Refactoring reccomendations:

As this is one of my first times working with refactoring code, I found the task a little intimidating: however I noticed a small inefficiency problem when constructing objects in the data model such as this segment from the file playlist.cpp:

```
Recording::Recording(const string & aTitle,
            const string & anArtist,
            const string & aProducer,
            const string & aYear,
            const int anID){
    cout << "Recording(string&, string&, String&, String&, int)" << endl;
    title = aTitle;
    artist = anArtist;
    producer = aProducer;
    year = aYear;
    id = anID;
    tracks = vector<Track*>(MAX_NUMBER_OF_TRACKS, NULL);
}
```

as it's written each of the members will have their values written twice: once to initialise the values to zero or null strings, and a second time to the constructed values. The solution? Base-member initialisation.

To improve runtime, each of the objects in the memory model (tracks, users, playlists, songs and recordings) now have their members initialised once through base-member initialisation rather than to be constructed and set again as in the current model.

So while a constructor may have looked like this before:

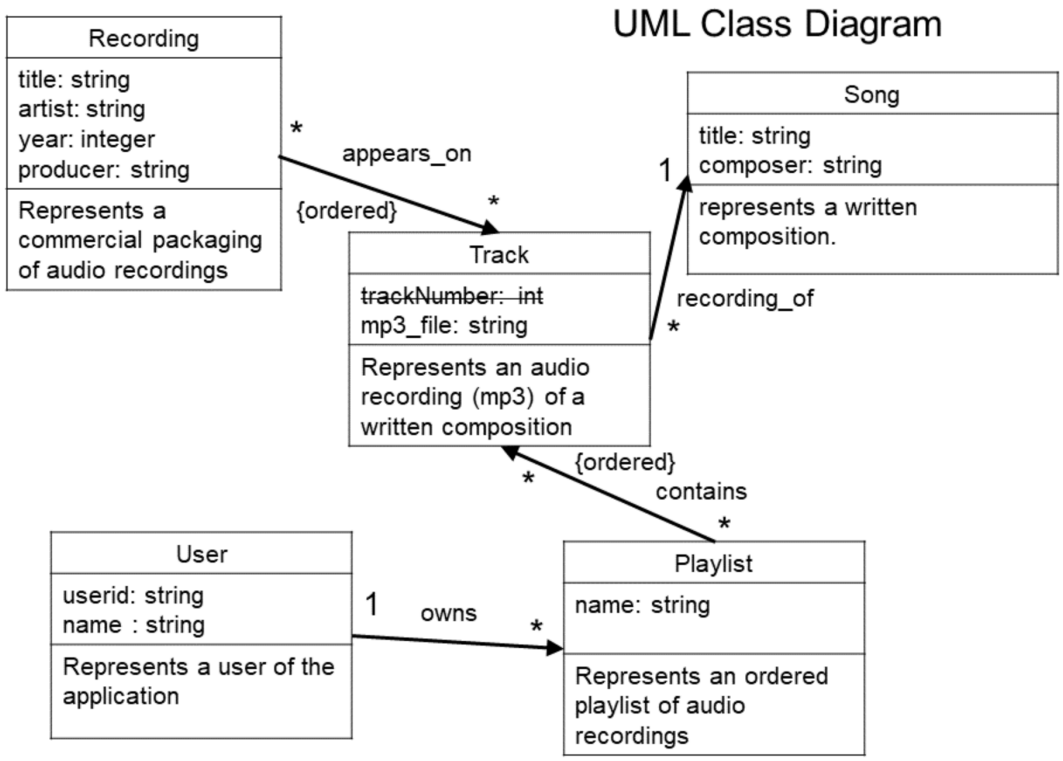
```
Class (attribute a, attribute b, attribute c) {
     //members written once to default (0) values
     A=a;
     B=b;
     C=c;
     //members written again
}
```

runtime is reduced by changing code to:

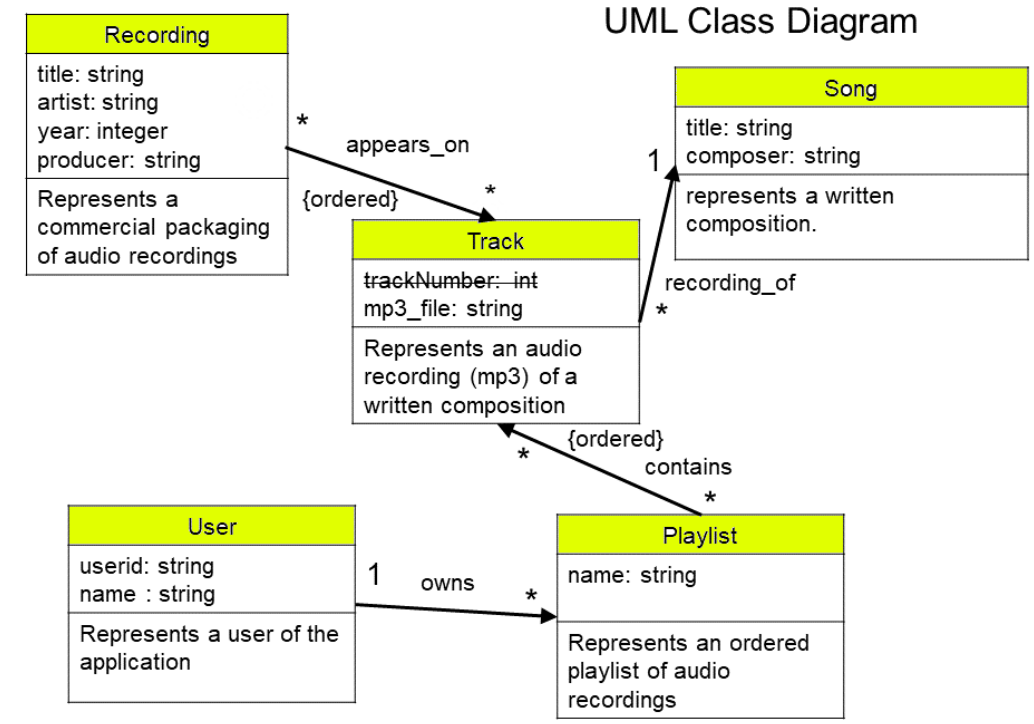
```
Class (attribute a, attribute b, attribute c): A{a}, B{b}, C{c}
{} //members written a single time to chosen values
```

As such the UML class diagram will look more or less the same:

Before:



After: (highlight = changed code)



The included text file which populates the Beatles database should be perfect for regression testing as it already adds various members of various kind to the data structure