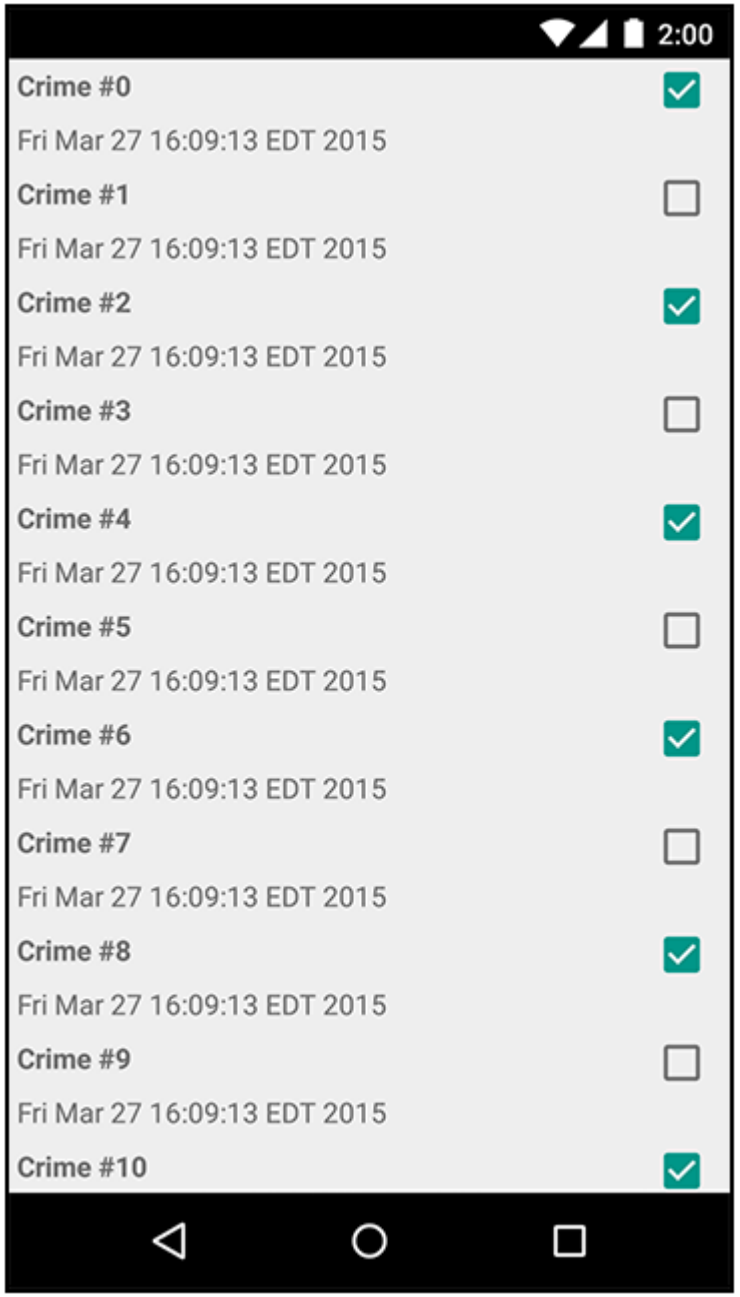
Android Lists

Purpose

Display a list of items, each with some internal structure (text, checkbox)

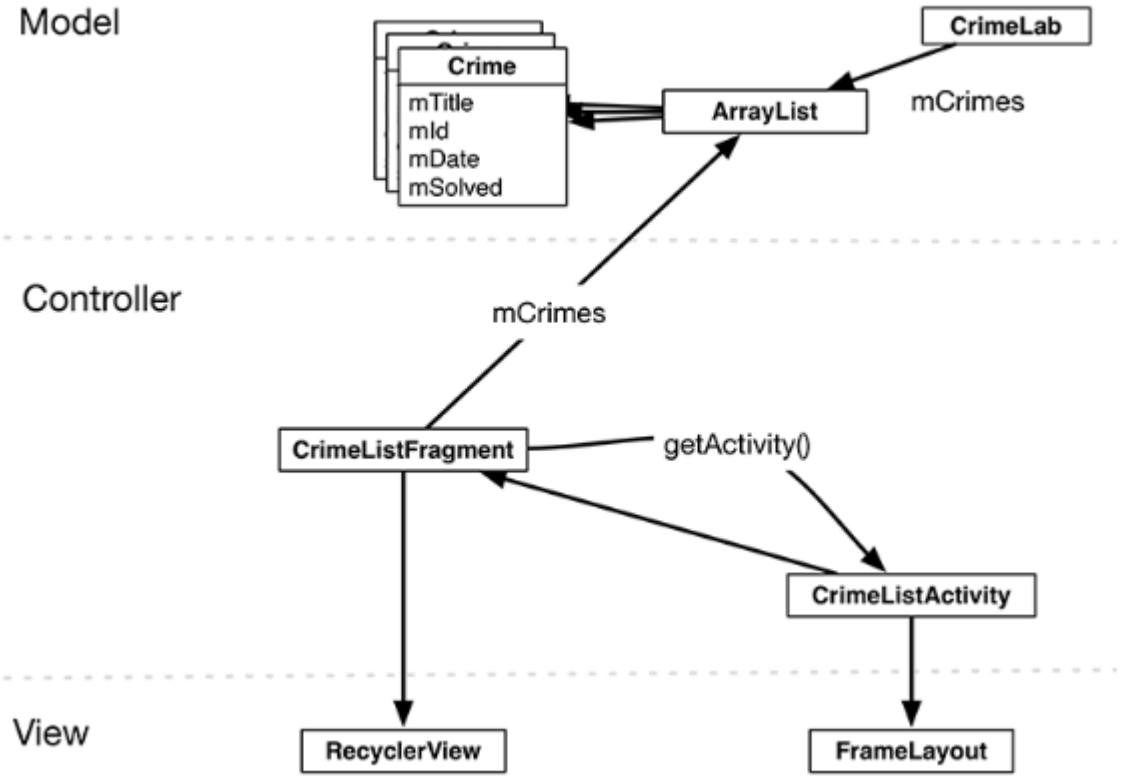


From ListView to Recycler View

- Older versions of Android (pre-5) accomplished this via **`ListView`** (and **`GridView`**)
- Although less intuitively named, **`RecyclerView`** makes it easier to do some more advanced operations
  - Horizontal scrolling
  - Animation (displaying removal/addition of items)

MVC

Figure 9.2 CriminalIntent with a list of crimes



Singleton classes

- Recall our experience with saved state instance bundles: **`Activity`** objects “go away” when we switch to another activity.
- We can solve this with our existing Java skills: create a special kind of *factory* class that will:
  - Give us a new instance of itself the first time we access it
  - Use that same instance thereafter (throughout the life of the app)

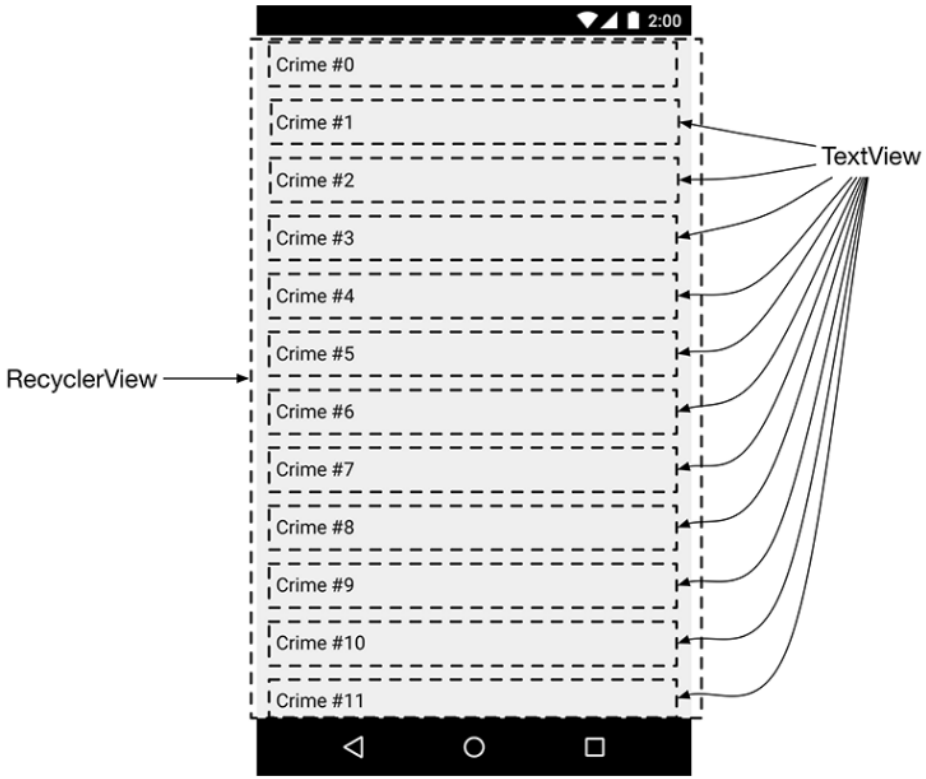
Singleton classes

```
public class CrimeLab {

    private static CrimeLab sCrimeLab;

    public static CrimeLab get(Context context) {

        if (sCrimeLab == null) {
            sCrimeLab = new CrimeLab(context);
        }

        return sCrimeLab;
    }

    private CrimeLab(Context context) {
        // Private, so nobody else call call constructor.
        // Context unused for now.
    }
}
```

Fleshing out `CrimeLab`

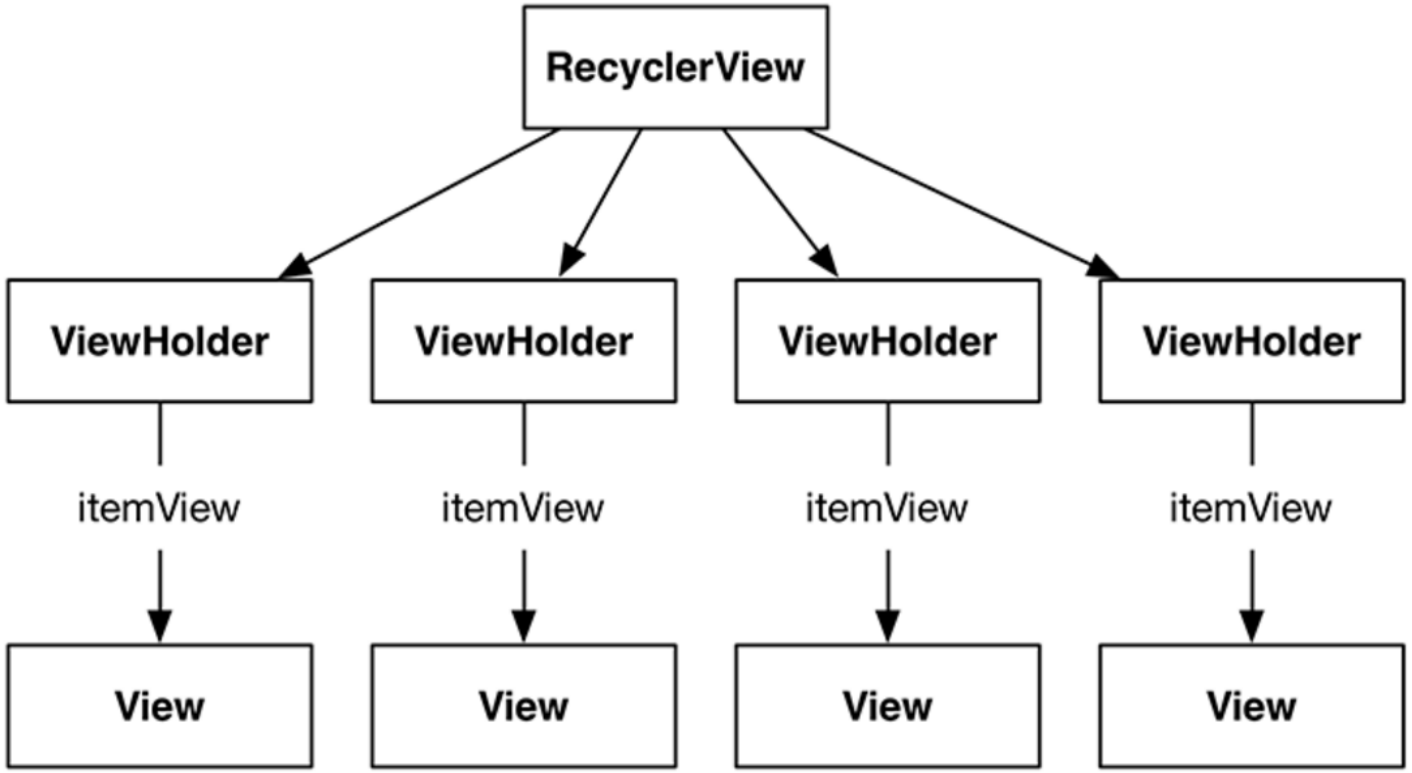
```
public class CrimeLab {

    private static CrimeLab sCrimeLab;
    private List<Crime> mCrimes;            // abstract List type

    // ...
    private CrimeLab(Context context) {
        mCrimes = new ArrayList<>();        // concrete implementation
    }

    public List<Crime> getCrimes() {        // abstract return type
        return mCrimes;
    }

    public Crime getCrime(UUID id) {        // UUID = unique identifier
        for (Crime crime : mCrimes) {
            if (crime.getId().equals(id)) {
                return crime;
            }
        }
        return null;
    }
}
```

Back to `RecyclerView`



Underneath the hood, it's a little more complicated than this ....

Each TextView needs a ViewHolder



Why we need `ViewHolder`

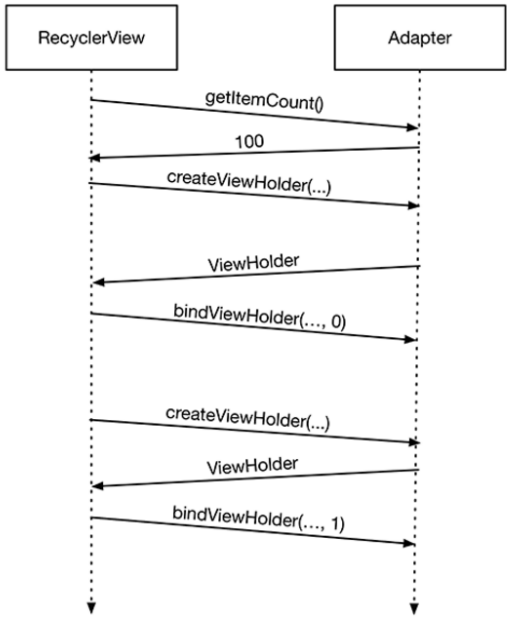
- Often there's some other stuff we need to put in the list item, like a thumbnail image:

Listing 9.13 A typical `ViewHolder` subclass

```
public class ListRow extends RecyclerView.ViewHolder {
    public ImageView mThumbnail;

    public ListRow(View view) {
        super(view);

        mThumbnail = (ImageView)
view.findViewById(R.id.thumbnail);
    }
}
```

But wait it gets worse: `RecyleView.Adapter`

- `RecyclerView` will "talk with" its `Adapter` to get the appropriate `ViewHolder`:



- Hence the name: `RecyclerView` will "recyle" old `ViewHolder` object once it no longer needs them (e.g., list scrolls down)
- `Holder` allows us to do the expensive `findViewById()` calls once, then stores results for future use.