Application structure - Krogkollen

The project is divided into packages with one feature or purpose each (package by feature). We've used the pattern Model-View-Presenter (MVP) when developing this application.

Package structure:

se.chalmers.krogkollen
- .adapter
- .backend
- .detailed
- .help
- .list
- .map
- .pub
- .search
- .sort
- .utils

The major packages are backend, detailed, list, map, pub and utils.

- backend: The backend package contains classes that handles our backend connection. It contains a class named BackendHandler which is a singleton class that's used throughout the program. BackendHandler is initialized with a backend that implements the IBackend interface, which makes the application highly portable to different backend solutions. We have implemented a Parse.com server class and a mock server that was used for testing purposes without generating API requests to our Parse.com server.
- detailed: A package containing classes that handles the detailed view. The detailed view is where the user can find more useful information about a pub.
- list: This package contains all classes that handles the list view. This is where the pubs are shown in a list sorted in different ways, by selecting a pub the user can get to the already mentioned detailed view.
- map: In this package all classes handling the Google Map are collected. This view shows, other than the actual map, a marker for each pub and a marker for the users location on the map. It also handles getting the user location and moving it on the map as it moves in real time.
- pub: The pub package contains all classes to do with the pubs. That includes requesting information about all saved pubs on the server.
- utils: Includes some small utility classes that often are used in several other packages.

Model-View-Presenter

Due to the nature of the Android operating system which relies heavily on the Activity class the Model-View-Presenter (MVP) design pattern is a good choice to use. MVP separates logic, data and UI into different classes, much like the more common Model-View-Controller (MVC) pattern. The main difference between the two patterns is that in MVP the view is completely unaware of the model. The view contains all UI elements, the presenter contains the UI logic and the model contains the data. In MVP the view is completely unaware of the model and only communicates with the presenter. MVP allows all parts of the program to be mocked, which allows testing which otherwise would be hard or sometimes impossible when using the MVC pattern. The view can be easily mocked with a class that e.g. generates artificial clicks and generates output to a console.
Our implementation of MVP isn't completely perfect and we've omitted some parts of the pattern to cut coding times. The code can however be easily refactored to fit the pattern when the time comes to extend the application.

Assets

All assets are saved in the /res/ folder of the project root directory.
Practically all layouts are written in XML while some parts are programmatically generated, for example, the pub markers. The pub markers are programmatically generated because they are supposed to be able to change if the pub they represent changes.

String, booleans, styles and dimensions are saved in XML-files where deemed appropriate.

Code standards

- All instance variables should be put at the very beginning of every class.
- All public classes and methods should have javadoc, public variables only need javadoc if they are very specific and hard to understand by themselves.
- All private and package private methods should have comments explaining the purpose of the method.
- Interfaces should be named with a capital "I" in the beginning, abstract classes with a capital "A" and classes using specific patterns for example the factory pattern should have the pattern name in the class name.
- Package by feature.
- Advanced algorithms and long segments should be well commented as well as all other code that is even slightly hard to understand.
- Strings and constant definition values should be put in a xml file.