| **Single Responsibility Principle** | A class should have **one and only one responsibility**. We should write, change or maintain **a class for only one purpose** which gives us the advantage of cleaner, robust code and fewer test cases as we know which functionality is covered in which class and which class would be modified in case of a change. |
| --- | --- |
| **Open-Closed Principle** | A class should be **open for extension but closed for modification** which essentially means we **do not want our existing code to be modified** causing potential issues or bugs . Other developers incase of some functionality change should be able to extend the class and override some methods. |
| **Liskov Substitution Principle** | **Derived classes must be substitutable for the base class**. In simple words, If class B is extending class A, we should be able to replace A with B, without disrupting the behaviour of our program. |
| **Interface Segregation Principle** | No one should be required to implement methods in their classes which they will not use. **Larger interfaces should be split into smaller ones**. This ensures that implementing classes only need to be concerned about the methods that are useful to them. |
| **Dependency Inversion Principle** | A **class should depend on abstractions** (interfaces and abstract classes) **instead of concrete implementations** (classes). The abstractions should not depend on details; instead, the details should depend on abstractions. |