**SOLID**

**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).

**Open/closed principle**: "software entities … should be open for extension, but closed for modification."

**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.

**Interface segregation principle:** many client-specific interfaces are better than one general-purpose interface."

**Dependency inversion principle**: one should "depend upon abstractions, [not] concretions.

**FOUR PILLARS OF OOPS: (PIE-A)**

**Encapsulation** - Hide your modules internal data and all other implementation details/mechanism from other modules. it is also a way of restricting access to certain properties or component.

**Abstraction** - the process of exposing essential feature of an entity while hiding other irrelevant detail.

**Inheritance** -passing on property, titles, debts, rights and obligations upon. Inheritance is when a 'class' derives from an existing 'class'.

**Polymorphism** - A subclass can define its own unique behavior and still share the same functionalities or behavior of its parent/base class. Polymorphism deals with how the program decides which methods it should use, depending on what type of thing it has. ie refers to changing the behavior of a super class in the subclass.

**Cohesion:** The degree to which elements inside a class belong together.

**Coupling:** Degree of which objects depend on each other.

**Composition over Inheritances:** Classes should achieve polymorphic behavior and code reuse by their composition (aka containing instance of other lasses) rather then inheritance from a base or parent class.

**Defensive Programming:** Proactively programming against potential unforeseen circumstances.

**Domain Driver Design:** Designing software that represents the business domain. Using language that reflects the business you are encompassing.

**Interface segregation principle:** Interfaces should be specific and limited when possible.

**Open Closed Rule:** Open for Extension Closed for Modification.

**Separation of Concerns/Single Responsibility:** Each class or module addresses a single responsibility.

**Domain Specific Language**- Gerkin