TOP 10 Object-Oriented and SOLID Desing Principles for Programmer

1. DRY (Don't repeat yourself)

It's important not to abuse it, duplication is not for code, but for functionality.

2. Encapsulate What Changes

It's easy to test and maintain proper encapsulated code.

3. Open Closed Design Principle

The key benefit of this design principle is that already tried and tested code is not touched which means they won't break.

4. Single Responsibility Principle (SRP)

The key benefit of this principle is that it reduces coupling between the individual component of the software and Code.

5. Dependency Injection or Inversion Principle

The beauty of this design principle is that any class which is injected by DI framework is easy to test with the mock object and easier to maintain because object creation code is centralized in the framework and client code is not littered with that.

6. Favor Composition over Inheritance

Some of you may argue this, but I found that Composition is the lot more flexible than Inheritance.

7. Liskov Substitution Principle (LSP)

If a class has more functionality than subclass might not support some of the functionality and does violate LSP.

8. Interface Segregation Principle (ISP)

Another benefit of this design principle in Java is, the interface has the disadvantage of implementing all method before any class can use it so having single functionality means less method to implement.

9. Programming for Interface not implementation

A programmer should always program for the interface and not for implementation; this will lead to flexible code, which can work with any new implementation of the interface.

10. Delegation Principles

In order to compare two objects for equality, we ask the class itself to do comparison instead of Client class doing that check.

Dr. Kleinhirn.eu

Source: 10 Coding Principles Every Programmer Should Learn - DZone Java