

**ASSIGNMENT COVER SHEET**

**Student’s Name: Ken Kilmartin & Catharine Phoenix**

**ID Number: B00091252 & B00091705**

**Course: Bachelor of Science (Honours) in Computing**

**Year: 3rd Year**

**Lecturer: Orla McMahon**

**Title of Assignment: Assignment 2**

**Due Date: 9th April**

**Date Submitted: 9th April**

The material contained in this assignment is the authors original work, except where work quoted is duly acknowledged in the text. No aspect of this assignment has been previously submitted for assessment in any other unit or course.

Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_/\_\_\_\_\_\_/\_\_\_\_\_\_

Signed: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Contents**

[**Introduction** 1](#_Toc510789656)

[**Methodology** 2](#_Toc510789657)

[**UML Diagrams** 3](#_Toc510789658)

[**Complete UML** 7](file:///C:\Users\Nutzer\Desktop\oodp\oodpAssignment2\Report%20for%20assingment2.docx#_Toc510789659)

[**Bibliography** 8](#_Toc510789660)

# **Introduction**

The Four design patterns we decided to implement within our project are as follows:

* The Singleton design pattern.
* The Abstract Factory design pattern.
* The Adapter design pattern.
* The Command design pattern.

We made our choice based on the design patterns we had thought would suit our purpose best for this assignment. We had also considered the Builder design pattern instead of the Command pattern due to the command creating a few more classes but we had both wanted to look at the Command in more detail. The group also decided to make use of GitHub for this assignment due to the time period we had to complete this assignment was over the Easter break. Our repository can be seen at:

<https://github.com/KenKilmartin/oodpAssignment2>

# **Methodology**

We had decided to build a program for a library to use. We choose to build a library program as we had thought the Abstract factory and Singleton design patterns would be a good fit to try use these design patterns. We then had thought if the library was donated a bunch of comics we could try making an adapter for the comics to fit in.

We used the Abstract factory to encapsulate a group of individual factories such as books, CDs and comic books. We choose these as they all share a common theme. We choose to do this design pattern as we both felt that in the first assignment we had it mixed up a little. This time we made our abstract factory and then had each factory look’s after a single class compared to the first assignment we had all classes in the one factory.

Each of the comics then uses an adapter design pattern to make it fit into the library program. The thought behind this was a comic book shop donated a bunch of old comics to the library and gave the database so then the library had to make adapters for each of the comics to make it fit into their program. We can see that for example on the Garfield comic it does not implement anything. An adapter was made to make it fit around Garfield called Garfield adapter and then it could go into the library system.

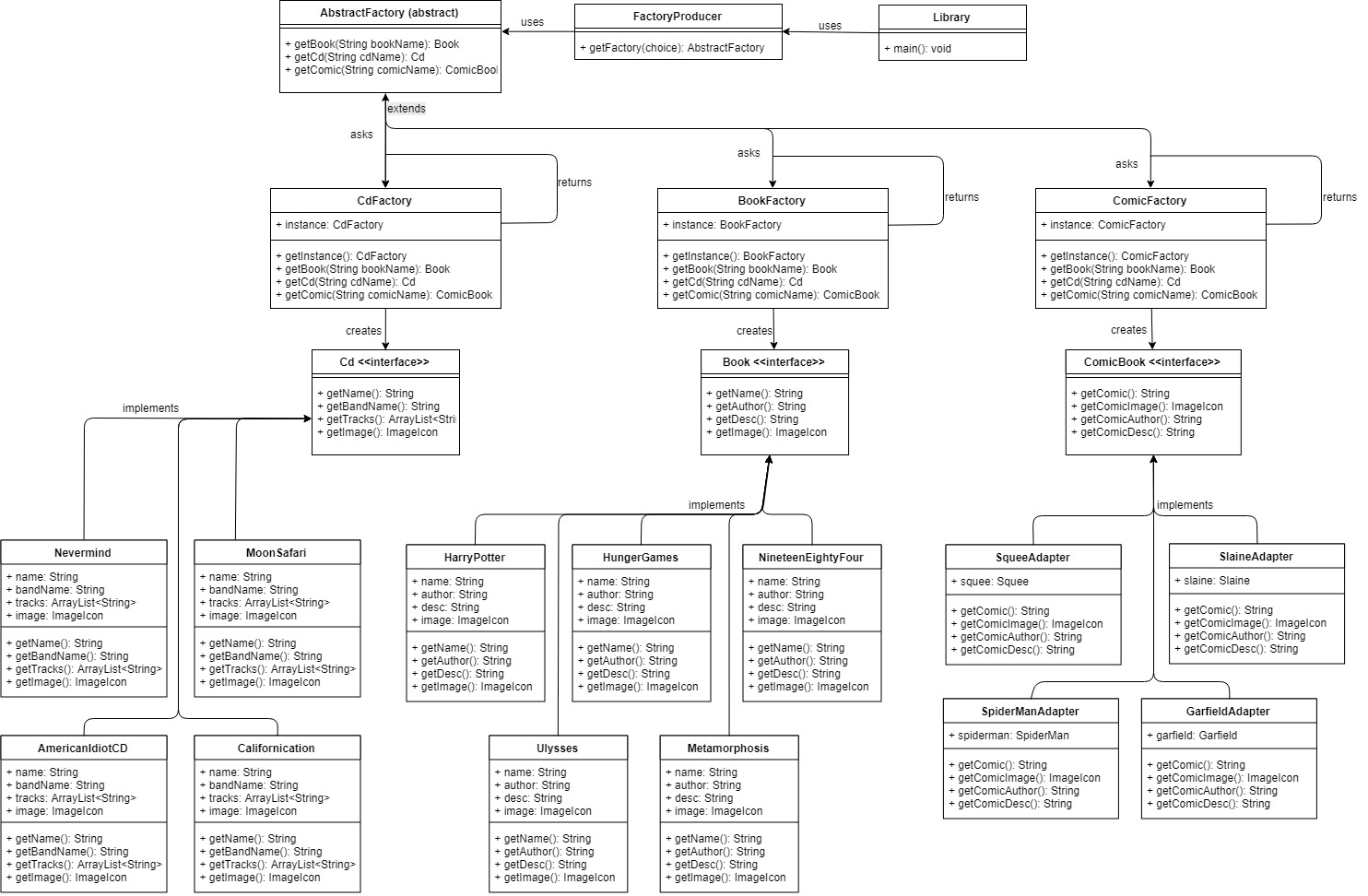
We had originally thought about not trying to use the Command design pattern as we had thought that it would just add too many classes to the project. However, we decided to apply this pattern to the main interface. We use this pattern in the library when you are deciding on which product you want to rent be it Book, CD, Comic or to Logout.  
  
The Singleton design pattern was used on each of the factories (Book, CD, Comic). We did this first by making it a private static object ensuring that there was a single object of the factory being made, then we made a constructor private so that this class cannot be instantiated and then we get the only object that is available.

It was a challenging to try think of a project to try fit the design patterns into but as soon as we had figured out what we would fit them into it was easier to program it. We both feel in real world situations it may be easier to apply patterns into problems.

# **UML Diagrams**

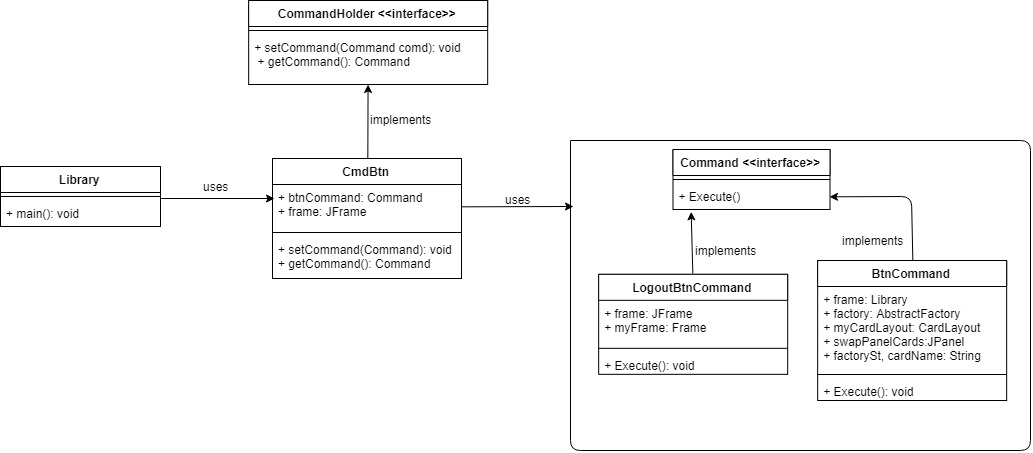
**Abstract Factory Pattern**

The Abstract Factory pattern has the “FactoryProducer” which in turn creates the other factories – CdFactory, BookFactory and ComicFactory. We used this pattern as it is understood to be one of the best ways to create an object. The interfaces are responsible for creating related objects. Each generated factory can give the objects as per the Factory pattern. (TutorialsPoint - Abstract Factory Pattern, 2018)



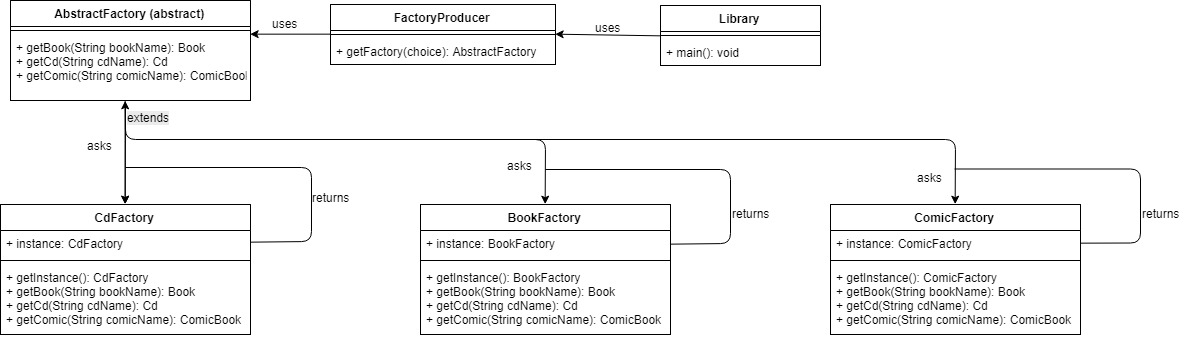
**Command Pattern**

In the command pattern we have the CommandHolder Interface which specifies just two methods – setCommand and getCommand. The CmdBtn (Command Button) class implements that interface. It uses the Command Interface, which specifies the method – execute. We have two types of commands in our Library, one command is a button to logout and the other command is a button command that will open one of three GUI interfaces depending on the parameters passed. Bothe these commands implement the Command interface, so they need to implement the method “execute”.



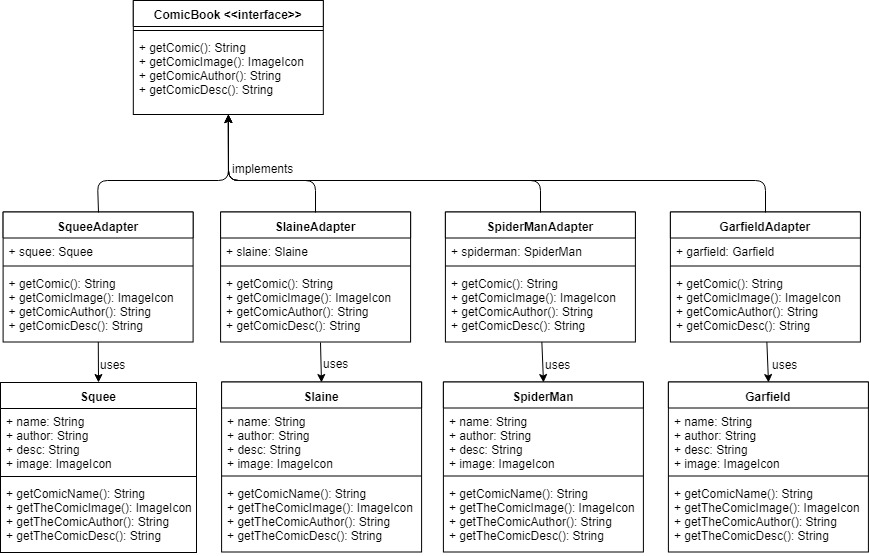
**Singleton Pattern**

The CarFactory, BookFactory and ComicFactory all use the singleton pattern. This means that only one instance of these classes can be made and it can be accessed directly without the need to instantiate the object.



**Adapter Pattern**

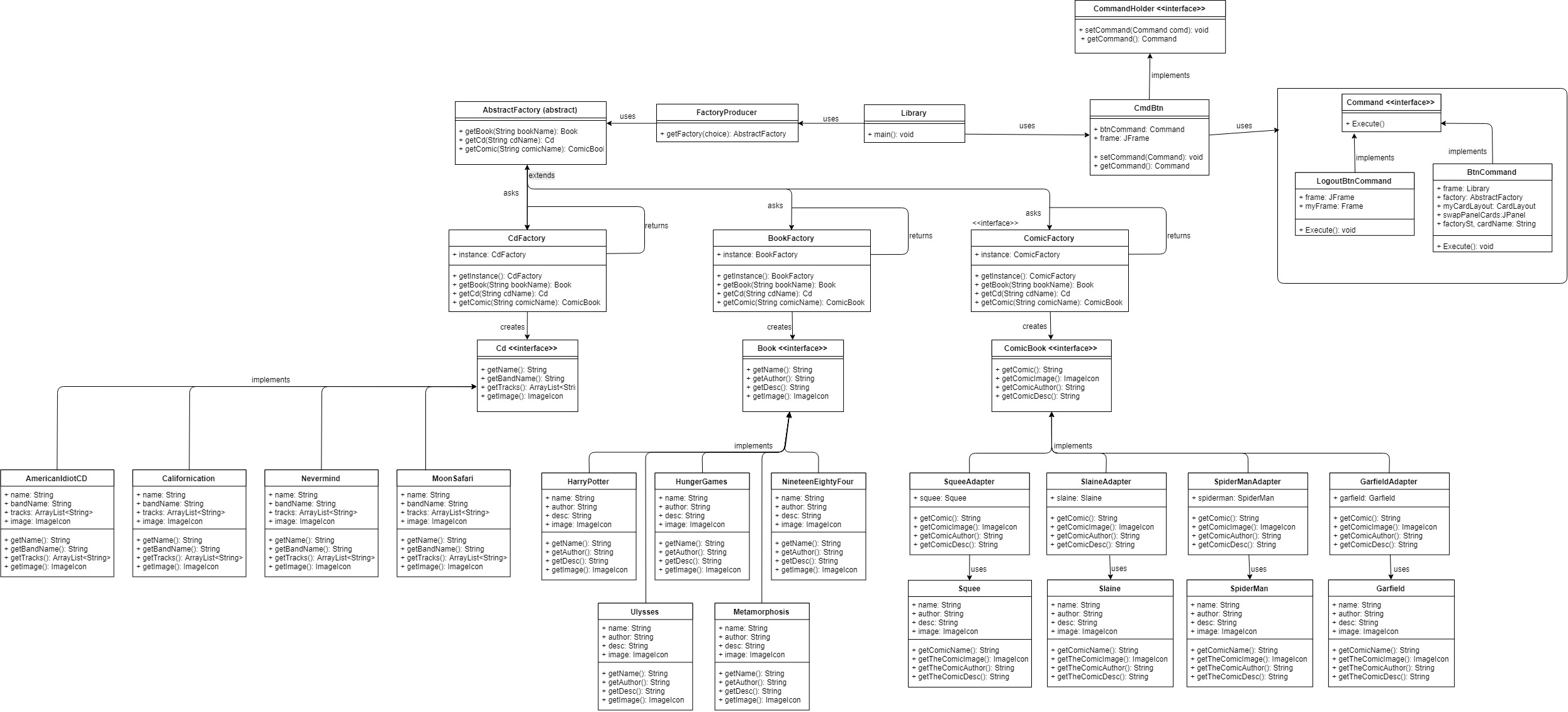
We decided to use the adapter pattern to practice building a bridge between two incompatible interfaces. As mentioned before, the story behind tis pattern is that an old comic book store donates all its comics to the library along with their own database. We made an adapter for each of the comics to be able to implement the ComicBook interface but also can access the information from the donated comic books. Each adapted creates an instance of the comic and accesses the information through the instance.



**Division of Work**

We both worked on the concept of the program together, and on deciding what design patterns would fit in to the software. Before leaving for Easter break, we worked together on a small scale of all the patterns within a basic “Library” program. We did this so that we knew that we were both on the same page knowing where the program was headed. From there Catharine took the lead on the Abstract Factory and the Command patterns and Ken took lead on the Singleton and Adapter patterns.

We are both happy with the division of work and feel that neither of us let the other down.



# **Complete UML**

# **Bibliography**

*Design Pattern - Abstract Factory Pattern*. (2018). Retrieved 3 12, 2018, from Source Making: https://sourcemaking.com/design\_patterns/abstract\_factory

*Design Pattern - Adapter Pattern*. (2018). Retrieved 3 12, 2018, from Source Making: https://sourcemaking.com/design\_patterns/adapter

*Design Pattern - Command Patern*. (2018). Retrieved 3 12, 2018, from Source Making: https://sourcemaking.com/design\_patterns/command

*Design Pattern - Singleton Pattern*. (2018). Retrieved 3 12, 2018, from Source Making: https://sourcemaking.com/design\_patterns/singleton