Design principles

Principle	**Description**
Keep it simple	Try to make your design simple to implement, simple to understand, simple to maintain, simple to test
Keep it flexible	It should be easy to change your design • Technology changes • Changes in the business logic
Loose coupling	If A talks to B, then A and B should have the least amount of dependency at each other. A and B can be objects, components or applications
Separation of concern	Separate • Technology from business • Stable code from changing code • Business process from application logic • Implementation from specification
Information hiding	Hide the internal details from the client.
Modularity	Divide the whole system in smaller, independent subsystems
Open Closed principle	Your design should be open for extension, but closed for change
Don't repeat yourself	Write functionality at one place and only at one place. No copy/paste of code Avoid code scattering
Program to an interface, not an implementation	If we program to an interface, our code is only dependent on that interface, instead of an implementation. This allows us to plug-in another implementation.
High cohesion, low coupling	We want a lot of interaction within an object, a component or an application, and we want very low interaction between objects, components or applications
Liskov Substitution principle	Subclasses should be substitutable for their base classes. The result of this is that you should only use inheritance in a real "IS-A" relationship
Single responsibility principle	A class has only one responsibility. There should never be more than one reason for a class to change
Interface Segregation Principle	Clients should not be forced to depend on methods they do not use
Dependency Inversion Principle	High-level modules should not depend on low-level modules. Both should depend on abstractions