android

- os for mobile devices
  - phone
  - phablet
  - tablet
  - netbook
  - notebook
  - others
    - smart watch
    - smart TV
    - smart fridge
    - smart car
- system software that contorls the hardware components

hardware components

desktop hardware

- input devices
  - keyboard
  - mouse, joystick
  - camera: video
  - mic : audio
  - scanner
- memory
  - type
    - static
      - which can not be replaced/upgraded/downgraded
      - faster/costiler than dynamic
      - e.g.
        - registers in CPU
    - dynamic
      - can be easily upgraded or degraded
      - types
        - DDR, DDR2, DDR3, DDR4 and DDR5
  - primary
    - RAM
  - secondary
    - storage
      - magnetic
        - hard disk drive
      - optical
        - CD, DVD, BluRay DVD, Laser Disk
      - electronic

        - SSD, flash
- output devices
  - monitor (display)
    - Cahod Ray Tube (CRT)
    - flat screen
      - Thin Film Transistor (TFT)
      - Light Emitting Diode (LED)
      - In Place Switching (IPS)
    - size
      - 13/15/17/21/27/32/34
  - printer, plotter
  - speker: audio
- power supply
  - SMPS: Switch Mode Power Supply
- processor
  - central processing unit (CPU)
    - manufactures
      - intel
      - AMD
    - family
      - x86 (32) or x64 (64)
      - Complex Instructions Set Computing (CISC)
      - consume more power
  - graphical processing unit (GPU)
  - co-processors
    - math
  - micro-controllers
    - hard disk
    - camera
- periphal ports
  - usb
    - versions
      - usb 1
      - usb 2
      - usb 3
    - types based on the connector
      - A
      - B
      - C
    - types based on the size
      - micro
  - graphical
    - hdmi
    - dvi
    - vga
- expansion slots

    - PCI: Peripheral Component Interconnect
        - Netwok Interface Card
        - grapics card
    - PCIe: PCI express
    - Accelerated Graphics Port
- motherboard
    - holds all the components

mobile hardware

processor

- family
    - Reduced Instructions Set Computing (RISC)
    - Advanced RISC Machine (ARM)
        - specification about the processor
    - comsume less power
- manufactureres
    - Apple: Ax, M1
    - Qualcomm: Snapdragon
    - Samsung: Exynos
    - Huawei: Kirin
    - MediaTek: MTek
- is known as SoC (Systems on Chip) from hardware perspective
- is known as PoP (Package on Package) from features perspective

motherboard

- Printed Circuit Board
- used to hold all the eletronics components
- form factor: mobile
- most of the components are soldered on the motherboard

display

- acts as both input and ouput device
- multi-touch screen
- oleo-phobic
- protected using gorilla glass
- type
    - registrive display
    - capacitive display (*)
- screen types
    - LED
        - Organic LED
        - Active Matrix OLED
    - IPS

- size
    - 3.5/4.5/5/5.5/6/6.7/6.9
- resolution
    - standard
    - high
    - ultra high

input components

- touch input: display/screen

- audio

    - 2 microphones
    - 1 for recording (taking audio in)
    - 2nd for noise cancellation

- video

    - front
        - VGA camera
            - record a VGA resultion
            - 640x480 or 1280x720
    - rear
        - HD/Ultra HD resoltuion
            - 1280x720 (half hd)
            - 1920x1080 (full hd)
            - 3840x2160 (ultra hd / 4K)

- **sensor**

    - tri-axial accelerator
    - gyroscope
    - proxymity
    - temperature
    - pressure
    - ambiant light sensor

output components

- display
- audio
    - speaker
        - mono
        - stero
        - dolby vision

memory

- primary
  - RAM
  - Low Power DDR
  - 1GB to 12GB
  - soldered / hard wired on the motherboard
  - can not upgrade or degrade the RAM
  - also known as embedde RAM
- secondary (storage)
  - internal
    - flash storage
    - electronic storage
  - external
    - Secure Digital card

network connectivity

- BlueTooth

- NFC

- internet connectivity

  - WiFi
  - cellular
    - 1G
    - 2G
    - 3G
    - 4G
    - 5G

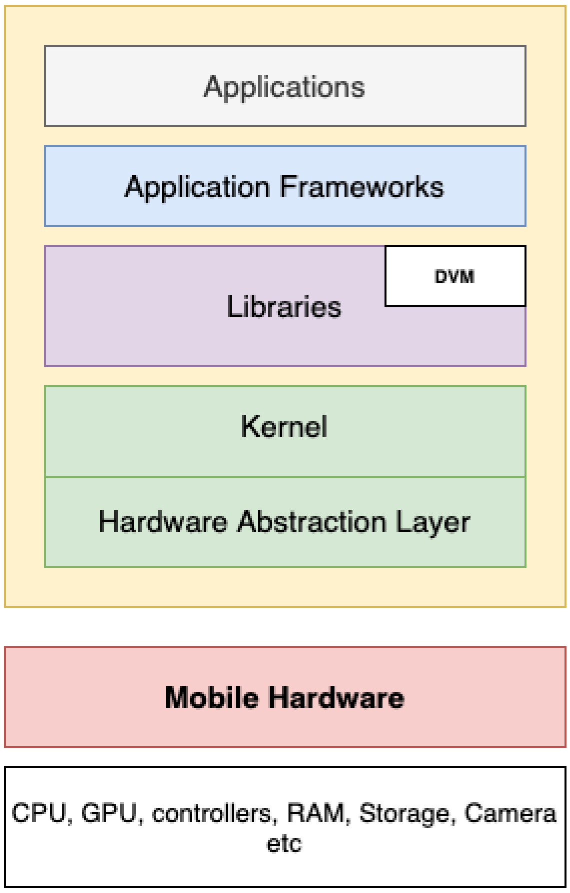
history

- andy rubin started Android Inc 2003
- andy met lary page in 2005
- google took over the android inc 2005
- google formed open handset alliance (OHA)
  - group of manfacturers - sony, samsung
  - cell providers - Orange
- google made the Android open source and free

android

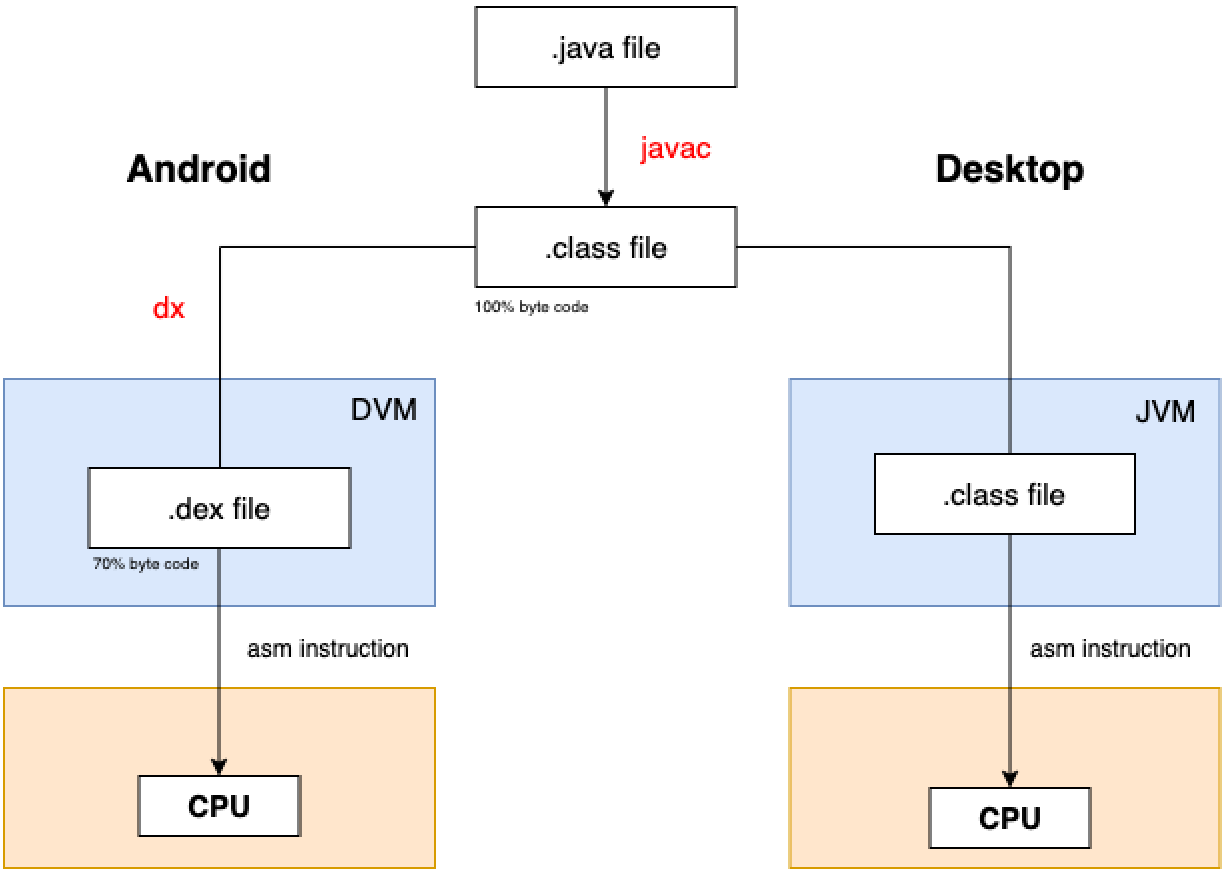
- os for mobile devices
- free and open source OS
- uses linux kernel at its core

architecture


Applications
Application Frameworks
Libraries
DVM
Kernel
Hardware Abstraction Layer
Mobile Hardware
CPU, GPU, controllers, RAM, Storage, Camera etc

- **kernel**
    - uses linux as kernel
    - responsibilities
        - file system management
        - CPU scheduling
        - process management
        - Hardware Abstraction Layer (HAL)
            - set of device drivers
            - camera driver, sd card driver etc
        - networking functionality
- **libraries**
    - exposes functionality provided by kernel
    - example
        - libsqlite: SQLite database
        - OpenGL: 3D graphics
        - SGL: standard graphics libray used for 2D graphics
        - SSL: secure socket layer (https)
    - dalvik
        - JVM for android
        - used to execute the android (java) applications
        - differences

        - JVM (Hotspot) is develoepd for desktop
          Dalvik is developed for android
        - JVM implementation is stack (memory) based
          Dalvik implementation is register based
        - class files developed for JVM contain 100% byte codes
          dex file developed for dalvik contains 70% byte codes and 30% native (assembly) code
    - .dex: dalvik executable



- **application frameworks**
    - Java application frameworks used for developing the android applications
    - e.g.
        - ActivityManager
        - PackageManager
- **applications**
    - all android applications run in this layer
    - contains
        - default / system applications like calculator/settings/calendar/phone etc
        - applications downloaded from play stores

android development

- **requirements**

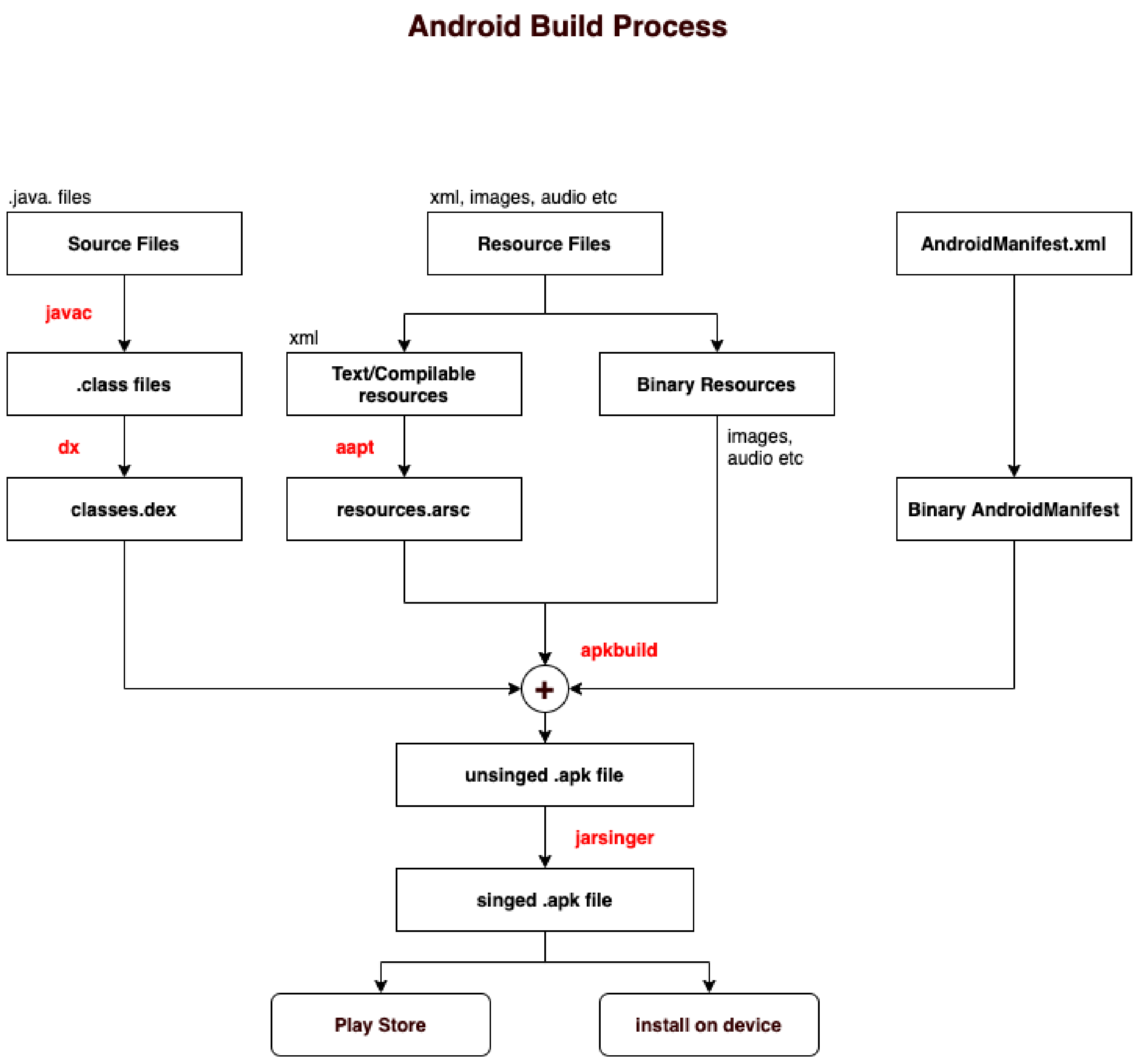
    - language: Java/Kotlin
    - sdk: software developent kit
        - collection of libraries: **libdalvik, libjar**
        - collection of header files/packages/namespaces: **packages (android.os, andorid.content)**
        - documentation: **https://developer.android.com**
        - toolchain
            - compiler/interpreter: **javac/dx**
            - linker
            - assmebler: **javac**
            - disassembler: **java -p**
            - decompiler:
            - debuggers: **jdb**
            - other tools
                - testing: **monkeyrunner**
                - android bridge: **adb**
                - emulator: **qemulator**
                - gradle: **used to build the application**
        - IDE: **Android Studio**
        - runtime/virtual machine/emulator/simulator: **real device/OS or Emulator (qemu/bluestack)**

android application

- android does not support console application
- any android application has to have at least one activity
- properties
    - application name
        - used only by the developer to differentiate the applications
        - different than the application label that is displayed on launcher activity
    - package name
        - package in android is representing an application
        - every application must have a unique package name across the playstore
        - e.g.
            - com..
            - com..
            - reverse dns format
        - never use example domain (otherwise the application will be rejected by playstore)
            - e.g. com.example.app1
    - minimum SDK
        - the minimum version of the android, the application will support
        - e.g.
            - if the minimum version is Marshmellow (6.0) then
                - application will support 6.0, 7.0 etc
                - application will NOT support 5.1, 5.0, 4.4 etc

android application hiearchy

- **app**
    - one of the modules in the project
    - represents android application (which will generate executable apk file)
    - module
        - collection of different files
        - represents one of the types of packages
    - **manifests**
        - **AndroidManifest.xml**
            - contains the application configuration
            - number of components (activities, services, broadcast receiver etc)
            - permissions required to execute the application
    - **java**
        - package name
            - java source code
    - **res**
        - directory represents resource files
        - **drawable**
            - one which can be drawn statically
            - e.g. images, xml
        - **layout**
            - represents activity's UI
            - xml file(s) where activitys ui code will be placed
        - **mipmap**
            - used to keep resolution dependent images
            - types
                - mdpi: mediam density per inch
                - hdpi: high density per inch
                - xhdpi: extra high density per inch
                - xxhdpi: extra extra high density per inch
                - xxxhdpi: extra extra extra high density per inch
        - **values**
            - contains different type of resources
                - **colors.xml**: contains the color resources
                - **strings.xml**: contains the string resources (used in localization)
                - **themes.xml**: contains different style / themes
        - **anim**: contains xml file defining the animation
        - **menu**: contains the xml file defining the activity menu items
- **Gradle Scripts**
    - gradle is used to build the android application
    - **build.gradle (Project)**: contains steps to build entire project
    - **build.gradle (Module)**: contains the steps to build the respective module
    - **proguard-rules.pro**: contains the rules to obfuscate the application source code



activity

- nothing but the GUI that is presented to the user so that user can interact with your application
- GUI container that holds different widgets (like button, textview etc)
- represents a screen

Widgets

- View
  - is a superclass of every widget in android
- TextView
  - used to display readonly text on the screen

  -
    ```
    // password
    textPassword = new TextView(this);
    ```

```
    textPassword.setText("Password");
    textPassword.setTextSize(19);
    layout.addView(textPassword);
```

- EditText
  - used to get input from user

  -
    ```
    // edit password
    EditText editPassword = new EditText(this);
    editPassword.setHint("password");
    layout.addView(editPassword);
    ```
- Button
  - used to perform an action
  - to peform an action set the onClickListener

  -
    ```
    // button save
    Button buttonSave = new Button(this);
    buttonSave.setText("Save");
    buttonSave.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Log.i("save button", "save button clicked");
        }
    });
    ```
- RadioButton
  - used to get an input in terms of options button
  - RadioGroup is used to group the radio buttons
    - it provides a functionality to select only one radio button at a time
    - by default orientation is set to vertical
    - it is a subclass of LinearLayout

  -
    ```

    <RadioGroup
        android:layout_weight="0.3"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <RadioButton
            android:text="Male"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    ```

```
    <RadioButton
        android:text="Female"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

</RadioGroup>
```

important methods

- activity
    - finish()
        - used to close the current activity
        - can close only the current activity

User interface

- dialog box
    - Toast
        - used to show a string message to the user
        - can not be customized

        -
          ```
          // param1: context
          // param2: message to be displayed on screen
          // param3: duration
          Toast.makeText(MainActivity.this, "Please enter email",
          Toast.LENGTH_SHORT).show();
          ```
    - Snackbar
    - AlerterDialog
    - FragmentDidalog
    - Custom Dialog

important points

- **Inflate XML file**
    - will be carried out by inflater
    - Inflater
        - LayoutInflater
            - used to inflate xml files for activities
            - used either explicitly or implcitly (by setContentView())
        - MenuInflater
            - used to inflate xml file for menu

- process
    - read the xml file
    - go through all the UI elements in the xml file
    - create an object for every element declared in xml file
    - build the hiearchy of the objects
    - return the object of root element

- **Log**
  - used to log errors/warnings etc
  - types
    - information (i)
    - debug (d)
    - warning (w)
    - error (e)
    - verbose (v)
  - params to the methods
    - tag: more info about the message
    - message: message to be added to the logcat

- **startup activity**
  - can be set by configuring intent-filter in AndroidManifest.xml

    -
      ```
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category
      android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
      ```
    - where
      - **action**: android.intent.action.MAIN means the activity is the main (startup) activity
      - **category**: android.intent.category.LAUNCHER will create a shortcut icon for this activity on launcher screen
  - the application with following manifest file will start with MainActivity
    - as the MainActivity has the intent-filter declaration

  -
    ```
      <application>
          <activity
              android:label="Second Activity"
              android:name=".SecondActivity">
          </activity>
          <activity
              android:label="Main Activity"
              android:name=".MainActivity">
    ```

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

- the application with following manifest file will start with SecondActivity
    - as the SecondActivity has the intent-filter declaration

-
  ```
    <application>
        <activity
            android:label="Second Activity"
            android:name=".SecondActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:label="Main Activity"
            android:name=".MainActivity">
        </activity>
    </application>
  ```

RecyclerView

- used to render the list like view
- replacement for ListView
- advantages
    - manages memory efficiently
    - simple to use
    - reasy to reuse
- **steps**
    - add the dependency
        - visit url :https://developer.android.com/jetpack/androidx/releases/recyclerview
        - add the following line(s) in build.gradle (app) file

-
      ```
      dependencies {
          implementation
      "androidx.recyclerview:recyclerview:1.2.1"
          implementation "androidx.recyclerview:recyclerview-
      selection:1.1.0"
      }
      ```

- add recyclerview in layout xml file

  -
      ```
        <androidx.recyclerview.widget.RecyclerView
          android:id="@+id/recyclerView"
          android:layout_width="match_parent"
          android:layout_height="match_parent"/>
      ```

- in activity java add reference and connect with ui element

  -
      ```
        // reference
        RecyclerView recyclerView;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            .
            .

            // connect with UI element
            recyclerView = findViewById(R.id.recyclerView);
        }
      ```

- create recyclerview adapter

  -
      ```
      // Adadpter
      public class ContactAdapter extends
      RecyclerView.Adapter<ContactAdapter.ViewHolder> {

          // Context used to create views
          Context context;

          // data source
          ArrayList<Contact> contacts;
      ```

```
    public ContactAdapter(Context context,
ArrayList<Contact> contacts) {
        this.context = context;
        this.contacts = contacts;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {

        // get the layout inflater to inflate recycler
itemxml file
        LayoutInflater inflater =
LayoutInflater.from(context);

        // inflate the recycler item xml file
        LinearLayout layout = (LinearLayout)
inflater.inflate(R.layout.recycler_item_contact, null);

        // create an object of view holder class
        return new ViewHolder(layout);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder,
int position) {

        // get the contact at the position from the data
source
        Contact contact = contacts.get(position);

        // display the contact details
        holder.textName.setText(contact.getName());
        holder.textAddress.setText(contact.getAddress());
        holder.textEmail.setText(contact.getEmail());
        holder.textPhone.setText(contact.getPhone());
    }

    @Override
    public int getItemCount() {
        return contacts.size();
    }

    // ViewHolder to hold the view that will be added inside
every item
    public static class ViewHolder extends
RecyclerView.ViewHolder {

        // references
        TextView textName, textEmail, textAddress,
textPhone;

        public ViewHolder(@NonNull View itemView) {
                super(itemView);

                // connect ui element with references
                textName = itemView.findViewById(R.id.textName);
                textAddress =
itemView.findViewById(R.id.textAddress);
                textEmail =
itemView.findViewById(R.id.textEmail);
                textPhone =
itemView.findViewById(R.id.textPhone);
            }
        }
}
```

- set the adapter to the recycler view

  -
    ```
    // adpter
    ContactAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
      .
      .
      .

      // connect with UI element
      recyclerView = findViewById(R.id.recyclerView);

      // set layout manager for recycler
      LinearLayoutManager layoutManager = new
    LinearLayoutManager(this);
      recyclerView.setLayoutManager(layoutManager);

      // instantiate adapter
      adapter = new ContactAdapter(this, contacts);

      // set adapter to recyclerview
      recyclerView.setAdapter(adapter);
    }
    ```

menu

- types
  - options menu
    - is provided for entire activity

        - steps to enable options menu programmatically

            -
              ```
              @Override
              public boolean onCreateOptionsMenu(Menu menu) {
                  // add the menu items here
                  menu.add("Add Contact");
                  menu.add("Close")
                  return super.onCreateOptionsMenu(menu);
              }

              @Override
              public boolean onOptionsItemSelected(@NonNull
            MenuItem item) {
                  // check which menu item is selected
                  if (item.getTitle().equals("Add Contact")) {
                      Intent intent = new Intent(this,
            AddContactActivity.class);
                      startActivityForResult(intent, 0);
                  } else if (item.getTitle().equals("Close")) {
                      finish();
                  }
                  return super.onOptionsItemSelected(item);
              }
              ```
    - context menu

Database

- Android supports a database named SQLite
- **SQLite**
    - lite version of SQL
    - supports only basic features
        - file as a database (create a file to create a new database)
        - tables (rows + columns)
        - functions
    - by default available on android (no external installation is needed)
    - is getting used by the defauly/system applications
        - like calendar, contacts, clock etc
- **steps**
    - create a subclass of SQLiteOpenHelper
        - used to manage the database
        - defines

        - onCreate()
            - used to initialize the schema (table(s))
            - gets called only once
        - onUpgrade()
            - used to upgrade the schema
            - gets called only once

    -
      ```
          public class DBHelper extends SQLiteOpenHelper {
          private static String TAG = "DBHelper";

          // database name
          private static String DB_NAME = "persondb.sqlite";

          // database version
          // - by default it has to start from 1
          private static int DB_VERSION = 1;

          public DBHelper(@Nullable Context context) {
              super(context, DB_NAME, null, DB_VERSION);
          }

          @Override
          public void onCreate(SQLiteDatabase sqLiteDatabase) {
              Log.i(TAG, "initializing schema");

              // initialize the schema
              // varchar -> TEXT
              sqLiteDatabase.execSQL("create table person (" +
                      "id integer primary key autoincrement, " +
                      "fullName TEXT, " +
                      "address TEXT, " +
                      "phone TEXT, " +
                      "email TEXT)");
          }

          @Override
          public void onUpgrade(SQLiteDatabase sqLiteDatabase, int
      i, int i1) {

          }
      }
      ```

- perform the required operation

    -
      ```
      // perform insert operation

      // step1: get the DBHelper object
      ```

```
DBHelper helper = new DBHelper(this);

// step2: get SQLiteDatbase reference
SQLiteDatabase db = helper.getWritableDatabase();

// step3: execute the query to perform the operator

// step4: close the datbase
db.close();
```

Service

- one of the components in android
- responsible for
    - sharing the logic
    - running the code in the background
- to create a service
    - create a class and extend it from android.app.Service

        -
          ```
          public class MyService extends Service {

              private static String TAG = "MyService";

              @Override
              public void onCreate() {
                  super.onCreate();
                  Log.i(TAG, "onCreate()");
              }

              @Nullable
              @Override
              public IBinder onBind(Intent intent) {
                  Log.i(TAG, "onBind()");
                  return null;
              }

          }
          ```
    - register the service into AndroidManifest.xml

        -
          ```
          <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
          ```

```
  android:label="@string/app_name"
  android:roundIcon="@mipmap/ic_launcher_round"
  android:supportsRtl="true"
  android:theme="@style/Theme.Application1">

  <!--
    registration of service
    - android:name
      - service class name
    - android:exported
      - whether the service can be used by other
applications(s)
      - by default the service is always exported
    - android:enabled
      - whether to enable the service
   -->
  <service
      android:enabled="true"
      android:exported="true"
      android:name=".service.MyService" />

</application>
```

- ways to start service
  - bound services
    - bound to the application for performing the task
    - lifecycle methods
      - onCreate()
        - get called when the service componets gets created
        - suppose to initialize the service
        - **do not perform long running jobs here**
      - onBind()
        - used to let the component know that the service has started
        - used to perform long running jobs
        - **long running jobs must be executed inside a thread**
      - onUnbind()
      - onDestroy()
    - application can control the service
      - starting

        -
          ```
            // intent: provides the service that needs to be
          started
            // connection: used as callback
          ```

```
  // flag: BIND_AUTO_CREATE
  bindService(intent, connection,
BIND_AUTO_CREATE);

  .
  .
  .

  // used to let the compoenent know when the
service gets
  // - connected => onServiceConnected()
  // - disconnected => onServiceDisconnected()
  ServiceConnection connection = new
ServiceConnection() {

      @Override
      public void onServiceConnected(ComponentName
componentName, IBinder iBinder) {
          Log.i(TAG, "onServiceConnected()");
      }

      @Override
      public void
onServiceDisconnected(ComponentName componentName)
{
          Log.i(TAG, "onServiceDisconnected()");
      }
  };
```

- stopping

  -
    ```
      // stop the service
      // which was started in bound mode
      unbindService(connection);
    ```

- unbound services
  - not bound to a specific application
  - always runs in the background
  - also known as started service
  - start when the android boots up
  - e.g.
    - LocationService

        - NotificationService
        - MusicService
        - BlueToothService
        - TelephonyService

        -
          ```
            public void startMyServiceUnbound(View view) {
                Log.i(TAG, "starting MySerivce in unbound mode");

                Intent intent = new Intent(this,
          MyService.class);

                // start the service in unbound mode
                startService(intent);
            }

            public void stopMyServiceUnbound(View view) {
                Log.i(TAG, "stopping MySerivce (started in unbound
          mode)");

                Intent intent = new Intent(this,
          MyService.class);

                // stop the service started in unbound mode
                stopService(intent);
            }
          ```

BroadcastReceiver

- the component used to receive broadcasted message(s)
- broadcasted message
    - intent sent by the same or different application or OS
- to add a broadcast receiver
    - create a class and extend from android.content.BroadcastReceiver

        -
          ```
          ```
    - register the BroadcastReceiver in AndroidManifest.xml

        -
          ```
          ```

questions

- what are the components in android?
    - activity
    - service
    - broadcast receiver
    - content provider
- what is Binder ?
    - it is IPC (interprocess communication) mechanism in android
- which are the ways to start the service in android
    - bound
    - unbound
- activity lifecycle
- service lifecycle
- how to add database support in the application
- what are the differences between sqlite and mysql
-