1. **The Strategy Pattern**: định nghĩa ra 1 family of algorithms (có thể là methods or các functions) và đóng gói chúng lại => để ta có thể hoán đổi chúng cho nhau trong lúc sử dụng tùy vào clients.
2. **Observer Pattern**: Sử dụng mqh 1-n, lấy 1 Subject làm nơi lưu trữ, chỉnh sửa data, và các Observers đăng kí với Subject đó sẽ đc update và notify thường xuyên về data đó (có thể sử dụng built-in Observable though not recommended) hoặc tự tạo mô hình riêng.
3. **Decorator pattern**: Khi muốn add thêm behaviours or functionailites cho 1 object dynamically aka chỉnh được trong runtime (chứ kp cho toàn class) có những “phụ kiện” đi kèm (như cốc coffee có toppings, player có items trên người) thì ta follow decorator pattern – build 1 object thông qua việc wrap các decorators ra bên ngoài layer by layer (hiện biết cách là thông qua constructor => wrap lại), và khi sử dụng tính toán thì sẽ được unbox đập hộp tính từ ngoài vào trong cùng.

Quan trọng:

* Decorators phải có cùng supertype với object mà chúng decorates.
* Decorators cũng có behaviours và attributes riêng.
* Decorators sẽ thay đổi behaviour components mà chúng wrap => nếu ta cần concrete component type (type object mà đc wrap) thì không nên dùng decorators.
* Should not overuse !!!

1. **Factory Method Pattern**: Defines 1 interface (or abstract class) for object creation aka calling constructors, và để subclasses – classes implement interface or abstract class đó decide việc instantiation. Pattern này giúp ta encapsulate instantiation of concrete types

Được sử dụng khi có nhiều instance của 1 super type cần đc instantiate mà ta không biết trong runtime phải khởi tạo loại nào !!!.

1. **Abstract Factory Pattern**: Provide 1 interface để tạo ra 1 families of dependent objects mà k cần specify concrete classes của chúng (Ví dụ ở trong eclipse Windows, trong package ducpad class PizzaStation)