SOLID

- S – Принцип единственной ответственности (Single Responsibility Principle),
- O – Принцип открытости/закрытости (Open-Closed Principle),
- L – Принцип подстановки Барбары Лисков (Liskov Substitution Principle),
- I – Принцип разделения интерфейсов (Interface Segregation Principle),
- D – Принцип инверсии зависимостей (Dependency Inversion Principle).

Принцип единственной ответственности

Принцип единой ответственности гласит, что у каждого класса должна быть только одна «ответственность» и он не должен брать на себя другие обязанности.

Принцип единственной ответственности требует от нас не добавлять дополнительные обязанности к классу, чтобы нам не приходилось менять класс, когда нам нужно изменить функционал.

У класса должна быть всего одна причина для изменения.

Принцип открытости/закрытости

Сущности программы (классы, модули, функции и т.п.) должны быть открыты для расширения, но закрыты для изменений.

Следование этому принципу гарантирует, что класс определен достаточно, чтобы делать то, что он должен делать. Добавление любых дополнительных функций может быть реализовано путем создания новых сущностей, которые расширяют возможности существующего класса и добавляют дополнительные функции самим себе.

Принцип подстановки Барбары Лисков

Объекты в программе должны быть заменяемы экземплярами их подтипов без ущерба корректности работы программы.

Этот принцип говорит нам о том, что если класс `Sub` является подтипом класса `Sup`, тогда в программе объекты типа `Sup` должны легко заменяться объектами типа `Sub` без необходимости изменения кода.

Следуя принципу подстановки Лисков, вы гарантируете, что сможете заменить экземпляры класса низкого уровня объектами класса высокого уровня без какого-либо негативного воздействия на приложение

Принцип разделения интерфейсов

Ни один клиент не должен зависеть от методов, которые он не использует.

Принцип разделения интерфейсов предполагает создание небольших интерфейсов, известных как «ролевые интерфейсы», вместо большого интерфейса, состоящего из нескольких методов. Разделяя методы по ролям на более мелкие интерфейсы, клиенты будут зависеть только от методов, которые имеют к ним отношение.

Принцип инверсии зависимостей

Принцип инверсии зависимостей гласит:

1. Модуль высокого уровня не должен зависеть от модулей низкого уровня. И то, и другое должно зависеть от абстракций.
2. Абстракции не должны зависеть от деталей реализации. Детали реализации должны зависеть от абстракций.

Чтобы следовать принципу инверсии зависимостей, нам необходимо убедиться, что класс высокого уровня
не зависит от конкретной реализации класса низкого уровня. Вместо этого он должен зависеть от некоторой абстракции.