I consider MVC to be easy to understand, well-known, and quite simple in implementing. However, in this project I did not manage to implement it fully due to reasons which will be explained later.

3.3 Design Patterns Used

After choosing the overall architecture of the application it was also necessary to assess which, if any, design patterns would be useful in the music player. The book I used for my research regarding the design patterns was *Java Design Patterns. A Tutorial.* [11], naturally, as it was recommended to me multiple times, both during my 2nd year of studies and during my industrial placement.
Even though the book provided description of numerous patterns, the only one I was immediately able to decide upon being used in my project was the singleton pattern. One instance of a singleton would be used for storing data of a list of currently selected songs. I imagined the user interface would be handled by the Activity classes which would, for example, obtain the data about the songs from the singleton, which would in turn obtain it from a FLAC decoder. The singleton's main feature utilised here would simply be the fact that it ensures only one instance of a specified class exists. Thus, I would be certain that when accessing the data in the singleton class, it would be the same data I set there with my decoder class.

3.4 Design – Detailed Description

3.4.1 Significant Data Structures

The data structure I used most widely in this project was Array List. It prove itself as an easy to use, fast and reliable Collection. I used it mostly in such objects as ArrayList<String> for a list of names of Files in a folder – in the Activity which gives the user the possibility to choose a song by directory. Another example of its use could be ArrayList<File> which was used both to hold the Files which had the "flac" extension in the abovementioned Activity and to hold the list of files currently played. Finally, ArrayList<Song> was also used to handle data of all songs after the SD card was scanned. Song is my own class which represents a FLAC track, with its File and all the important meta-data extracted from the file.

3.4.2 Use Case Diagram

The use case diagram for my design, seen in Figure 7, did not actually change since I wrote my progress report. The list of actions the user can perform is exactly the same. All of them are shown on the use case diagram below.

Every action is a concrete function, which means each one symbolises a direct connection between the user and a feature. What is more, the "Select" functionalities are available from the application's starting (main) screen, the "Playlist" ones, along with the "Sort songs" functionality, will be located in a menu which will be invoked by pressing a device's "Menu" button. Finally, the "Song" actions will be available to the users from the screen where the currently played song will be shown.