Если **Намерение** запрашивает выполнение какого-либо действия с указанным набором данных, то системе нужно уметь выбрать приложение (или компонент) для обработки этого запроса.

Для этого применяются **Фильтры-Намерений**, которые используются для регистрации *Деятельностей*, *Служб* и *Приёмников широковещательных Намерений* в качестве компонентов, способных выполнять заданные действия с конкретным видом данных.

В **Фильтре-Намерений** декларируется только три составляющих объекта **Intent**: *действие*, *данные*, *категория*.

Структура приложения под платформу Андроид

Файл манифеста AndroidManifest.xml

Назначение: предоставляет системе основную информацию по приложению:

- определяет имя пакета приложения (уникальный идентификатор для приложения);
- описывает компоненты приложения – Activities, Services, Broadcast Receivers и Content Providers (определяет имена классов, реализующих каждый из компонентов и объявляет возможности);
- содержит список необходимых разрешений для обращения к защищенным частям API и взаимодействия с другими приложениями;
- объявляет разрешения, которые сторонние приложения обязаны иметь для взаимодействия с компонентами данного приложения;
- объявляет минимальный уровень API Android, необходимый для работы приложения;
- перечисляет связанные библиотеки.

Структура приложения под платформу Андроид

Например,

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
       ...
   </application>
</manifest>
```

Структура приложения под платформу Андроид

Для определения компонентов используются:

<activity> для **Activity** (деятельности)

<service> для **Service** (службы)

<receiver> для **Broadcast Reciever** (приемника широковещательных сообщений)

<provider> для **Content Providers** (поставщики данных)

Если компоненты не заявлены в манифесте, то они не видны системе и, следовательно, никогда не могут быть запущены.

Кроме, **Broadcast Reciever** может создаваться динамически в коде и регистрироваться с помощью вызова *registerReceiver()*.

Структура приложения под платформу Андроид

РЕСУРСЫ

Использование ресурсов даёт возможность изменять некоторые части приложения без модификации исходного кода, а также позволяет оптимизировать приложение для различных устройств (с различным языком интерфейса или размером экрана)

Типы ресурсов:

- Изображения;
- Слои GUI (XML файлы);
- Объявления меню (XML файлы);
- Текстовые строки.

Структура приложения под платформу Андроид

Для каждого ресурса, включённого в приложение *Android*, определяется уникальный идентификатор (целое число) в файле **R.java**, которое можно использовать для ссылки на ресурс из кода или из других ресурсов определённых в XML.

Этот класс **R** генерируется на основе заданных ресурсов и создается во время компиляции приложения (нет смысла его редактировать вручную, потому что все изменения будут утеряны при повторной генерации).

Виджеты

Виджет – это графический элемент управления (объект класса **View**), который служит интерфейсом для взаимодействия с пользователем.

Типы виджетов:

- кнопки,
- текстовые поля,
- флажки,
- переключатели,
- списки.