**1. 단일책임의 원칙 : SRP (Single Responsibility Principle)**

- **하나의 클래스는 하나의 책임만 가지도록 해야한다.**

- 어떤 변화에 의해 클래스를 변경해야 하는 이유는 오직 하나뿐이어야 함을 의미한다.

- 즉, 각 클래스는 다른 클래스에 대해 의존성이 낮아야 한다.

**2. 개방폐쇄의 원칙 : OCP (Open Closed Principle)**

- **클래스는 높은 응집도와 낮은 결합도를 가져야 한다.**

- 응집도가 높다는건 하나의 모듈, 클래스가 하나의 책임 또는 관심사에만 집중되어 있다는 뜻이다.

- 같은 책임, 관심사를 기반으로 하나의 객체로 설계하기 때문에 객체의 변경이 발생하더라도 다른 곳에 미치는 영향이 제한적이다.

- 책임과 관심사가 다른 객체 또는 모듈과는 낮은 결합도를 유지해야한다. 이는 높은 응집도보다 더 민감한 원칙이다.

- 결합도란 하나의 오브젝트가 변경이 일어날 때 관계를 맺고 있는 다른 오브젝트에게 변화를 요구하는 정도이다.

- **객체를 한번 생성을 했을 때, 구성요소(컴포넌트, 클래스, 모듈, 함수)는 확장에 열려있고, 변경에는 닫혀있어야 한다.**

**3. 리스코프 치환 원칙 : LSP (Liskov Substitution Principle)**

**- 객체는 프로그램의 정확성을 깨지 않으면서 하위 타입의 인스턴스로 바꿀 수 있어야 한다.**

**- 하위 클래스는 인터페이스 규약을 지켜서 작성되어야 한다.**

- 서브 타입은 언제나 기반 타입으로 교체할 수 있어야 한다. 즉, 서브 타입은 언제나 기반 타입과 호환 될 수 있어야 한다.

**- 상속을 통한 재사용은 기반 클래스와 서브 클래스 사이에 IS-A관계가 있을 경우로만 제한되어야 한다.**

- 클래스의 상속, 다형성과 관련된 원칙이다.

**4. 인터페이스 분리 원칙 : ISP (Interface Segregation Principle)**

**- 범용 인터페이스 하나보다는 특정 클라이언트를 위한 여러 개의 인터페이스 분리가 더 좋다.**

- SRP가 클래스의 단일책임을 강조한다면 ISP는 인터페이스의 단일책임을 강조한다.

- 하지만 ISP는 어떤 클래스 혹은 인터페이스가 여러 책임 혹은 역할을 갖는 것을 인정한다.

- 인터페이스는 교육정책과 비교하면 이해하기 수월, **인터페이스를 분리 할 때는 절대 인터페이스가 변경되면 안 된다는 전제조건이 깔려야 한다.**

**5. 의존관계 역전 원칙 : DIP (Dependency Inversion Principle)**

- 프로그래머는 구체화가 아니라 추상화에 의존해야 한다고 한다. 즉, **구현 클래스(구현체)가 아니라 인터페이스(역할)에 의존해야 한다.**

- 변수에 구상 클래스의 레퍼런스를 저장하지 말아야 한다.

- 구상 클래스에서 유도된 클래스를 만들지 말아야 한다.

- 베이스 클래스에 이미 구현되어 있는 메소드는 되도록이면 오버라이딩 하지 말자.

**- \*훅 메소드 (슈퍼클래스에서 디폴트 기능을 정의해두거나 비워뒀다가 서브클래스에서 선택적으로 오버라이드 할 수 있도록 만들어둔 메소드를 훅(hook) 메소드라고 한다.)**

- 구조적 디자인에서 발생하던 **하위 레벨 모듈의 변경이 상위 레벨 모듈의 변경을 요구하는 위계관계를 끊는 의미의 역전**이다.