Stuff to remember:

Views:

* + two types: UIComponents and Container
    - UIComponents
      * TextView
        + Append(): adds on to previous text
        + setText(): overwrites what was previously there as text
    - Container: examples:
      * FrameLayout
      * ScrollLayout
      * RelativeLayout
      * ConstrainedLayout
  + properties: described inside of an XML file 🡪 in the res/layout folder
  + setContentView(R.layout.activity\_main);
    - R: dynamically pointing to the res folder
      * E.g. R.layout, R.string
    - setContentView: inflates the layout
* how to import an existing project: File 🡪 New 🡪 Import Project
* android:id="@+id/tv\_toy\_names"
  + @refers to something
  + + : create if not already there
* findViewById(R.id.[nameOfId]); 🡪 dynamically have access to a certain View in the source code
* Responsive Design: layouts that adjust to the size of the screen
* Logging
  + Command: log.x(String tag, String message)
    - Tag: usually class name
      * Error (x=e): crash
      * Warn (x=w): can still run
      * Debug (x=d):
      * Verbose:
      * Wtf: errors that should never happen
* Res directory
  + Values: strings or integers
  + Drawable: bitmap types and shapes
  + Layouts: XML layouts for the app
  + Menu
  + Color
  + …
* Menu
  + Create separate “menu directory” in res
  + Inside this directory, create a menu res file
  + Specify the menu in XML
    - Item
    - app:showAsAction “ifRoom”
    - orderInCategory
  + use the Menu in Java
    - onCreateMenu
    - onOptionsItemSelected



* Toasts
  + Context context = MainActivity.this;  
    String message = "Search clicked"  
    Toast.*makeText*(context, message, Toast.*LENGTH\_SHORT*);
* Uri
  + Uri.parse(<Base URL>).buildUpon().appendQueryParameter(PARAM\_..., param). … .build();



* URL
  + new URL(Uri.toString());
  + doing a connection:



* Permissions
  + to be added in the manifest: <uses-permission..>
* Async task (<https://developer.android.com/reference/android/os/AsyncTask.html> )
  + AsyncTask <Params, Progress, Result>
  + 🡪 created as an inner class of the activity in which the UI works
* JSON:
  + in Java: class: JSONObject
  + one JSONObject can include other JSONObjects (“arrays” of data)
    - getJSONObject(String object) to call the sub-objects
  + key-value pairs
    - getString(String key) to call the values of a certain key
* ProgressBar
  + more like a turning wheel, doesn’t really show any progress, just indicates that something is still loading.

RecyclerView

* why?
  + data has to be stored into memory if directly loaded
  + no list formats so far, rather everything loaded in one TextView
* principle:
  + list items are not created while scrolling, but are preloaded in a queue/recycling bin 🡪 unused list items (which are no longer in the visible area of the list) are recycled by putting them back into the queue 🡪 do not have to be reloaded again, put are simply binded with new data.
* components:
  + RecyclerView
  + Adapter
  + DataSource
  + ViewHolder: holds View items (like TextView) so that they don’t have to be looked up (with findViewById) everytime again
  + LayoutManager

- difference between px, dip, dp, sp: http://stackoverflow.com/questions/2025282/what-is-the-difference-between-px-dp-dip-and-sp-on-android

- static methods do not need an object generation

- always use auto complete when coding!! --> quicker and less mistakes

Best practices:

* Member variables (Instanzvariablen) are called m…