Institute of Technology Blanchardstown
Institiúid Teicneolaíochta Baile Bhlainséir

Assignment Report
Object Oriented Design Patterns

Joseph Tierney	B00092923	Lecturer: Orla McMahon
		COMP H3012
		Object Oriented Design Patterns

Table of Contents

Abstract

This paper describes the technologies and approach that can be used to create and maintain a music application using the abstract factory method pattern. The paper will discuss the design of the application, going into detail about why certain styles and layouts were used. It will also go into detail about how the application works, with the major focus being on how the design pattern that was chosen was applied to this application. The paper will also illustrate the use of the design pattern with the aid of a UML diagram, and, finally, will describe the how the user could extend the application by adding a fifth or a sixth band, and the possibility of adding different requirements, such as upcoming events, new single releases or a new album release.

Design of the Application

The design of the application is a rather simple one. It makes use of JFrames in Java. This application includes features such as JButtons, JLabels, JTextFields, JMenuBars and Icon images. The simplistic nature of this design was chosen specifically with the user in mind, as it makes the application easy to navigate and understand what it is doing.

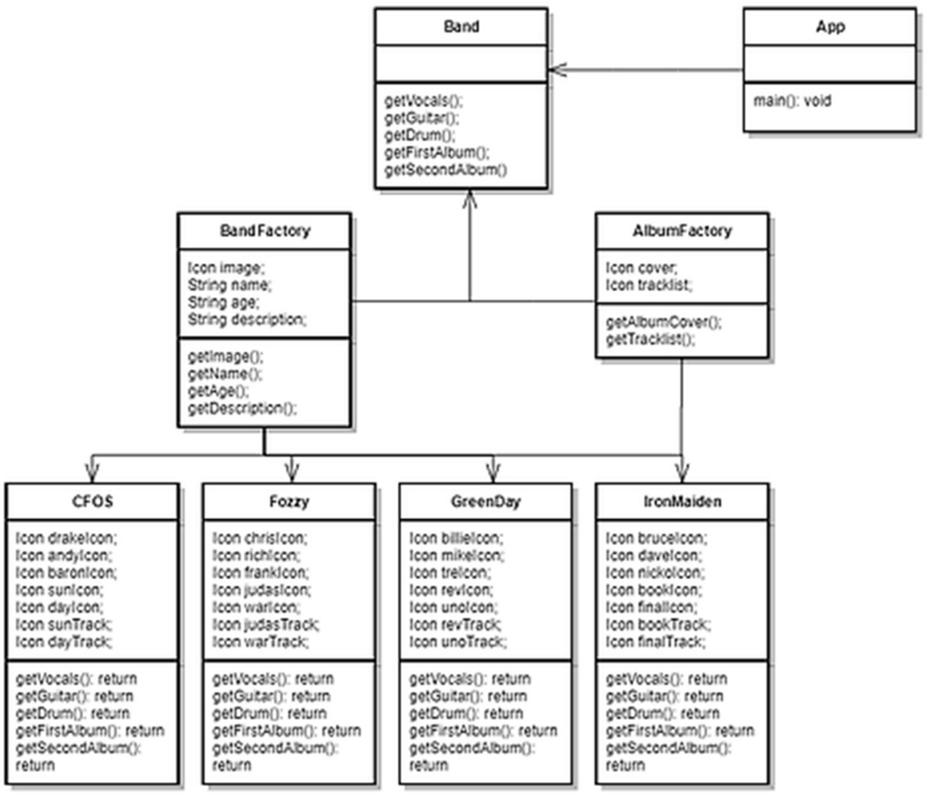
To start off, when running the application, the user will be greeted to a simple looking Graphical User Interface (GUI). At the top of the GUI, they will find two menu items; “File” and “Sample Tracks”. When clicking “File”, the user will see a sub-menu titled “About”. Clicking this will allow the user to see who made the application. Clicking “Sample Tracks” will show four sub-menus, each titled after the four sample segments of a song from each of the bands included in this application. Clicking on any one of these sub-menus will play a 30-40 second sample of the band’s latest hit single.

Following on from this, on the main panel of the application the user will see a label titled “Bands”, with four radio buttons underneath, each with the bands name next to it. To the right of that, there are three labels, titled “Name:”, “Age:”, and “Bio:”, respectively, with blank text fields next to them. Underneath this is a blank space reserved for each of the band members images, album covers and track lists. At the bottom of the application is six buttons that the user can click on. These six are “Vocalist”, “Guitarist”, “Drummer”, “Album One”, “Album Two” and “Quit”. Clicking on either “Vocalist”, “Guitarist” or “Drummer”, will display that band members information in the blank text fields. Clicking “Album One” and “Album Two” will display a picture of the album with a track list next to the image. Finally, clicking “Quit” will close the application.

The text fields in the application have been disabled, this was a design choice made early into production of the application, as it prevents the user from editing a band members information. Also, the application is not able to be resized or maximized to full screen, this design choice was chosen because the application was not intended for full screen use and keeps everything clear and concise for the user to see.

Abstract Factory Method

The abstract factory method is used to return one of several related classes of objects, each of which can return several different objects on request. For this application, I felt the abstract factory method was the best choice, because with the bands, we are asking the same questions, such as who is the vocalist, who is the guitarist, who is the drummer, what albums does this band have, and what is the track list of this album? The abstract factory class created is the Band, which holds the get methods that will return these results. BandFactory and AlbumFactory, which are not factories, simply contain and return the image, name, age, description, cover and track list. The bands themselves, CFOS, Fozzy, Green Day and Iron Maiden, are concreate factories – meaning they implement the methods used in the original abstract class. App is the main class that creates the Graphical User Interface and displays all the information from the factories.



Extending the Application

If the user wishes to extend the application, be it with a new band, a new album or an upcoming event, it’s a very simple process. If the user is adding an upcoming event, they should could either create an EventFactory or go to the Band.java file and add a public abstract method for one of the existing factory. If they chose to do the former, they should create a new file called EventFactory, declare the variables required for the event, and then create the constructor with the required variables. Then they should add their get methods for their variables. Once this is done, they should add an abstract method to the Band class called public abstract EventFactory getUpcomingEvent();. Finally, in the App.java class, add the necessary labels, text fields and buttons that will allow them to view upcoming events. In the app class they should create a setEvent method, calling the abstract factory, for example, eventName.setText(band.getUpcomingEvent.getName());.

Adding a new band is a simpler task, as it just requires using the previously existing bands as a backbone to what is needed.