Clean code:

* It should be easily accessible and understandable to others, i.e. straightforward, clear intent, good abstractions.
* It is made for the real-world, i.e. has a clear error-handling strategy
* It had minimal dependency (does one thing, has minimal dependencies)
* Variable and method name should be a meaningful name.

Solid Principle:

Solid principle includes the 5 concept,

1. **S**ingle Responsibility
2. **O**pen/Closed
3. **L**iskov Substitution
4. **I**nterface Segregation
5. **D**ependency Inversion

Single Responsibility: **a class should only have one responsibility. Furthermore, it should only have one reason to change.**

**Open/Closed: classes should be open for extension, but closed for modification.**

Liskov Substitution**:**  **if class B is a subtype of class A, then we should be able to replace A with B without disrupting the behavior of our program.**

**Interface Segregation: larger interfaces should be split into smaller ones. By doing so, we can ensure that implementing classes only need to be concerned about the methods that are of interest to them.**

**Dependency Inversion: This says, instead of high-level modules depending on low-level modules, both will depend on abstractions.**