ASSIGNMENT 1 FRONT SHEET

Qualification	**TEC Level 5 HND Diploma in Computing**		
Unit number and title	**Unit 43: Internet of Things**		
Submission date		**DateReceived1stsubmission**	
Re-submissionDate		**DateReceived2ndsubmission**	
Student Name	Bui Dinh Kha	**Student ID**	GCH17468
Class	GCH 0709	**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	

Grading grid

P5	P6	P7	M5	M6	D2	D3

Contents

Figure

I. OOP definition and characteristics

1) Definition of OPP

According to (Rouse, 2019)Object-oriented programming is a programming technique that allows programmers to create objects in code that abstract real-life objects. This approach is currently very successful and has become one of the software development paradigms, especially for enterprise software.

When developing applications using OOP, we will define classes to model real objects. In the application these classes will be instantiated as objects and throughout the application run, the methods of this object will be called.

2) Encapsulation

As a way to hide the inner handling properties of an object, other objects cannot be directly manipulated to change state that can only be accessed through the public methods of that object.

Any access to this internal state is required through a public API to ensure the state of the object is always valid because the public APIs are responsible for performing validity checks as well as the update order. Update the status of the object. (Rouse, 2019)

3) Inheritance

Inheritance and reusing methods and properties of the base class. And the inheritance class is called the subclass, it will inherit what **the parent class has and allows.**

For example

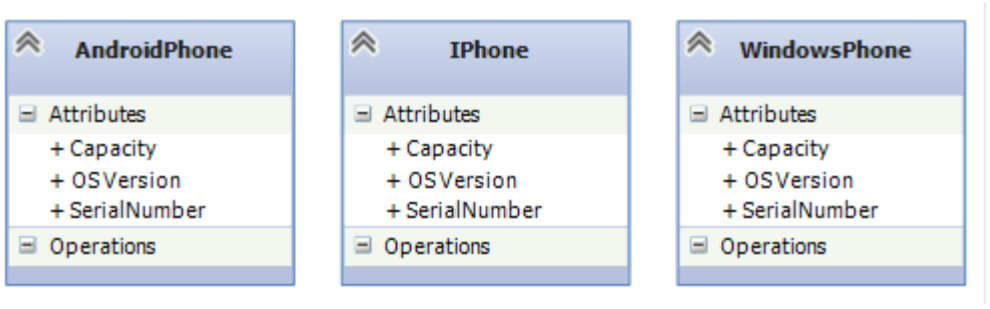


Figure Sample 1 (ungnt185, 2019)

Each class represents a different type of smartphone but has the same properties. Instead of copying these properties, it is better to put them in a place that can be used by other classes.

4) Polymorphism

Polymorphism allows different functions to be executed differently on different objects.Put it simply: Polymorphism is a concept that two or more classes have the same methods but can be implemented in different ways.

5) Abstraction

An abstraction method that defines the actions and properties of certain types of objects**.** The abstraction here is primarily intended to abstract and define the behavioral properties required to determine what object it is based on the behavioral nature of the object.

Example for more clearly. We have classes that are phones (phones), there are many types of phones, nokia, samsung, iphone ....with different shapes and functions, so how can we determine where is the opposite? phone icon. But you just need to imagine what they all have in common, listening, calling, texting. So, the object of the new phone will inherit the listening, calling, and texting properties of the class (Phone)

Figure 1 Example 2 (Dreamtime, 2019)

6) Advantages of OPP

-Because object-oriented programming came later, it overcome all the weaknesses of the previous programming methods. Specifically, the following advantages:

- Easily manage code when there is a program change.
- Easy to expand the project.
- Save significant resources for the system.
- High security.
- Highly reusable. (Puri, 2019)

II. Assumed Scenario for OOP

1) Introduction scenario.

Currently there are many interesting games to help people entertain and reduce stress. Prior to this demand, we were required to quickly introduce and develop a simple and convenient game. Game Snake is a game that does not require high intelligence, the important thing is skill, it helps you feel comfortable when playing games.

Game instructions:

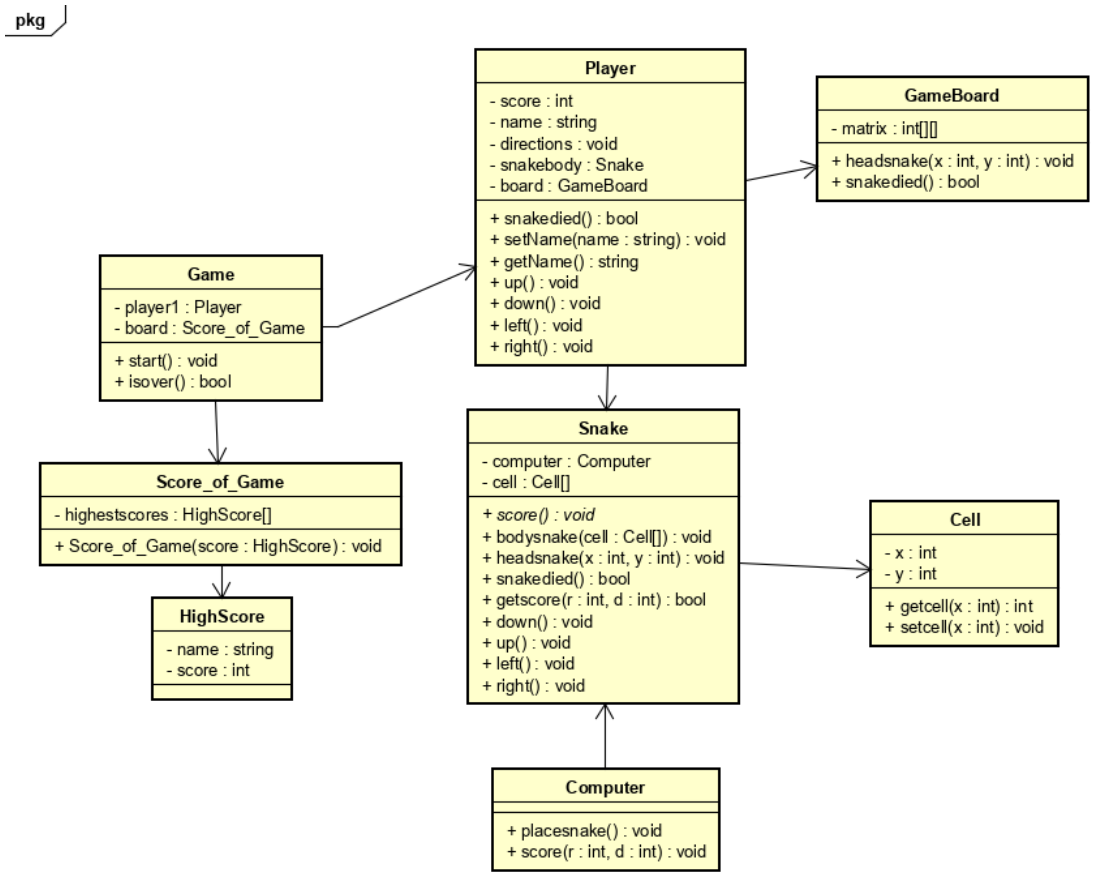
1. At first you will have a snake with the smallest length and food shown on the screen.

2. The way the snake eats food within a screen limit, can go through walls and extend the body when eating 1 food.

3. How to move: you will control the snake with buttons up, down, left and right so that the snake will eat the food in the map and not bite into its body.

4. You will end the game when your snake bites itself or it spans the map. Your points will then be saved and displayed on the screen. You can see the player's queue of points at the scoreboard.

2) Class diagram.



I designed this class diagram with 9 classes. In which the Game class is associated with the Class Player and Score_of_Game, the Game allows the program to run with the start () function: void, and the program ends.

The player's score will be saved at Score_of_Game class, this class will output the player's score and the highest score in the game.

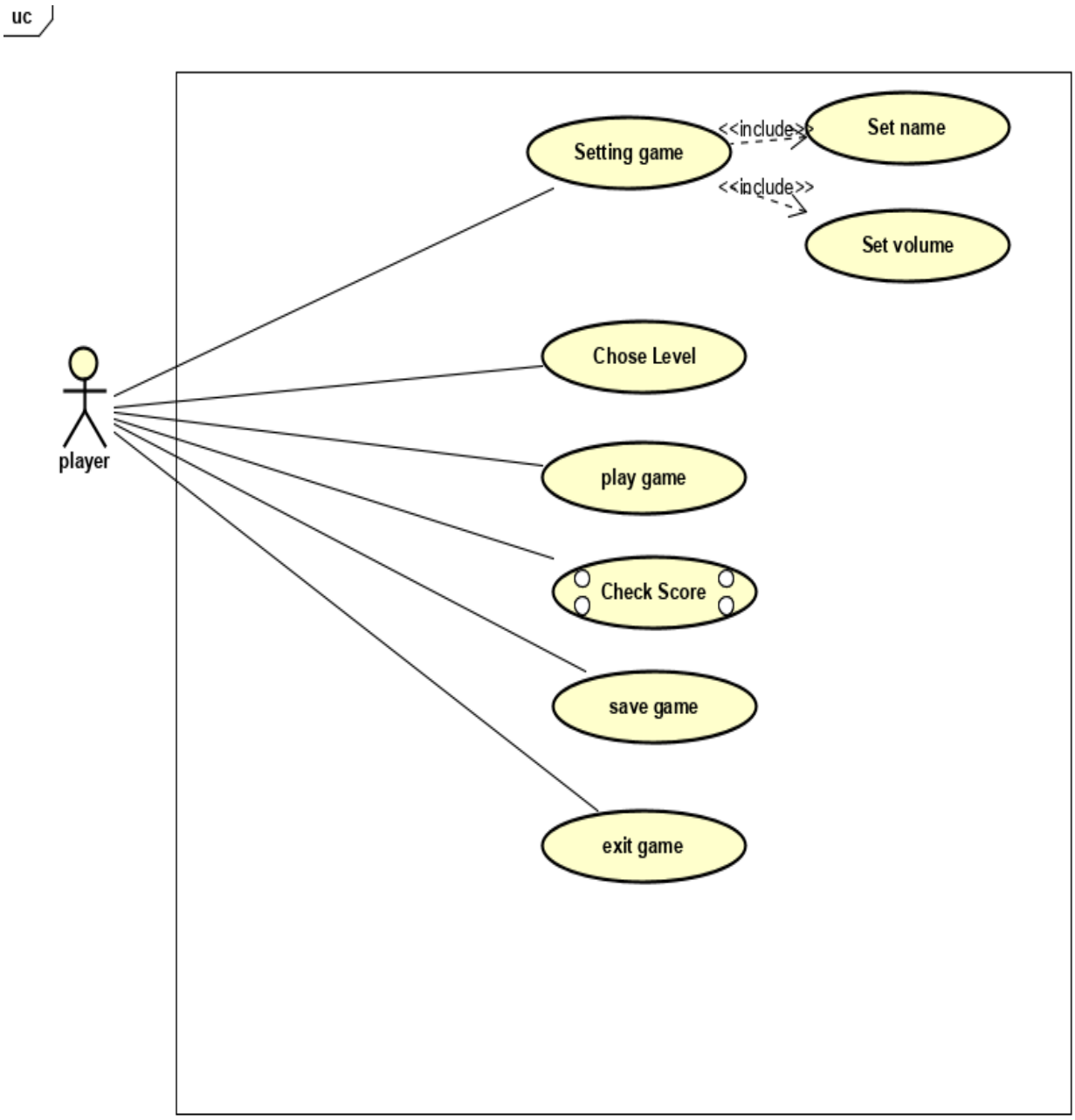
Class Player is the task of setting the player's name, receiving keys to move from the player using the up (), down (), left (), right () functions.

The GameBoard class is where the game board is printed with the matrix: int [] [] function, in addition this class also determines the snake's head.

Snake class is an important class, it locates the head and body of the snake by links with Class Cell, snake body will be a collection of Cell []. Run up (), down (), left (), right () functions to change the snake's movement direction.

Class Computer has an Association relationship with Class Snake. The computer is responsible for determining the original snake's position and the snake's food.

3) Use Case diagram

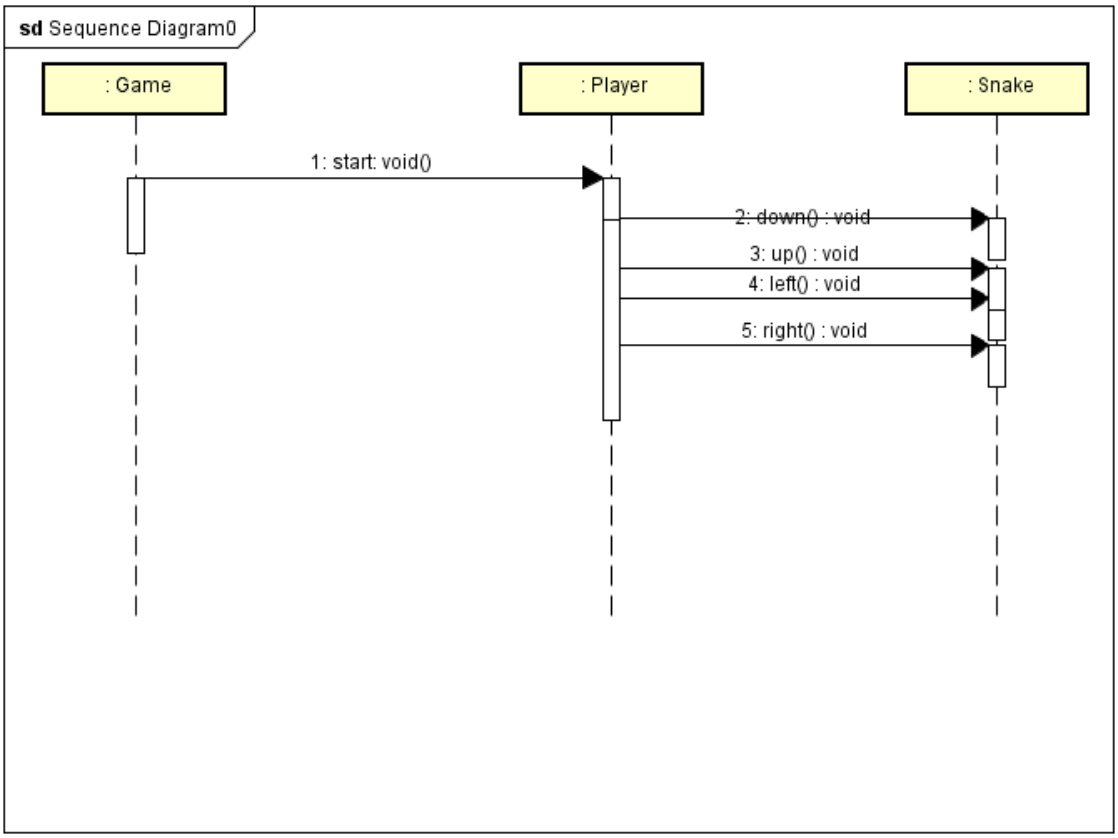


Players have the function of setting features for the game such as Set name, Set volune. Chose level, play Game, go to Check Score to check their score and finally Save game and Exit game

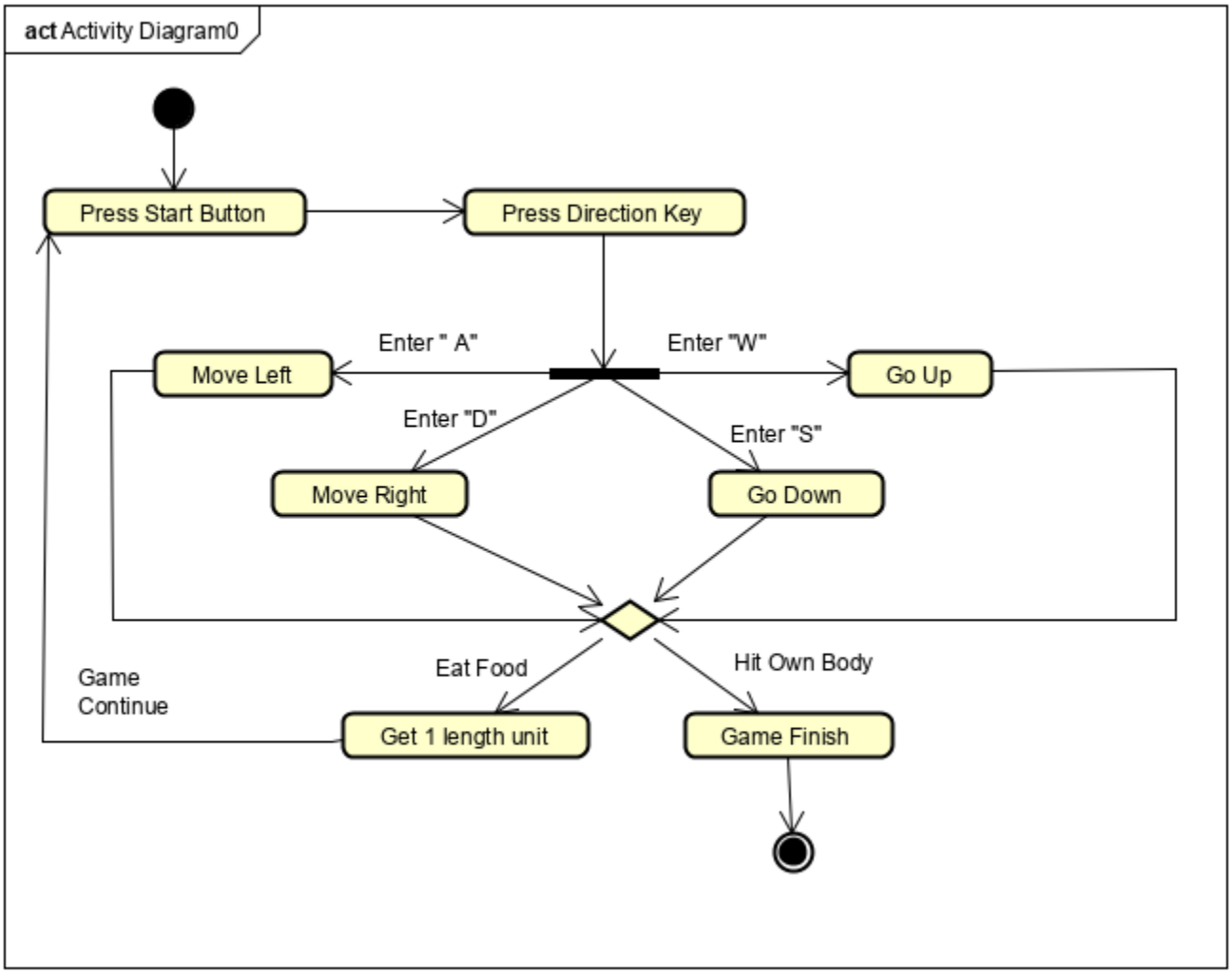
4) Sequence Diagram.

First player will press one of the buttons up, down, left, right , then Game will receive a request and respond to a solid command to start running. After the snake has run, the player keeps pressing, up, down, right, leftbut in it, the snake will run based on the key pressed.

When the player controls the coordinates of the snake head and the food coordinates overlap, the Game will increase the snake length. And if the snake's head is stabbed and its body, the Game will end.



5) Activity diagram for snake game



Explain activity: on the display screen, the player will start snake game by selecting start button. The goal of player is to move the snake to eat food by controlling 4 dicrection key: turn left, turn right, go up and go down. Each time snake eats the food, it will get 1 more unit length and the score of player will increase.The game keeps continuing. The game will end if player controls snake to touch the border (wall) or eat itself body.

III. Introductions and Design Patterns

Design Patterns are an object-oriented programming technique that is quite important and every good programmer must know. Design patterns are frequently used in OOP languages. It gives you "designs", solutions to solve common problems, common in programming. The problems you encounter may come up with a solution yourself, but it may not be optimal. Design Pattern helps you solve problems in the most optimal way, providing you with solutions in OOP programming.

1) Creational patterns

This design patterns provide a solution for creating objects and concealing the logic of its creation, instead of creating objects directly using the new method. This makes the program more flexible in deciding which objects need to be created in given situations.

Creational Pattern include

- Factory Method :
  Define Interface to generate objects but let the subclass decide which class to use to generate the Factory method object, which allows a class to pass object initialization process to the subclass.
- Builder:
  Separate the construction of a complex object from its representation so that the same construction process can create different representations.
- Abstract Factory:
  Provides an interface for creating objects (related to each other) without specifying a class when or specifying a specific class (concrete) to create each object
- Prototype:
  Specifies the type of objects to create using a template object, creating a new one by copying this pattern object.
- Singleton:
  Make sure a class has only one instance and provide a global access point to it.

For more understand I will take the example with Abstract Factory :

A company that produces stuffed toys about animals and people.

With the current position, the company decided to expand the production of more shapes, for the animal class the company produced more dogs and in, for the human class the company developed in the form of superheroes and actors. . The production process for animals and humans is different, so there will be 2 companies 1 for animals, 1 for humans, based on the parent company's methods.

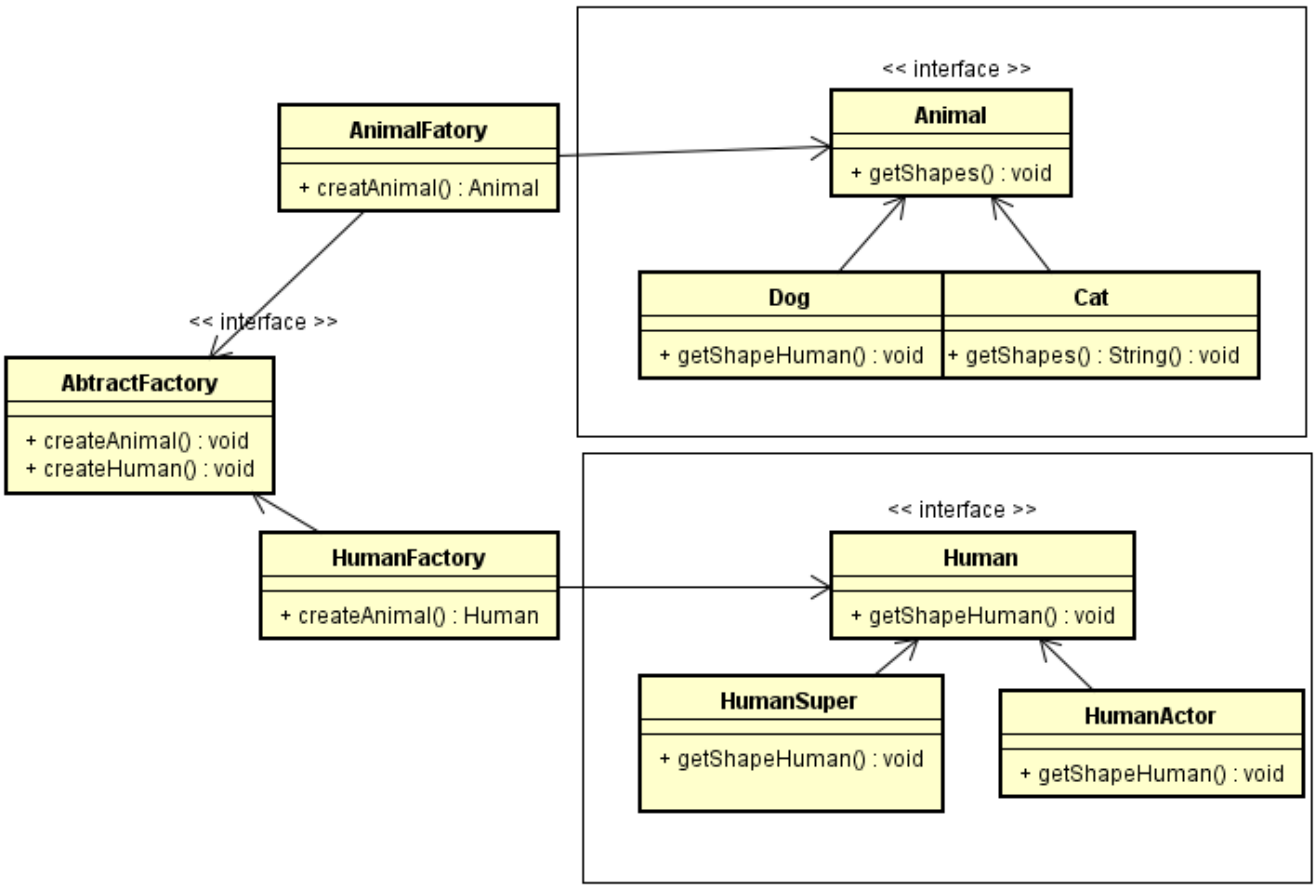


Figure 2 Sample Abstract Factor

2) Observer in behavioral design patterns

Definition: the observer is one kind of behavioral design pattern in which an object (subject) maintains a list of its dependencies (observer) and automatically notifies them of any changes, usually by calling one of their functions

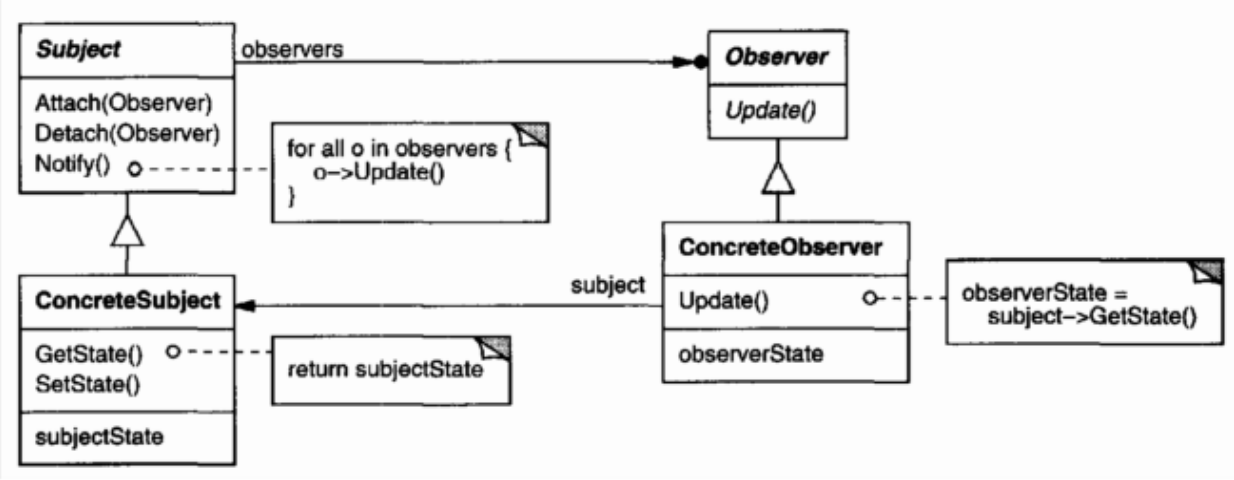
Purpose

- Define a one-to-many relationship between objects so that when one object changes state, all its dependents are notified and updated automatically
- An object can notify an unlimited number of other objects

Structure

When a subject has a status change, it will browse through its list of observers and call the status update method for each observer.

Here is the structure which according to (Tung D Trung, *Advance Programming*, lecture note, FPT university):



Subject: knows its observer. Any number of Observer object may observer an object. It provides an interface for attaching and detaching Observer object

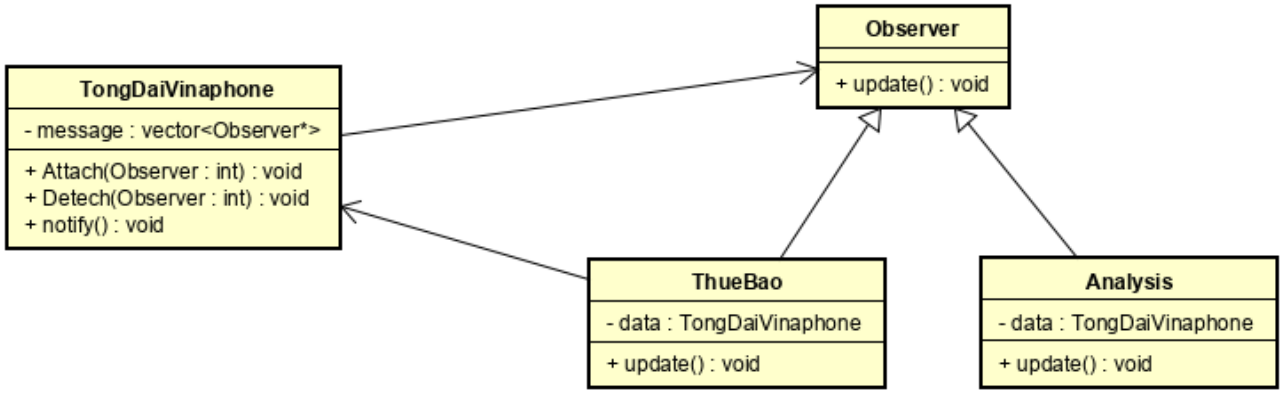
Observer: define an updating interface for object that should be notified of changes in a subject

ConcreteSubject: store state of interest to ConcreteObserver object and send notification to its observer when its state changes

ConcreteObserver: maintain a reference to a ConcreteSubject object, save state of subject and implement the Observer updating interface to keep its state consistent with the subject's

Example

The telecommunication network Vinaphone wants to send message to all users each time it wants or attach or delete more users numbers. In this case, the observer pattern is fit. The class diagram below will ilustrate:



3) Structural Patterns.

Structural design 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.

Types of structural design patterns:

There are following 7 types of structural design patterns.

Adapter Pattern: Adapting an interface into another according to client expectation.

Bridge Pattern: Separating abstraction (interface) from implementation.

Composite Pattern: Allowing clients to operate on hierarchy of objects.

Decorator Pattern: Adding functionality to an object dynamically.

Facade Pattern: Providing an interface to a set of interfaces.

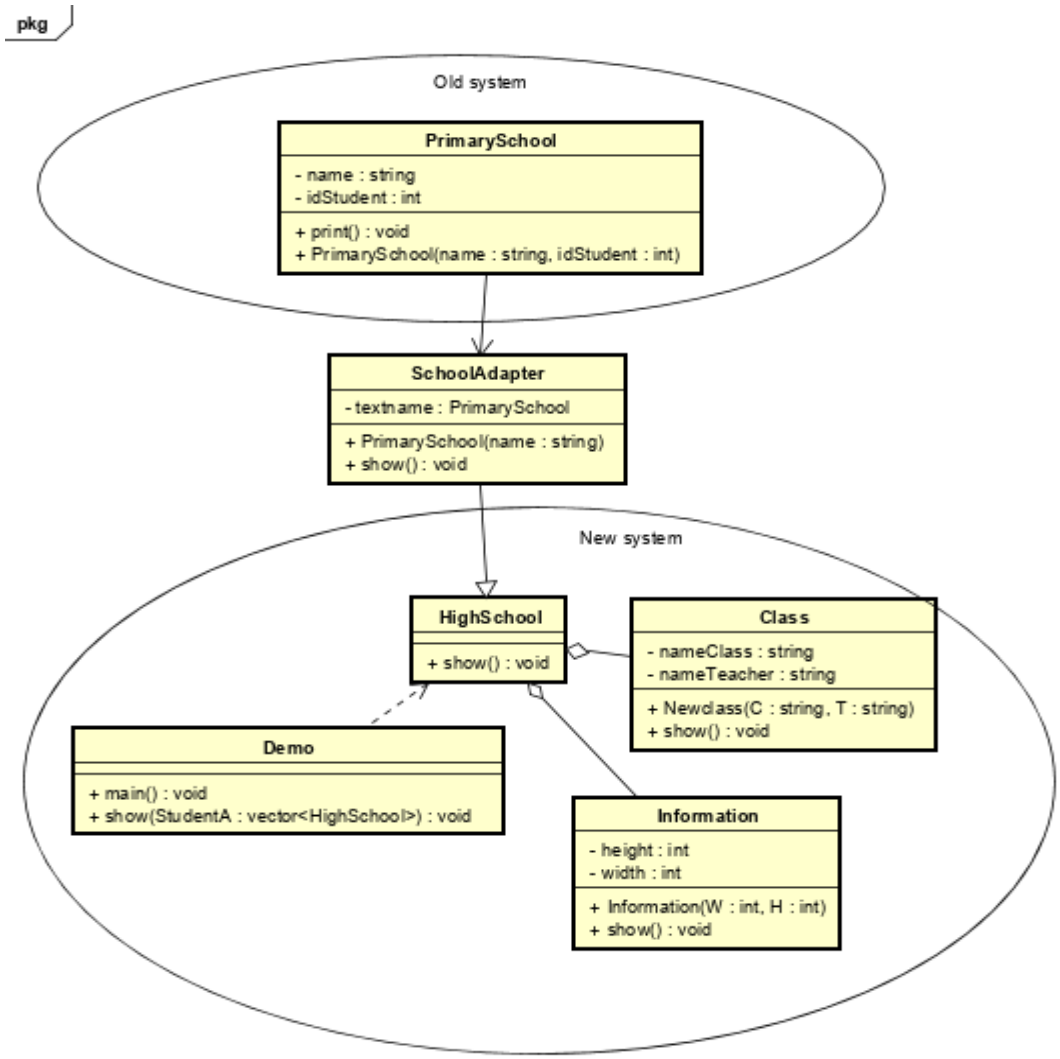
Flyweight Pattern: Reusing an object by sharing it.

Proxy Pattern: Representing another object.

Example: Adapter

In software engineering, the adapter pattern is a software design pattern (also known as a wrapper, an alternate naming way shared with a decorative pattern) that allows the interface of an existing class to be use as another interface. It is often used to make existing classes work with other classes without modifying their source code.

This system helps support two systems when they are not in the same interface. In the new interface, the system wants to re-use the Name data in the system, but not the same interface cannot be accessed. Adapter class created to inherit the old system and link to the new system, from which data can be used from the old system.



IV. Bibliography

Dreamtime. (2019). Retrieved from https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjg6JyDqc7lAhVZE4gKHSgnB58QjRx6BAgBEAQ&url=https%3A%2F%2Fwww.dreamstime.com%2Froyalty-free-stock-photography-phone-its-functions-three-phones-different-icons-image36183247&psig=AOv

Patterns, D. (2019). Retrieved from https://sourcemaking.com/design_patterns/creational_patterns

Puri, D. (2019). Retrieved from https://www.quora.com/What-are-the-advantages-of-OOP

Rouse, M. (2019). Retrieved from https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP

ungnt185. (2019). Retrieved from https://tungnt.net/cac-nguyen-ly-cua-lap-trinh-huong-doi-tuong-object-oriented-programming/

❒ **Summative Feedback:**

❒ **ResubmissionFeedback:**

Grade:	AssessorSignature:	Date:
InternalVerifier'sComments:		
Signature&Date:		