RADICAL

RecyclerView

GENIUS

+**Lisa Wray** Zeitouni

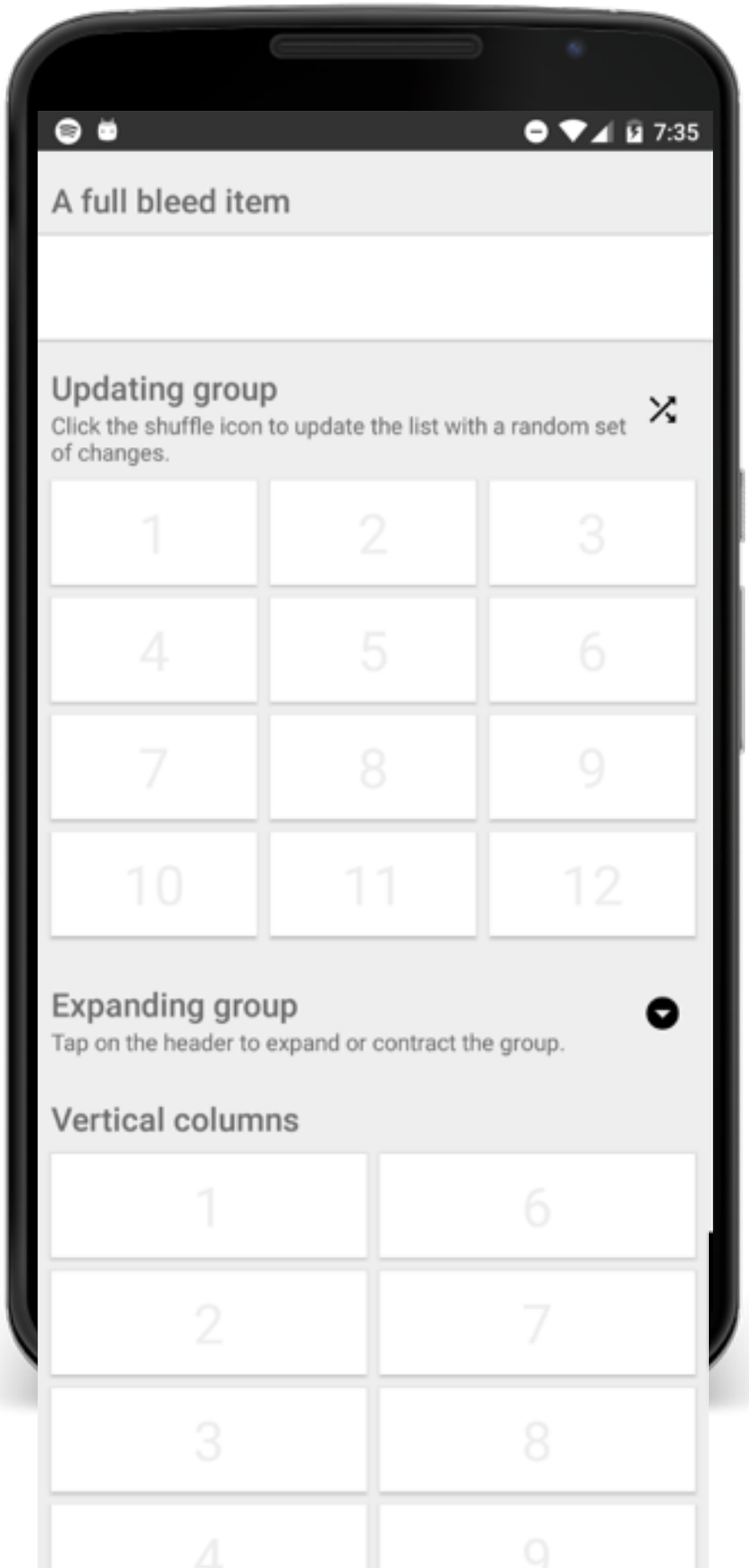
@lisawrayz

A practical guide to complex layouts in a RecyclerView

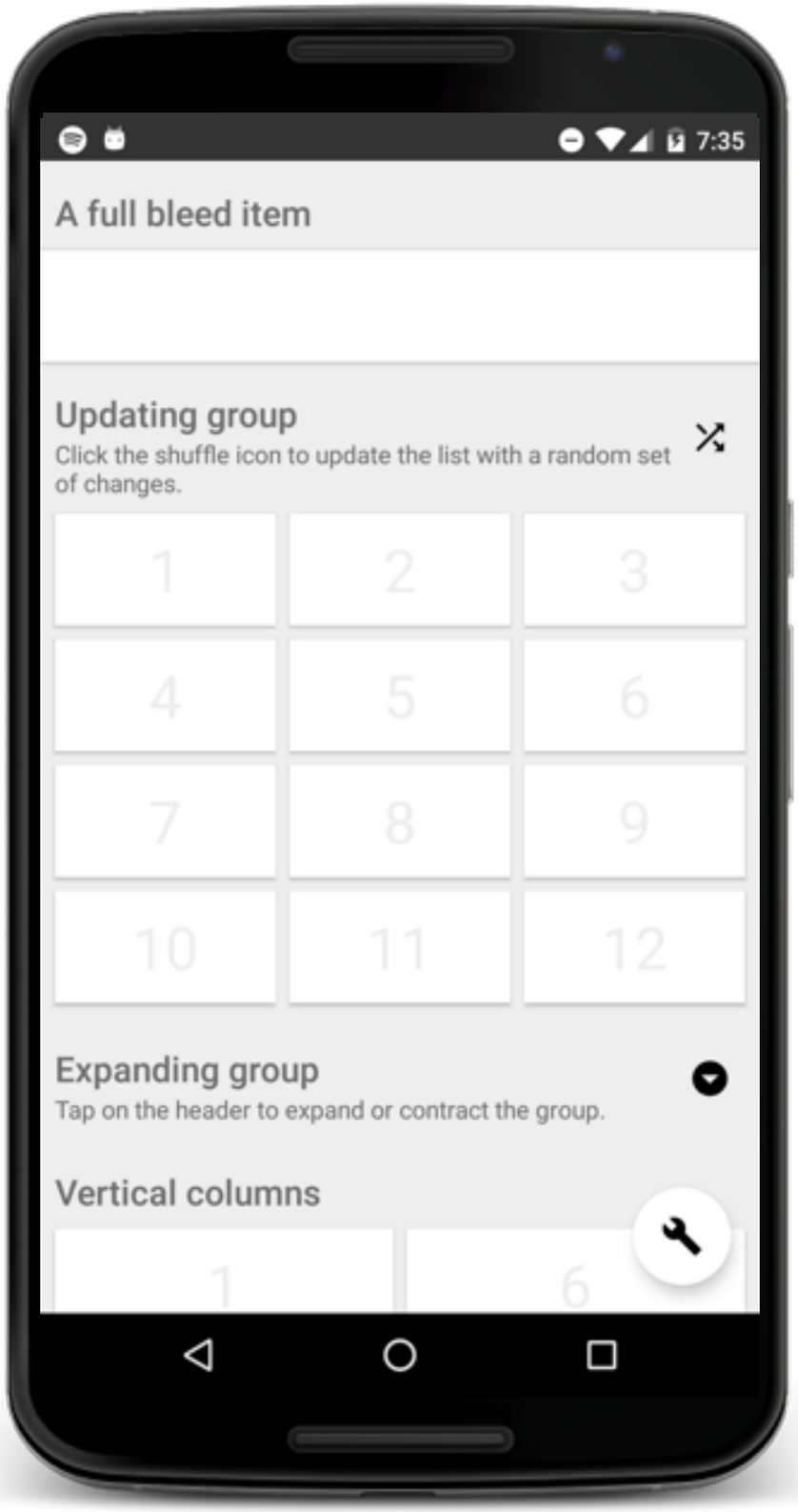
@lisawrayz

What’s special about RecyclerView?

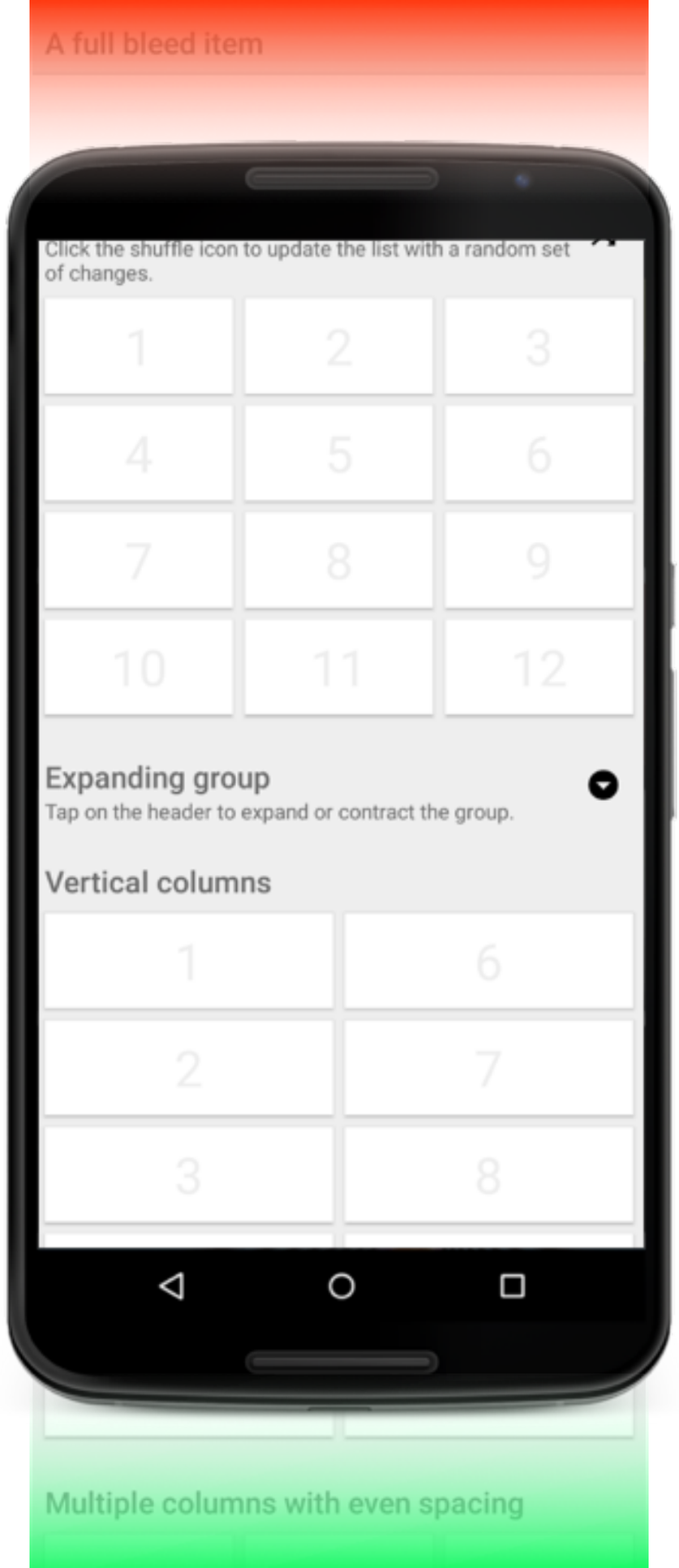
Viewport into a huge virtual layout



What is an “Item” anyway?

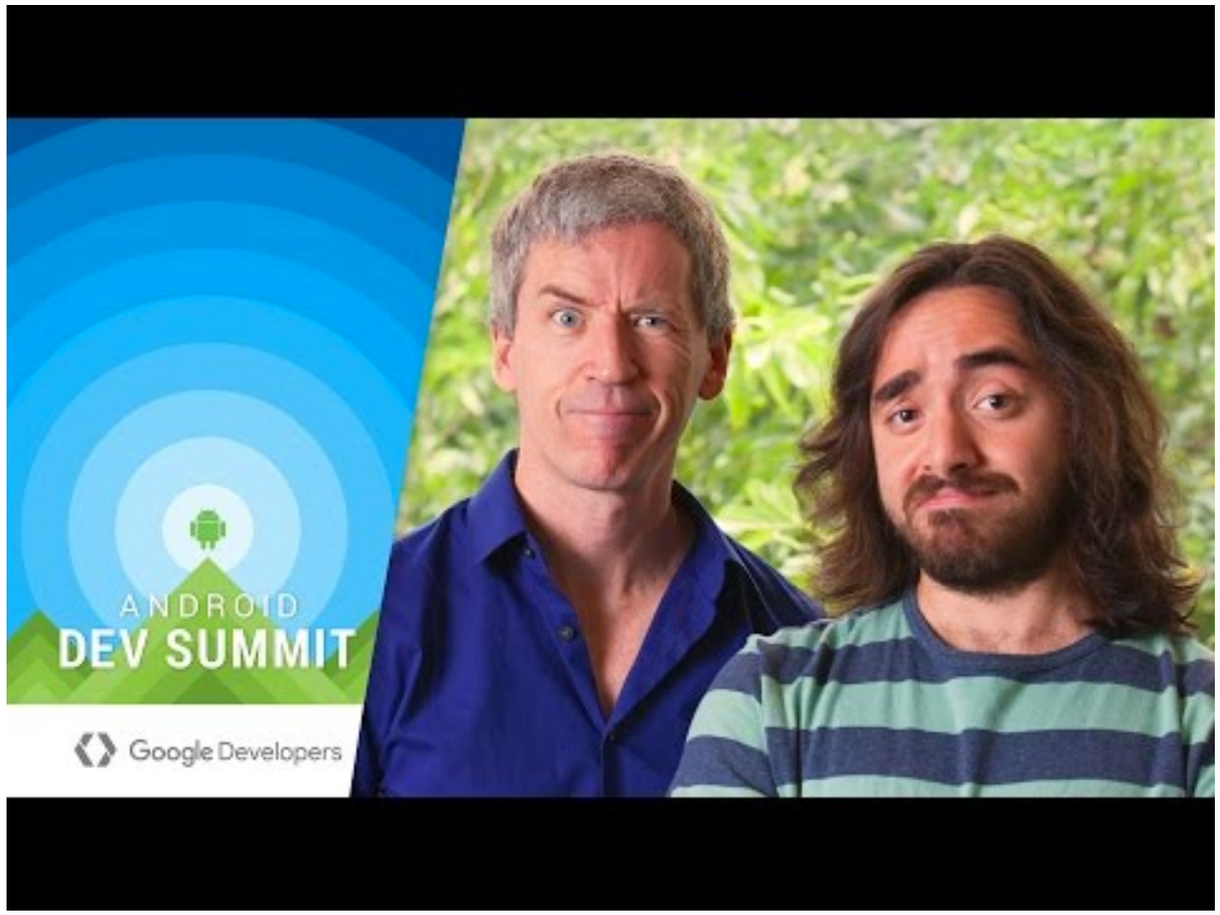


New lifecycle — items live, die, live again



“RV Animations & Behind the Scenes”

Android Dev Summit 2015



youtube.com/watch?v=imsr8NrIAMs

“Pro RecyclerView”

360|AnDev

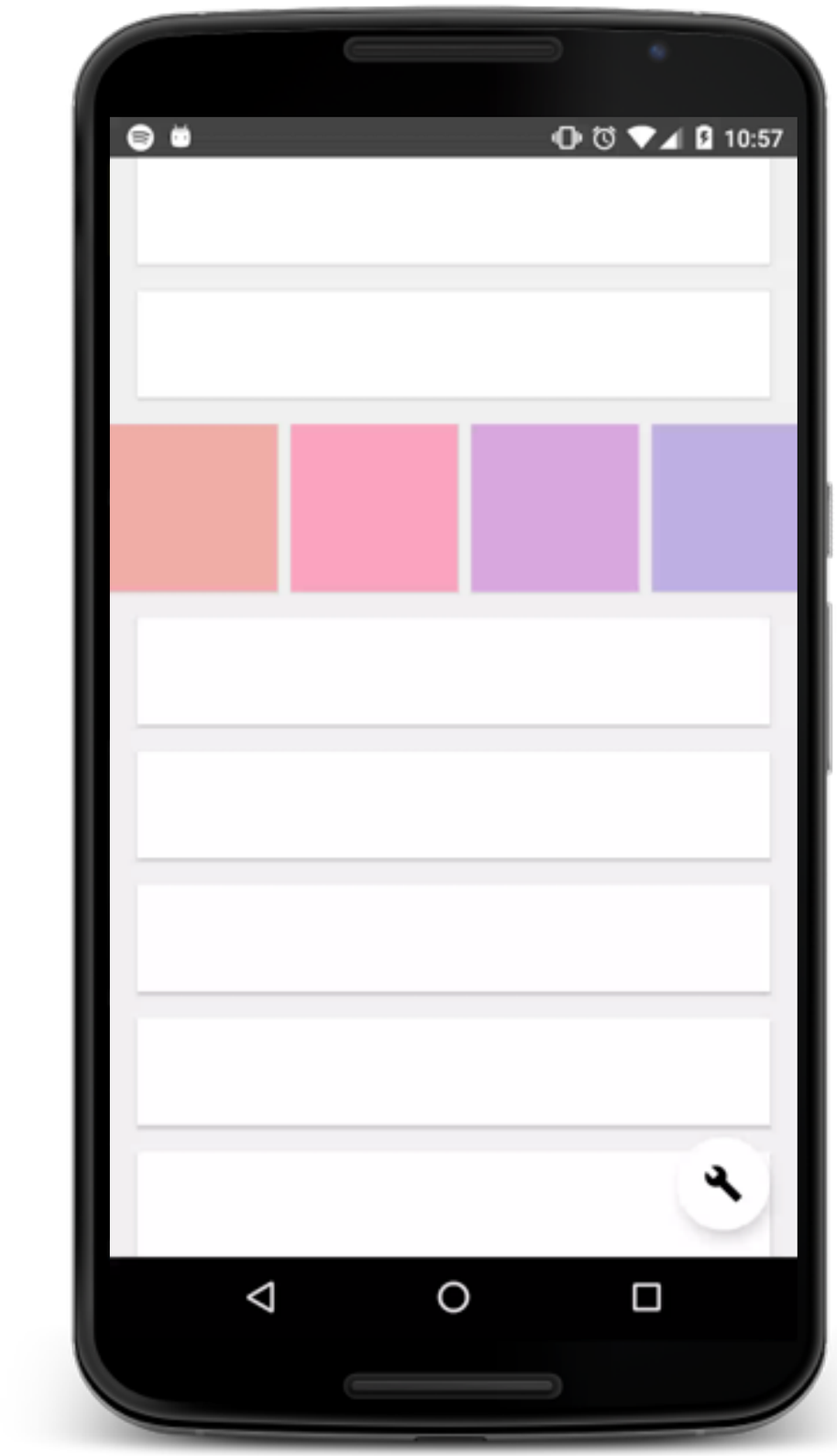
speakerdeck.com/yigit/pro-recyclerview

components 101

- **RecyclerView**: Creates, binds, recycles
- **Adapter**: Provides data
- **LayoutManager**: Lays out & positions
- **ItemDecoration**: Adds offsets, draws over / under
- **ItemAnimator**: Animates changes
- **ItemTouchHelper**: Handles drag&drop, swipe-to-delete
- **SnapHelper**: Creates ViewPager-like scrolls & flings
- **DiffUtil**: Calculates changes for you

- **RecyclerView**: `RecyclerView`
- **Adapter**: `RecyclerView.Adapter`
- **LayoutManager**: `Linear-` / `GridLayoutManager`
- **ItemDecoration**: nope
- **ItemAnimator**: `DefaultItemAnimator`
- **ItemTouchHelper**: `SimpleItemTouchHelper` / `SimpleCallback`
- **SnapHelper**: `LinearSnapHelper`
- **DiffUtil**: `DiffUtil`

looks hard,
actually easy!



carousel

Item with a RecyclerView and a horizontal LinearLayoutManager

Not like ListView — it just works!

layout/item_carousel.xml

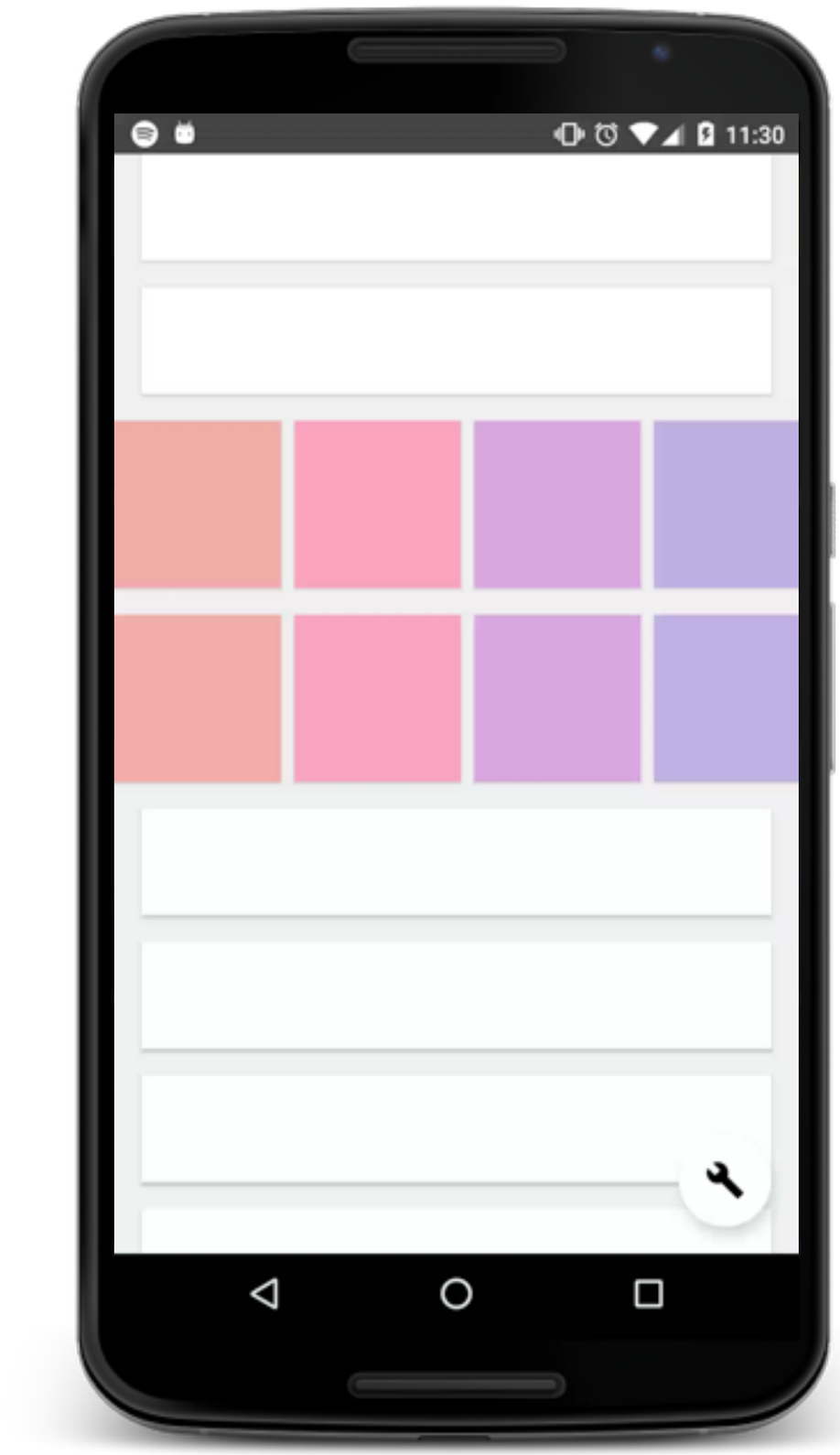
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
</android.support.v7.widget.RecyclerView>
```

onCreate:

```
…
lm = new LinearLayoutManager(context, HORIZONTAL, false);
recyclerView.setLayoutManager(lm);
recyclerView.addItemDecoration(carouselDecoration);
```

onBind:

```
recyclerView.setAdapter(adapter);
```



Gravity.START

snapping

SnapHelper
support lib 24.1

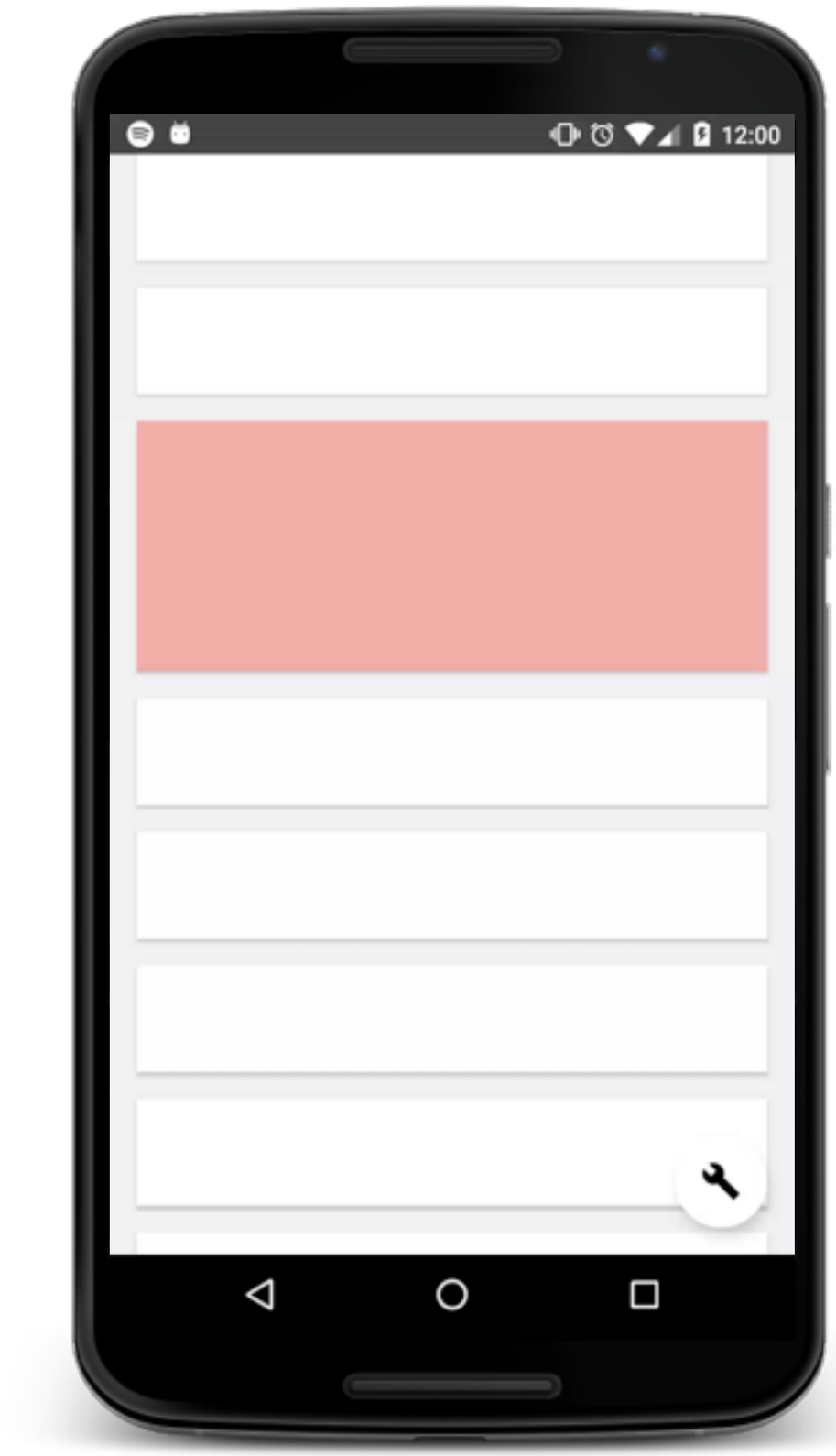
snapping

Base class: `SnapHelper`

`LinearSnapHelper`: center snapping

```
SnapHelper snapHelper = new GravitySnapHelper(Gravity.START);
snapHelper.attachToRecyclerView(recyclerView);
```

GravitySnapHelper:
rubensousa.github.io/2016/08/recyclerviewsnap



viewpager (-like)

width=MATCH_PARENT item
LinearSnapHelper

no fragments
fling is allowed

swipe-to-delete

Drag & drop uses the same mechanism

```
private TouchCallback touchCallback =
        new SwipeTouchCallback();
ItemTouchHelper itemTouchHelper =
        new ItemTouchHelper(touchCallback);

itemTouchHelper.attachToRecyclerView(recyclerView);
```

```
public class SwipeTouchCallback
    extends ItemTouchHelper.SimpleCallback {

    public SwipeTouchCallback() {
        super(0, ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT);
    }
}
```

```java
public class SwipeTouchCallback
    extends ItemTouchHelper.SimpleCallback {

    public SwipeTouchCallback() {
        super(0, 0);
    }

    @Override public int getSwipeDirs(RecyclerView recyclerView,
            RecyclerView.ViewHolder viewHolder) {

        if (viewHolder.getItemViewType() == R.layout.item_card) {
            return ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT;
        } else {
            return super.getSwipeDirs(recyclerView, viewHolder);
        }
    }
}
```

```
public class SwipeTouchCallback
    extends ItemTouchHelper.SimpleCallback {

…

    @Override public void onSwiped(
        RecyclerView.ViewHolder viewHolder, int direction) {

        int position = viewHolder.getAdapterPosition();
        // remove & notify
    }

    @Override public void onChildDraw(Canvas c, RecyclerView
recyclerView, RecyclerView.ViewHolder viewHolder, float dX,
float dY, int actionState, boolean isCurrentlyActive) {
        View child = viewHolder.itemView;

        // Fade out the item
        child.setAlpha(1 - (Math.abs(dX) / child.getWidth()));

        super.onChildDraw(…);
    }
}
```

DiffUtil

@lisawrayz


8:07


8:14

1. thou shalt notify as precisely as possible

```
notifyItemChanged(…);
notifyItemRangeChanged(…);
notifyItemAdded(…);
notifyItemRangeAdded(…);
notifyItemRemoved(…);
notifyItemRangeRemoved(…);
notifyItemMoved(…);
```

```
… and notifyDatasetChanged();
```

```
DiffUtil.DiffResult diffResult =
    DiffUtil.calculateDiff(
        new Callback(items, newItems));

// Actually change adapter
adapter.clear();
adapter.addAll(newItems);

// Notify
diffResult.dispatchUpdatesTo(adapter);
```

```
private class Callback extends DiffUtil.Callback {

    …

    @Override public boolean areItemsTheSame(
        int oldItemPosition, int newItemPosition) {}


    @Override public boolean areContentsTheSame(
        int oldItemPosition, int newItemPosition) {}
}
```

multiple view types

naïve way

```
public class ViewTypes {

    public static final int HEADER = 0;
    public static final int CARD = 1;
    public static final int FULL_BLEED_CARD = 2;
    public static final int SQUARE_CARD = 3;
    public static final int SMALL_CARD = 4;

}
```

item creation

```
@Override public RecyclerView.ViewHolder
    onCreateViewHolder(ViewGroup parent, int viewType) {
    LayoutInflater inflater = … ;

    View view;
    switch (viewType) {
        case HEADER:
            view = inflater.inflate(
                    R.layout.item_header, parent, false);
            return new HeaderViewHolder(view);
        case CARD:
            view = inflater.inflate(
                    R.layout.item_card, parent, false);
            return new CardViewHolder(view);
        case FULL_BLEED_CARD:
        case SQUARE_CARD:
        case SMALL_CARD:
         …
    }
}
```

item bind

```
@Override public void onBindViewHolder(
    RecyclerView.ViewHolder viewHolder, int position) {

    Model model = models.get(position);
    switch (viewHolder.getItemViewType()) {
        case HEADER:
            HeaderViewHolder headerVH = (HeaderViewHolder) viewHolder;
            headerVH.title.setText(model.getTitle());
            if (model.getSubtitle() != null) {
                headerVH.subtitle.setText(model.getSubtitle());
            }
            headerVH.subtitle.setVisibility(
                    model.getSubtitle() != null ?
                    View.VISIBLE : View.GONE);
            headerVH.icon.setImageDrawable(model.getIcon());
            break;
        case CARD:
            CardViewHolder cardVH = (CardViewHolder) viewHolder;
            …
            break;
        case FULL_BLEED_CARD:
        case SQUARE_CARD:
        …
    }
}
```

This is your adapter on switch statements

delegate — an ok way

```
public interface AdapterDelegate {
    void onBind(RecyclerView.ViewHolder viewHolder,
        int position);
    boolean handles(ViewHolder viewHolder);
}
```

```
public class Adapter extends RecyclerView.Adapter {
    List<AdapterDelegate> delegates;

    public Adapter() {
        delegates.add(new HeaderDelegate());
        delegates.add(new CardDelegate());
    }
```


```
    @Override public void onBindViewHolder(
            RecyclerView.ViewHolder viewHolder, int position) {
        for (AdapterDelegate delegate : delegates) {
            if (delegate.handles(viewHolder)) {
                delegate.onBind(viewHolder, position);
            }
        }
    }
}
```

better way

```
public class ItemTypes {

    public static final int HEADER = 0;
    public static final int CARD = 1;
    public static final int FULL_BLEED_CARD = 2;
    public static final int SQUARE_CARD = 3;
    public static final int SMALL_CARD = 4;

}
```

better way

R.layout.*item_header*

R.layout.*item_card*

Item

```
public class SongItem extends Item {
    private final Song song;

    public SongItem(Song song) {
        this.song = song;
    }

    @Override public void bind(ViewHolder viewHolder,
        int position) {
        // binding logic here
    }

    @Override public int getLayout() {
        return R.layout.song;
    }
}
```

Epoxy is an Android library for building complex screens in a RecyclerView

Epoxy

Epoxy is an Android library for building complex screens in a RecyclerView. It abstracts the boilerplate of view holders, item types, item ids, span counts, and more, in order to simplify building screens with multiple view types. Additionally, Epoxy adds support for saving view state and automatic diffing of item changes.

We developed Epoxy at Airbnb to simplify the process of working with RecyclerViews, and to add the missing functionality we needed. We now use Epoxy for most of the main screens in our app and it has improved our developer experience greatly.

Epoxy

```
EpoxyAdapter epoxyAdapter = new EpoxyAdapter();
EpoxyModel headerModel = new HeaderModel();
epoxyAdapter.addModel(headerModel);
```

Epoxy

```
public class PhotoModel extends EpoxyModel<PhotoView> {
  private final Photo photo;

  public PhotoModel(Photo photo) {
    this.photo = photo;
 }

  @LayoutRes
  public int getDefaultLayout() {
    return R.layout.view_model_photo;
  }

  @Override
  public void bind(PhotoView photoView) {
    photoView.setUrl(photo.getUrl());
  }
}
```

Epoxy

```
public class PhotoModel extends EpoxyModel<PhotoView> {
  private final Photo photo;

  public PhotoModel(Photo photo) {
    this.photo = photo;
 }

  @LayoutRes
  public int getDefaultLayout() {
    return R.layout.view_model_photo;
  }

  @Override
  public void bind(PhotoView photoView) {
    photoView.setUrl(photo.getUrl());
  }
}
```

Epoxy

```
public class PhotoModel extends EpoxyModel<PhotoView> {
  private final Photo photo;

  public PhotoModel(Photo photo) {
    this.photo = photo;
 }

  @LayoutRes
  public int getDefaultLayout() {
    return R.layout.view_model_photo;
  }

  @Override
  public void bind(PhotoView photoView) {
    photoView.setUrl(photo.getUrl());
  }
}
```

Epoxy

```
public class PhotoModel extends EpoxyModel<PhotoView> {
  private final Photo photo;

  public PhotoModel(Photo photo) {
    this.photo = photo;
 }

  @LayoutRes
  public int getDefaultLayout() {
    return R.layout.view_model_photo;
  }

  @Override
  public void bind(PhotoView photoView) {
    photoView.setUrl(photo.getUrl());
  }
}
```

need a custom view or view holder for each item

consider data binding

/MyAdapter.java

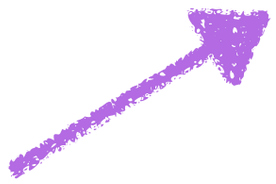
```
@Override public RecyclerView.ViewHolder onCreateViewHolder(
    ViewGroup parent, int layoutResId) {
    LayoutInflater inflater = LayoutInflater.from(
        parent.getContext());
    ViewDataBinding binding = DataBindingUtil.inflate(
        inflater, layoutResId, parent, false);
    return new ViewHolder<>(binding);
}

public class ViewHolder<T extends ViewDataBinding> extends
RecyclerView.ViewHolder {
    public final T binding;

    public ViewHolder(T binding) {
        super(binding.getRoot());
        this.binding = binding;
    }
}
```

consider data binding

```
/MyAdapter.java
@Override public RecyclerView.ViewHolder onCreateViewHolder(
    ViewGroup parent, int layoutResId) {
    LayoutInflater inflater = LayoutInflater.from(
        parent.getContext());
    ViewDataBinding binding = DataBindingUtil.inflate(
        inflater, layoutResId, parent, false);
    return new ViewHolder<>(binding);
}

public class ViewHolder<T extends ViewDataBinding> extends
RecyclerView.ViewHolder {
    public final T binding;

    public ViewHolder(T binding) {
        super(binding.getRoot());
        this.binding = binding;
    }
}
```

multiple columns

```
new GridLayoutManager(context, spanCount);
```



total number
of columns

choosing a spanCount

- Least common multiple (LCM) of all your desired column splits
    - I want single, double, triple & quad columns LCM(1, 2, 3, 4) = 12

- No performance hit from having a large num of columns. (Large num of items might be)

span size lookup

```
final int spanCount = 12;
layoutManager = new GridLayoutManager(this, spanCount);
layoutManager.setSpanSizeLookup(
    new GridLayoutManager.SpanSizeLookup() {
    @Override public int getSpanSize(int position) {
        int viewType = adapter.getItemViewType(position);
        switch (viewType) {
            case HEADER:
                return spanCount;
            case CARD:
                return spanCount / 2;
            case FULL_BLEED_CARD:
                return spanCount;
            case SMALL_CARD:
                return spanCount / 3;
            default:
                return 1;
        }
    }
```

```java
final int spanCount = 12;
layoutManager = new GridLayoutManager(this, spanCount);
layoutManager.setSpanSizeLookup(
    new GridLayoutManager.SpanSizeLookup() {
        @Override public int getSpanSize(int position) {
            Item item = groupAdapter.getItem(position);
            return item.getSpanSize(spanCount, position);
        }
});
```

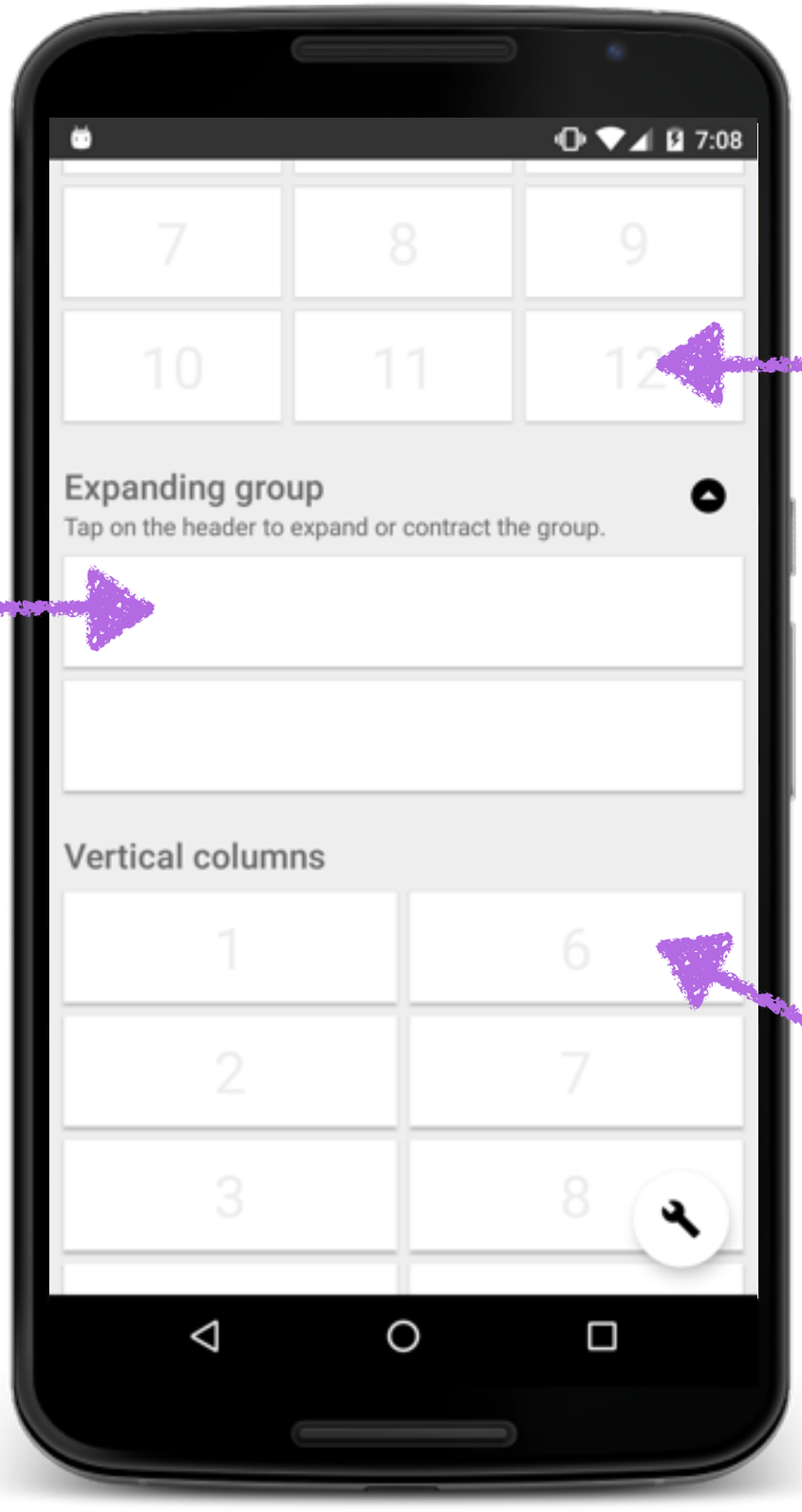
Item

```
public class SongItem extends Item {
    private final Song song;

    public SongItem(Song song) {
        this.song = song;
    }

    @Override public void bind(ViewHolder viewHolder,
        int position) {…}

    @Override public int getLayout() {
        return R.layout.song;
    }

    @Override public int getSpanSize(int spanCount,
        int position) {
        // individual item’s span size
    }
}
```

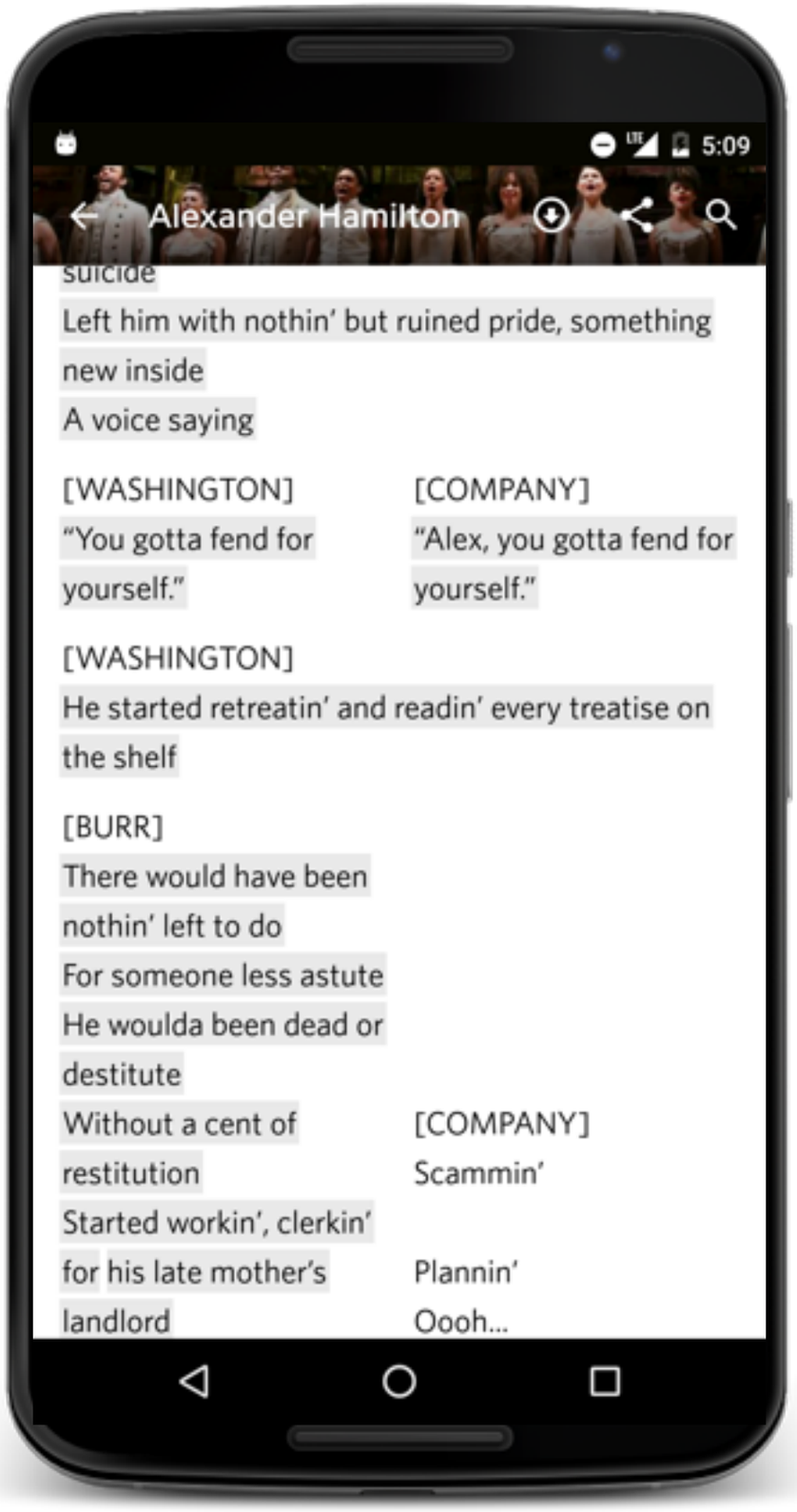
Item

```
public class SongItem extends Item {
    private final Song song;

    public SongItem(Song song) {
        this.song = song;
    }

    @Override public void bind(ViewHolder viewHolder,
        int position) {…}

    @Override public int getLayout() {
        return R.layout.song;
    }

    @Override public int getSpanSize(int spanCount,
        int position) {
        // individual item’s span size
    }
}
```

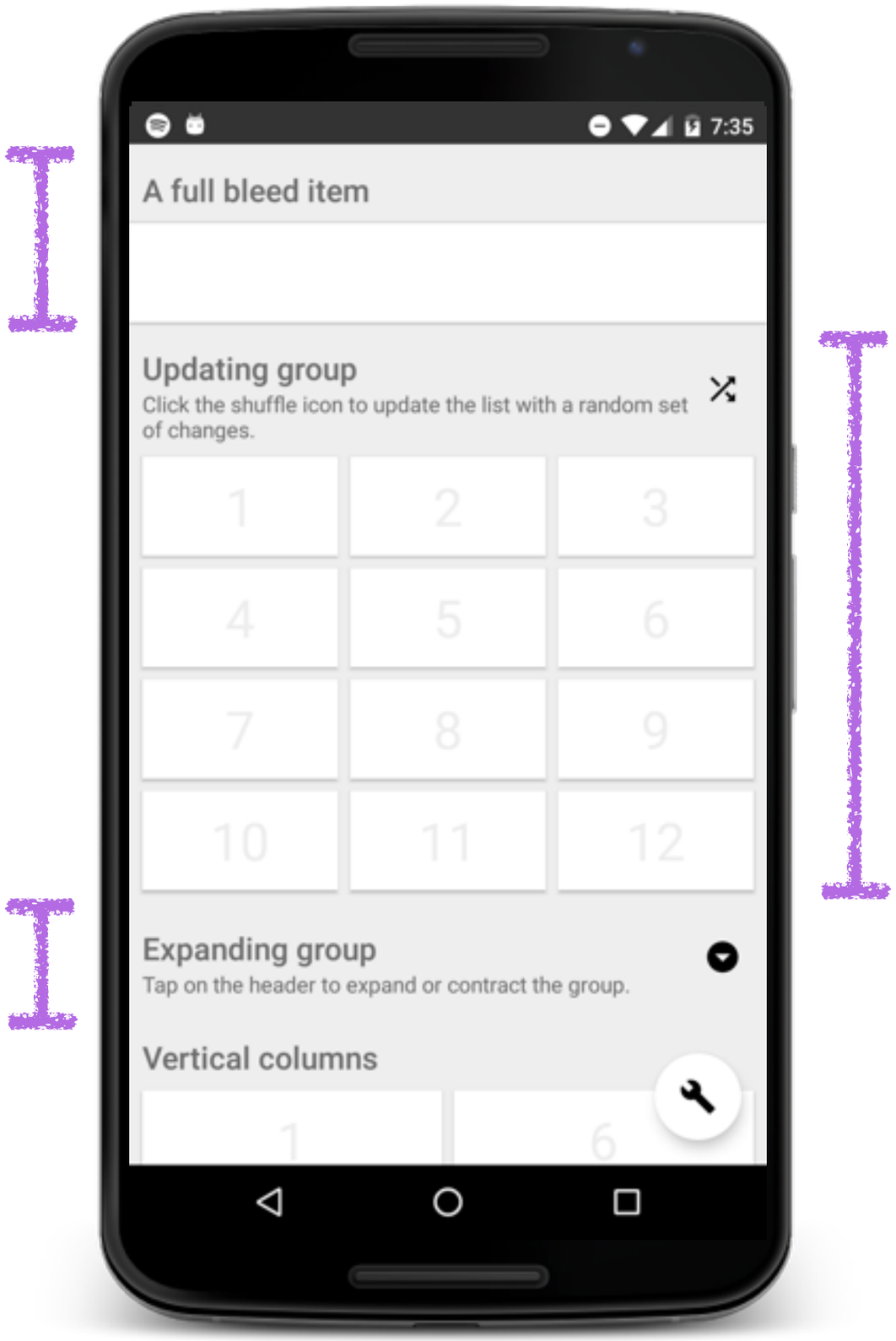
All the same view type



Columns of text



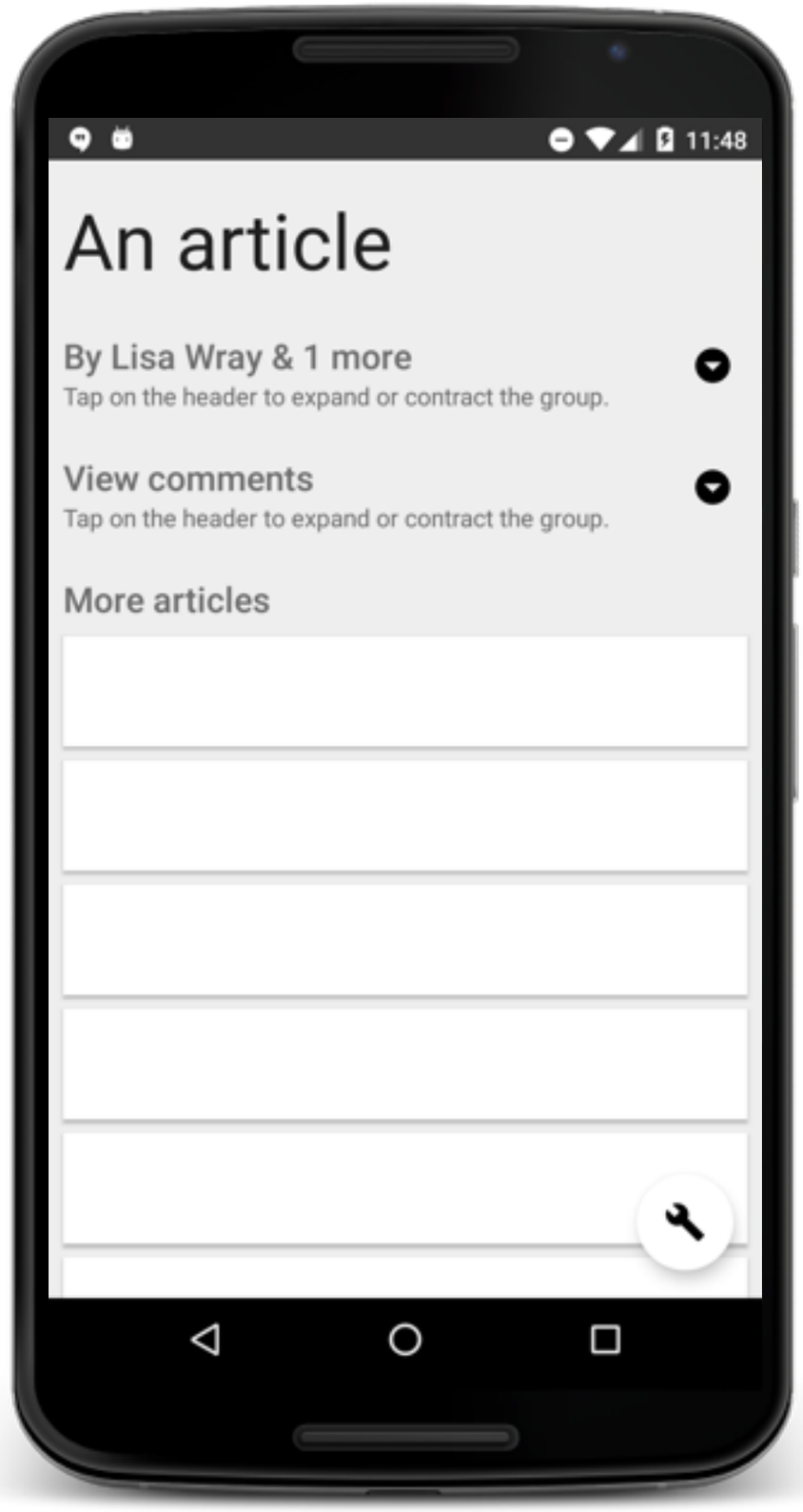
Groups


7:35
A full bleed item
Updating group
Click the shuffle icon to update the list with a random set of changes.
1
2
3
4
5
6
7
8
9
10
11
12
Expanding group
Tap on the header to expand or contract the group.
Vertical columns
1
6

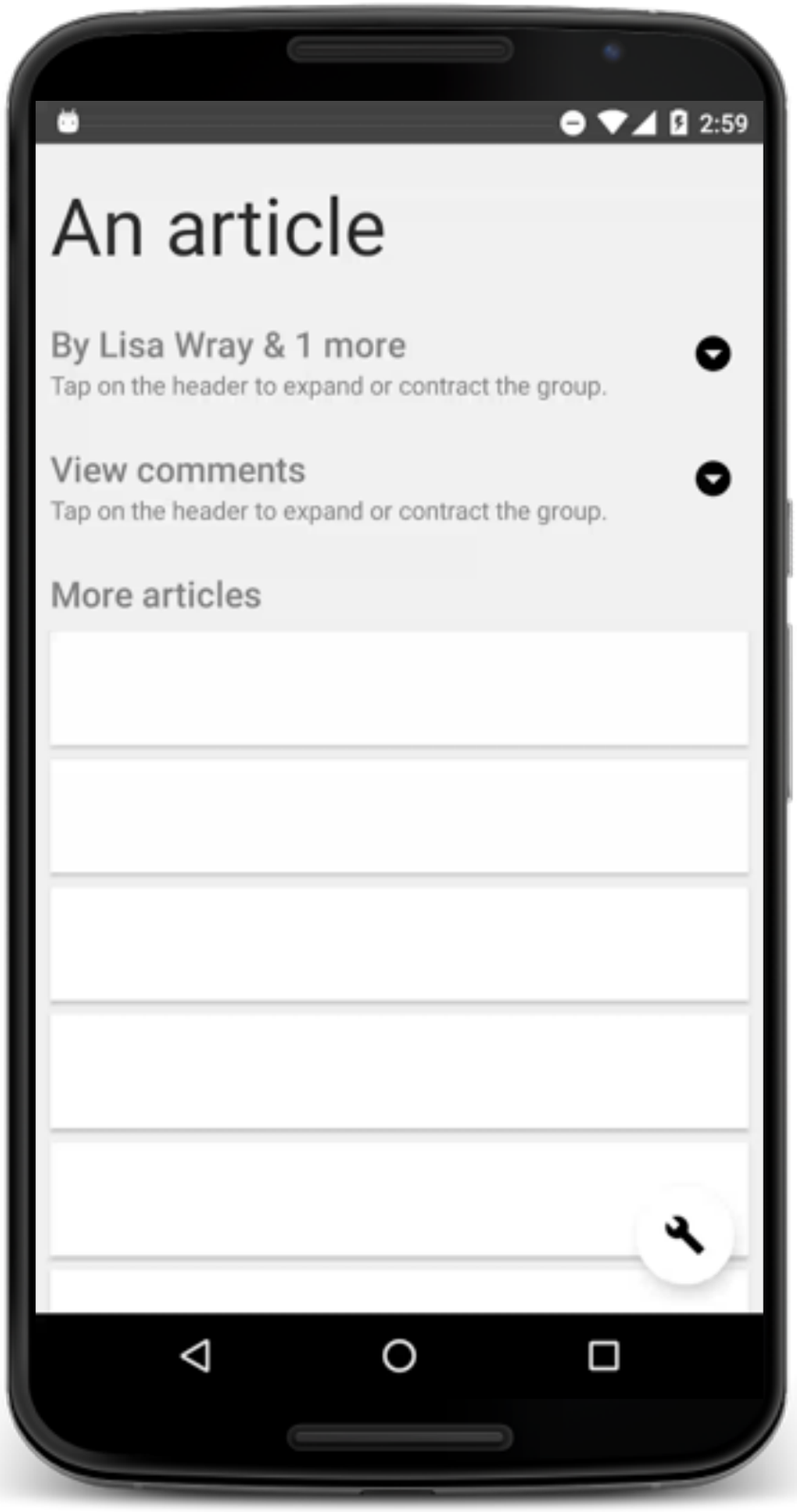
0

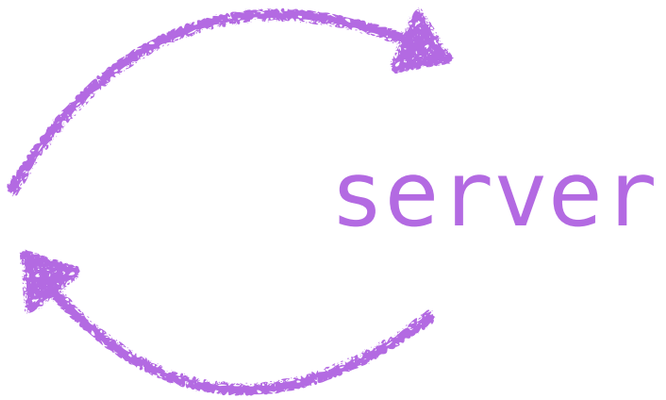
1

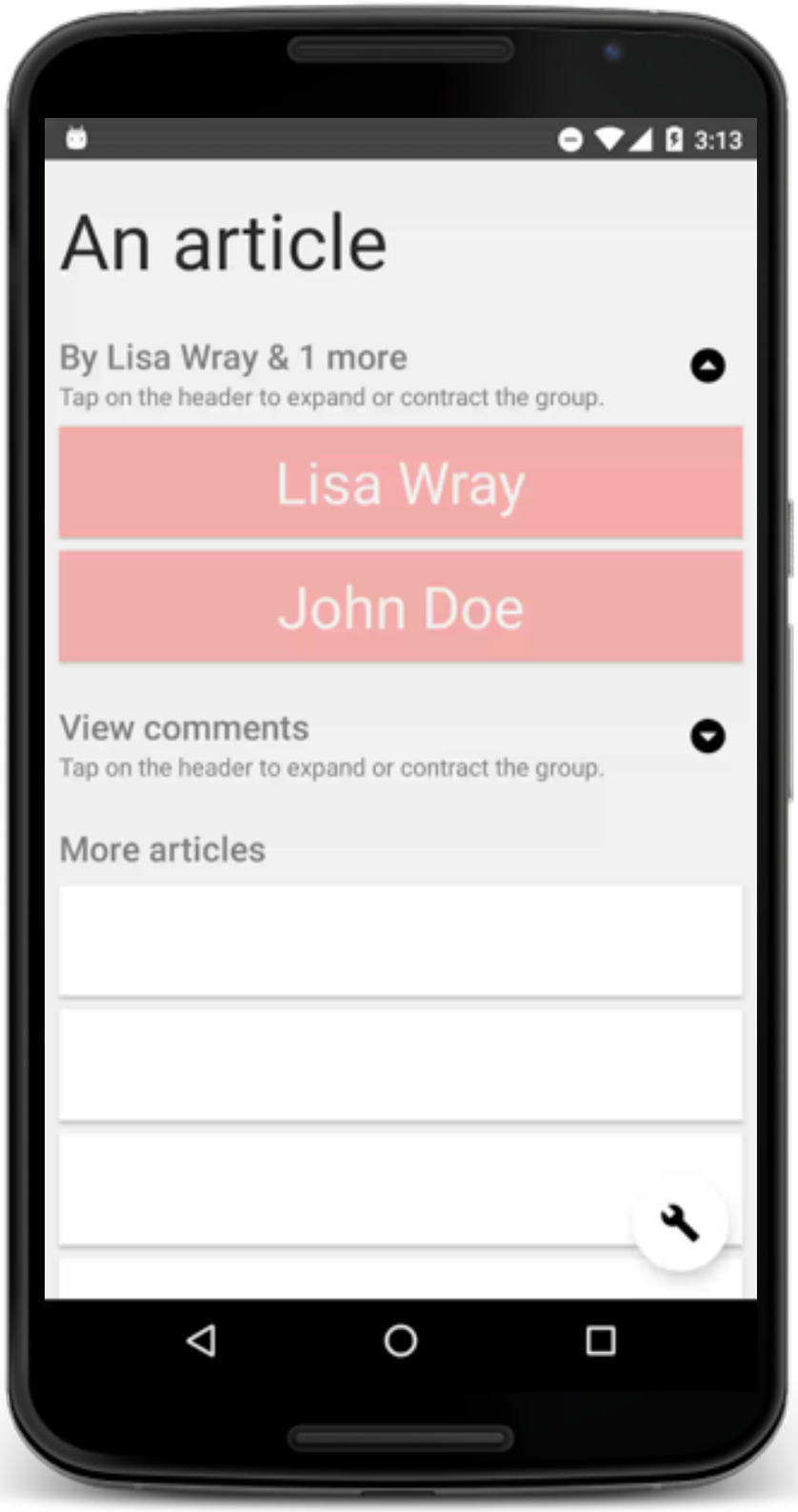
onClick, pos=2



3


2:59
An article
By Lisa Wray & 1 more
Tap on the header to expand or contract the group.
View comments
Tap on the header to expand or contract the group.
More articles


server



3

5

@lisawrayz

Don’t hold adapter position!

```
Header commentHeader;
List<Comment> comments;

int index = adapter.getPosition(commentHeader) + 1;
adapter.addAll(index, comments);
adapter.notifyInsert(index, comments.size());
```

Use references

```
Header commentHeader;
List<Comment> comments;

int index = adapter.getPosition(commentHeader) + 1;
adapter.addAll(index, comments);
adapter.notifyInsert(index, comments.size());
```

List.indexOf()

Use references

```
Header commentHeader;
List<Comment> comments;

int index = adapter.getPosition(commentHeader) + 1;
adapter.addAll(index, comments);
adapter.notifyInsert(index, comments.size());
```

Epoxy

```
EpoxyModel commentHeaderModel;
List<EpoxyModel> commentModels;

for (int i = commentModels.size - 1; i >= 0 ; i--) {
    expoxyAdapter.insertModelAfter(commentHeaderModel);
}
```

Epoxy

```
EpoxyModel commentHeaderModel;
List<EpoxyModel> commentModels;
```

```
for (int i = commentModels.size - 1; i >= 0 ; i--) {
    expoxyAdapter.insertModelAfter(commentHeaderModel);
}
```

Epoxy

```
EpoxyModel commentHeaderModel;
List<EpoxyModel> commentModels;

for (int i = commentModels.size - 1; i >= 0 ; i--) {
    expoxyAdapter.insertModelAfter(commentHeaderModel);
}

epoxyAdapter.hideModels(commentModels);
```

Genius / groupie

Watch 15 | Star 1 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs | Settings

Groupie helps you display and manage complex RecyclerView layouts. — Edit

11 commits | 1 branch | 0 releases | 2 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

lisawray committed on GitHub Updates to README — Latest commit c656416 a minute ago

README.md

groupie

Groupie helps you display and manage complex RecyclerView layouts. It lets you treat your content as logical groups and handles change notifications for you -- think sections with headers and footers, expandable groups, blocks of vertical columns, and much more. It makes it easy to handle asynchronous content updates and insertions and user-driven content changes. At the item level, it abstracts the boilerplate of item view types, item layouts, viewholders, and span sizes.

Groupie was developed as an internal library at Genius because none of the existing solutions for multiple view types or groups were simple or flexible enough for us. We decided to make it public at this early stage in order to encourage discussion about managing and developing complex, rich user interfaces with RecyclerView.

groupie

```
GroupAdapter groupAdapter;

Item item = new TitleItem();
groupAdapter.add(item);

HeaderItem header = new HeaderItem(“Comments”);
ExpandableGroup commentGroup =
    new ExpandableGroup(header);
groupAdapter.add(commentGroup);
```

groupie

```
GroupAdapter groupAdapter;

Item item = new TitleItem();
groupAdapter.add(item);

HeaderItem header = new HeaderItem(“Comments”);
ExpandableGroup commentGroup =
    new ExpandableGroup(header);
groupAdapter.add(commentGroup);
```

groupie

```
GroupAdapter groupAdapter;
Item item = new TitleItem();
groupAdapter.add(item);

HeaderItem header = new HeaderItem(“Comments”);
ExpandableGroup commentGroup =
    new ExpandableGroup(header);
groupAdapter.add(commentGroup);
```

groupie

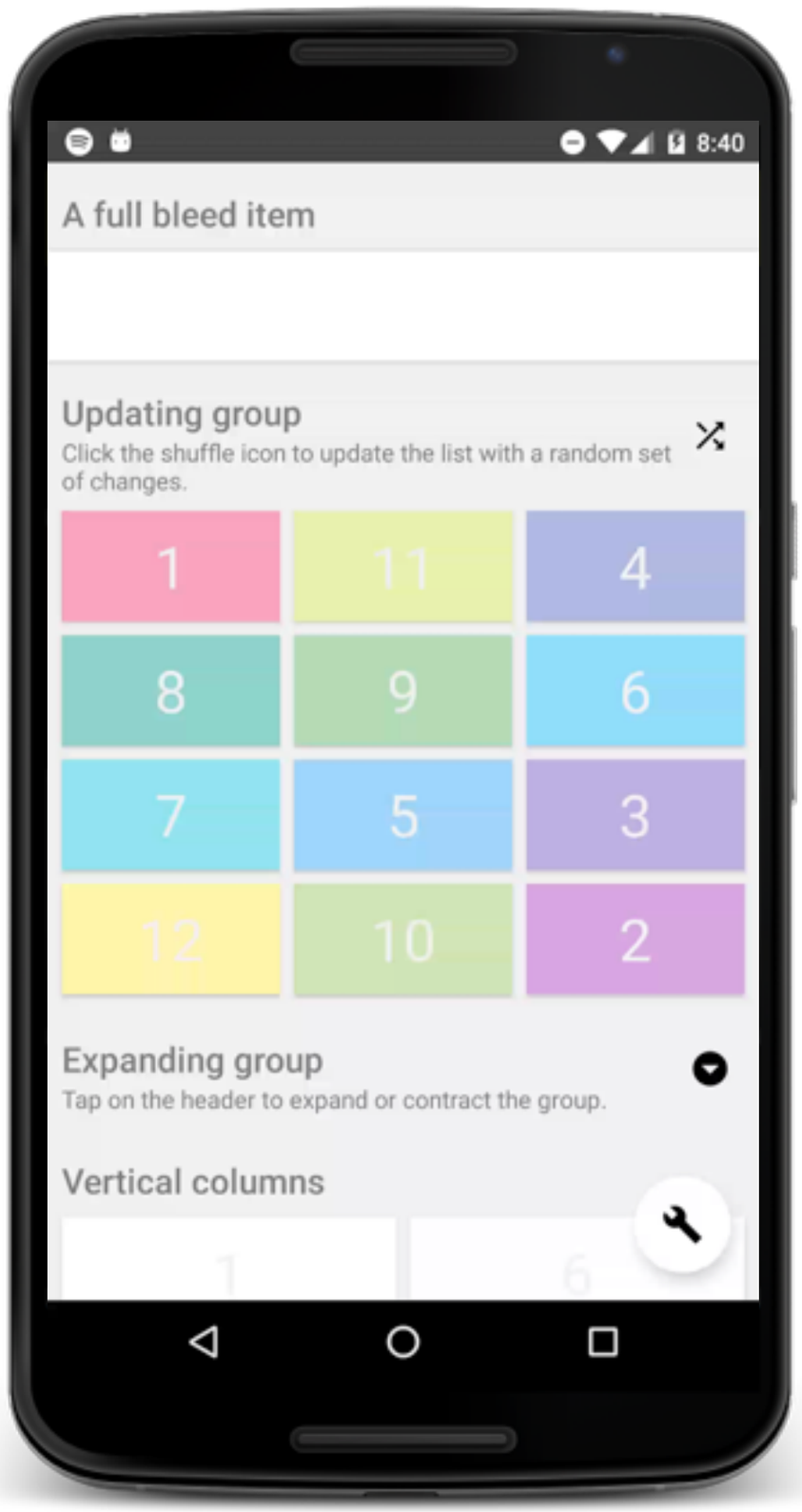
```
List<Item> commentItems;
ExpandableGroup commentGroup;
commentGroup.addAll(commentItems);
commentGroup.toggleExpanded();
```

groupie

```
List<Item> commentItems;
ExpandableGroup commentGroup;
commentGroup.addAll(commentItems);
commentGroup.toggleExpanded();
```

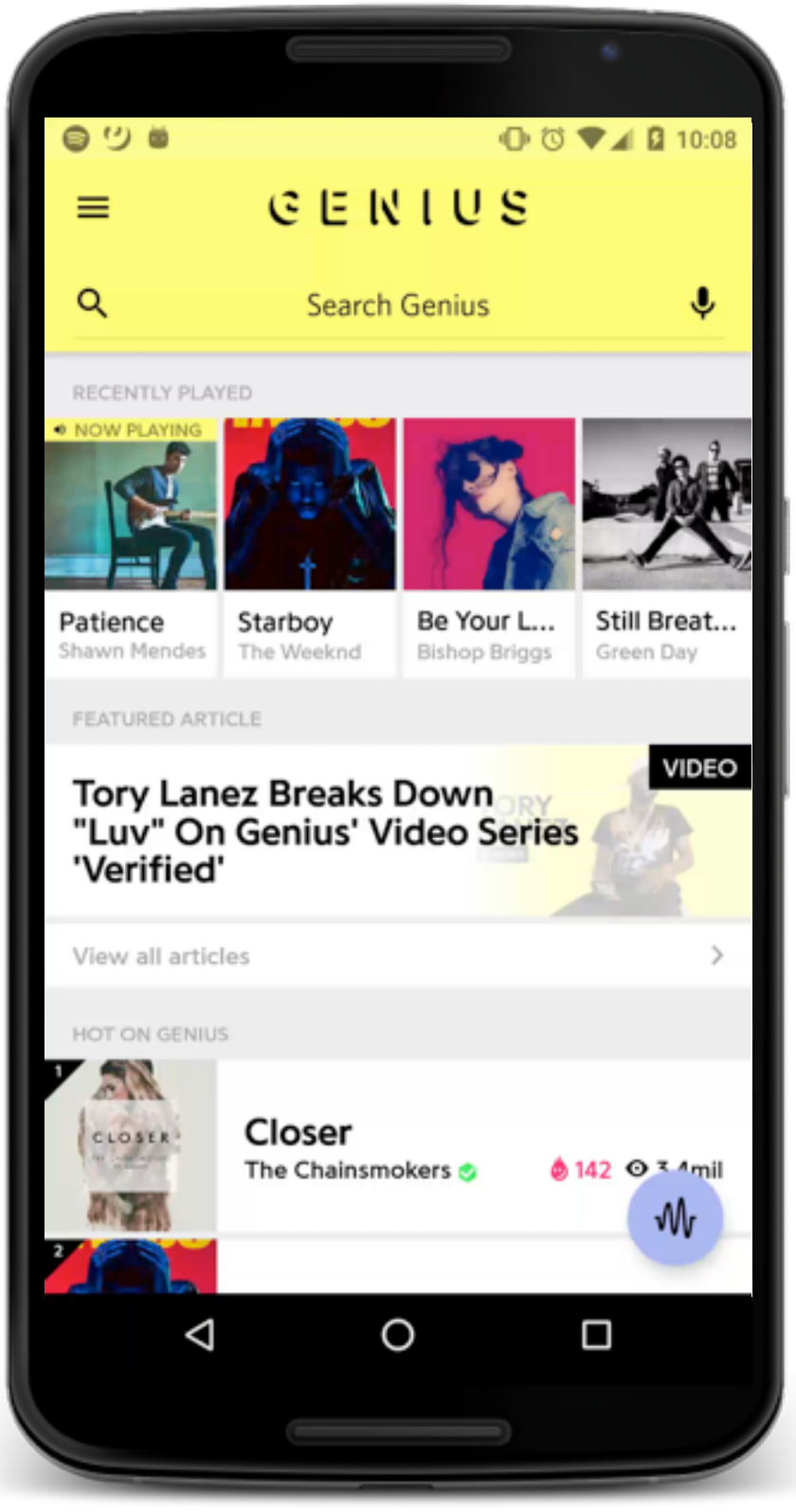
groupie

```
List<Item> commentItems;
ExpandableGroup commentGroup;
commentGroup.addAll(commentItems);
commentGroup.toggleExpanded();

commentGroup.toggleExpanded();
```

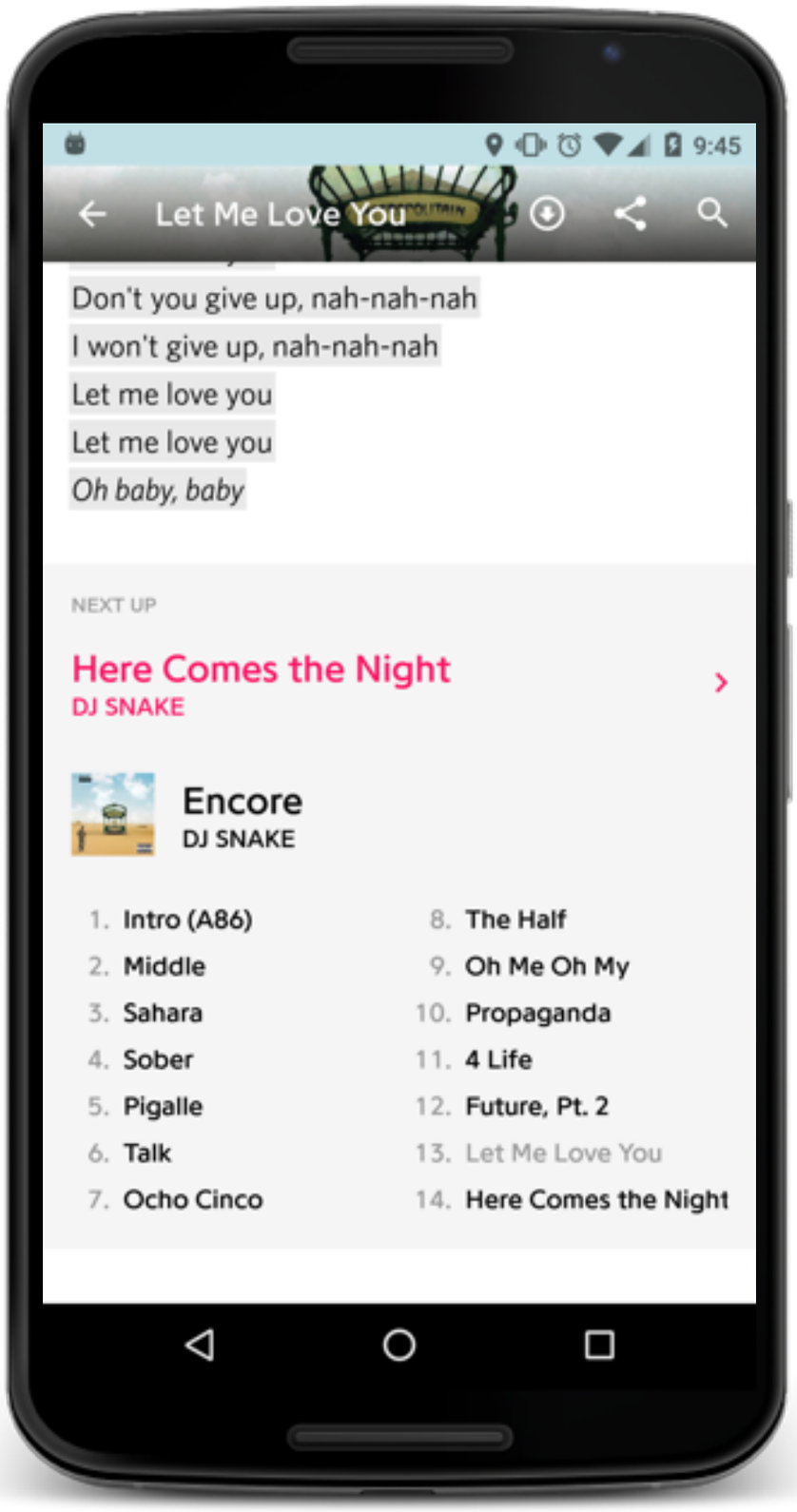


DiffUtil x
Groupie.**UpdatingGroup**

@lisawrayz


10:08
GENIUS
Search Genius
RECENTLY PLAYED
NOW PLAYING
Patience
Shawn Mendes
Starboy
The Weeknd
Be Your L...
Bishop Briggs
Still Breat...
Green Day
FEATURED ARTICLE
VIDEO
Tory Lanez Breaks Down "Luv" On Genius' Video Series 'Verified'
View all articles
HOT ON GENIUS
1
CLOSER
Closer
The Chainsmokers
142
2

Groups are like a mini adapter — can fool GLM into vertical columns



Encapsulation along with efficient recycling (instead of one large item)

Don't you give up, nah-nah-nah
I won't give up, nah-nah-nah
Let me love you
Let me love you
Oh baby, baby

NEXT UP

Here Comes the Night ›
DJ SNAKE

Encore
DJ SNAKE

1. Intro (A86)
2. Middle
3. Sahara
4. Sober
5. Pigalle
6. Talk
7. Ocho Cinco
8. The Half
9. Oh Me Oh My
10. Propaganda
11. 4 Life
12. Future, Pt. 2
13. Let Me Love You
14. Here Comes the Night

Don't you give up, nah-nah-nah
I won't give up, nah-nah-nah
Let me love you
Let me love you
Oh baby, baby

NEXT UP

Here Comes the Night
DJ SNAKE ›

Encore
DJ SNAKE

1	1. Intro (A86)	8. The Half
2	2. Middle	9. Oh Me Oh My
3	3. Sahara	10. Propaganda
4	4. Sober	11. 4 Life
5	5. Pigalle	12. Future, Pt. 2
6	6. Talk	13. Let Me Love You
7	7. Ocho Cinco	14. Here Comes the Night

examples here!
(whether or not you use the lib)

github.com/Genius/groupie

ItemDecoration

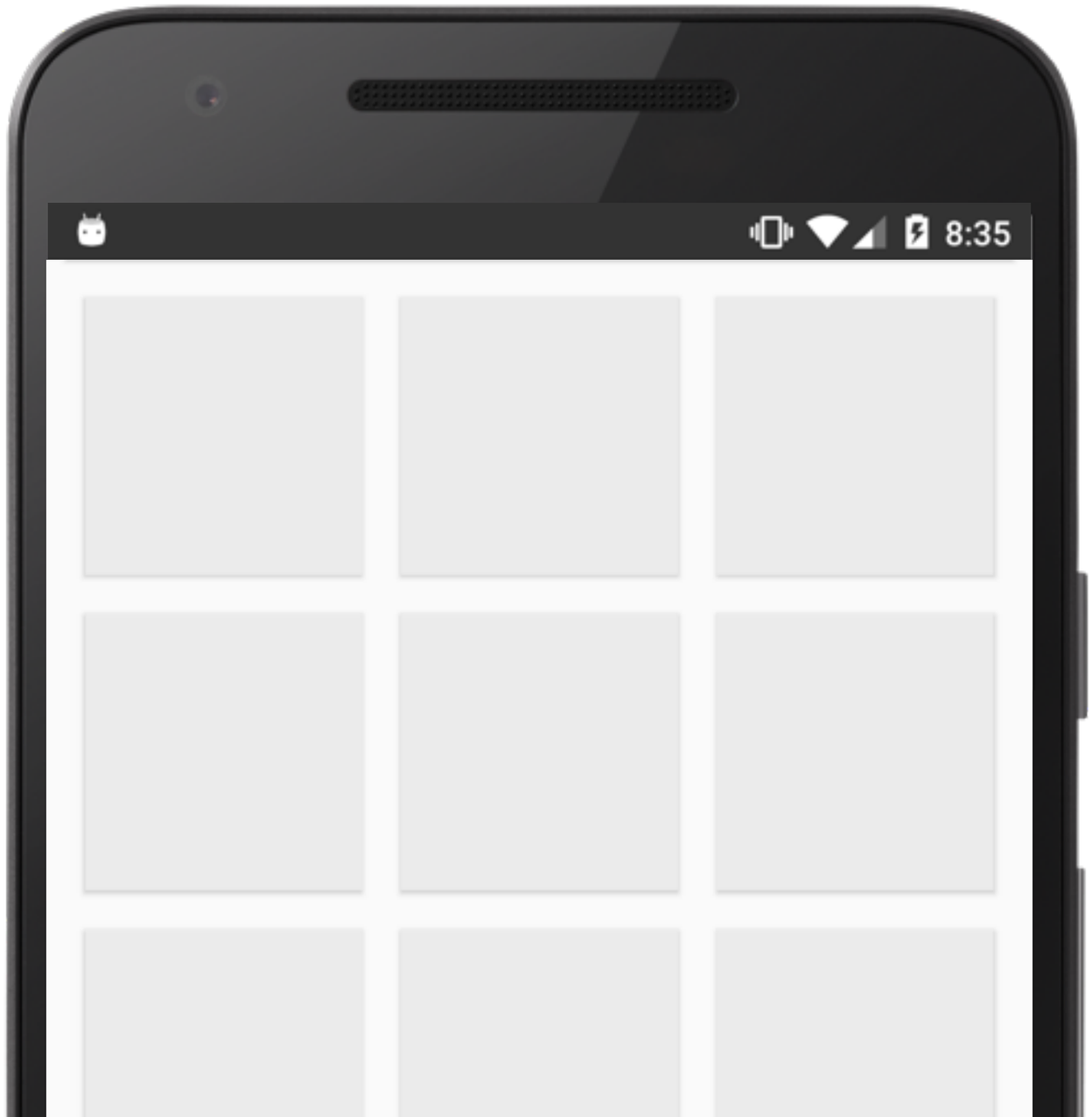
```
public class ItemDecoration {

    public void getItemOffsets(…) {}

    public void onDraw(…) {}

    public void onDrawOver(…) {}

}
```

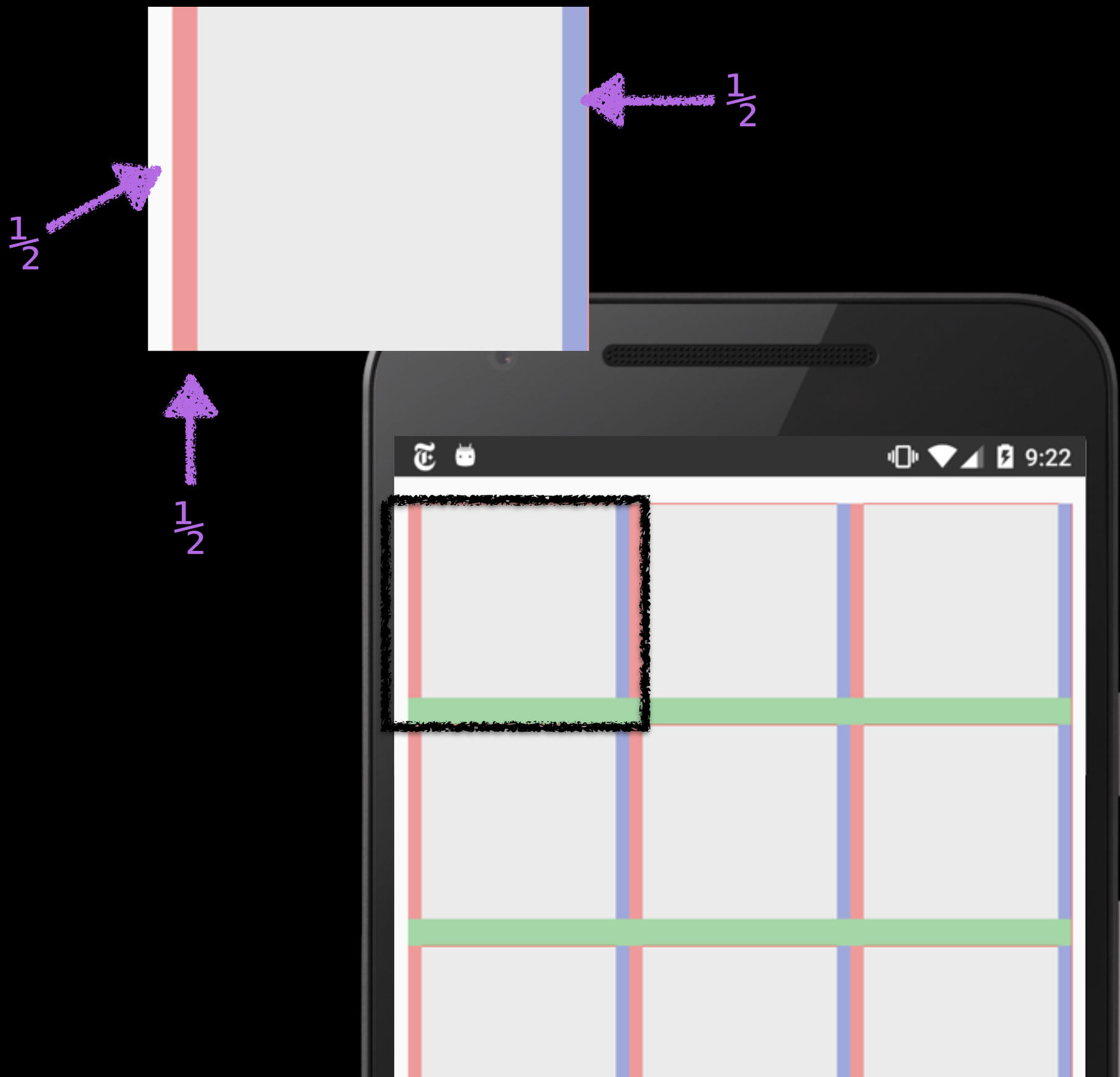
```
public class ItemDecoration {

    public void getItemOffsets(…) {}

    public void onDraw(…) {}

    public void onDrawOver(…) {}

}
```

common request:
space my columns
evenly

simple solution

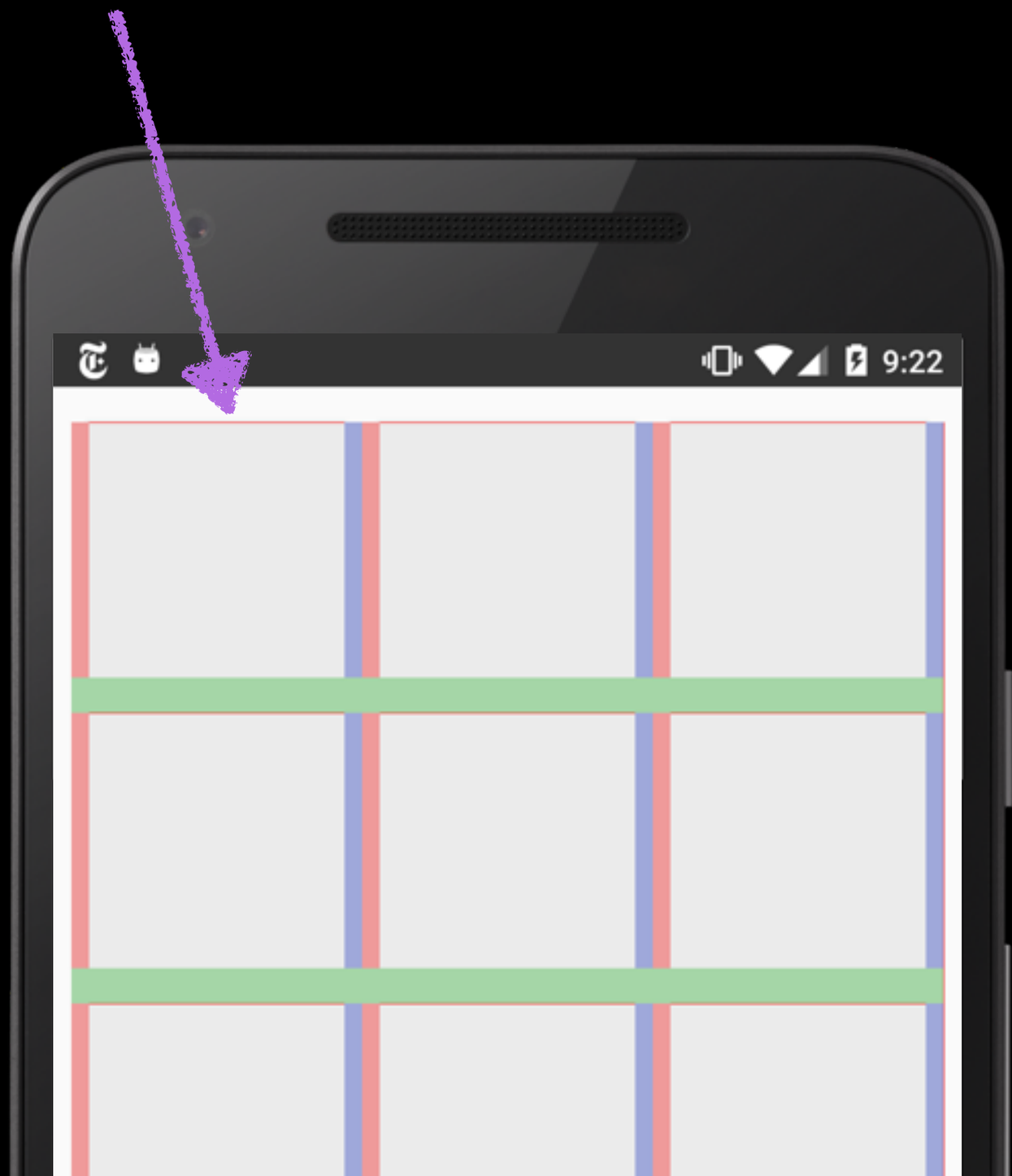
1/2 padding on outsides of RV, 1/2 padding on each side of item





@lisawrayz

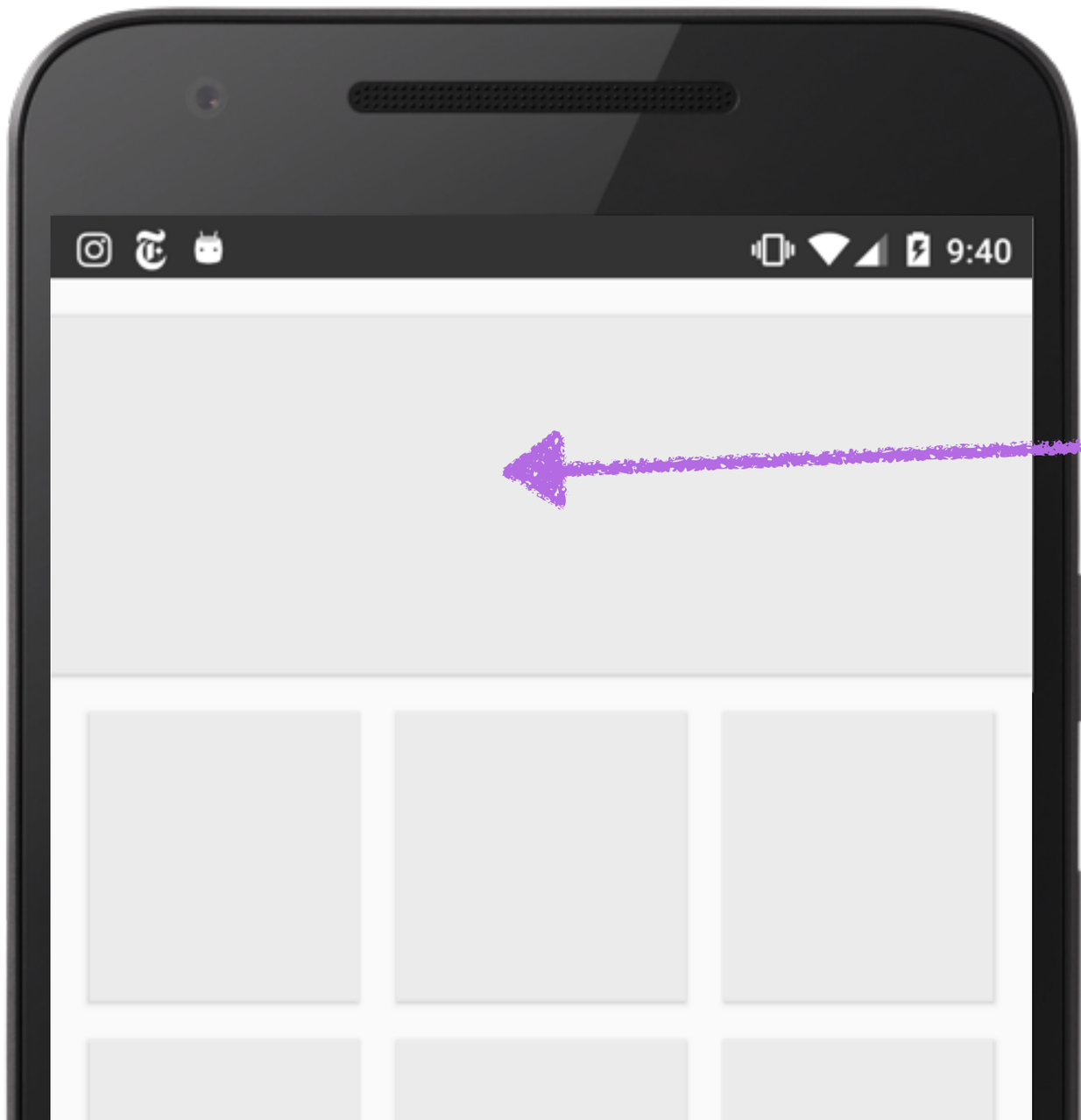
```
android:paddingTop=“@dimen/padding”
android:clipToPadding=“false”
```



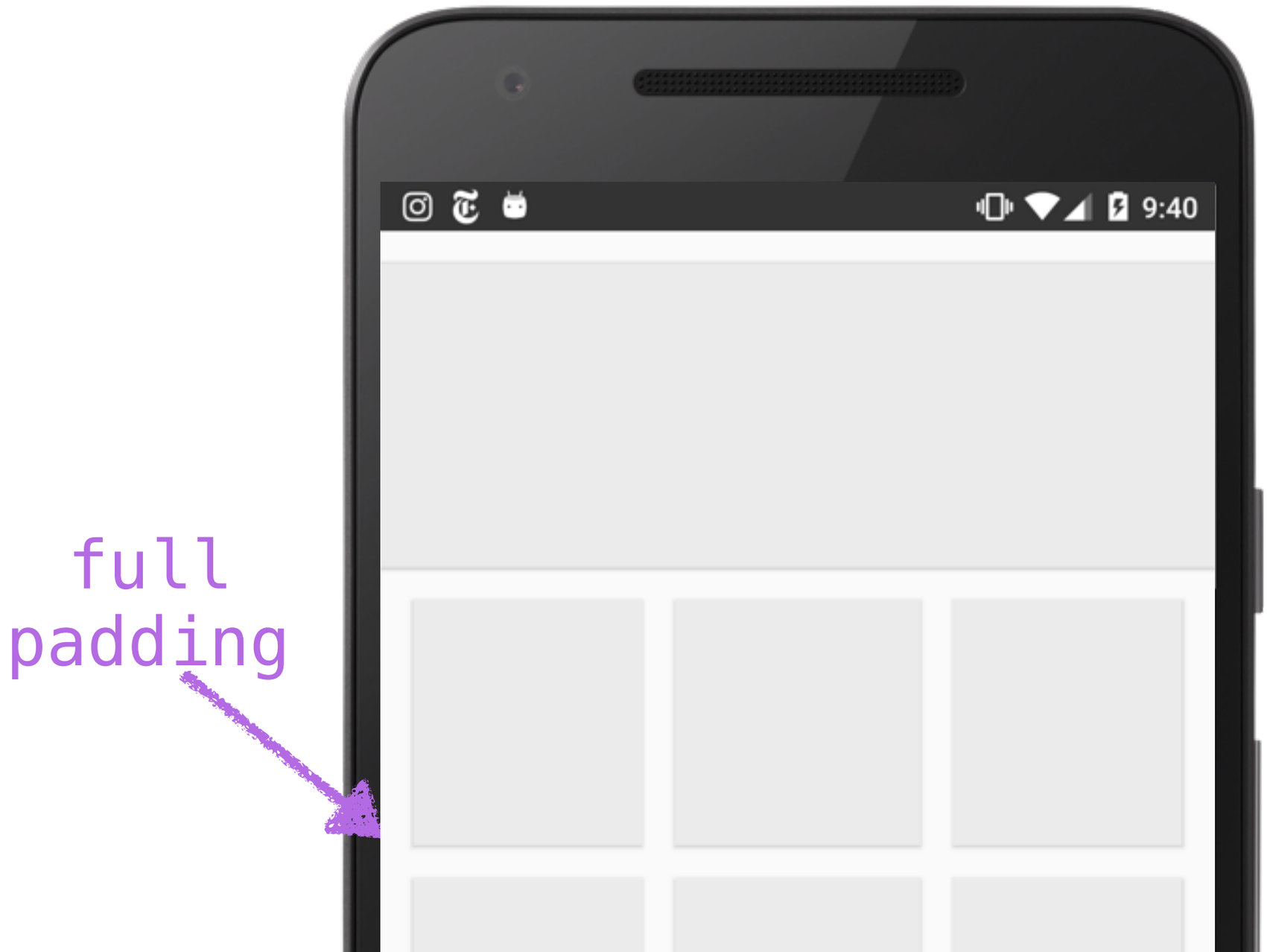
@lisawrayz

complex solution

what if we can't use padding?



2x offsets on edges?



Is this item on the left edge, right edge, middle, …?

```
@Override public void getItemOffsets(Rect outRect, View view,
    RecyclerView parent, RecyclerView.State state) {

    RecyclerView.ViewHolder viewHolder = parent.getChildViewHolder(view);

    GridLayoutManager.LayoutParams layoutParams = view.getLayoutParams();
    GridLayoutManager gridLayoutManager = parent.getLayoutManager();

    int spanSize = layoutParams.getSpanSize();
    int totalSpanSize = gridLayoutManager.getSpanCount();

    if (spanSize + layoutParams.getSpanIndex() == totalSpanSize) {
        // Item reaches to right edge of list
        outRect.right = padding;
    }
    if (layoutParams.getSpanIndex() == 0) {
        // Item's left edge is on left edge of list
        outRect.left = padding;
    }
}
```

Is this item on the left edge, right edge, middle, …?

```
@Override public void getItemOffsets(Rect outRect, View view,
    RecyclerView parent, RecyclerView.State state) {

    RecyclerView.ViewHolder viewHolder = parent.getChildViewHolder(view);

    GridLayoutManager.LayoutParams layoutParams = view.getLayoutParams();
    GridLayoutManager gridLayoutManager = parent.getLayoutManager();

    int spanSize = layoutParams.getSpanSize();
    int totalSpanSize = gridLayoutManager.getSpanCount();

    if (spanSize + layoutParams.getSpanIndex() == totalSpanSize) {
        // Item reaches to right edge of list
        outRect.right = padding;
    }
    if (layoutParams.getSpanIndex() == 0) {
        // Item's left edge is on left edge of list
        outRect.left = padding;
    }
}
```

Is this item on the left edge, right edge, middle, …?

```
@Override public void getItemOffsets(Rect outRect, View view,
    RecyclerView parent, RecyclerView.State state) {

    RecyclerView.ViewHolder viewHolder = parent.getChildViewHolder(view);

    GridLayoutManager.LayoutParams layoutParams = view.getLayoutParams();
    GridLayoutManager gridLayoutManager = parent.getLayoutManager();

    int spanSize = layoutParams.getSpanSize();
    int totalSpanSize = gridLayoutManager.getSpanCount();

    if (spanSize + layoutParams.getSpanIndex() == totalSpanSize) {
        // Item reaches to right edge of list
        outRect.right = padding;
    }
    if (layoutParams.getSpanIndex() == 0) {
        // Item's left edge is on left edge of list
        outRect.left = padding;
    }
}
```

Is this item on the left edge, right edge, middle, …?

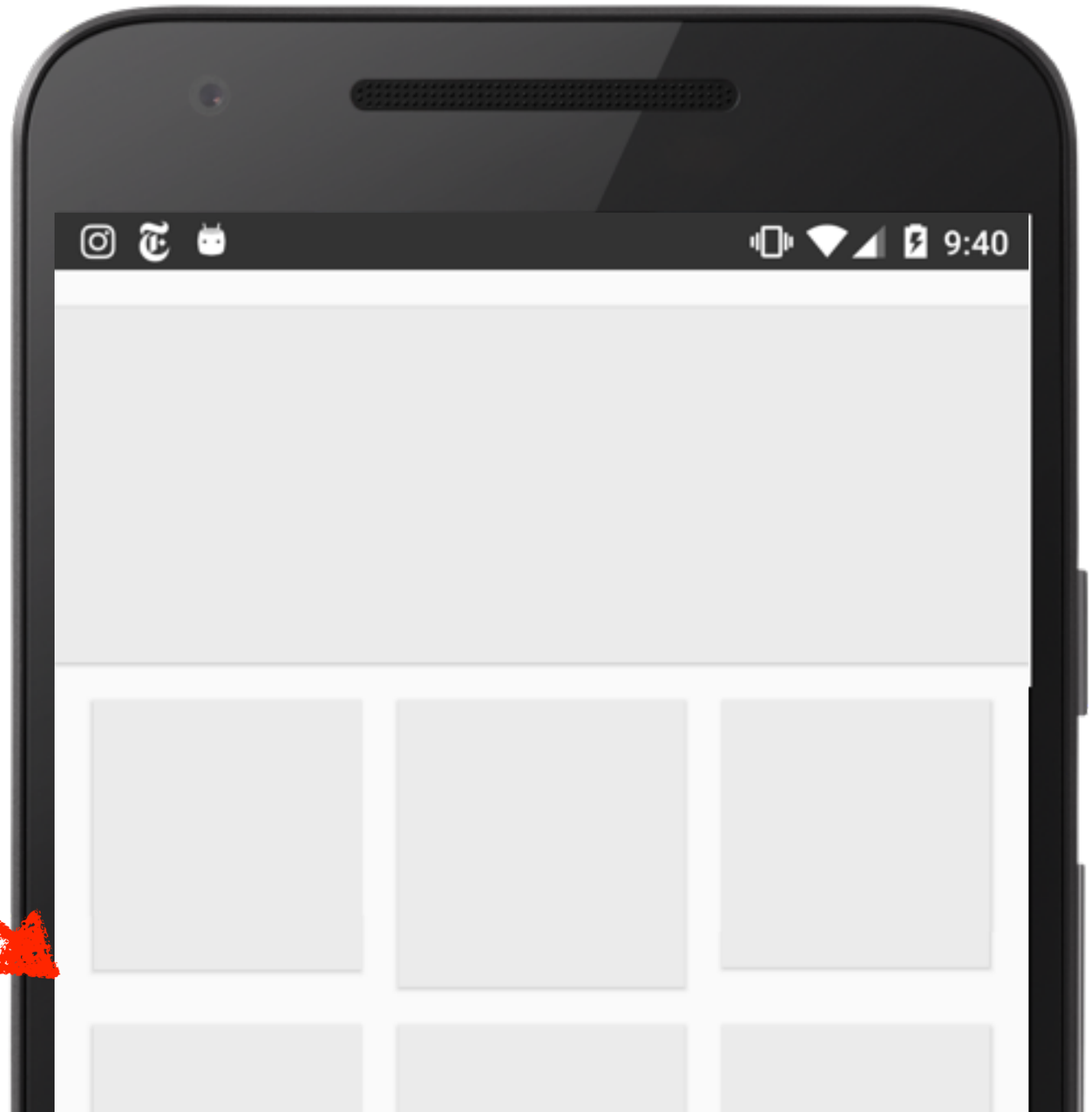
```
@Override public void getItemOffsets(Rect outRect, View view,
    RecyclerView parent, RecyclerView.State state) {

    RecyclerView.ViewHolder viewHolder = parent.getChildViewHolder(view);

    GridLayoutManager.LayoutParams layoutParams = view.getLayoutParams();
    GridLayoutManager gridLayoutManager = parent.getLayoutManager();

    int spanSize = layoutParams.getSpanSize();
    int totalSpanSize = gridLayoutManager.getSpanCount();

    if (spanSize + layoutParams.getSpanIndex() == totalSpanSize) {
        // Item reaches to right edge of list
        outRect.right = padding;
    }
    if (layoutParams.getSpanIndex() == 0) {
        // Item's left edge is on left edge of list
        outRect.left = padding;
    }
}
```

Is this item on the left edge, right edge, middle, …?

```
@Override public void getItemOffsets(Rect outRect, View view,
    RecyclerView parent, RecyclerView.State state) {

    RecyclerView.ViewHolder viewHolder = parent.getChildViewHolder(view);

    GridLayoutManager.LayoutParams layoutParams = view.getLayoutParams();
    GridLayoutManager gridLayoutManager = parent.getLayoutManager();

    int spanSize = layoutParams.getSpanSize();
    int totalSpanSize = gridLayoutManager.getSpanCount();

    if (spanSize + layoutParams.getSpanIndex() == totalSpanSize) {
        // Item reaches to right edge of list
        outRect.right = padding;
    }
    if (layoutParams.getSpanIndex() == 0) {
        // Item's left edge is on left edge of list
        outRect.left = padding;
    }
}
```

2x offsets on edges?

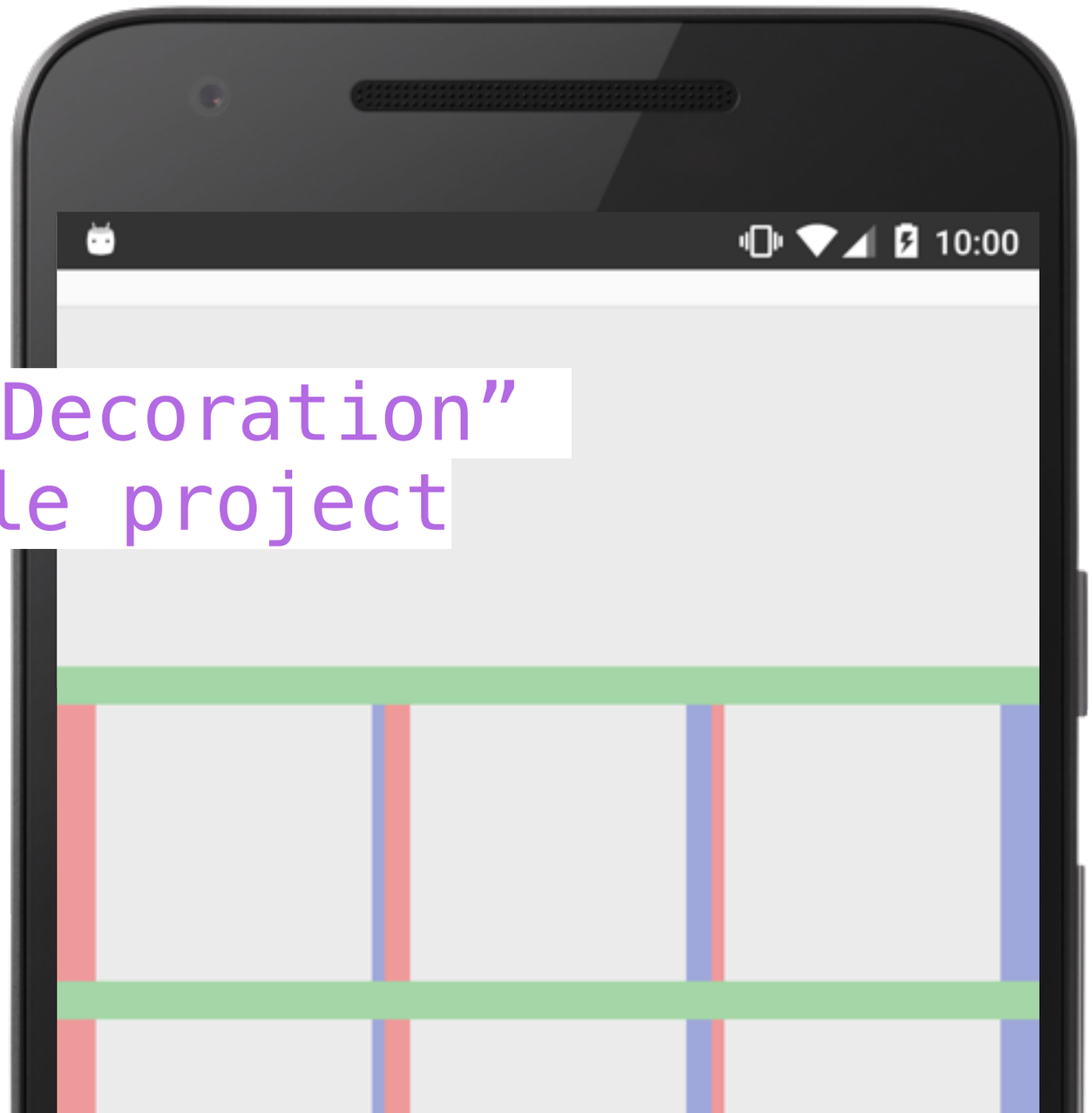
uneven item widths — offsets don't change measured item width



eek!
different
size
squares!!

each item needs same total padding …

just differently distributed



“DebugItemDecoration”
in example project

even column padding

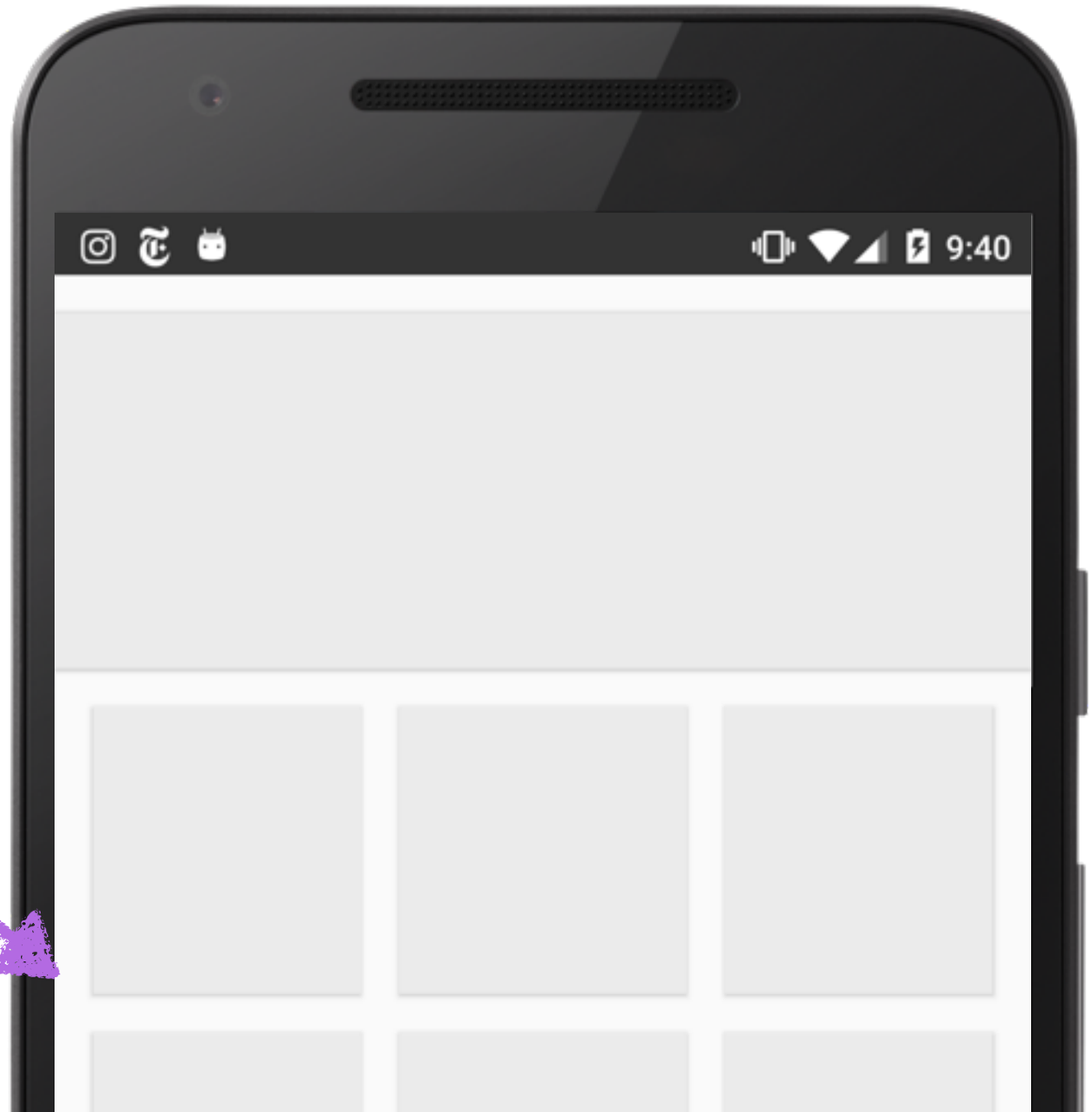
```
@Override public void getItemOffsets(Rect outRect, View view,
    RecyclerView parent, RecyclerView.State state) {

    GridLayoutManager.LayoutParams layoutParams = view.getLayoutParams();
    GridLayoutManager gridLayoutManager = parent.getLayoutManager();
    float spanSize = layoutParams.getSpanSize();
    float totalSpanSize = gridLayoutManager.getSpanCount();

    float n = totalSpanSize / spanSize; // num columns
    float c = layoutParams.getSpanIndex() / spanSize; // column index

    float leftPadding = padding * ((n - c) / n);
    float rightPadding = padding * ((c + 1) / n);

    outRect.left = (int) leftPadding;
    outRect.right = (int) rightPadding;
}
```

even column padding

```
@Override public void getItemOffsets(Rect outRect, View view,
    RecyclerView parent, RecyclerView.State state) {

    GridLayoutManager.LayoutParams layoutParams = view.getLayoutParams();
    GridLayoutManager gridLayoutManager = parent.getLayoutManager();
    float spanSize = layoutParams.getSpanSize();
    float totalSpanSize = gridLayoutManager.getSpanCount();

    float n = totalSpanSize / spanSize; // num columns
    float c = layoutParams.getSpanIndex() / spanSize; // column index

    float leftPadding = padding * ((n - c) / n);
    float rightPadding = padding * ((c + 1) / n);

    outRect.left = (int) leftPadding;
    outRect.right = (int) rightPadding;
}
```

even column padding

```
@Override public void getItemOffsets(Rect outRect, View view,
    RecyclerView parent, RecyclerView.State state) {

    GridLayoutManager.LayoutParams layoutParams = view.getLayoutParams();
    GridLayoutManager gridLayoutManager = parent.getLayoutManager();
    float spanSize = layoutParams.getSpanSize();
    float totalSpanSize = gridLayoutManager.getSpanCount();

    float n = totalSpanSize / spanSize; // num columns
    float c = layoutParams.getSpanIndex() / spanSize; // column index

    float leftPadding = padding * ((n - c) / n);
    float rightPadding = padding * ((c + 1) / n);

    outRect.left = (int) leftPadding;
    outRect.right = (int) rightPadding;
}
```



phew …
nice
and even

@lisawrayz

ItemDecorations are additive

```
recyclerView.addItemDecoration(
    new SpacingItemDecoration());

recyclerView.addItemDecoration(
    new HeaderItemDecoration(blue));
```

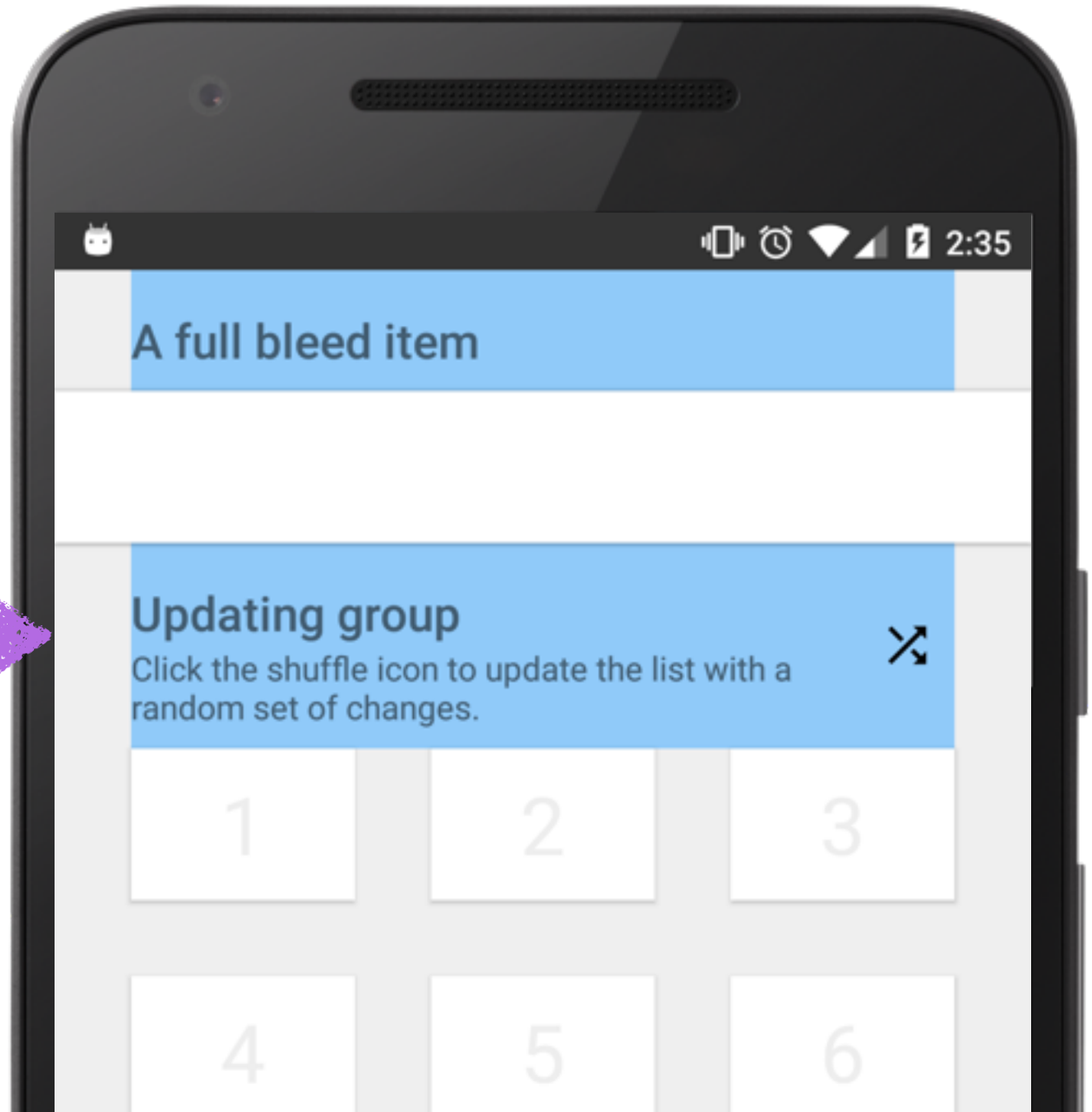
```
public class ItemDecoration {

    public void getItemOffsets(…) {}

    public void onDraw(…) {}

    public void onDrawOver(…) {}

}
```

```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {

    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);
        if (!isHeader(child, parent)) continue;

        float top = child.getTop();
        float bottom = child.getBottom();
        float right = child.getRight();
        float left = child.getLeft();
        c.drawRect(left, top, right, bottom, paint);
    }
}
```

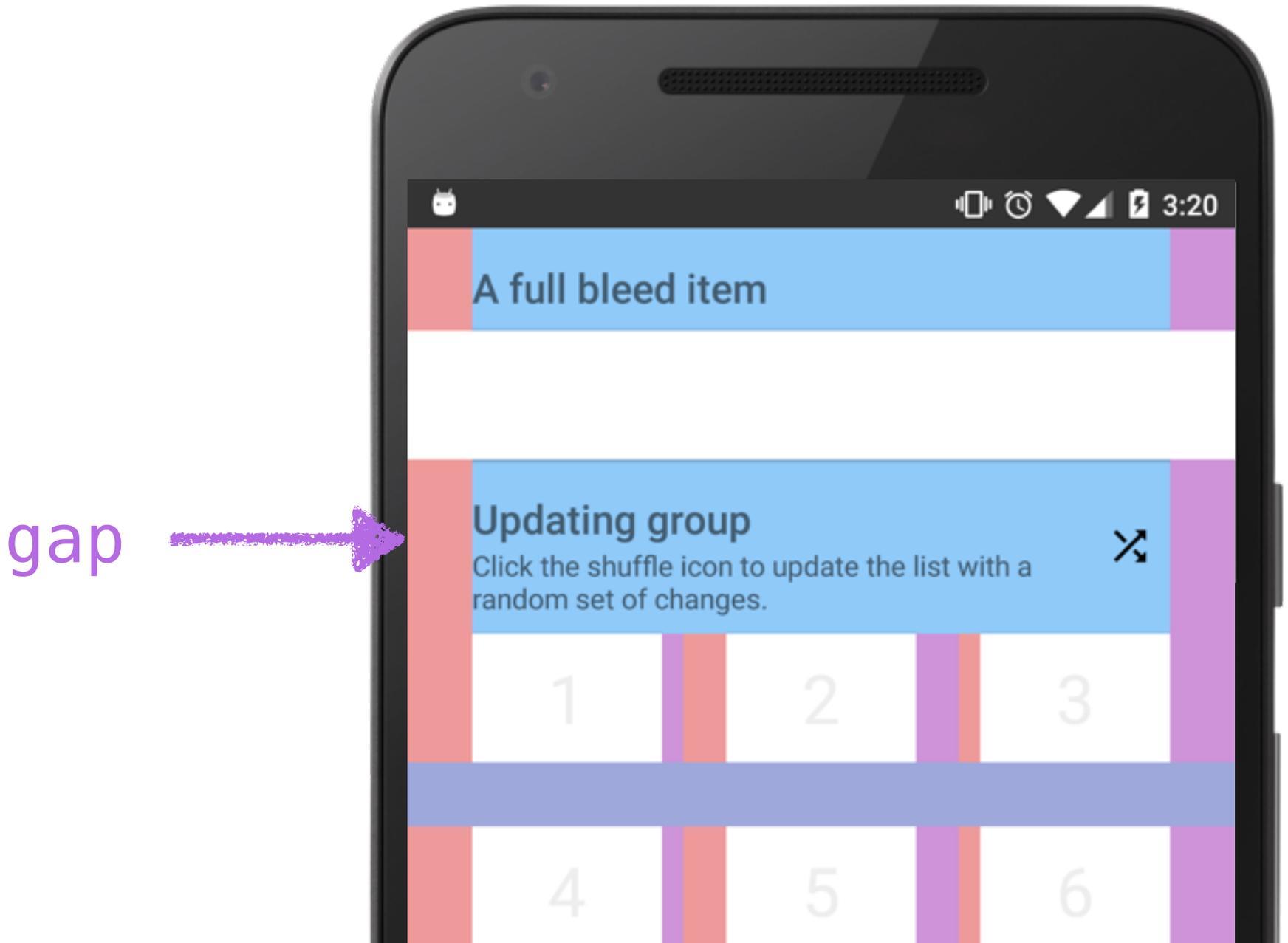
```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {

    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);
        if (!isHeader(child, parent)) continue;

        float top = child.getTop();
        float bottom = child.getBottom();
        float right = child.getRight();
        float left = child.getLeft();
        c.drawRect(left, top, right, bottom, paint);
    }
}
```

```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {

    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);
        if (!isHeader(child, parent)) continue;

        float top = child.getTop();
        float bottom = child.getBottom();
        float right = child.getRight();
        float left = child.getLeft();
        c.drawRect(left, top, right, bottom, paint);
    }
}
```

```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {

    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);
        if (!isHeader(child, parent)) continue;

        float top = child.getTop();
        float bottom = child.getBottom();
        float right = child.getRight();
        float left = child.getLeft();
        c.drawRect(left, top, right, bottom, paint);
    }
}
```



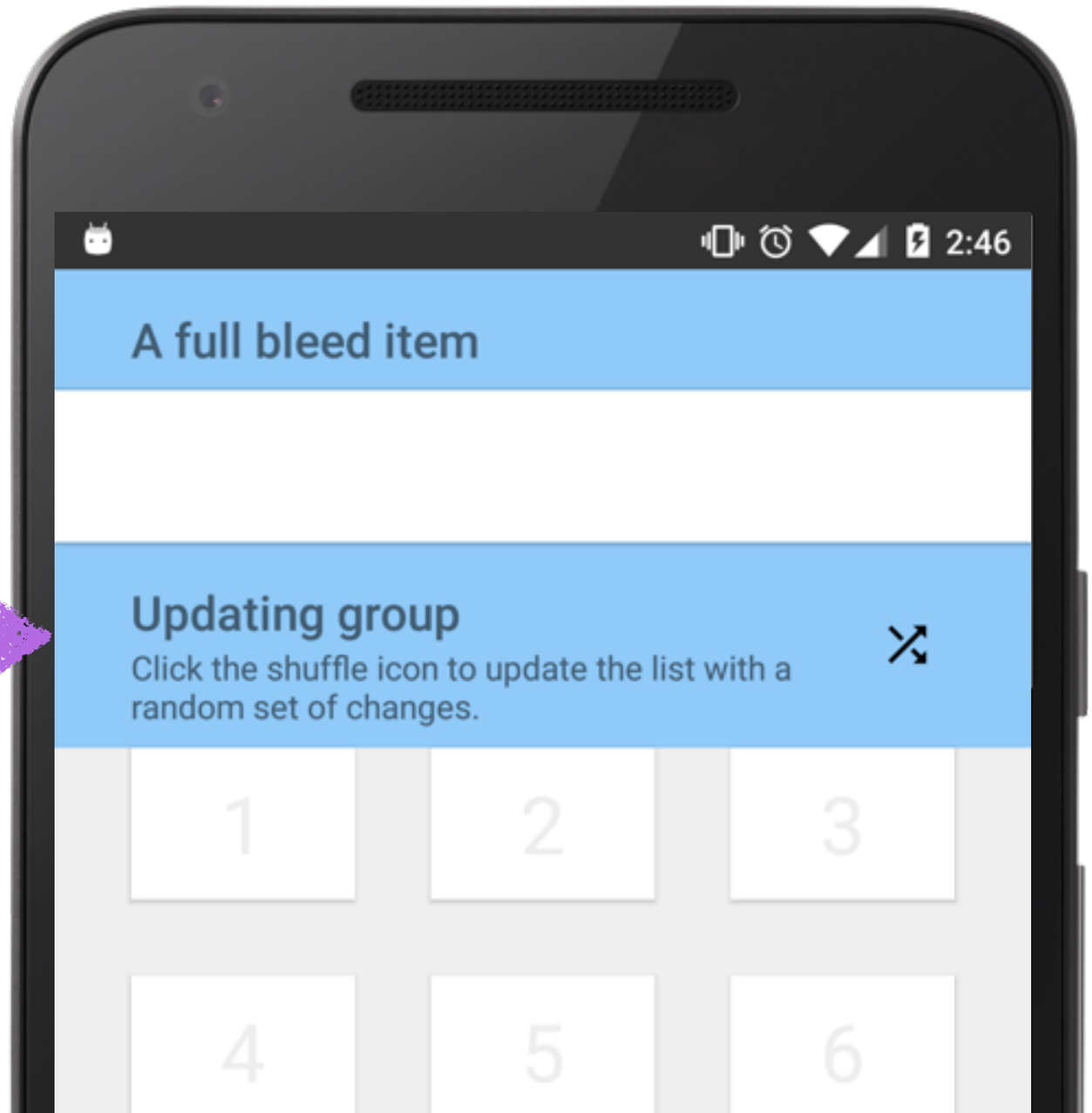
gap


3:20
A full bleed item
Updating group
Click the shuffle icon to update the list with a random set of changes.
1
2
3
4
5
6
gap

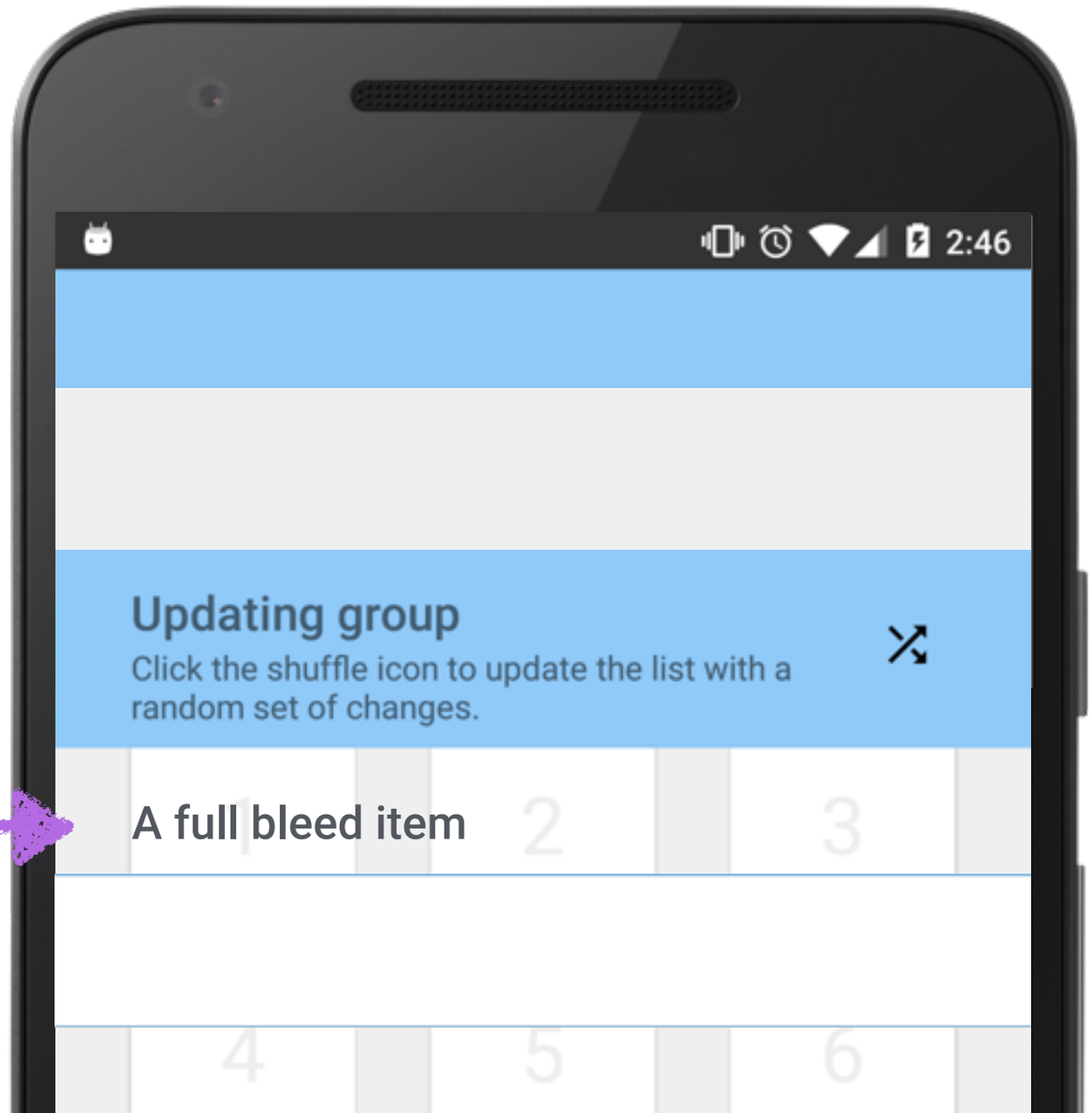
use decorated bounds

```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {
    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);
        if (!isHeader(child, parent)) continue;

        RecyclerView.LayoutManager lm = parent.getLayoutManager();
        float top = lm.getDecoratedTop(child);
        float bottom = lm.getDecoratedBottom(child);
        float right = lm.getDecoratedRight(child);
        float left = lm.getDecoratedLeft(child);
        c.drawRect(left, top, right, bottom, paint);
    }
}
```

much better



Items move!



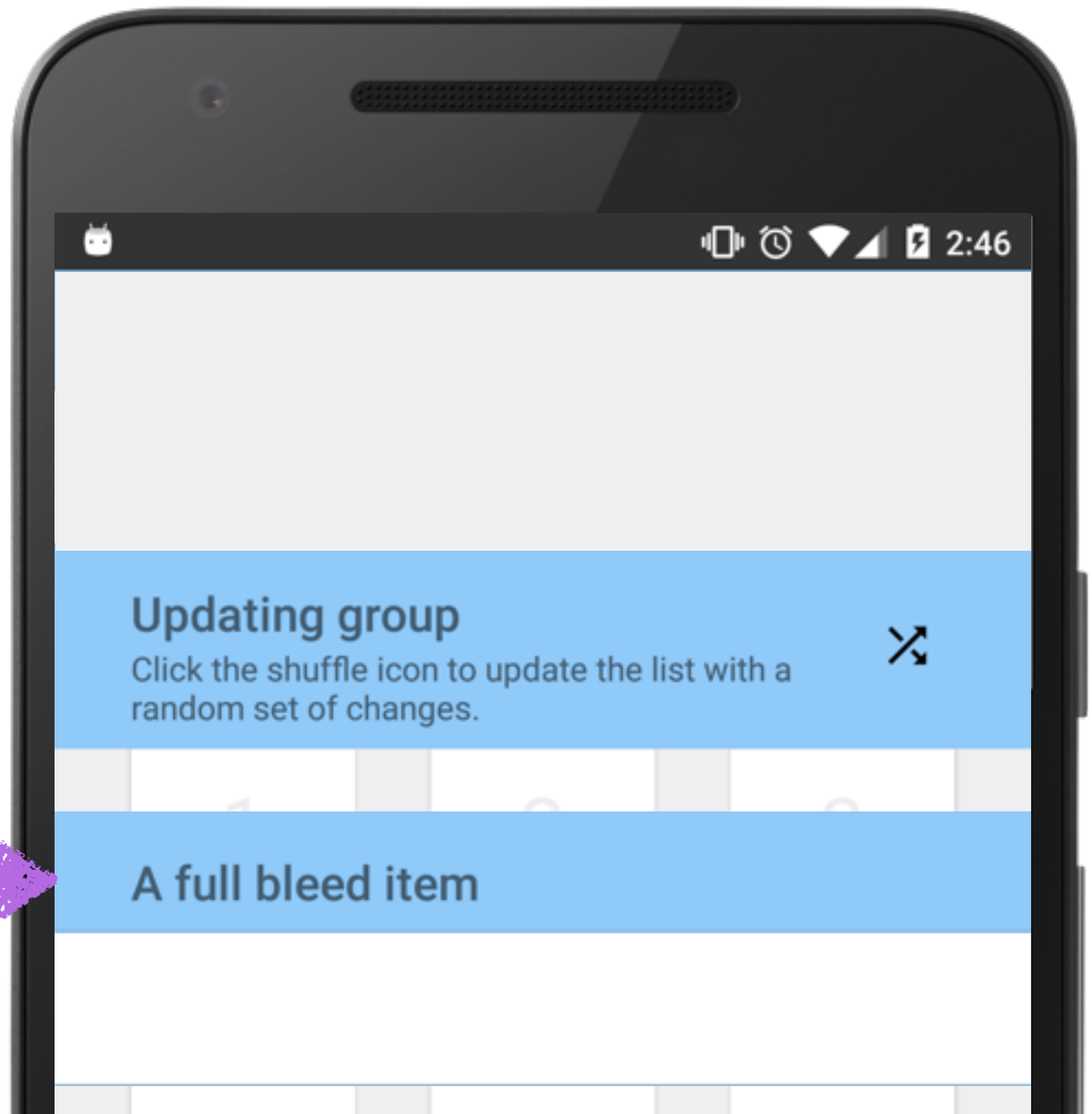
yikes

use translation

```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {
    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);
        if (!isHeader(child, parent)) continue;

        RecyclerView.LayoutManager lm = parent.getLayoutManager();
        float top = lm.getDecoratedTop(child)
            + child.getTranslationY();
        float bottom = lm.getDecoratedBottom(child)
            + child.getTranslationY();
        float right = lm.getDecoratedRight(child)
            + child.getTranslationX();
        float left = lm.getDecoratedLeft(child)
            + child.getTranslationX();
        c.drawRect(left, top, right, bottom, paint);
    }
```

use translation

```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {
    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);
        if (!isHeader(child, parent)) continue;

        RecyclerView.LayoutManager lm = parent.getLayoutManager();
        float top = lm.getDecoratedTop(child)
            + child.getTranslationY();
        float bottom = lm.getDecoratedBottom(child)
            + child.getTranslationY();
        float right = lm.getDecoratedRight(child)
            + child.getTranslationX();
        float left = lm.getDecoratedLeft(child)
            + child.getTranslationX();
        c.drawRect(left, top, right, bottom, paint);
    }
```



better

@lisawrayz

use layout position

```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {
    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);

        int position = parent.getChildAdapterPosition(child);
    }
}
```

use layout position

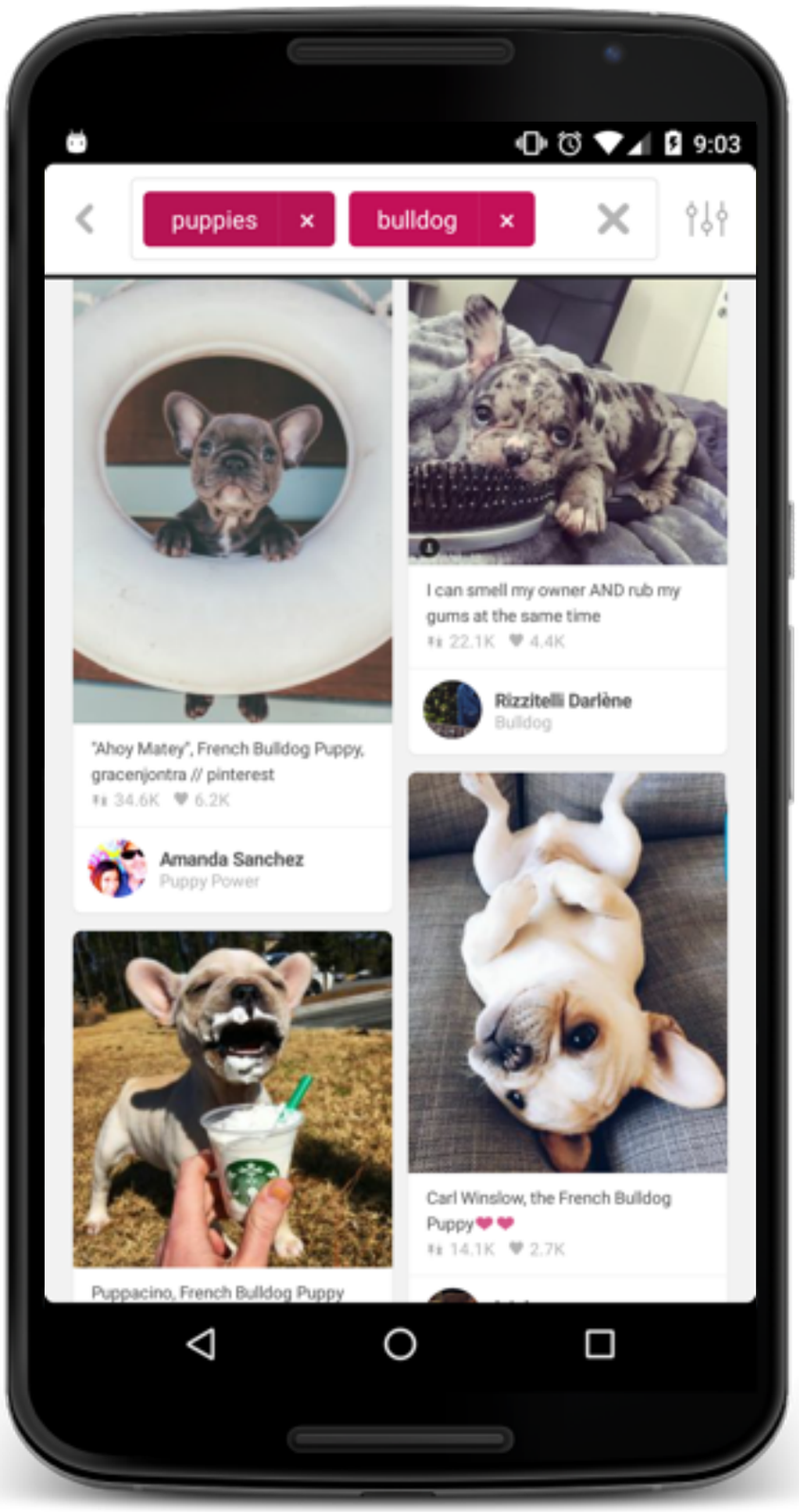
```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {
    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);

        int position = parent.getChildAdapterPosition(child);
    }
}
```

can be NO_POSITION
during animation

use layout position

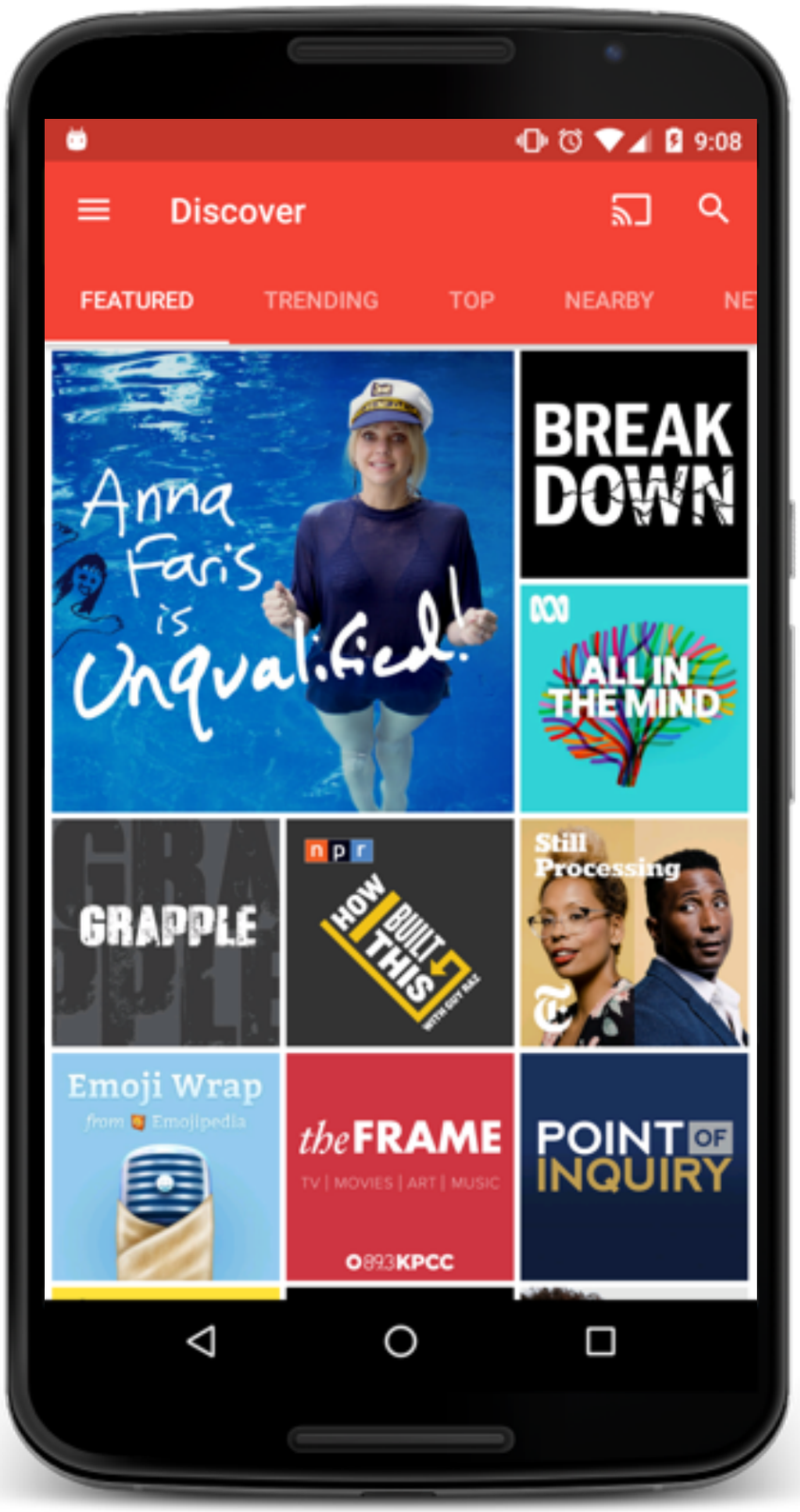
```
@Override public void onDraw(Canvas c, RecyclerView parent,
    RecyclerView.State state) {
    for (int i = 0; i < parent.getChildCount(); i++) {
        View child = parent.getChildAt(i);

        int position = parent.getChildLayoutPosition(child);
    }
}
```

other/custom layout managers



included in Android:

`StaggeredGridLayoutManager`

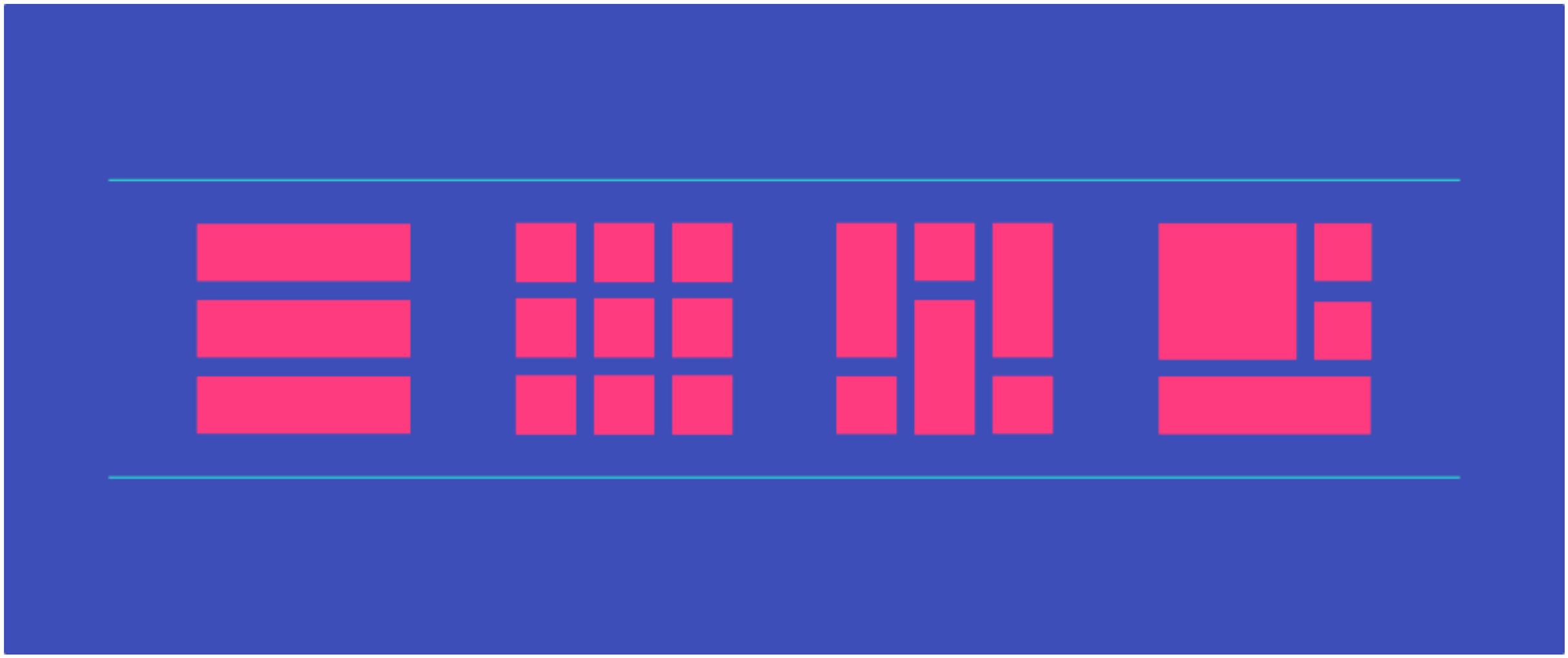


not in Android:

`SpannedGrid LayoutManager ?`

Two way view

only maven snapshots right now

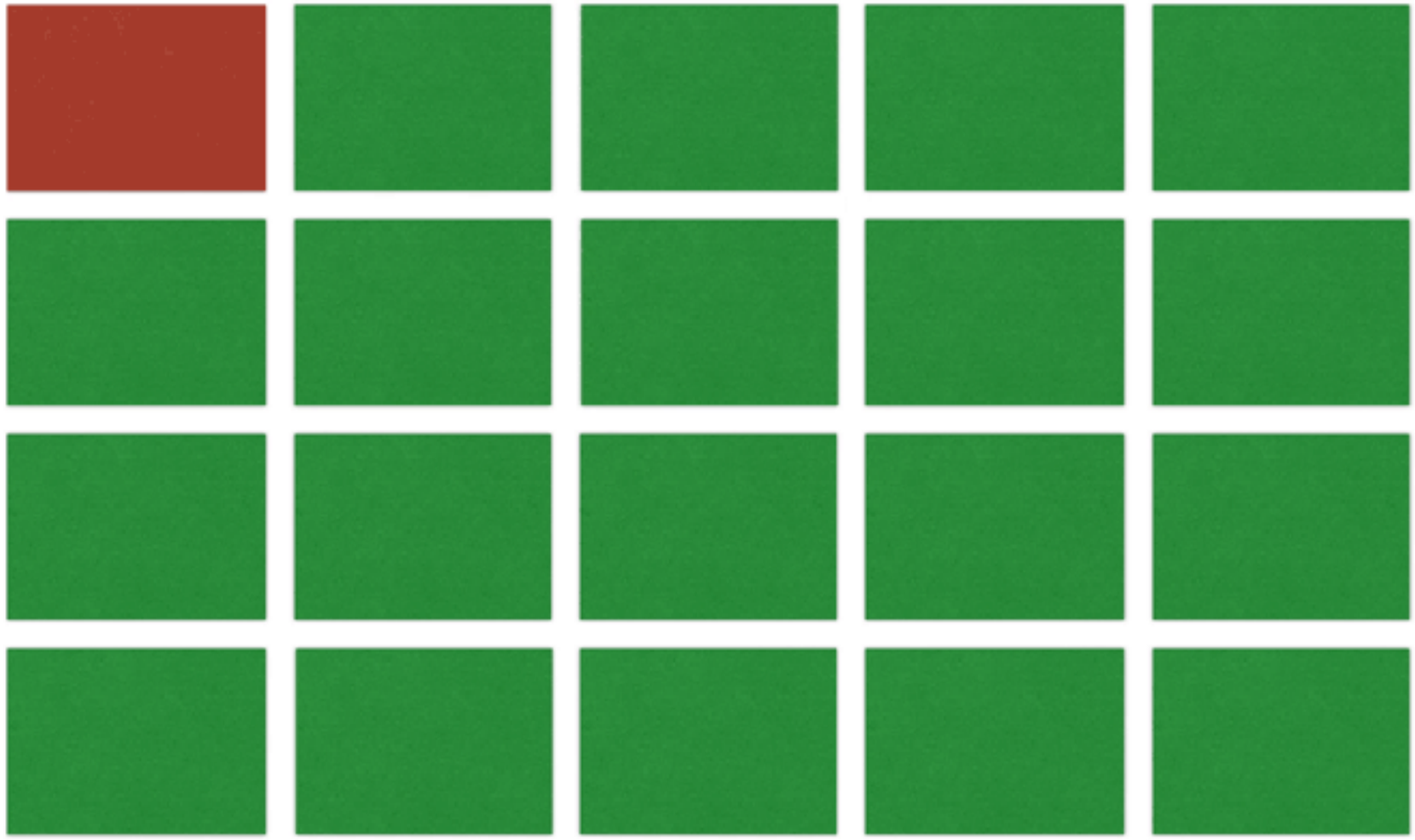


lucasr.org/2014/07/31/the-new-twowayview/

github.com/lucasr/twoway-view

“Building a RecyclerView LayoutManager: Part 1–3”, Dave Smith

in-depth tutorial



wiresareobsolete.com/2014/09/building-a-recyclerview-layoutmanager-part-1/

with great power comes great responsibility

-recyclerview

GENIUS

github.com/Genius/groupie

+**Lisa Wray** Zeitouni

@lisawrayz