MusicPhone Solution layout

app – Package containing the UI resources and classes that provide data from Last.FM.

commons – Package containing the data models, interfaces and logic modules used by the MusicPhone UI and tests. *You will implement the missing logic inside this project.* You may not change the interfaces or the `DeviceManager`. You may extend the behaviour of the data classes, but without changing their existing behaviour or their interfaces. You will need to add one or more classes here.

commons.dataClasses – Package containing the basic classes representing the entities present in MusicPhone

commons.interfaces – Package containing the interfaces from that are implemented by the different parts of this system. You should not change any of those. In particular, the methods exposed by IConnector and their implementation will be useful to undertake Task D.

commons.xmlData – Package containing the dump of Last.FM response for the API necessary to solve the task in XML format. Those are useful for testing purposes.

commons.dataConnectors – Package containing the implementation of IConnector (LastFmXmlConnector) methods to parse the Last.FM XML files. You may want to instantiate such class for testing purposes.

gps, **player** and **recommender** – Packages containing the UIs for the application main components. The UI is bounded to a class (e.g. GpsUI.java and Gps.java) which is the implementation of the interfaces present in commons.interfaces. The UIs button events handlers (e.g. click) need to be developed in order to bound each button the its functionality.

Hints:
http://en.wikipedia.org/wiki/Observer_pattern
http://docs.oracle.com/javase/7/docs/api/java/util/Observer.html
http://docs.oracle.com/javase/7/docs/api/java/util/Observable.html

http://docs.oracle.com/javase/tutorial/uiswing/events/actionlistener.html
http://docs.oracle.com/javase/tutorial/uiswing/events/intro.html

http://en.wikipedia.org/wiki/Singleton_pattern

What you should know before you start

Initialization of components

The application project creates concrete instances of `IPlayer`, `IGps`, and `IRecommender` objects. These instances persist when the application is running. application will set the `Connector` property of

the `Recommender` object to an instance of `LastFmXmlConnector,` which implements the `IConnector` interface.

The `IConnector` interface

Defines access to XML data from Last.FM. The `IConnector` class you need to test your implementation with preloaded XML data is `LastFmXmlConnector.` This class has a 0-argument constructor To access the XML data from **Commons**, use the `Connector` property of the `Recommender class`.

The XML data for testing

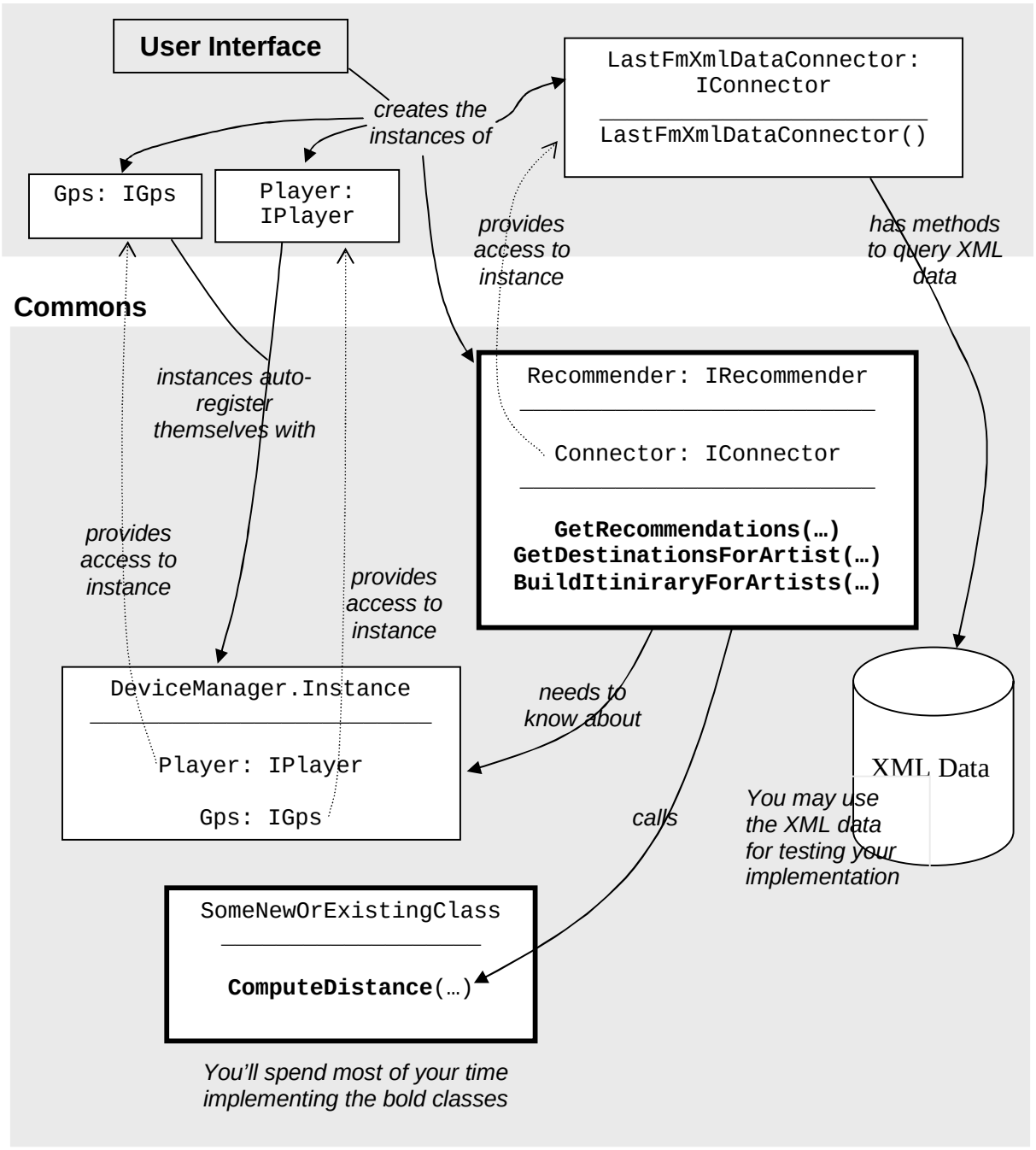
Located in the **XmlData** folder of the **Commons** project. The `LastFmXmlConnector` class accesses the files in this folder.

`DeviceManager.Instance`

Provides singleton access to instances of the `IPlayer` and `IGps` objects. When these objects are instantiated by the **Application**, they register themselves with the `DeviceManager`.

Architecture

Review the block digram on the next page.



UnitTests

- *You may instantiate and test any of the classes defined in* **Commons** *here.*
- *You may also instantiate the concrete classes* `Player`, `Gps`, *and* `LastFmXmlConnector` *defined in* **application**.