Solid включает в себя 5 разных подпринципов:

Принцип единой ответственности

Принцип открытостей и закрытостей

Принцип подстановки

Разделение интерфейсов

Инверсия зависимости

Single Responsibility помогает разбивать общие конструкции на независимые модули, уменшать межмодульную свзяь, помогает изменять меньшее их кол-во при внесении изменений.

Open-Closed помогает проектировать модули, выполняющие только одну задачу, вводит абстракцию через которую происходит связь с сущностью, выносит место взаимодействия разных модулей в отдельную прозрачную сущность.

Liskov Substitution помогает выявлять проблемные абстракции и скрытые связи между сущностями, делает поведение модулей предсказуемым, помогает вводить ограничение на наследование(чтобы потомки не противоречили базовому поведению).

Interface Segregation помогает снизить зависимость между модулями, при наследовании нет ненужной функциональности, при внесении измненений затрагиваются только нужные части, а не все зависящие модули, помогает разделять сущности.

Dependency Inversion - верхнеуровневые сущности не должны зависеть от нижнеуровневых реализаций.

Абстрактные классы

Эти классы описывают сущности, которые не имеют в реальности конкретного воплощения

Абстрактный класс задается вот так: abstract class имя класса.

При наследовании от абстрактного класса можно быть увереным в том, что экземпляр абстрактного класса нигде не будет создан.

Абстрактные методы - это методы, которые должны быть реализованы в классах наследниках.

Полиморфизм - процесс переопределения реализации метода базового класса

Абстрактный класс может задавать интерфейс реализации дочерних классов

Чтобы в дочерних классах работали абстрактные методы основного класса, то они должны быть реализованы абстрактным методом и там