**Los principios SOLID**

Para el desarrollo y diseño de aplicaciones, PRINCIPIOS SOLID es uno de los fundamentos de la arquitectura y desarrollo de software.

## 1. Principio de Responsabilidad Única

“Una clase debería tener una y solo una, para cambiar”

Si una clase tiene varias responsabilidades esto quiere decir que el cambio de una responsabilidad afectara o modificara a otra responsabilidad de esa clase.

## 2. Principio de Abierto/Cerrado

"Debería poder extender el comportamiento de una clase, sin modificarlo".

Establece que las entidades de software como ser las clases, módulos y funciones tendrían que estar abiertos para su extensión, pero tendrían que estar cerrados para su modificación

## 3. Principio de Sustitución de Liskov

## “Las clases derivadas deben ser sustituibles por sus clases base”.

## Declara que una subclase debe ser sustituible por su superclase, y si al hacer esto, el programa falla, estaremos violando este principio.

## 4. Principio de Segregación de la Interfaz

## “Haz interfaces que sean específicas para un tipo de cliente”, es decir, para **una finalidad concreta**.

## Este principio establece que los clientes no deberían verse forzados a depender de interfaces que no usan.

## 5. Principio de Inversión de Dependencias

## “**Depende de abstracciones**, no de clases concretas”.

## En algún momento nuestro programa o aplicación llegará a estar formado por muchos módulos. Cuando esto pase, es cuando debemos usar inyección de dependencias, lo que nos permitirá controlar las funcionalidades desde un sitio concreto en vez de tenerlas esparcidas por todo el programa.

## Conclusión

## Seguir estos 5 pasos puede que sea tedioso, pero vayamos echándole practica nos ayudara a la hora de programar, nuestro código será más accesible para nosotros o para otros programadores ya que nuestro código tendrá una estructura clara y definida.

## Extends

## Es una palabra reservada que indica a la clase hija cual va ser su padre por ejemplo tenemos la clase padre Estudiante y la clase hija Persona y al poner:

## “public class Estudiante extends Persona”

Con esto estamos diciendo a la clase Estudiante que copie todos los atributos y métodos ya sean públicos o protegidos de la clase Persona aquí viene la definición de herencia