**SOLID Principles**

SOLID is a mnemonic acronym for five design principles intended to make software designs more understandable, flexible and maintainable.

The principles are a subset of many principles promoted by Robert C. Martin. Though they apply to any object-oriented design, the SOLID principles can also form a core philosophy for methodologies such as agile development or adaptive software development.

1. **Single Responsibility Principle**

A class should have only a single responsibility (i.e. changes to only one part of the software's specification should be able to affect the specification of the class).

1. **Open/closed Principle**

Software entities … should be open for extension, but closed for modification.

1. **Liskov Substitution Principle**

Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program." See also design by contract.

1. **Interface Segregation Principle**

Many client-specific interfaces are better than one general-purpose interface.

1. **Dependency Inversion Principle**

One should "depend upon abstractions, [not] concretions.

**SOLID Principles motivation:**

1. Maintainability
2. Testability
3. Flexibility and Extensibility
4. Parallel Development
5. Loose Coupling

SOLID Principles and Design Patterns plays key role in the above motivations.

# END