OOP and DP – Advanced Programming

Team 2:

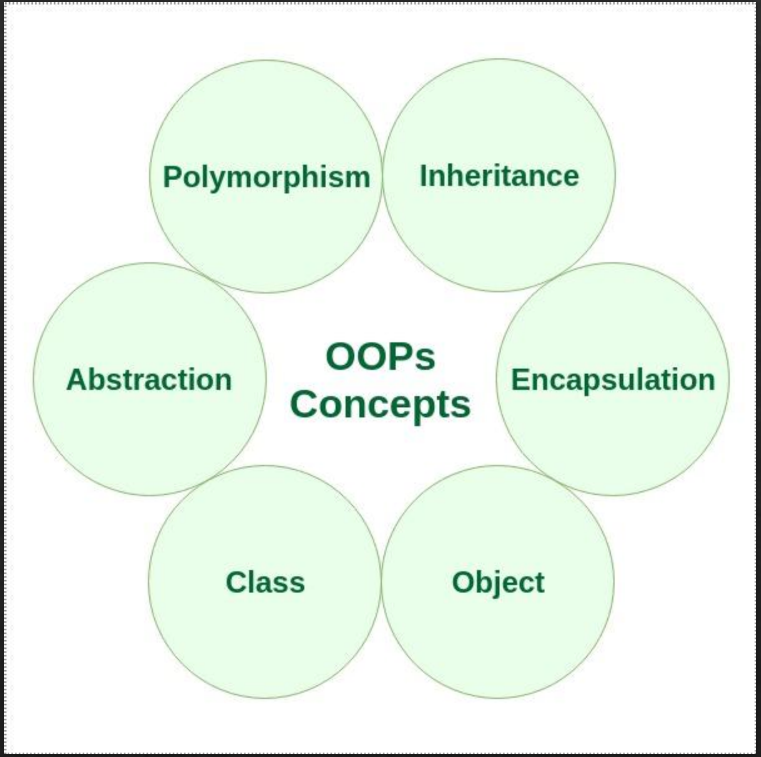
Le Do Thai Son
Do Van Truong
Tran Minh Khoi

Object-Oriented Programming (OOP)

Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

Characteristics of an Object-Oriented Programming language:

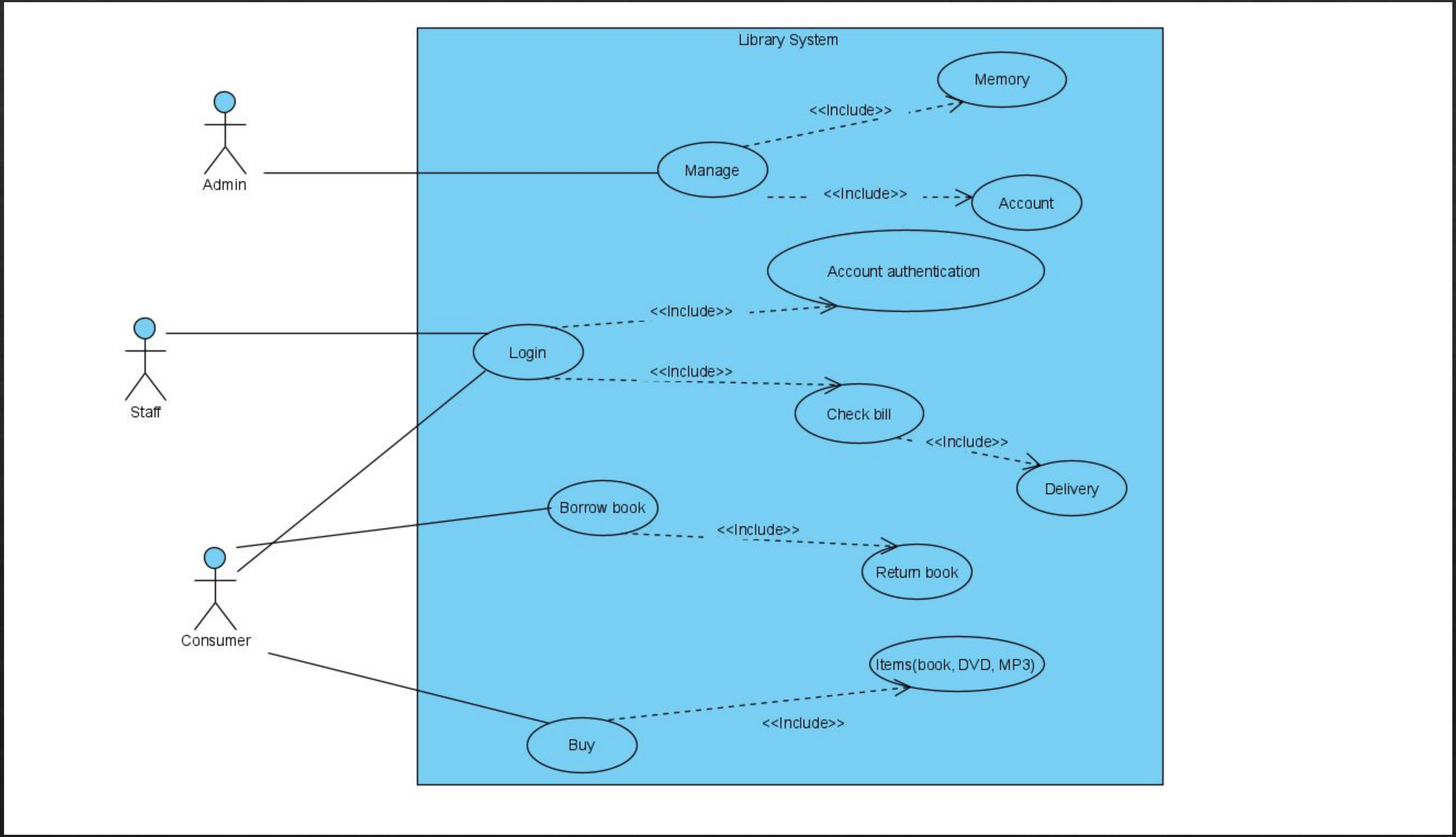
Polymorphism
Inheritance
Encapsulation
Abstraction



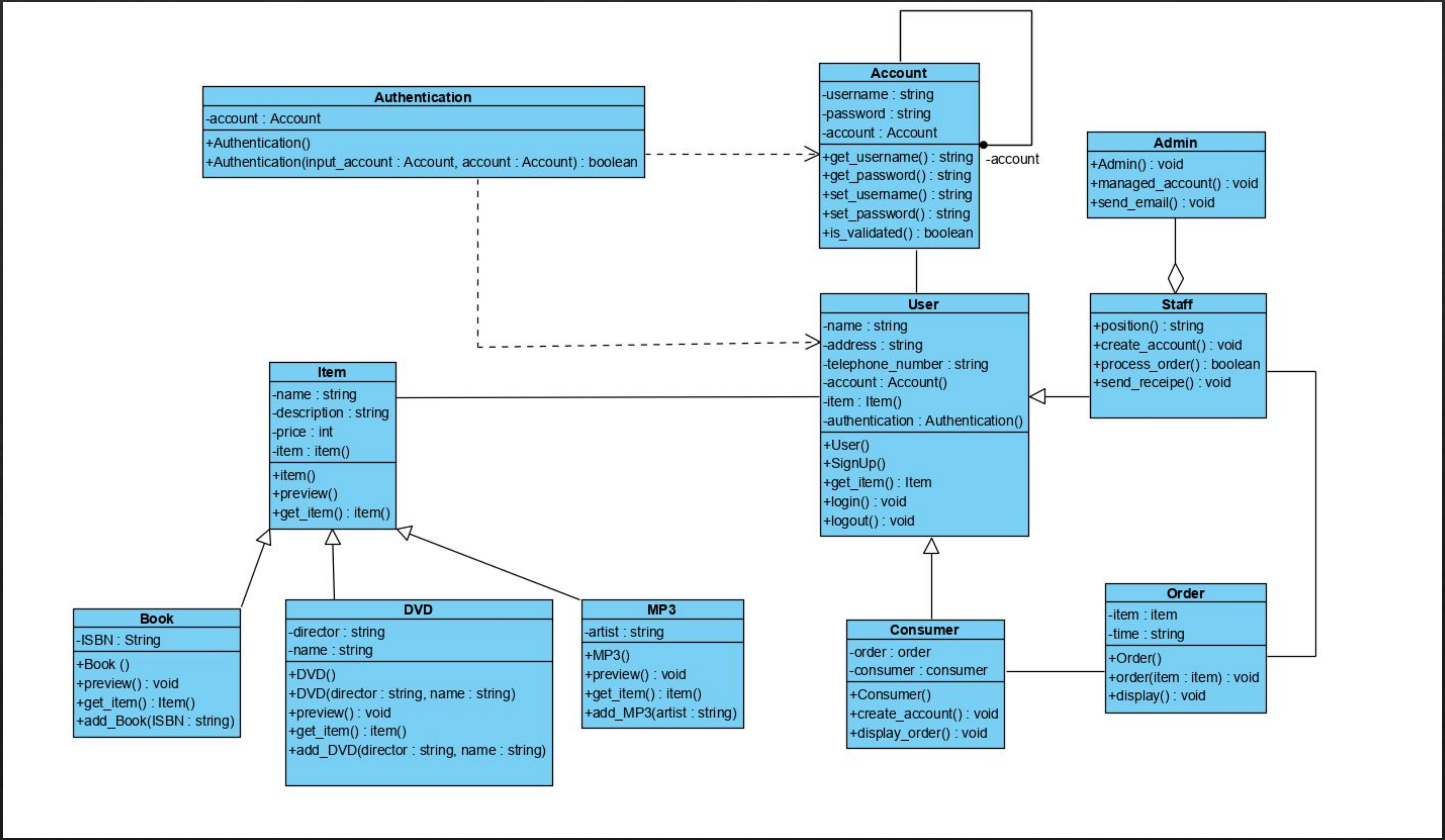
Scenario

- A profile system for keeping all the library files that takes in files for input. The user can search for the file. Files can be inputted into the library. The user can access the search engine's interface. The staff can do research on files, manage files, suggest file based on the file relevancy and quoted times. There will be a memory of all the files to record all the locations the files have been to. Memory can be managed by the admin but cannot be managed by the user but the data of memory cannot be deleted. File password service allows for access for only staff. User can borrow books in library and register an account for reading books in library.
- Besides, it sells products related to entertainment products such as, mp3, and DVD. Users can refer to the products of the store (see product information, see product overview). If you decide to buy DVD or download MP3 the user will need to register an account at the store with personal information (name, address, telephone number, username, password) then the user will order products that they like. The store will check orders and deliver goods to users in 3 – 5 days.

Use case diagram (Scenario)



Class diagram (Scenario)



Creational Patterns

- Creational design patterns are design patterns that deal with object creation mechanisms, trying to create objects in a manner suitable to the situation. The basic form of object creation could result in design problems or added complexity to the design. Creational design patterns solve this problem by somehow controlling this object creation.
- Characteristic of Creational Patterns:
  - Factory Method
  - Abstract Factory
  - Builder
  - Prototype
  - Singleton

Singleton Pattern

- The singleton pattern is a software design pattern that restricts the instantiation of a class to one "single" instance. This is useful when exactly one object is needed to coordinate actions across the system. The term comes from the mathematical concept of a singleton.

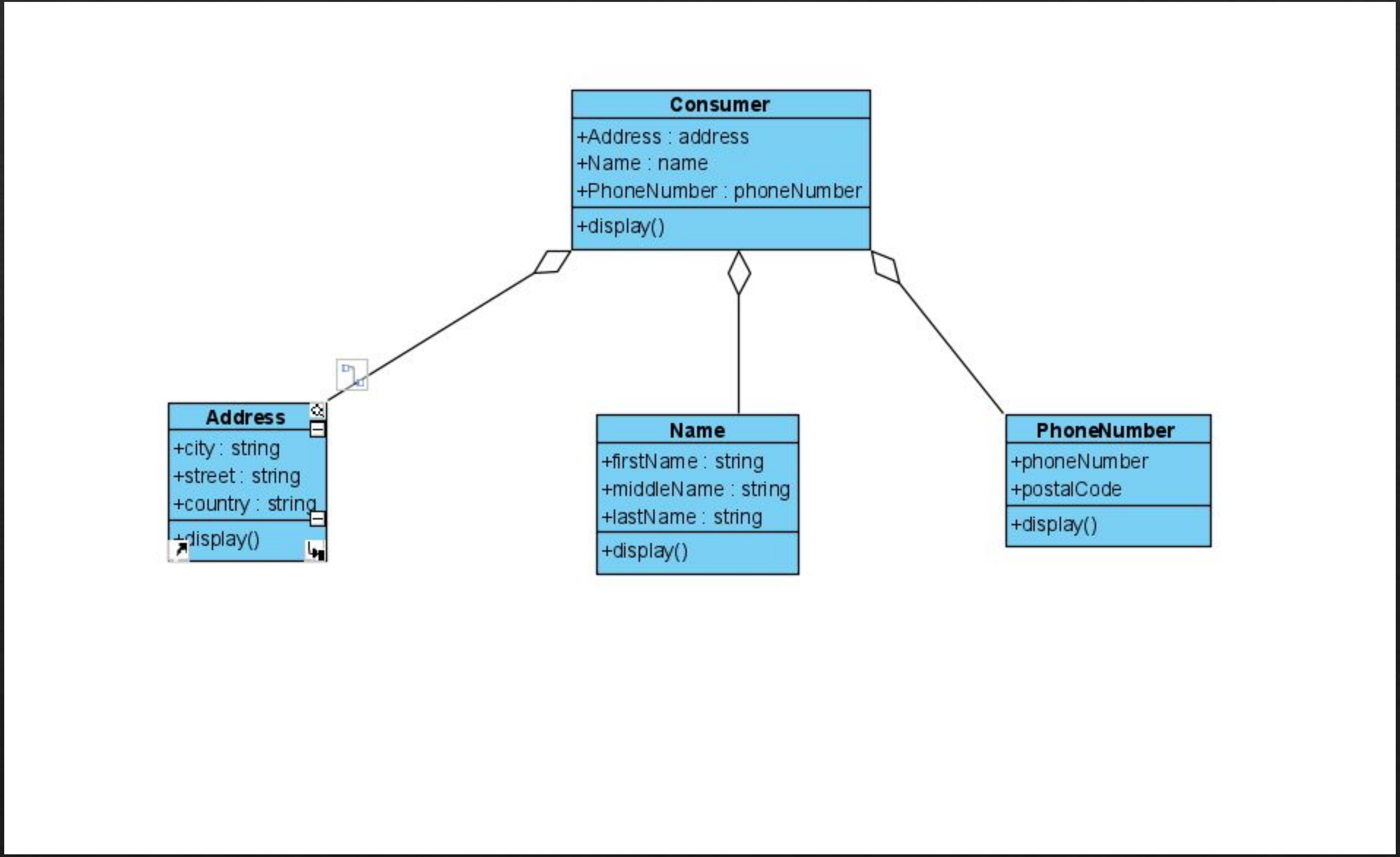
Structural Patterns

- Structural Patterns
  - Facade Pattern is to create a wrapper class for a subsystem (be it a library or a module consisting of many different classes), helping to simplify the interface of that subsystem. As a result, other modules can work with that subsystem more easily through the wrapper, the complexity of the subsystem itself will not affect the outside world.
  - Characteristic of Structural Patterns:
    - Adapter
    - Bridge
    - Composite
    - Decorator
    - Façade
    - Flyweight
    - Proxy

Façade Pattern

- Facade Pattern is to create a wrapper class for a subsystem (be it a library or a module consisting of many different classes), helping to simplify the interface of that subsystem. As a result, other modules can work with that subsystem more easily through the wrapper, the complexity of the subsystem itself will not affect the outside world.
- Scenario: A system used to store customer information after buying clothes at a store to contact customers for promotions. These stores include customer names, customer addresses, phone numbers to contact

Class diagram (Façade Pattern)



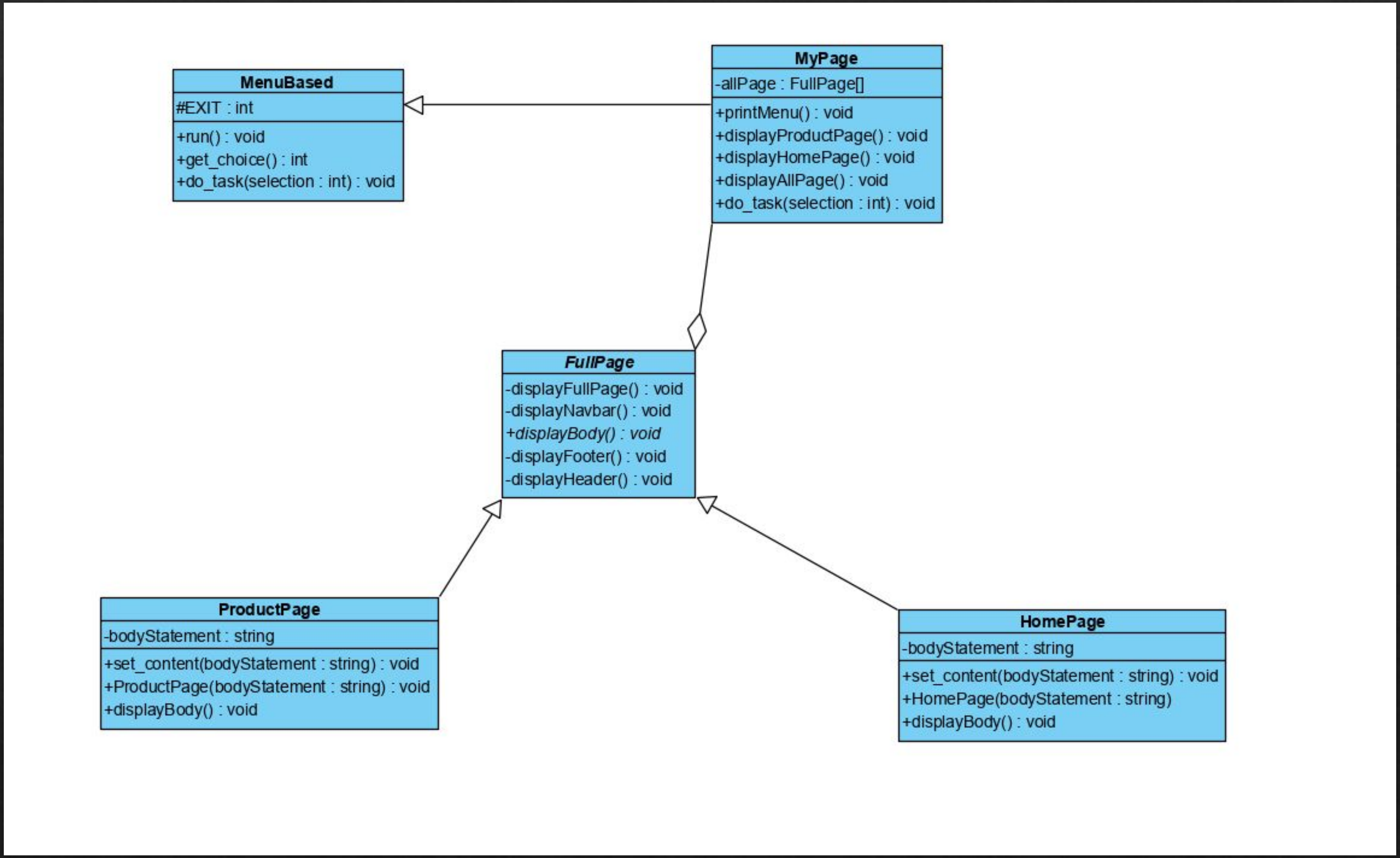
Behavioral Patterns

- Behavioral patterns are involving algorithm and the relation between objects. It describes not only the behavior between objects but also the behavior between the interaction between them
- Characteristic of Behavioral Patterns:
  - Chain of Responsibility
  - Command
  - Iterator
  - Mediator
  - Memento
  - Observer
  - State
  - Strategy
  - Template Method
  - Visitor

Template Method

- Determine the structure of an algorithm in an operation, inherit some steps to subclasses. Template method lets subclasses re-determine some steps of an algorithm but do not affect to the structure of algorithm. Using template method helps developers to avoid the duplication in coding.
- Scenario: The structure of an HTML page has a title, navigation bar, body, and footer. Usually a website always have the same title and footer for every site, but the body and possibly the navigation bar can change depends on the content and purpose of each page. Instead of headers and footers for each page, they should be placed in a template.

Class diagram (Template Method)



ASSIGNMENT 1 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title	Unit 20: Advanced Programming		
Submission date	23/06/2020	**Date Received 1st submission**	23/06/2020
Re-submission Date		**Date Received 2nd submission**	
Student Name	Đỗ Văn Trường	**Student ID**	
Class	GCH0714	**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	Truong

Grading grid

P1	P2	M1	M2	D1	D2

❒ Summative Feedback:		❒ Resubmission Feedback:
Grade:	Assessor Signature:	Date:
Lecturer Signature:		

Table of Contents

I. Introduction

This report will explain the characteristic of the Object-oriented programming paradigm by applying Object-oriented analysis and design on a given (assumed) scenario. The scenario will be able to present various characteristic of OOP. Then some design patterns will be provided with real case scenarios, corresponding patterns illustrated by UML class diagrams.

II. OOP – Object Oriented Programming

II.1. Definition

Object Oriented Programming (OOP) is a programming model based on the concept of "object technology".

An object consists of:

- The data called attribute.
- Source code is organized into methods (Method).

Methods that allow the object to access and edit the data fields of another object, which the current object has interacted with (objects that support the "this" or "self" methods).

In object-oriented programming, computer programs are designed by separating it from the range of objects interacting with each other. Object-oriented programming languages are quite diverse, mostly class – based programming languages, meaning that objects in these languages are treated as entities of a class, used to define a class datatypes (Clark, 2013).

Why we should use OOP: (Clark, 2013)

- Easily to maintain and expand and implement changes in the programs.
- Code can be reused in other programs to solve a different problem.
- The ability to create software systems mode effectively, better designs and fewer flaws using a team process because the OOP forces the designers to have a long and extensive design phase
- Better integration with loosely coupled distributed computing systems.
- Improved integration with modern operating systems.

II.2. Characteristic of OOP

Encapsulation: In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them. Example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial transactions and keeps records of all the data related to finance. Similarly, the sales section handles all the sales-related activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of the sales section and the employees that can manipulate them are wrapped under a single name "sales section". (Clark, 2013)

Inheritance: in OOP is the relationship between classes in which subclasses inherit all the properties and methods in the superclass. Using inheritance in OOP allows us to classify objects in the program according to their common characteristics and functions. But the subclasses only have access to and use the properties and methods of the superclass in the form of "public" and "protected". (Clark, 2013)

Abstraction: Data abstraction is one of the most essential and important features of object-oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. Example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car. This is what abstraction is. Abstraction using Classes: We can implement Abstraction in C++ using classes. The class helps us to group data members and member functions using available access specifiers. A Class can decide which data member will be visible to the outside world and which is not. Abstraction in Header files: One more type of abstraction in C++ can be header files. For example, consider the pow() method present in math.h header file. Whenever we need to calculate the power of a number, we simply call the function pow() present in the math.h header file and pass the numbers as arguments without knowing the underlying algorithm according to which the function is actually calculating the power of numbers. (Clark, 2013)

Polymorphism: the word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism. (Clark, 2013)

An operation may exhibit different behaviours in different instances. The behaviour depends upon the types of data used in the operation.

C++ supports operator overloading and function overloading.

- *Operator Overloading*: The process of making an operator to exhibit different behaviours in different instances is known as operator overloading.
- *Function Overloading*: Function overloading is using a single function name to perform different types of tasks. Polymorphism is extensively used in implementing inheritance.

II.3. Specific Scenario

A profile system for keeping all the library files that takes in files for input. Store file in categories and alphabetically. Files can be inputted into the library. If the files is used. There will be a categorizer that will address the file to the coding program so that the file can be accessed by the coding program. Files read will be highlighted in purple, non-read files will be highlighted in blue. Scholary files will have a separated engine dedicated for that file type. AI will be used for that engine. The AI can do research on files, manage files, suggest file based on the file relevancy and quoted times. Search engine can be managed by the programmer to search for files and the programmer can ask questions that the search engine will provide help. There will be a memory of all the files to record all the locations the files have been to. Memory can be managed by the admin but cannot be managed by the librarian but the data of memory cannot be deleted. File password service allows for access for only authorized users. File sharing allows the files to be shared via a media for better communication and information sharing. User can borrow books in library and register an account for reading books in library. Besides, it sells products related to entertainment products such as, mp3, and DVD. Users can refer to the products of the store (see product information, see product overview). If you decide to buy DVD or download MP3 the user will need to register an account at the store with personal information (name, address, telephone number, username, password) then the user will order products that they like. The store will check orders and deliver goods to users in 3 – 5 days.

II.4.1. Use-case Diagram

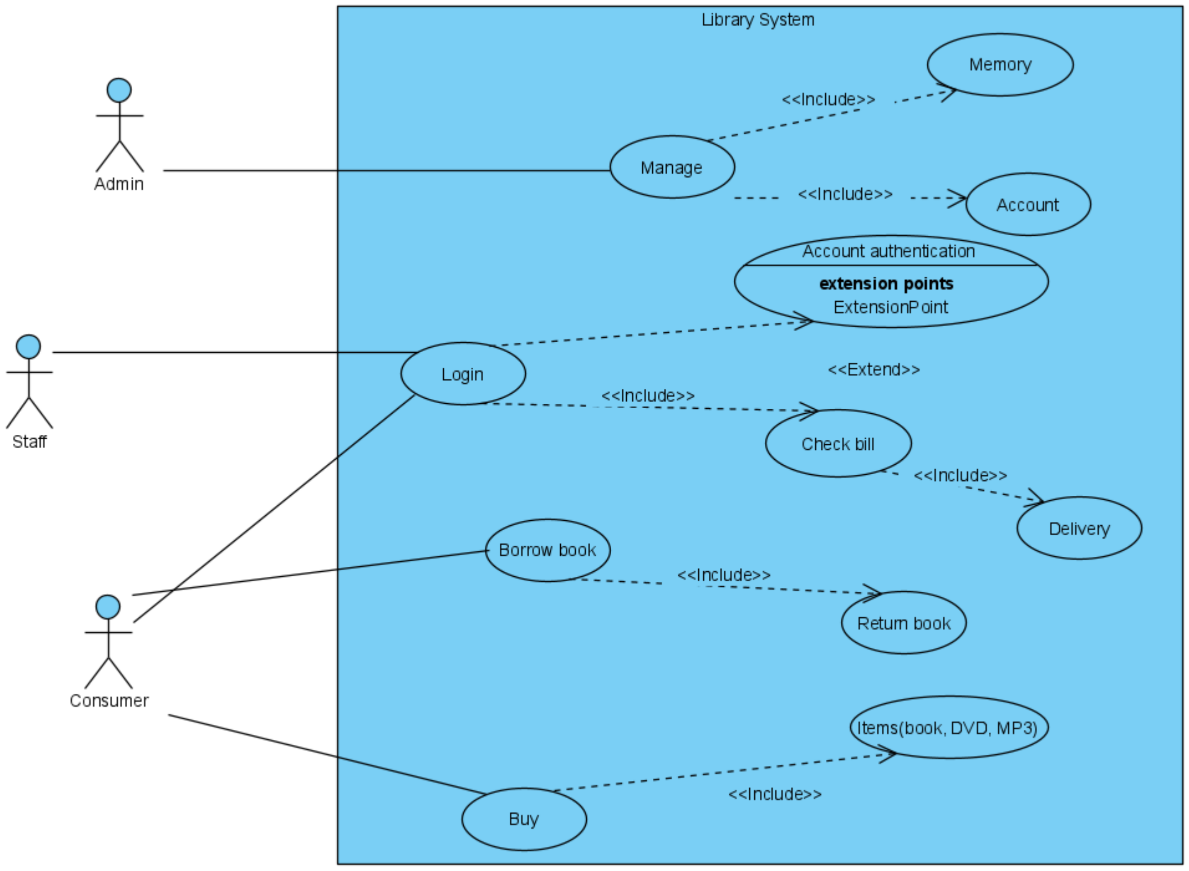


Figure 1: Use-case diagram

The whole system will contain three actors:

- The Admin: Admin can manage memory and manage account
- The Staff: Staff can login, account authentication, check bill, delivery
- The Consumer: Consumer can login, borrow book, buy book. Customers can check orders by logging into the system and delivery. Besides, customers can return books and buy DVDs and MP3.

II.4.2. Class Diagram

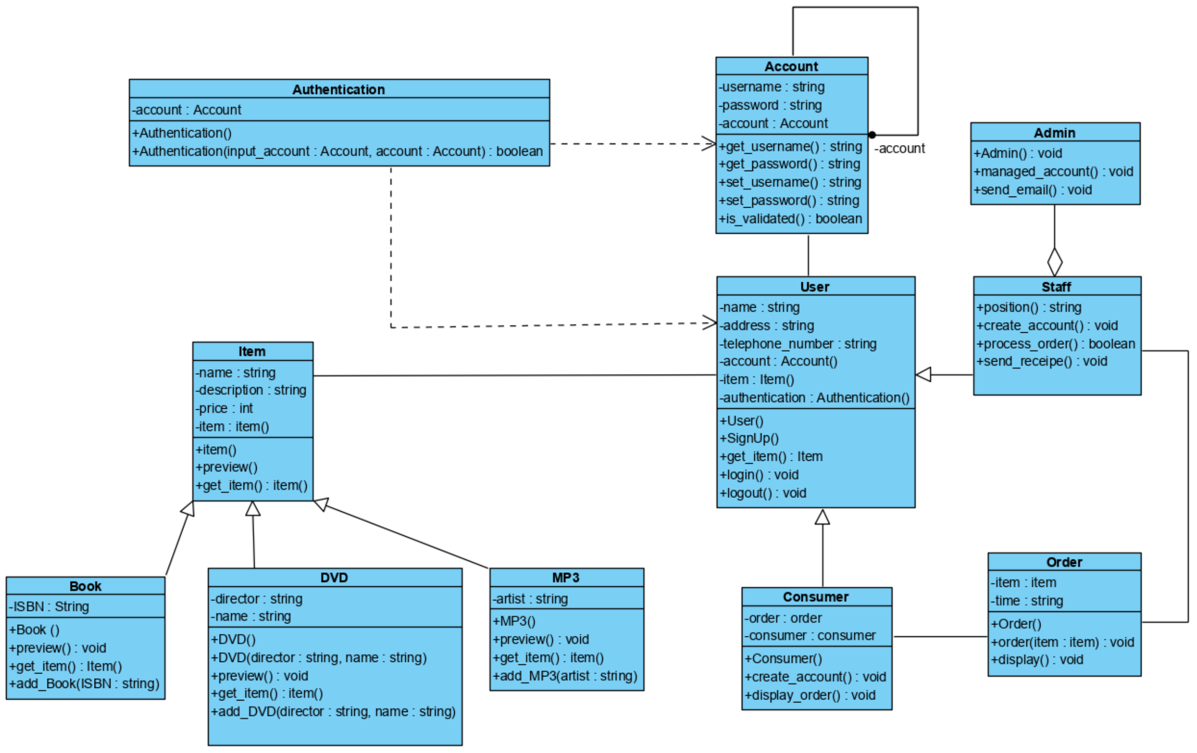


Figure 2: Class diagram

- Class User: This is an essential class for the system. This class includes some basic functions that a user needs including login, logout and some other functions.
- Class Account: With this class, the user can create an account with full username and password, in addition to users who want to create an account, this class will send a confirmation form to the account.
- Class Admin: The admin layer can manage the entire account from how many accounts have been created and will then send a confirmation email.
- Class Staff: This class can do the following tasks: create an account so that the user can use that account to log in to the system, besides it can handle orders according to customer requirements.
- Class Order: With this order class, it is easier for customers who want to order online.
- Class Consumer: As analyzed above, the staff will create a new account and come here, the customer will be the user of the allocated account to log in to the system. At the same time, customers can book books and DVDs more conveniently with their account.
- Class Item: With this class, users or customers can watch DVDs, books and MP3s conveniently through functions such as price, name or description. In my opinion, this class is quite important in the system

II.5. Object Oriented Programming (OOP) in the scenario

II.5.1. Inheritance

Looking at the class diagram, the inheritance characteristic is shown in the Consumer and Staff class. These two classes can inherit methods from User class. Though Consumer and Staff do not having login() and logout() method but they can inherit from Employee as long as login() and logout() method are set under public.

II.5.2. Abstraction

The Employee class contains the Consumer () method in abstract form. In these two classes there is also one the method is called Consumer () but is used for different purposes and situations. The manager through Consumer () method can do a number of things like managing staff

II.5.3. Polymophism

Both class Consumer and class Staff are inherited method account() from class User. In different circumstances they using method account() for specific purposes of each class. In this class diagram, method account() is set as virtual method so that it can be overridden

II.5.4. Encapsulation

III. Design Pattern

III.1. Behaviors patterns

Behavioral patterns are involving algorithm and the relation between objects. It describes not only the behavior between objects but also the behavior between the interaction between them. (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, 1995, p. 221)

III.1.2. Temple Method

Definition

Template **method** – behavioral pattern: Determine the **structure** of an **set of rules** in an operation, inherit **a few** steps to subclasses. Template **approach shall we** subclasses re-**determine a few** steps of an set rules however do no longer affect to the shape of set of rules. Using template approach helps builders to keep away from the duplication in coding. (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, 1995, p. 325)

Applicability:

- Execute the invariant parts of the algorithm once and leave it to subclasses to execute behavior may vary.
- When not unusual conduct amongst subclasses should be factored and localized in a commonplace class to keep away from code duplication.
- Controlling subclasses' extensions.

Structure

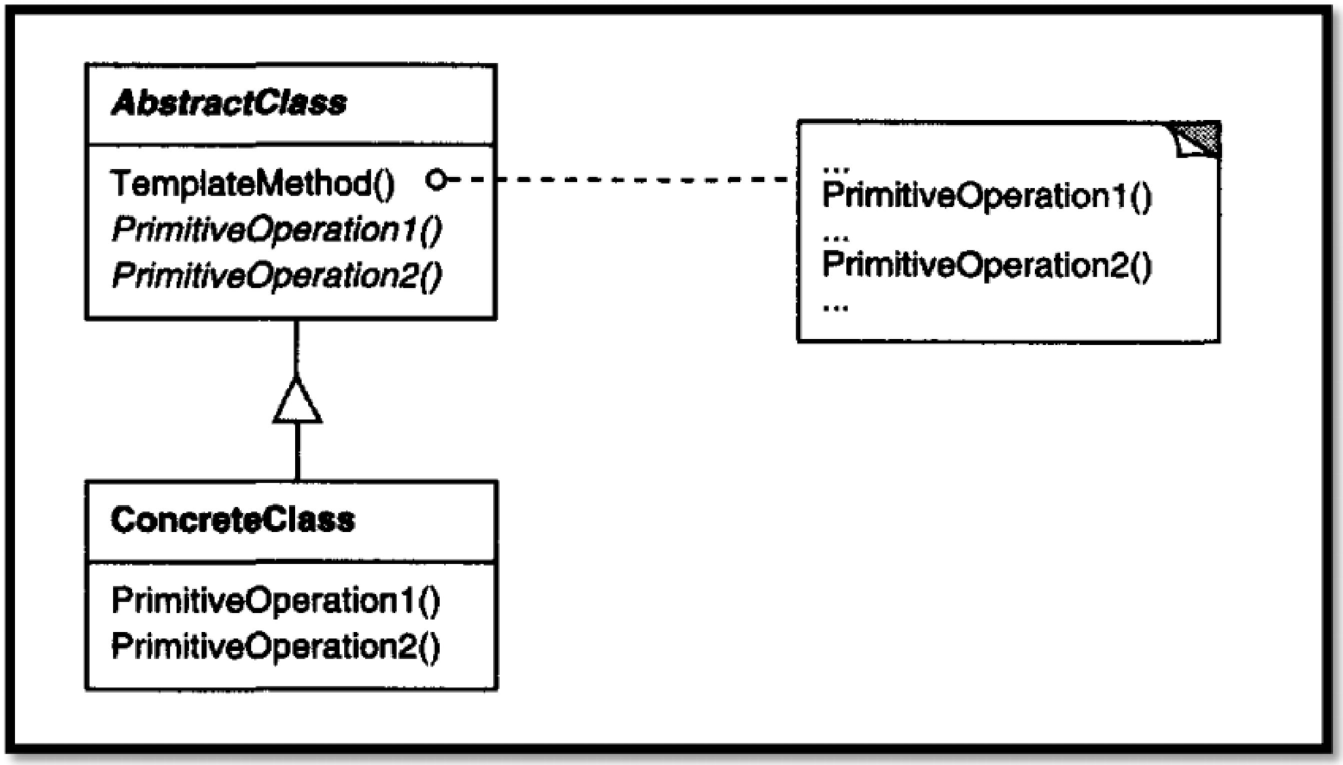


Figure 3: Structure of Template method

There are 2 basic classes of template method.
• AbstractClass: this class contains one or many operations for the concreteClass to re-define those operations.
• ConcreteClass: this class inherits the abstract operations from super class then re-determine the abstract operations of super class

Scenario

The structure of an HTML page has a title, navigation bar, body, and footer. Usually a website always have the same title and footer for every site, but the body and possibly the navigation bar can change depends on the content and purpose of each page. Instead of headers and footers for each page, they should be placed in a template.

Diagram

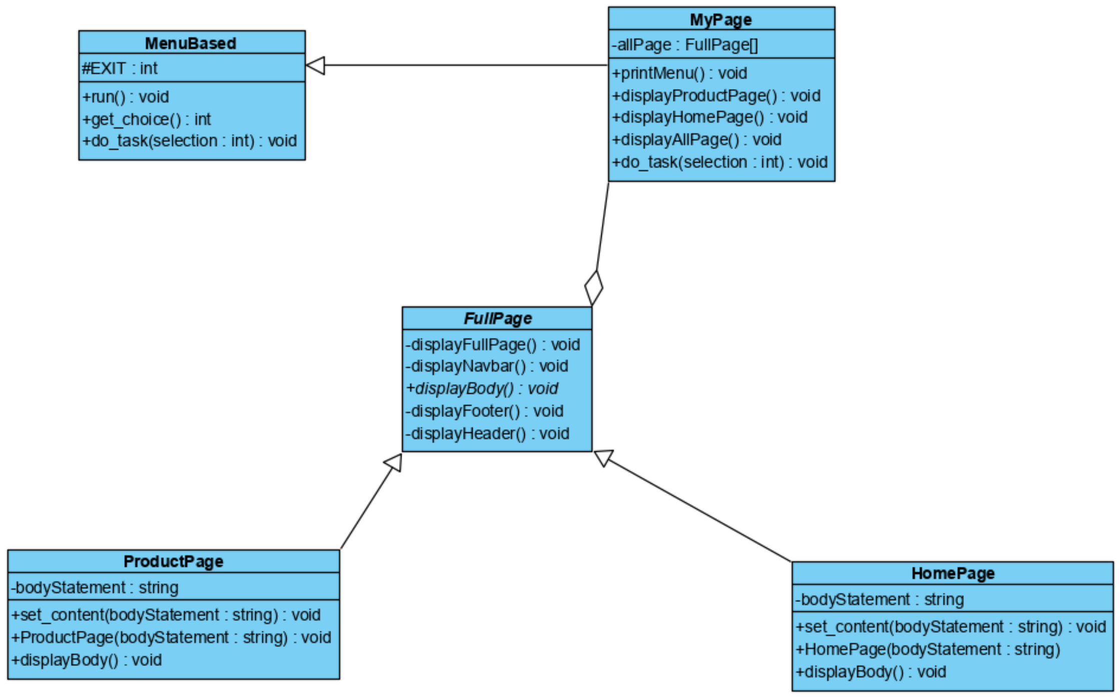


Figure 4: Template method

In this scenario, FullPage is the incredible class that is inherited to ProductPage magnificence and HomePage elegance. It carries displayFullPage, displayNavBar, displayFooter, displayHeader and an abstract operation that is displayBody operation.

ProductPage and HomePage training are sub-instructions inherit the displayBody() lessons from super elegance. So that, they could re-outline this class after and show exclusive HTML pages after override it.

III.2. Structural Patterns

Structural design patterns are concerned with how classes and objects can be composed, to form larger structures. The structural design patterns simplifies the structure by identifying the relationships. These patterns focus on, how the classes inherit from each other and how they are composed from other classes.

III.2.1. Facade Pattern

Definition: Facade Pattern is to create a wrapper class for a subsystem (be it a library or a module consisting of many different classes), helping to simplify the interface of that subsystem. As a result, other modules can work with that subsystem more easily through the wrapper, the complexity of the subsystem itself will not affect the outside world.

Characteristic of Structural Patterns:

- Adapter
- Bridge
- Composite
- Decorator
- Façade
- Flyweight
- Proxy

Scenario: A system used to store customer information after buying clothes at a store to contact customers for promotions. These stores include customer names, customer addresses, phone numbers to contact

Class Diagram:

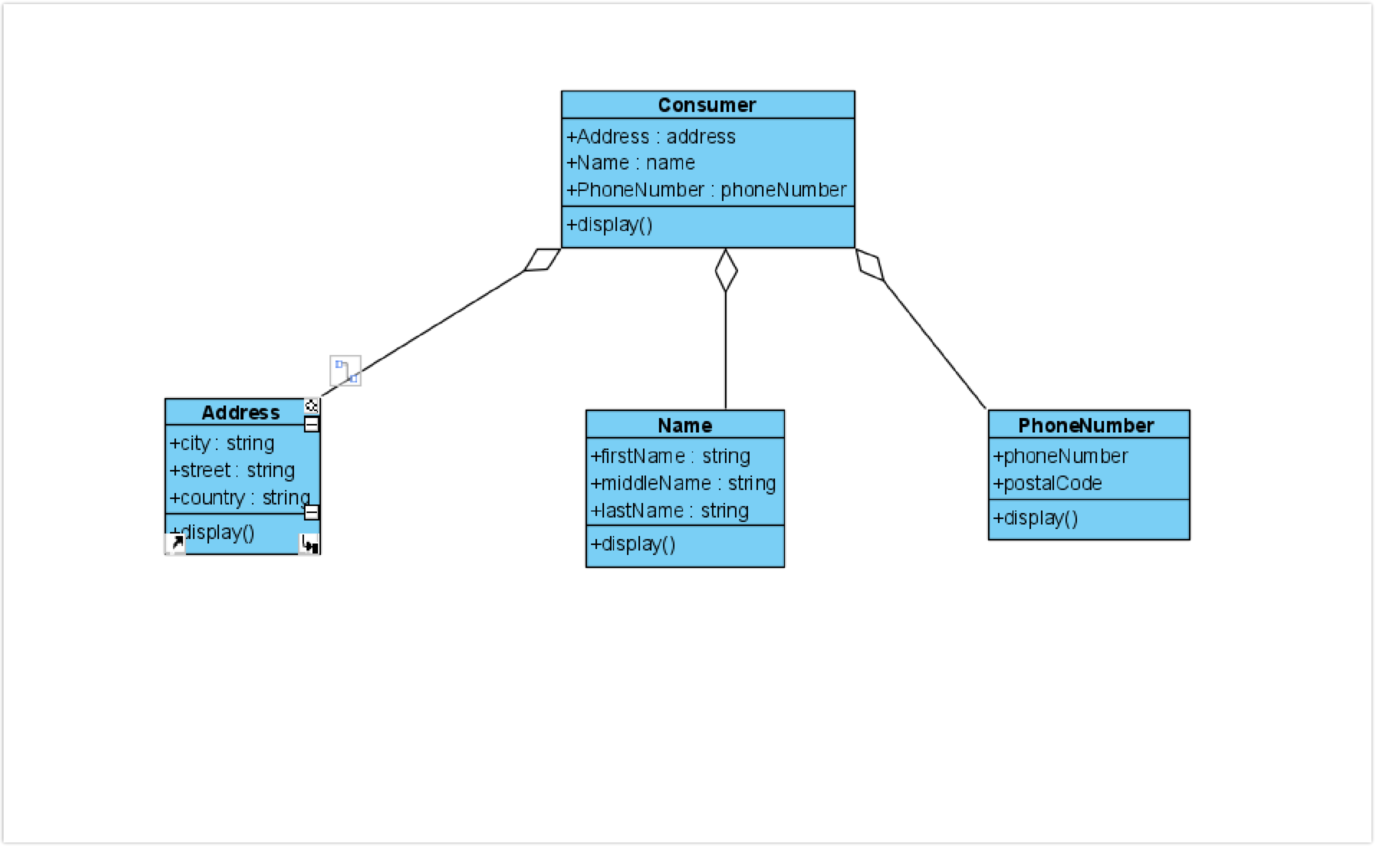


Figure 5: Facade Pattern

IV. Conclusion

This file consists of full know-how of OOP and Design Patterns with unique scenarios. Each layout pattern is explained cautiously and provided with a class diagram the usage of UML tools. Overall, the report has a typically achieved intention for the given scenario.

References

Clark, 2013. *Beginning C# Object-Oriented Programming.* Apress: s.n.

Clark, D., April 21, 2020. *Beginning C# Object-Oriented Programming. Apress.* s.l.:s.n.

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. (1995). *Design Patterns - Element of Reusable Object-Oriented Software.* Oxford University Press . Retrieved April 25, 2020

Index of comments

1.1 OOP:
- Truong: explain general concepts and definitions of OOP
- Khoi: explain about scenario => quite detailed
- Truong: Use case diagram => features of scenario are use cases, several incorrect relationships (include), should have extend relationship

Design class diagram:
- Truong: explain class diagram => quite detailed. Understand and apply OOP concept on diagram. However, some incorrect relationship (Admin vs Staff)
- Khoi: explain part of diagram, cannot explain relationship among Staff vs Order, very struggle when drawing constructor

Patterns:
- Khoi: Façade pattern => seem suitable scenario but don't understand class and component of Façade pattern.
- Truong: Template method => suitable scenario => can explain component of Template method pattern in diagram.