# Aufgabe 6.2

Paket *de.moviemanager.data:*

- Von welcher Klasse erbt die Klasse *Movie*? **-** *Portrayable*
- Wofür ist diese abstrakte Klasse zuständig? – sie implementiert die Interfaces *Identifiable, Nameable, Rateable, Parcelable*, damit erbende Klassen die dazu gehörigen Methoden verwenden können (Objekt mit Namen identifziieren, Rating vergeben, etc.).
- Erben weitere Klassen von dieser abstrakten Klasse, wenn ja, welche? – Ja, die Klasse *Performer*.

Manifest der App:

- Wie wird im Manifest eine Aktivität markiert, die beim Start der App ausgeführt wird? – mit dem folgenden Boilerplate Code:

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

- Welche Aktivität ist auf diese Weise gekennzeichnet? – die Aktivität *.ui.MasterActivity.*
- Welche implementierten Lebenszyklus-Operationen dieser Aktivität werden beim Start der Movie-Manager App aufgerufen? - *onCreate()*, *onStart().*
- Was geschieht in diesen Operationen? - *onCreate()* implementiert Startup-Logik, die nur einmal im Lebenszyklus der Aktivität geschieht; *onStart()* macht die Aktivität sichtbar für den Nutzer und initialisiert den Code, der die UI aufrechterhält.

Paket *de.moviemanager.android:*

- Vergleichen Sie diesen ResultHandling-Mechanismus mit der von Android bereitgestellten Version von *startActivityForResult* ohne *ResultHandling* - **???**

Paket *de.moviemanager.core.storage:*

- Welche Klassen der Movie Manager App werden in der Klasse *RuntimeStorage* verwendet? – *Movie*, *Performer* und *ImagePyramid.*

- Vergleichen Sie die Verwendung von (reversiblen) Transaktionen mit der Verwendung des Storage im Pokemon Manager.

| | Reversible Transaktionen | PokemonManager Storage |
|---|---|---|
| Einfachheit der Umsetzung | -- | + |
| Nutzerfreundlichkeit | ++ („Back-Button“) | -- (keine Fehlertoleranz nach Usability-Grundsätzen!) |

- Was müsste man, unabhängig von der UI, tun, wenn man eine neue Datenklasse hinzufügt?
  Mit de.moviemanager.core.json.__NEWCLASS__FromJsonObject arbeiten und entsprechend den JSON-Storage für die neue Klasse implementieren

Paket *de.moviemanager.ui.detail:*

- Was fällt Ihnen bei den Attributen für die Darstellung der Attribute eines Films auf? - die Datenobjekte werden mit den entsprechenden Views in der *MovieDetailActivity* “gebunden” (R.bind…)
- Welchen Zweck haben diese in *MovieDetailEditActivity*? – die Attribute werden dort an die Views mithilfe der Methode *bindViews*() gebunden.

Klasse *PortrayableRVAdapter:*

- Wie wird der *PortrayableRVAdapter* in diesen Klassen verwendet? – der Adapter wird mittels der Methode createAdapter() mit dem snetprechenden Fragmenten als Host initialisiert und zurückgegeben. Der Adapter wird auch in der Methode *afterUpdate()* verwendet, um Veränderungen in Daten zu signalisieren/markieren (mithilfe der Adapter-Methode *reselectOrder()*).