Outline

- Logistics
- Layouts review
- ImageView
- RecyclerView

Outline

- **Logistics**
- Layouts review
- ImageView

- RecyclerView

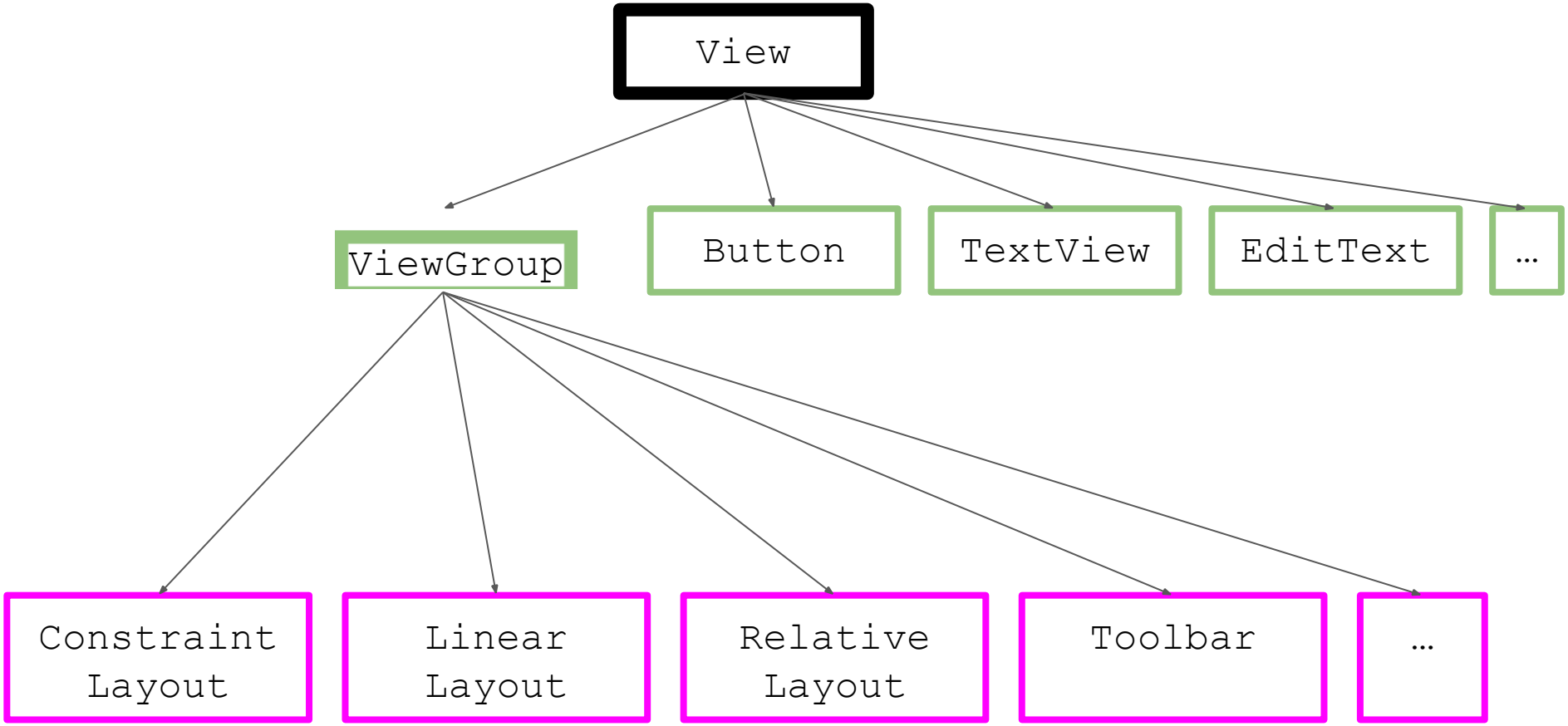
Outline

- Logistics
- **Layouts review**
- ImageView
- RecyclerView

Kotlin exercises?

- Two Sum

- Water Container

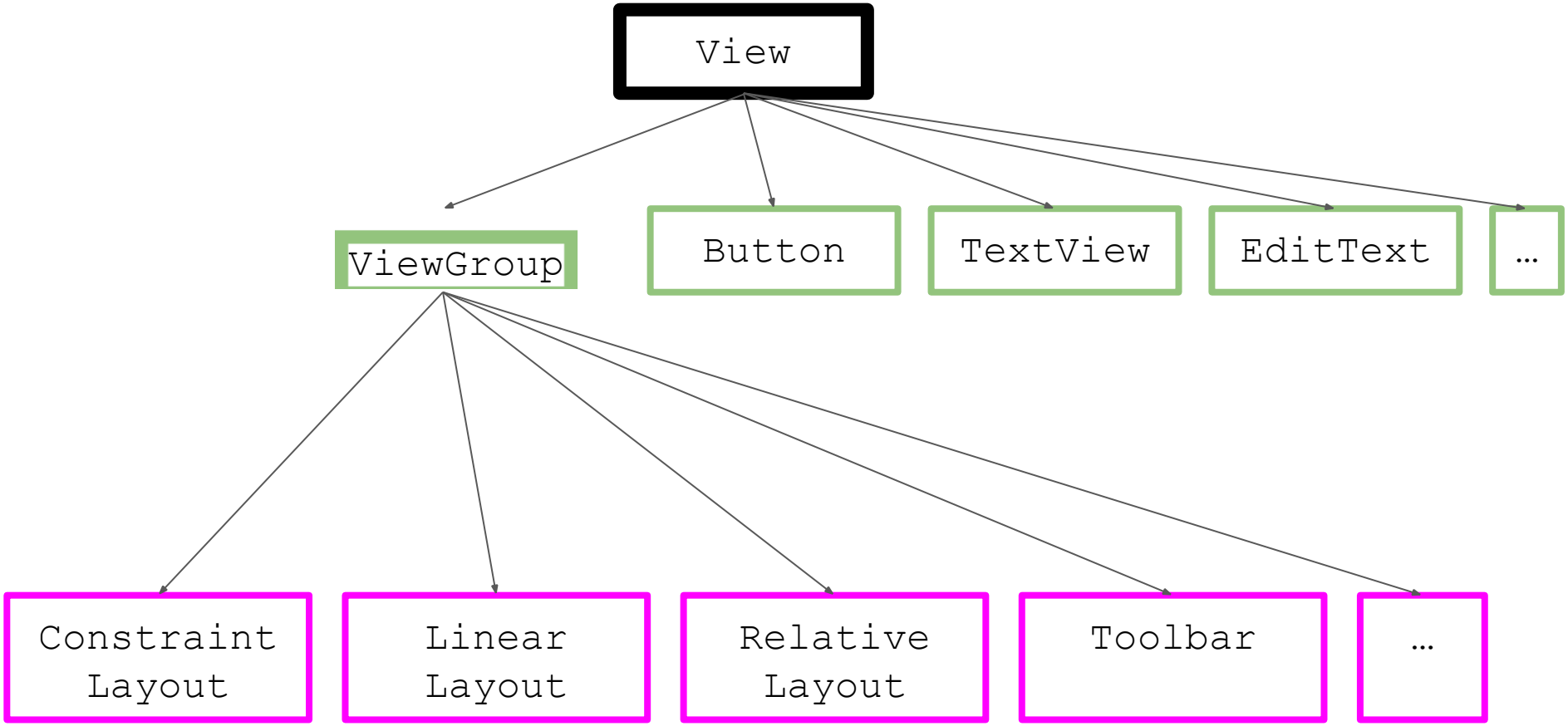

View
ViewGroup
Button
TextView
EditText
…
Constraint Layout
Linear Layout
Relative Layout
Toolbar
…

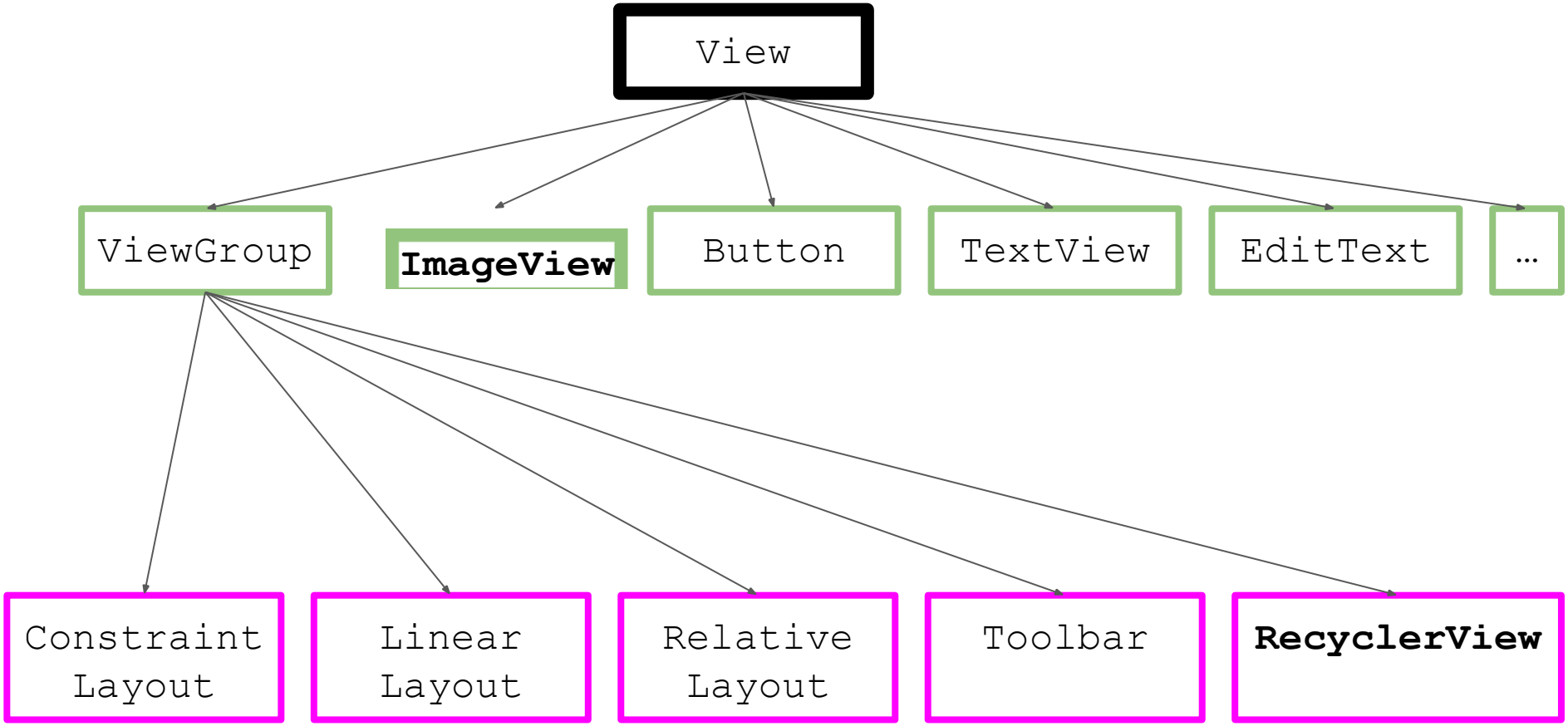
px, dp, sp

- **px**: actual pixels on the screen
- **dp**: density-independent pixel, based on pixel density of the phone (e.g. 160 dpi vs 440 dpi)
  - Use this for view padding/margins/width/height
- **sp**: scale-independent pixels, scaled by user's font preferences.
  - Use this for text

Views and ViewGroups

- Can be declared in layout XML file
- Don’t waste time memorizing attribute names
- Added programmatically in Kotlin


View
ViewGroup
Button
TextView
EditText
…
Constraint Layout
Linear Layout
Relative Layout
Toolbar
…


View
ViewGroup
ImageView
Button
TextView
EditText
…
Constraint Layout
Linear Layout
Relative Layout
Toolbar
RecyclerView

Outline

- Logistics
- Layouts review
- **ImageView**
- RecyclerView

ImageView

- Displays image resources
- Why does it require special handling?

- Memory usage
- Scale type
- Fetching remote images

ImageView - memory usage

- Be mindful of memory usage
  - Pixel phone: 12 MP camera
  - 4048 * 3036 pixels, 4 bytes per pixel = 48 MB memory
  - This can crash your app on older devices!
- Libraries such as **Glide** will handle this for you

ImageView - scale type

CENTER	center the image in the view, no scaling
CENTER_CROP	Scale the image uniformly so both width/height are **greater than or equal** to the view width/height
CENTER_INSIDE	Scale image uniformly so both width/height are **less than or equal** to the view width/height
FIT_CENTER	(**Default**) Ensure the image fits entirely inside the view, similar to CENTER_INSIDE

FIT_XY	Scale X and Y dimensions of the image independently to exactly match the view dimensions (may **change the aspect ratio**)

ImageView - remote images

- We'll often want to render an image based on a URL
- Network requests are an example of an asynchronous operation
  - Downloading the image must be down on a background thread
  - Translate the image into a bitmap
- Stanford image url

Image loading libraries

- Can fetch, decode, and display bitmaps in your app

- Can also do image transformations, e.g. rounded corners
  - Popular libraries
    - **Glide** - from Google
    - **Picasso** - from Square
    - **Fresco** - from Facebook

Outline

- Logistics
- Layouts review
- ImageView

- **RecyclerView**

RecyclerView

- The recommended way to display a list of items in Android
- Released with Android Lollipop in 2014
- Supercedes the ListView component

RecyclerView- why so complex?

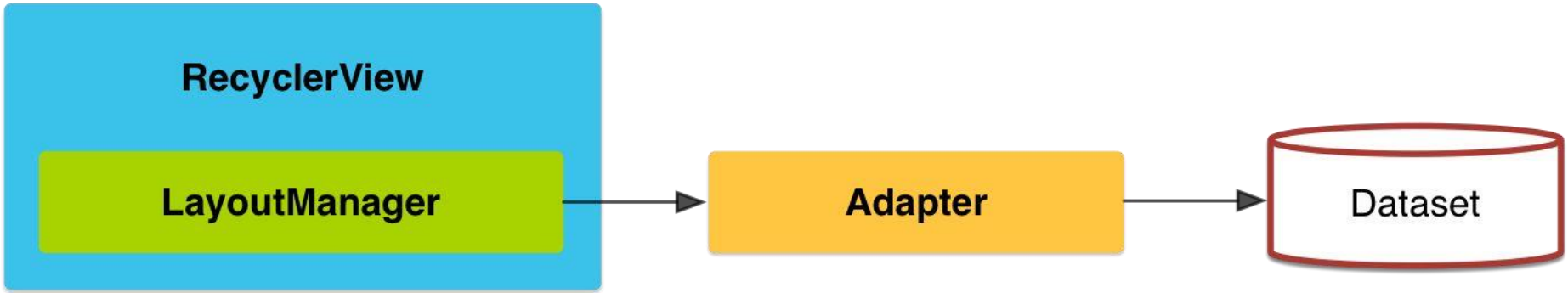
- Memory usage concerns
- Variety of ways to display and animate
- Need fine-grained control over individual list elements and click listeners

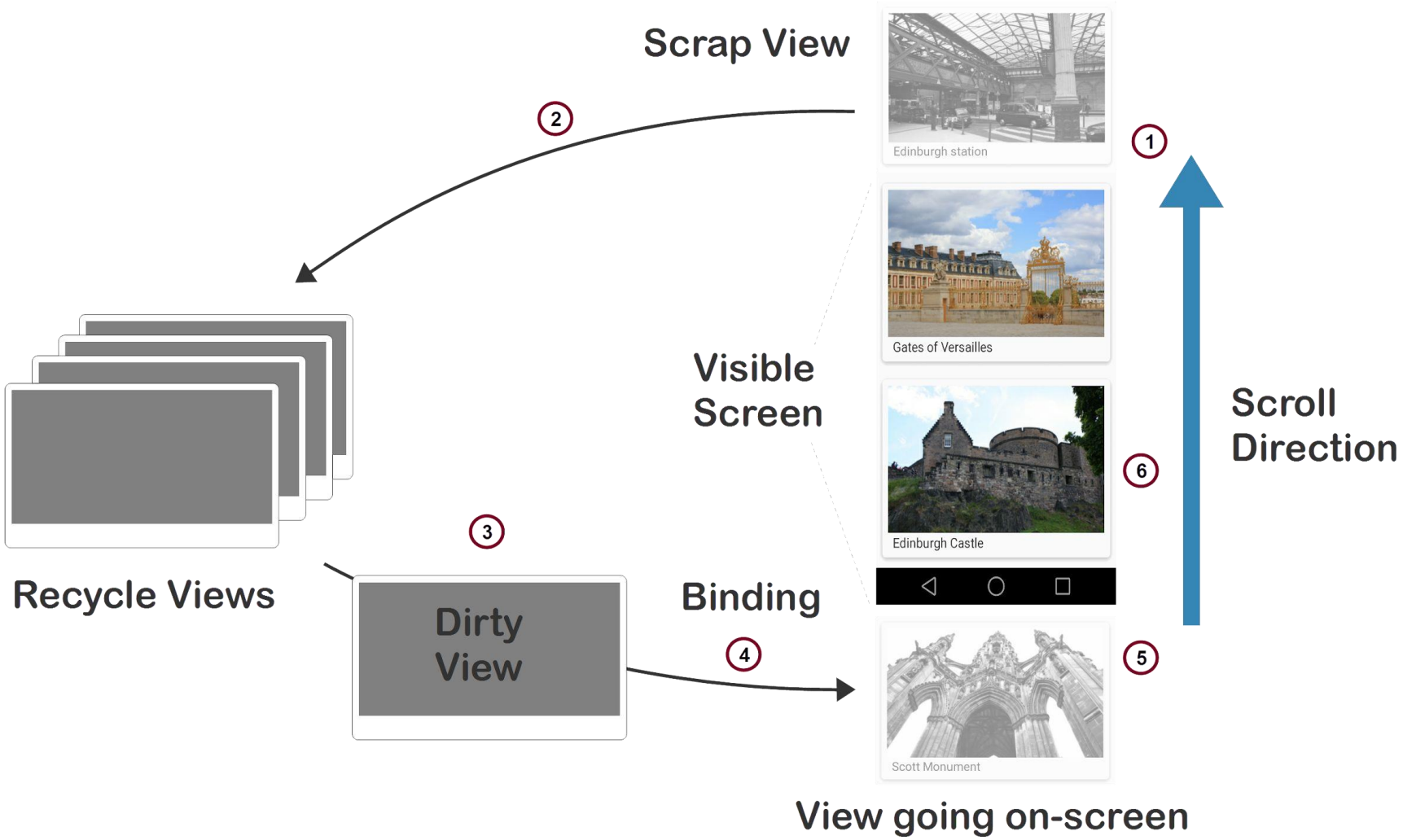
RecyclerView vs ListView

- (+) More efficient by default (use the ViewHolder pattern)
- (+) More flexible for styling + animations

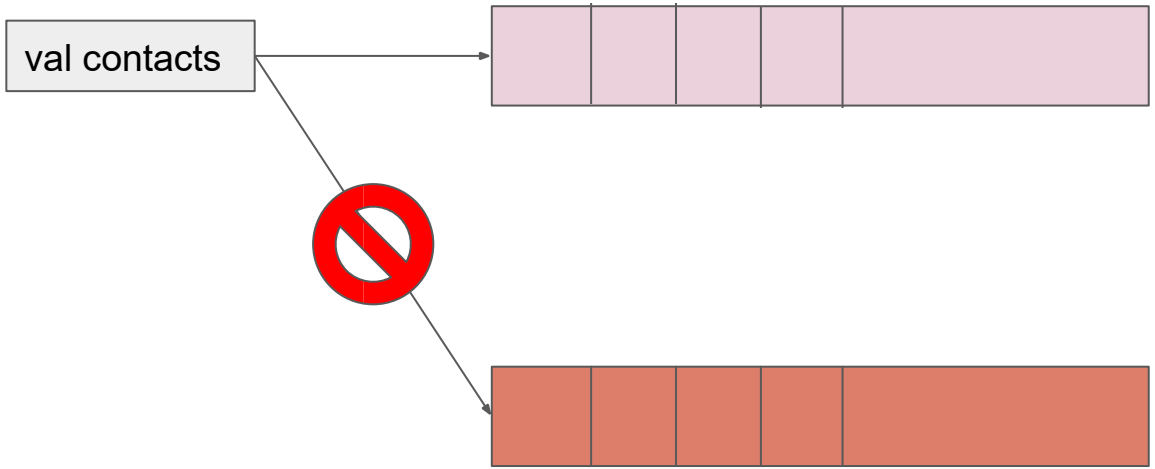
- (+) Separation of concerns
- (-) More complicated

RecyclerView Components




Scrap View
2
Edinburgh station
1
Gates of Versailles
Visible Screen
Edinburgh Castle
6
Scroll Direction
Recycle Views
3
Dirty View
Binding
4
5
Scott Monument
View going on-screen

val vs var



Further reading

- RecyclerView in <50 lines of code: videos
- Codepath guide: link
- Android developer guide: link

Prep for next week

- Complete extensions for the Tip Calculator by Sunday

- Submit feedback for your partner before next lecture