1. **Program to an interface, not implementation** – ý là không nên hard code behaviours or methods vào 1 class nào đó như thế sẽ khó sửa và lúc sửa sẽ mất tgian Thay vào đó sử dụng interface để define trước 1 set of behaviours sẽ dung, sau đó tùy vào mục đích sử dụng mà ta implement. Ngoài ra, việc sử dụng constructor cx là instantiate concrete class => program to implementation mất rồi.
2. **Identify aspects của app mà vary, dễ thay đổi and separate them from what stays the same –** chú ý những thứ mà cần thay đổi ở runtime để nhét riêng ra 1 chỗ chứ k gộp chung !
3. **Favor composition / aggregation over inheritance –** vì composition giúp ta encapsulate các methods or algorithms vào chung set of classes, ngoài ra còn có thể change behavior at runtime
4. **Strive for loosely coupled designs between objects that interact –** Design những objects mà ít phụ thuộc vào nhau => flexible hơn
5. **Classes should be open for extension, but close for modification**
6. **Depend upon abstractions. Do not depend on concrete classes.** Tương tự như cái principle 1, ngoài ra có 1 vài guidelines giúp học theo để áp dụng principle:

* **No variable should hold a reference to a concrete class.** Chắc chỉ có class chứa hàm main là có. Còn đâu kể cả object creation cũng không cần reference, mà chỉ cần return object đó.
* **No class should derive (extends) from a concrete class.** Nếu derive thì 1 concrete class => phụ thuộc vào nó mất r, không nên => avoid as possible.
* **No class should override an implemented method of any of its base classes.** Lí do vì base classes cần phải là abstract, nếu như có implemented method thì nó nên đc áp dụng cho mọi subclasses chứ kp chỉnh.

1. **Principle of Least Knowledge:** talk only to your immediate friends. Có 1 guideline để làm theo principle này. Chỉ invoke methods mà belong to:

* **The object itself –** Những methods thuộc class của object
* **Objects passed in as a parameter**
* **Objects that the method creates or instantiates (local object)**
* **Any components (Instance variable mà là object type) of our object**

1. **Hollywood principle:** Don’t call us, we will call you !!. Đại khái là client phụ thuộc vào superclass thôi, những subclass sẽ được superclass sử dụng lúc cần chứ kp ngược lại => tăng tính decouple.