**1. Уровни архитектуры Android**

* **Linux Kernel**: Базовый уровень. Управляет аппаратной частью.
* **Hardware Abstraction Layer (HAL)**: Связь между ядром и Android.
* **Native Libraries & Android Runtime**: Библиотеки и виртуальная машина (ART).
* **Application Framework**: API для разработки приложений.
* **Applications**: Ваши приложения.

**2. Архитектура приложения, основные компоненты**

* **Activity**: Экран приложения.
* **Service**: Фоновая задача.
* **Broadcast Receiver**: Обработка системных событий.
* **Content Provider**: Доступ к данным.

**3. Манифест приложения**

* Файл AndroidManifest.xml.
* Описывает приложение: имя, разрешения, компоненты (Activity, Service).

**4. Ресурсы приложения**

* Все, что не код: изображения, строки, стили.
* Сохраняются в папке res.

**5. Activity. Жизненный цикл**

* **Основные стадии**:
  1. onCreate() – создание.
  2. onStart() – видна пользователю.
  3. onResume() – активна.
  4. onPause() – неактивна.
  5. onStop() – скрыта.
  6. onDestroy() – уничтожение.

**6. Службы (Service)**

* Фоновая задача (например, музыка).
* **Жизненный цикл**: onCreate(), onStartCommand(), onDestroy().
* **Типы**:
  + Стартуемые (startService()).
  + Привязанные (bindService()).

**7. Контент-провайдеры (Content Providers)**

* Доступ к данным между приложениями.
* Пример: контакты телефона.

**8. Broadcast Receivers**

* Получают системные события (например, заряд батареи).
* Легкие, выполняют код и завершаются.

**9. Фрагменты. Жизненный цикл**

* **Фрагмент** – часть экрана.
* Жизненный цикл похож на Activity: onAttach(), onCreate(), onCreateView(), onDestroyView(), onDetach().

**10. Статическое и динамическое использование фрагментов**

* **Статическое**: описывается в XML.
* **Динамическое**: создается и управляется в коде.

**11. Списки и адаптеры**

* **ListView, RecyclerView** – для отображения списков.
* **Адаптеры**: связывают данные с элементами списка.

**12. Кастомные адаптеры для списков**

* Создаются для нестандартных элементов списка.
* Наследуются от BaseAdapter или RecyclerView.Adapter.

**13-14. Пользовательские компоненты**

* **Custom View**: создается с нуля, переопределяется метод onDraw().
* **Composite Component**: состоит из нескольких стандартных Views (например, кнопка с текстом и иконкой).

**15-16. Многопоточность**

* **Thread + Handler**: для работы в потоке и передачи данных в UI.
* **AsyncTask**: упрощает выполнение фоновых задач (устарел).

**17-19. Хранение данных**

* **Preferences**: настройки, хранят ключ-значение.
* **Файлы**: любые данные.
* **SQLite**: встроенная база данных.

**20-21. SQLiteDatabase и Cursor**

* SQLiteDatabase – класс для работы с БД.
* Cursor – для чтения результатов запроса.

**22-24. ORM и базы данных**

* **ORM (Object-Relational Mapping)**: автоматическое преобразование объектов в БД.
* Примеры: Room, ORMLite, GreenDAO.

**25-27. Воспроизведение и запись звука**

* **MediaPlayer**: для воспроизведения звука (диаграмма состояний: Инициализация → Подготовка → Воспроизведение).
* **SoundPool**: для коротких звуков.
* **AudioRecorder/MediaRecorder**: для записи звука.

**28-31. Android NDK**

* Позволяет использовать C/C++ в Android.
* **JNI**: интерфейс для связи Java и C.
* **Android.mk, Application.mk**: конфигурация сборки.
* Компиляция: через ndk-build.

**32-34. Сенсоры**

* **SensorManager**: управление сенсорами.
* Пример сенсоров: акселерометр, гироскоп.
* События обрабатываются через onSensorChanged().