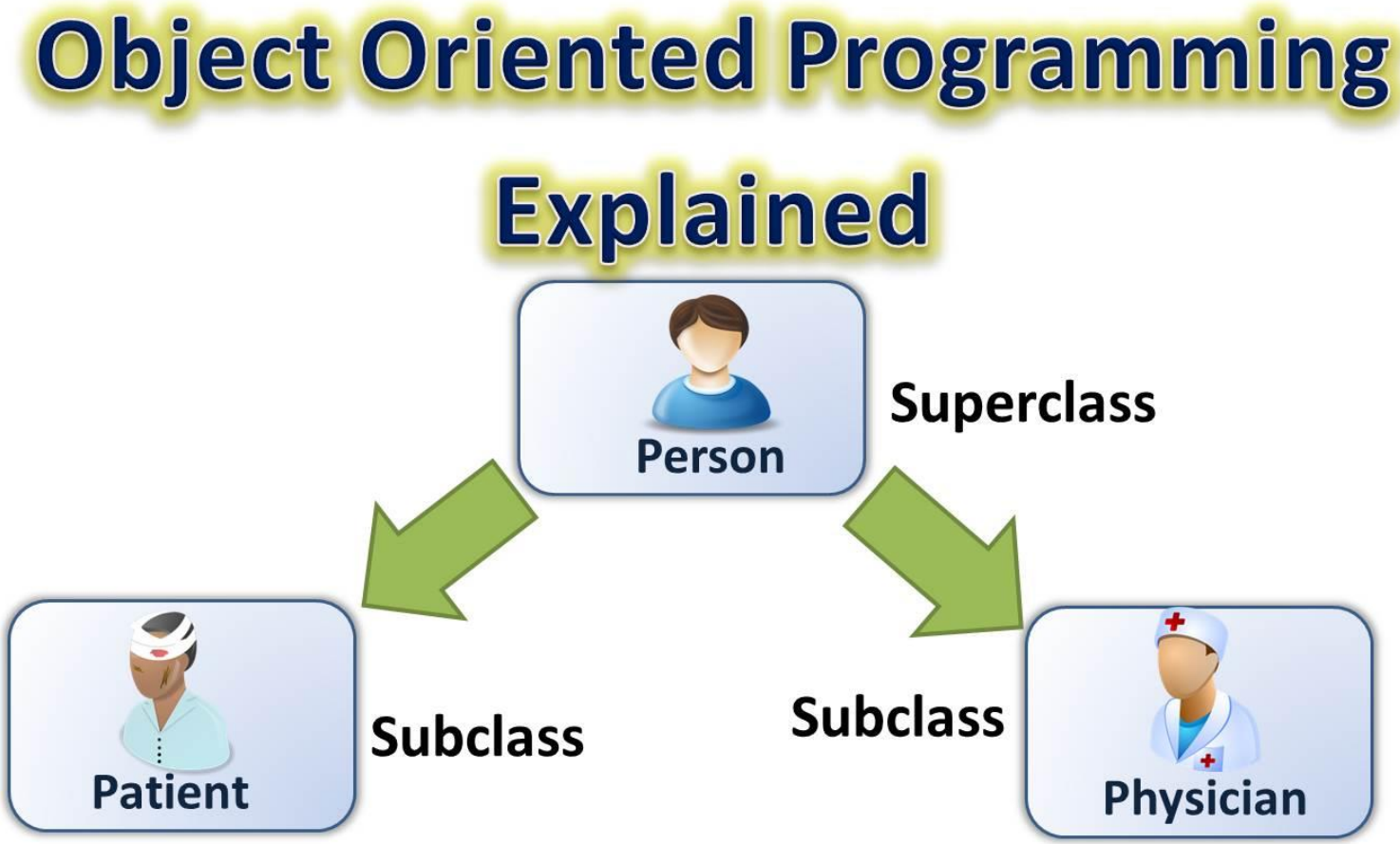
Advanced programming with C ++

Member: Bui Dinh Kha : GCH17468
Ngo Viet Duy: GCH 17574
Nguyen Huu Loc: GCH 15001

Table conten

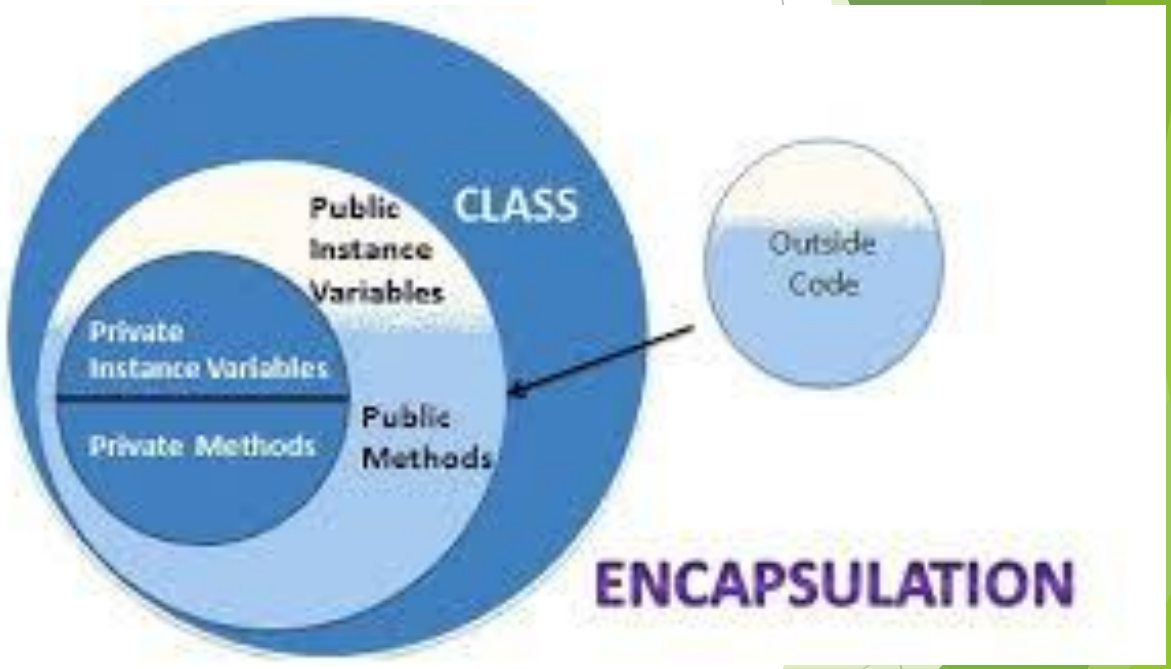
I.OOP definition and characteristics

1. Definition of OPP

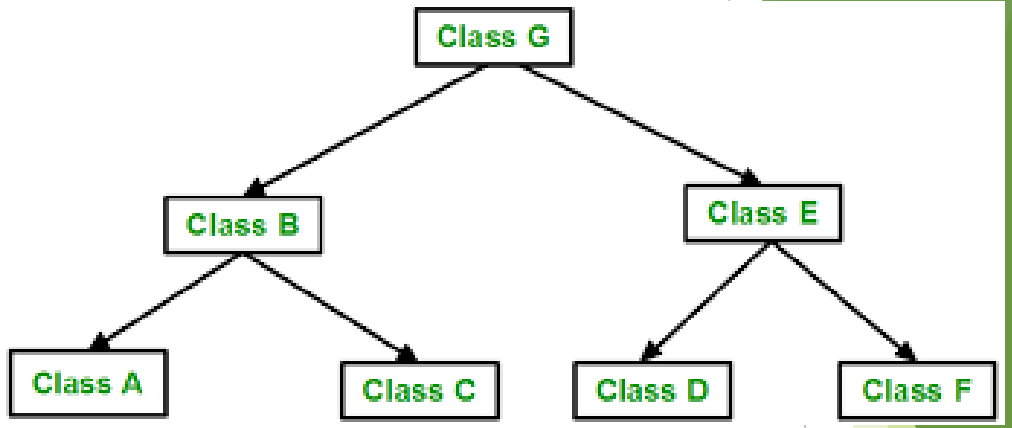


2) Encapsulation

As a way to hide the inner handling properties of an object, other objects cannot be manipulated directly to change state that can only be accessed through the object's public methods. .
All access to this internal state is required through the public API



3) Inheritance

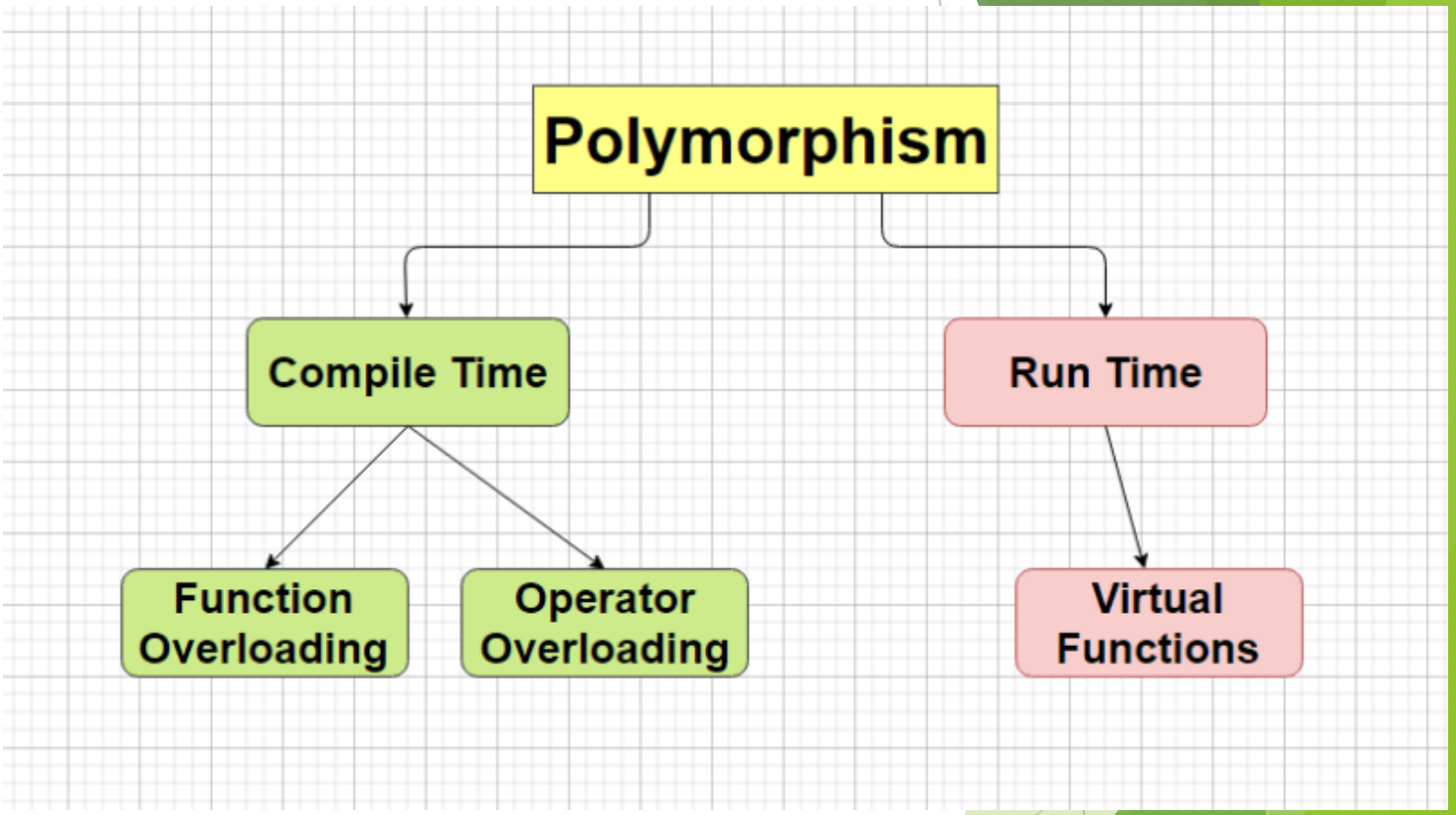


Inheritance and reusing methods and properties of the base class. And the inheritance class is called the subclass, it will inherit what **the parent class has and allows.**

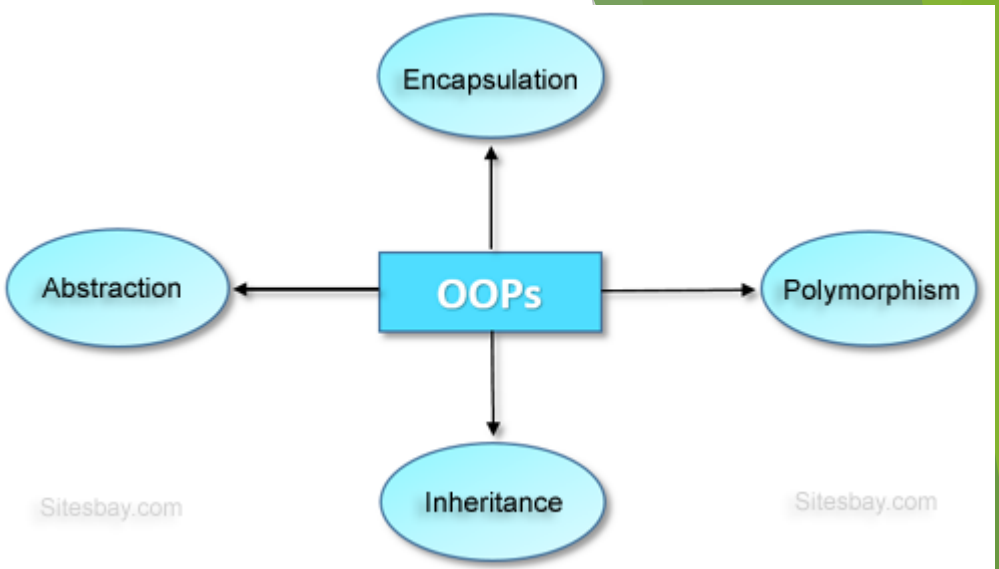
For example:

4) Polymorphism

Polymorphism allows different functions to be executed differently on different objects.
Another way to say: Polymorphism is a concept that two or more classes have the same methods but can be using in different ways.



5) Abstraction



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

Example

6) Advantages of OPP

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

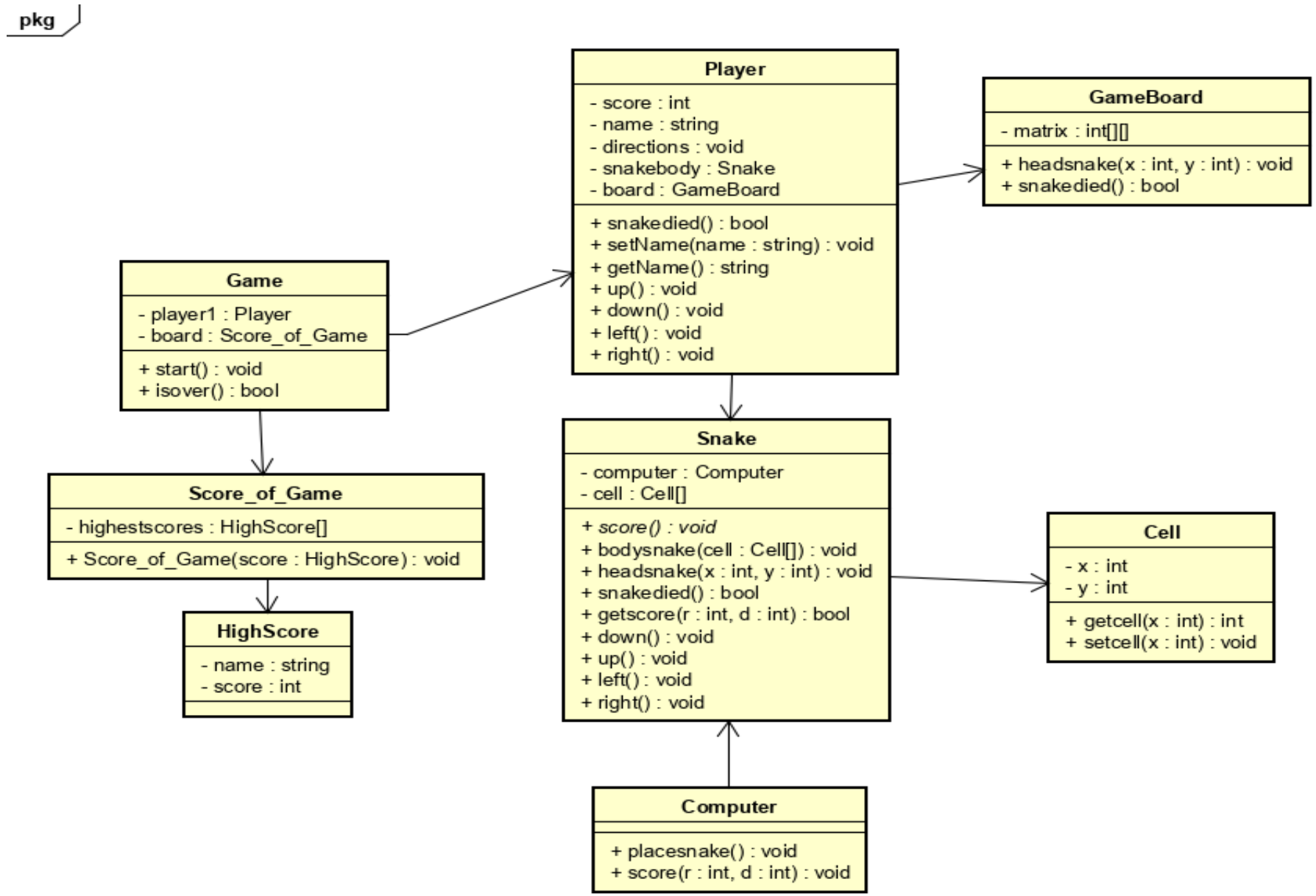
- Easily manage code when there is a program change.
- Easy to expand the project.
- Save significant resources for the system.
- High security.
- Highly reusable. (Puri, 2019)

II. Assumed Scenario for OOP

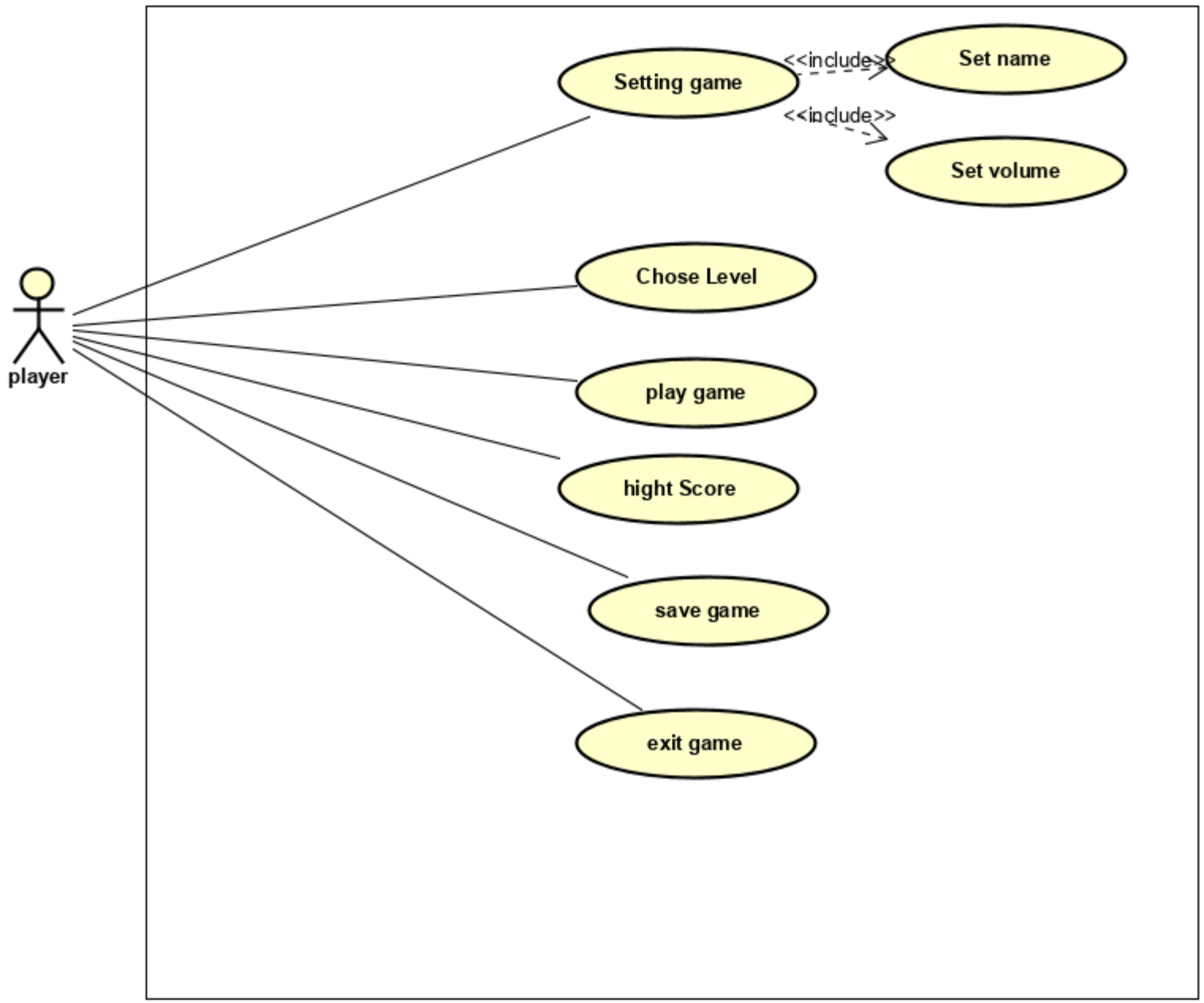
1. Introduction scenario

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.

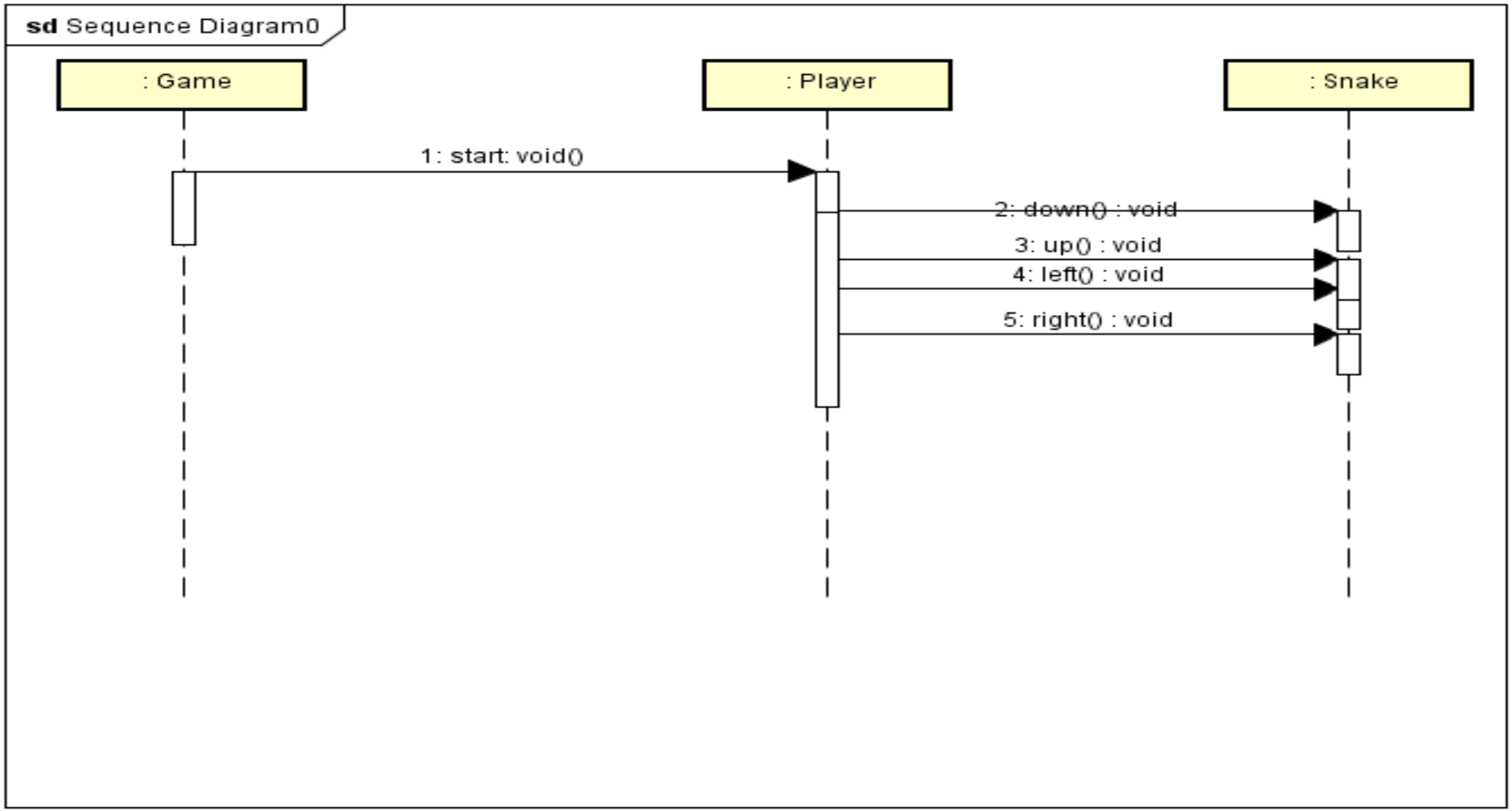
Class diagram.



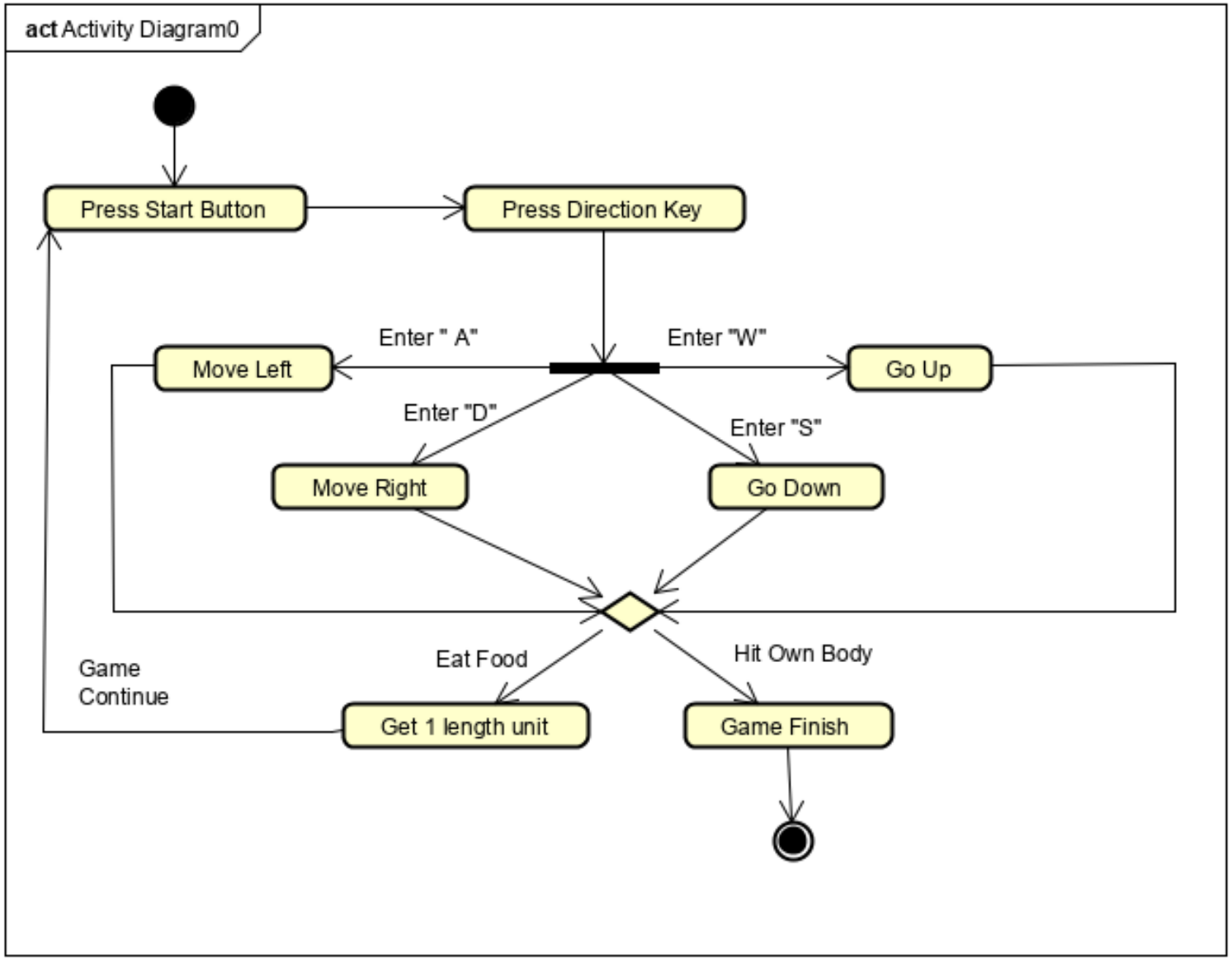
Use Case diagram



4) Sequence Diagram



5) **Activity diagram for snake game**

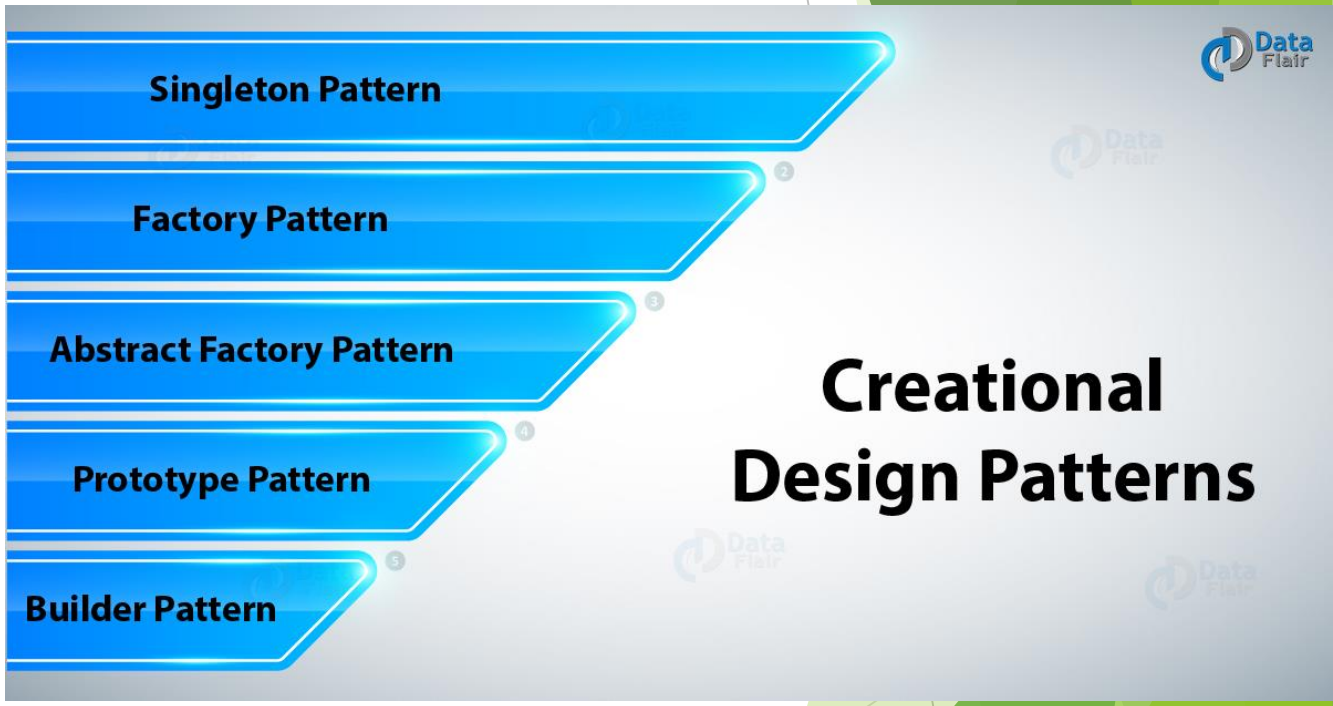


III. Introductions and Design Patterns

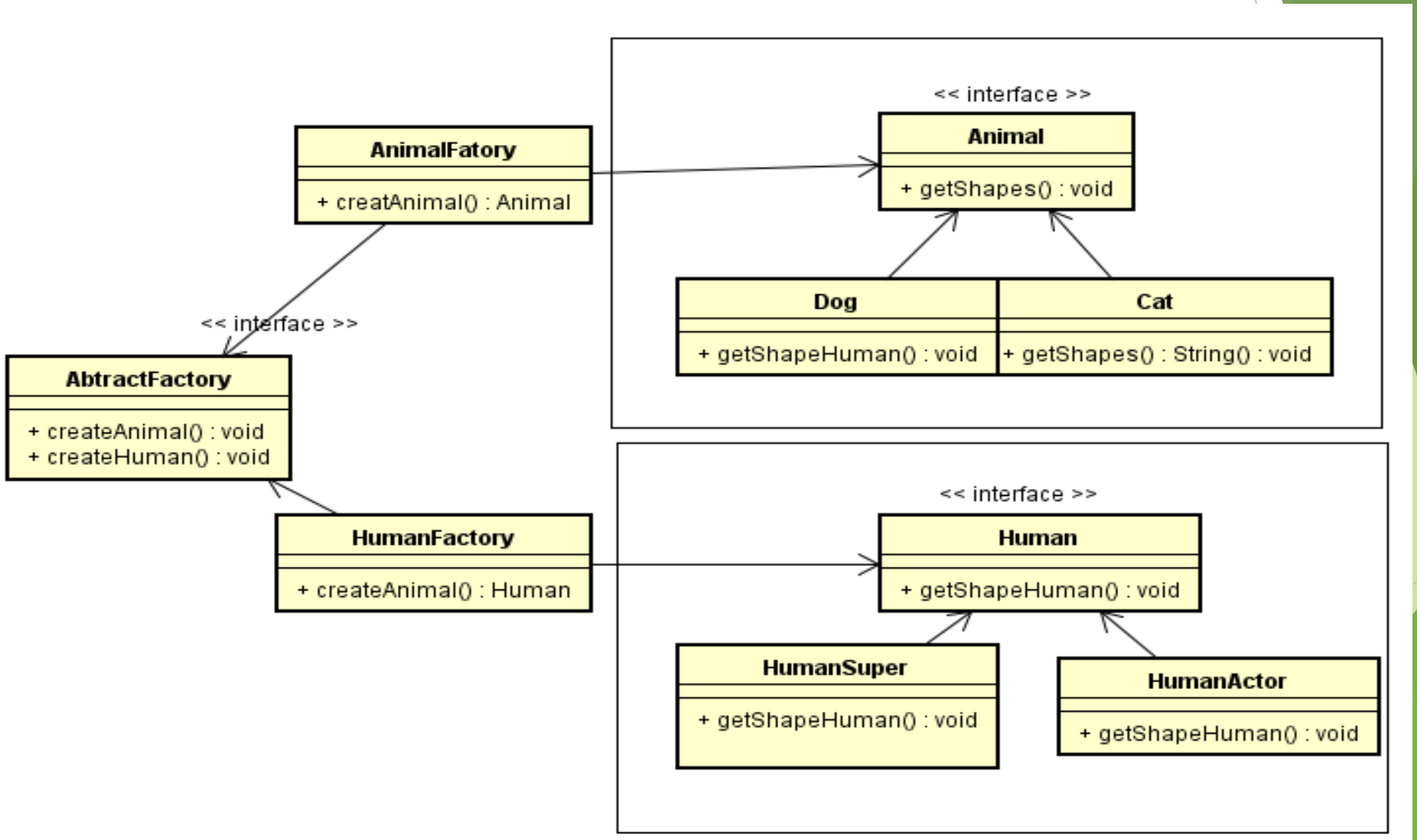
1) Creational patterns

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 :
- Builder:
- Abstract Factory:
- Prototype:
- Singleton:

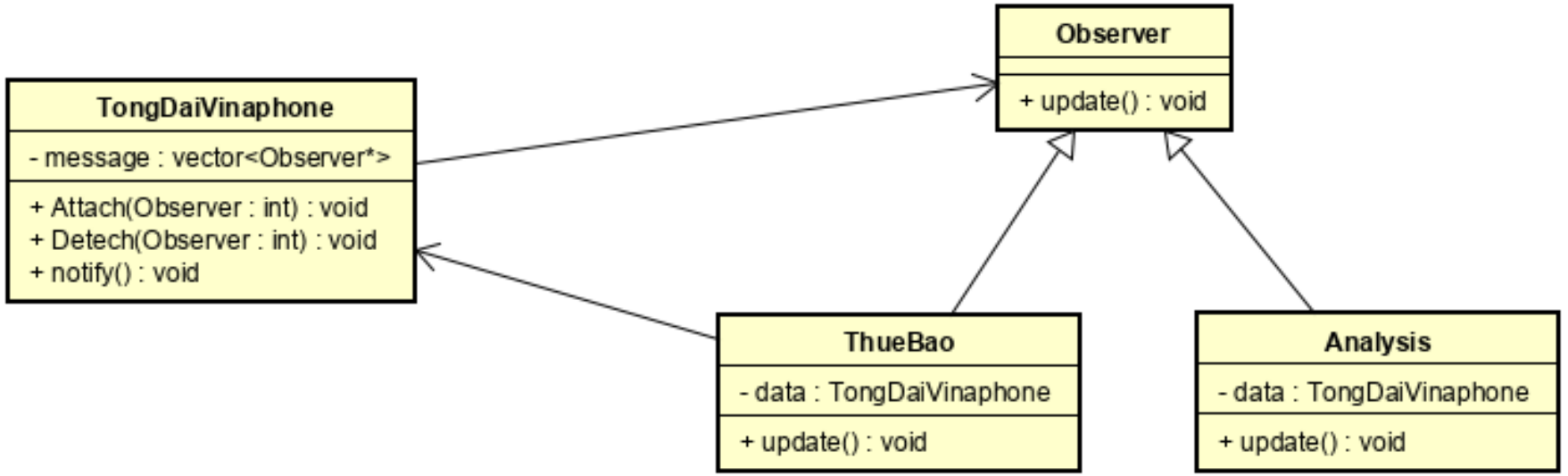


For more understand I will take the example with Abstract Factory :



Example: Observer in behavioral design patterns

➢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:



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

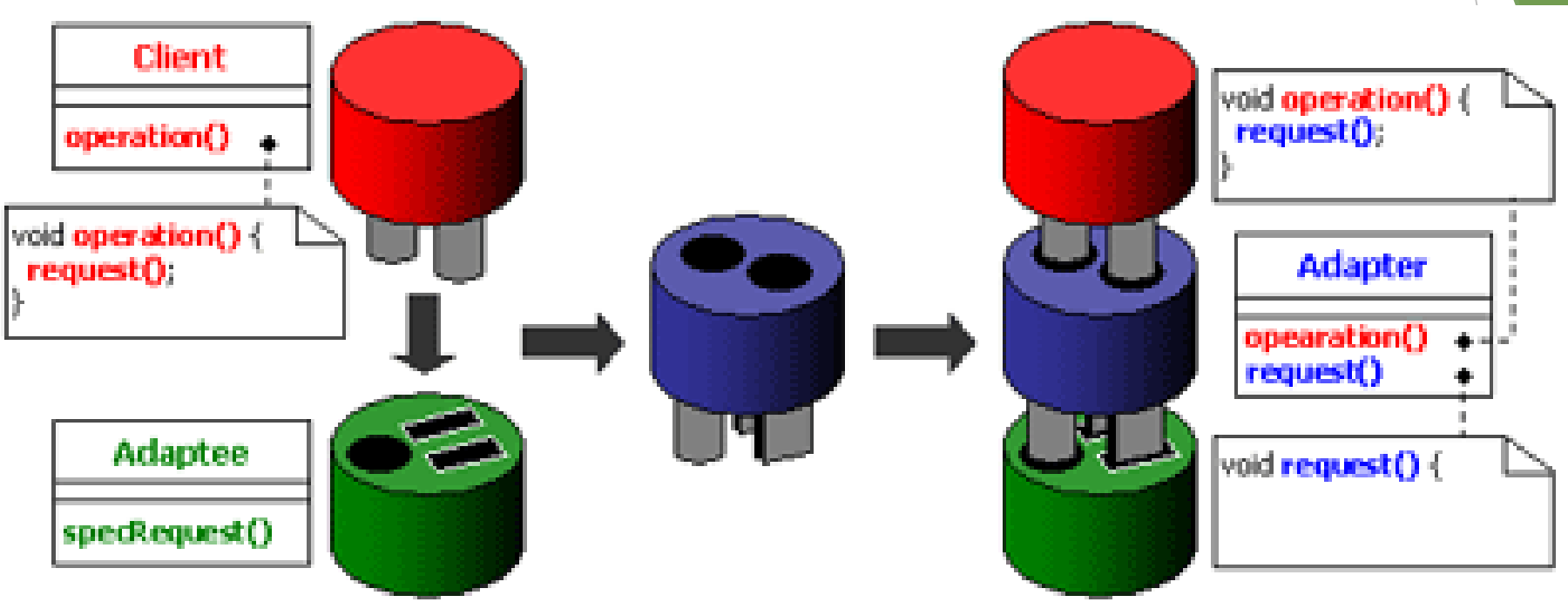
• Define a one-to-many relationship between objects so that when one object changes state, all its dependents are notified and updated automatically

• An object can notify an unlimited number of other objects

Structure

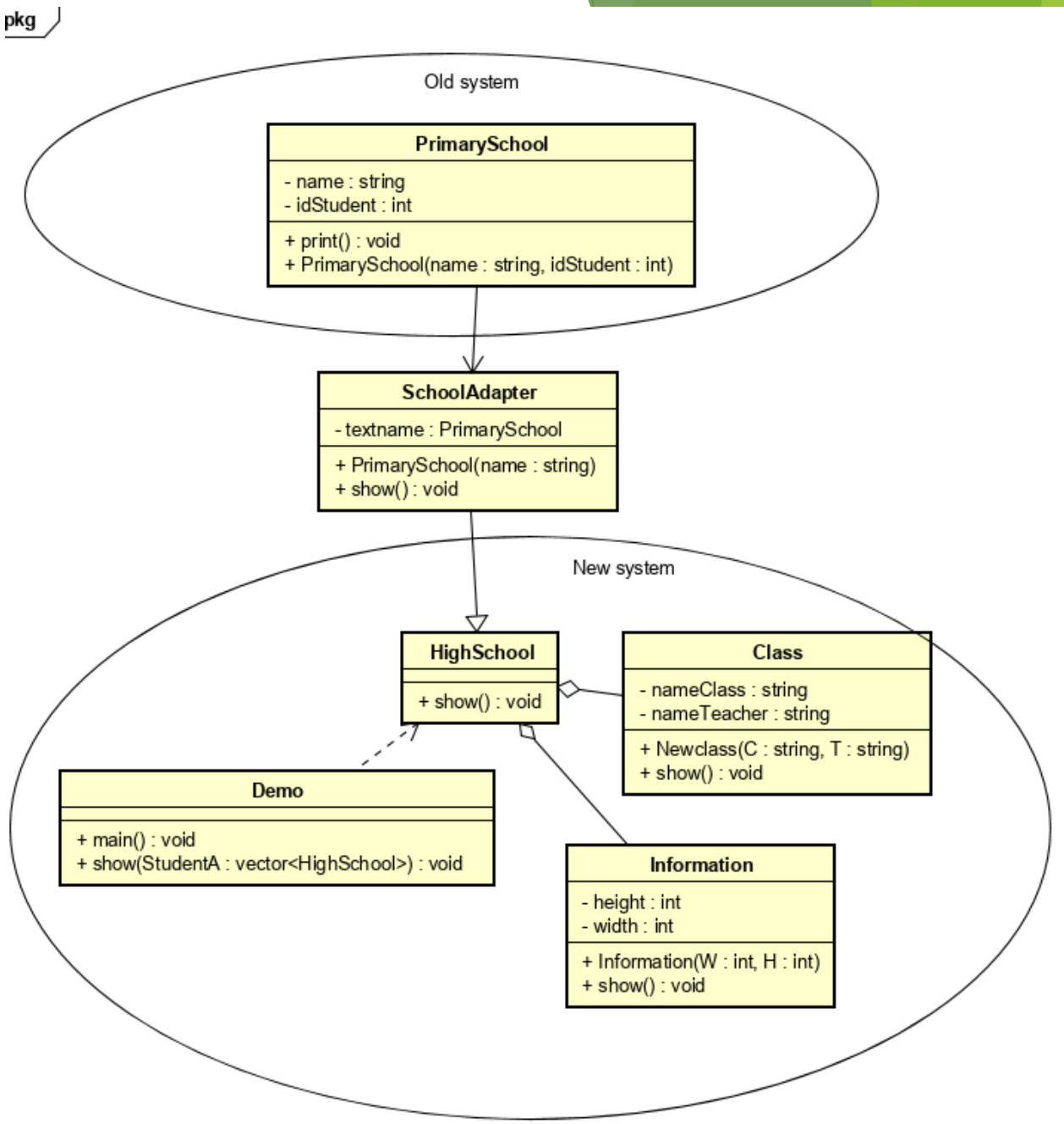
When a subject has a status change, it will browse through its list of observers and call the status update method for each observer.

3) Adapter in Structural Patterns.



Example: Adapter

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/

Thank you