# Solid Design Principles:

1. Single Responsibility Principle: Each class should have only one responsibility or reason to change. Promotes modularity and maintainability.
2. Open/Close Principle: Open to extension, close for modification. Once a code is tested and run, it should not be modified, the code should be written as such to allow adding new features without changing existing code.
3. Liskov Substitution Principle: The subclass object type should be substitutable in place of the base class. There might be multiple methods in an interface, it is necessary to implement each one true to its concept in the implementation.
4. Interface Segregation: Interfaces should have only the minimum number of related methods inside. Because it is necessary to implement all the methods, it might not be necessary to do so in every implementation. YAGNI.
5. Dependency Inversion: Principle-
   1. High level modules should not depend on low level modules. They should depend on abstraction.
   2. Abstraction should not depend on details, but details should depend on abstraction.
   3. Abstraction means interface, abstract class.
   4. High level modules inside which there is business logic.
   5. Low level modules don’t have business logic, it might have storage like lists, arrays.