Design Pattern

1st Chapter

SimUDuck Game Simulation

Case :

The programmer is asked to make Duck game simulation. The game show variety of ducks which is able to swim and make quack sound . Initial system designer creates standard OO technique by using one Duck superclass, and other specific Duck will inherits. Then, there is new requirement that the duck needs to fly above. Then it come solution again from the programmer to add new method called fly within the program. But the problem is, not all ducks can fly. So that, solution will not work. Not only that, but also not all ducks quack cause some of them squeak and mute. In order to solve that kind of problem, based on Design Pattern principal : “Identify the aspect of your application that vary and separate them from what stays the same.” Is by encapsulating vary things. The way to do that is by making interface flyable and quack in order to be implemented on required specific classes.

Class Diagram Solution :



Implementation

Class Duck as superclass has 2 method swim and display that will be implemented on all subclasses object : Mallard, Red Head, Rubber and Decoy. Flyable and quack will be implemented on each specific subclasses depends on requirement it self.

HAS A Relationship is better than IS A. means creating system using composition gives a lot more flexibility, not only encapsulate a family of algorithm into their own set of classes but also change behavior runtime. The pattern above is called STRATEGY pattern.