**ASSIGNMENT 2 FRONT SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **BTEC Level 5 HND Diploma in Computing** | | |
| **Unit number and title** | Unit 20: Advanced Programming | | |
| **Submission date** | 4/7/2022 | **Date Received 1st submission** | 4/7/2022 |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** | Nguyen Trong Dat | **Student ID** | GCH200102 |
| **Class** | GCH0906 | **Assessor name** | Doan Trung Tung |
| **Student declaration**  I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice. | | | |
|  |  | **Student’s signature** | Diagram  Description automatically generated |

**Grading grid**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P3 | P4 | M3 | M4 | D3 | D4 |
|  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **❒ Summative Feedback: ❒ Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Lecturer Signature:** | | |

Table of Contents

[Table of Figures: 3](#_Toc107861968)

[1. Introduction: 4](#_Toc107861969)

[2. Scenario analysis: 4](#_Toc107861970)

[2.1 Scenario: 4](#_Toc107861971)

[2.2 Diagram: 5](#_Toc107861972)

[3. Implementation: 6](#_Toc107861973)

[3.1 Code: 6](#_Toc107861974)

[3.2 Program screenshots: 14](#_Toc107861975)

[4. Discussion: 16](#_Toc107861976)

[4.1 Range of similar patterns: 16](#_Toc107861977)

[4.2 Usage of pattern: 16](#_Toc107861978)

[5. Conclusion: 17](#_Toc107861979)

[References: 17](#_Toc107861980)

# Table of Figures:

[Figure 1: Abstract factory design pattern 5](#_Toc107861942)

[Figure 2: IT abstract class 6](#_Toc107861943)

[Figure 3: setter 7](#_Toc107861944)

[Figure 4: Senior IT 8](#_Toc107861945)

[Figure 5: Fresher IT 8](#_Toc107861946)

[Figure 6: Project Factory 9](#_Toc107861947)

[Figure 7: Fresher Project Factory 10](#_Toc107861948)

[Figure 8: Senior Project Factory 11](#_Toc107861949)

[Figure 9: Main program **Error! Bookmark not defined.**](#_Toc107861950)

[Figure 10: Enter Choice function 12](#_Toc107861951)

[Figure 11: Add employee to Project function 12](#_Toc107861952)

[Figure 12: Enter numbers of employees 13](#_Toc107861953)

[Figure 13: Print employee in project2 14](#_Toc107861954)

[Figure 14: Choose project type 14](#_Toc107861955)

[Figure 15: Enter wrong format 14](#_Toc107861956)

[Figure 16: Input numbers of employees 15](#_Toc107861957)

[Figure 17: Show list of employees in project 15](#_Toc107861958)

# Introduction:

In this assignment, I will introduce about abstract factory and implementation. I will provide the class diagram and the program of scenario. Finally, I will discuss about the similar design pattern and use of design pattern.

# Scenario analysis:

## 2.1 Scenario:

For the project management, customer can request project for senior or project for fresher employees. The company has 2 departments which are IT and Design. The employee has 4 basic information which are id, name, age, role and 1 characteristic to separate the type of employee which is experience. Employee with more than 5 years experiences will become senior employee. Each employee can view their information. Because there are 2 types of projects for employees, project for senior requires the senior employees with more than 5 years experiences and fresher require fresher employees with less than 5 years. The manager can choose the type of project and add suitable employee into project.

## 2.2 Diagram:

Graphical user interface, diagram

Description automatically generated

Figure : Abstract factory design pattern

The same as the structure of abstract factory I have analysis in assignment 1. In this scenario, the client will only use the Project Factory which are Fresher Project Factory and Senior Project Factory. User can choose the type of project and the program will initialize the employee suitable for project. For instance, the fresher project will add fresher IT and fresher design when use the method addIT and AddDesign. Clients do not need to worry about the process of choosing the suitable employee into the project. In the abstract class IT, the IT employee has 4 basic information: id, name, age, and their role. To separate the senior and fresher employee, there is 1 attribute is experiences, if employee has more than 5 years exp, they become a senior. Same as IT, design employee also has basic information and the separate of senior and fresher.

# Implementation:

## 3.1 Code:

Text

Description automatically generated

Figure : IT abstract class

In the IT class, there are 5 attributes which are id, name, age, experiences and role of employee. In addition, the id is auto increment when new object has been declared by the variable count. The Id and Role properties only have the method get which mean read only, user cannot modify the value. In addition, the abstract method Show Info will be implemented by the class which inherit from IT.

Text

Description automatically generated

Figure : setter

In the method set, i have handle the exceptions such as in property Name, if user enter a blank for name, there are a message that warning user enter invalid value. The same with the Exp and Age, the value can not be negative.

Text

Description automatically generated

Figure : Senior IT

The class Senior IT is inherited from abstract IT class. In the constructor, the Senior IT has changed 1 attribute which is the role become Senior IT to separate with Fresher. The function ShowInfo() has been implement and overrive the method in IT class. The function will print the information of IT employee which are id, name, age, exp and role.

Text

Description automatically generated

Figure : Fresher IT

The class Fresher IT also inherit from abstract class IT. The constructor of Fresher is diffirent with senior by the role Fresher IT.

Text

Description automatically generated

Figure : Project Factory

In the abstract class Project Factory, there are 2 abstract functions AddIT and AddDesign for inheritance class can implement. In additon, there are a function that help the user can enter the experience of employee, that program can filter the suitable employee.

Text

Description automatically generated

Figure : Fresher Project Factory

As we can see, the fresher project factory inherit from abstract class Project Factory. For the function AddIT overide from abstarct class, the program will request user to input the exp of employee. Because the fresher project only suitable with fresher employee, thus if user enter the exp greater than 5 years or invalid such as a negative numbers, the program will ask for input again. I have use try catch to handle if user input a character not a number to avoid crash program. The same with the age when the age less than 18 and more than 80, user need to enter agian. For the name of employee, if user do not enter anything, user needs to input again because the while loop. Finally, the function will return a Fresher IT with the parameters name, age and exp that user input. The same with function AddDesign return a Fresher Design.

Text

Description automatically generated

Figure : Senior Project Factory

The same with Fresher Project Factory, the senior project factory inherit from abstract class Project Factory. For the function AddIT overide from abstarct class, the program will request user to input the exp of employee. Because the senior project only suitable with senior employee, thus if user enter the exp lower than 5 years or invalid such as a big numbers, the program will ask for input again. I have use try catch to handle if user input a character not a number to avoid crash program. The same with the age when the age less than 18 and more than 80, user need to enter agian. For the name of employee, if user do not enter anything, user needs to input again because the while loop. Finally, the function will return a senior IT with the parameters name, age and exp that user input. The same with function AddDesign return a senior Design.

Text

Description automatically generated

Figure : Enter Choice function

In the function Enter choice, the program will print a menu for user to choose. I have use try catch to handle if user do not enter a number into program and this will make program crash. The function will return the int value which number 1 is for senior project and other is for fresher.

Text

Description automatically generated

Figure : Add employee to Project function

In this function, the parameter choice will decide which project will be chosen. If user input number, the Senior project factory will be declared. After that, the program will request user to input the numbers of IT employees and design will take part in this project. When finishing add the suitable employees to project, the infomation of employees will be stored in the list.

Text

Description automatically generated

Figure : Enter numbers of employees

As we can see, the functiosn enters numbers into project, I have use try catch to handle exceptions when user may enter wrong format such as enter a character. And when user input an invalid value such as number of employees lower than 0, user will be asked for input again. The enter numbers of designers function is the same with enter numbers its. Both two functions return the integer numbers.

Text

Description automatically generated

Figure : Print employee in project2

In the print function, I have use foreach to loop the list that store the data of employee in project. Each employee calls the function show info to see their information.

## 3.2 Program screenshots:

Text

Description automatically generated

Figure : Choose project type

First of all, user needs to choose 1 type of project they want to create. If user chooses number 1, the project for senior will be initialized.

Text

Description automatically generated

Figure : Enter wrong format

In case user input wrong format of number that lead program can be crack, the program will ask user to enter again with a warning message that user need to be enter number.

Text

Description automatically generated

Figure : Input numbers of employees

After choosing the suitable project, the suitable employee will be initialized. User needs to enter the numbers of It employees will join into project. If user choose senior project and enter the employee that has the exp lower than 5, the program will ask user to enter again. After entering the IT employees, the program will continue ask for the number of designers will be taken part in the project. Same with IT when user enter the exp, program will ask user to enter exp again.

Text

Description automatically generated

Figure : Show list of employees in project

When user finish enters the information of employees into project, the program will show the list employees which have full information of workers including both ID and role of employees.

# Discussion:

## Range of similar patterns:

* Some similar design patterns:

Builder and Abstract Factory are comparable in that both may create complicated objects. The main distinction is that the Builder pattern emphasizes building a complex product piece by piece. The focus of Abstract Factory is on families of product objects (either simple or complex). Builder completes the process by returning the item, but the Abstract Factory pattern returns the item right away (Gamma, Helm, Johnson, & Vlissides, 1994).

Abstract Factory is often implemented with factory methods. Factory Method is also demonstrated in the Abstract Factory pattern's Motivation example. Typically, factory methods are referred to as template methods. It's not necessary to subclass Creator for prototypes. They frequently call for an initialize action on the Product class, though. The object is initialized by Creator using Initialize. Such an action is not required by the Factory Method (Gamma, Helm, Johnson, & Vlissides, 1994).

* Explain why your pattern is the most suitable for your scenario:

The main players in the Abstract Factory pattern are factories and products. This technique illustrates how to build groups of similar product objects without directly instantiating classes. It works best when there are variations within particular product families while the quantity and broad types of product objects remain consistent. By instantiating a specific concrete factory and then constantly using it to produce things moving forward, we can choose amongst families. By substituting a different instance of the concrete factory for it, we can even switch out entire families of products. The Abstract Factory design stands out from other creational patterns because it prioritizes families of goods rather to just one type of product object (Gamma, Helm, Johnson, & Vlissides, 1994).

## 4.2 Usage of pattern:

Advantages of abstract factory:

* Concrete (implementation) classes are separated from client code via the abstract factory pattern.
* Exchange of object families is made simpler.
* It encourages object consistency.

Disadvantages:

* The complexity and writing the code for set up the factory during the initial stages.

# Conclusion:

After finishing 2 assignments, I have understood more and more about design patterns and its usage. Moreover, I can apply the design pattern into a small project especially abstract factory pattern. In the future, I will learn more and more about design patterns and learn how to apply them into project.

# References:

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.*