Temario

Unidad 4

a)Herencia y Polimorfismo

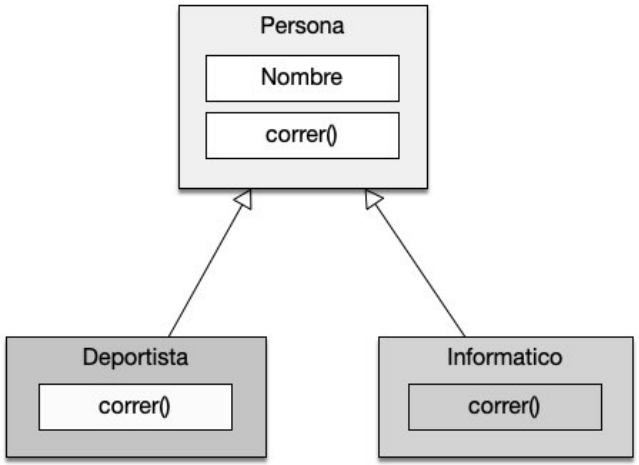
b)Principios básicos de diseño Orientado a Objetos

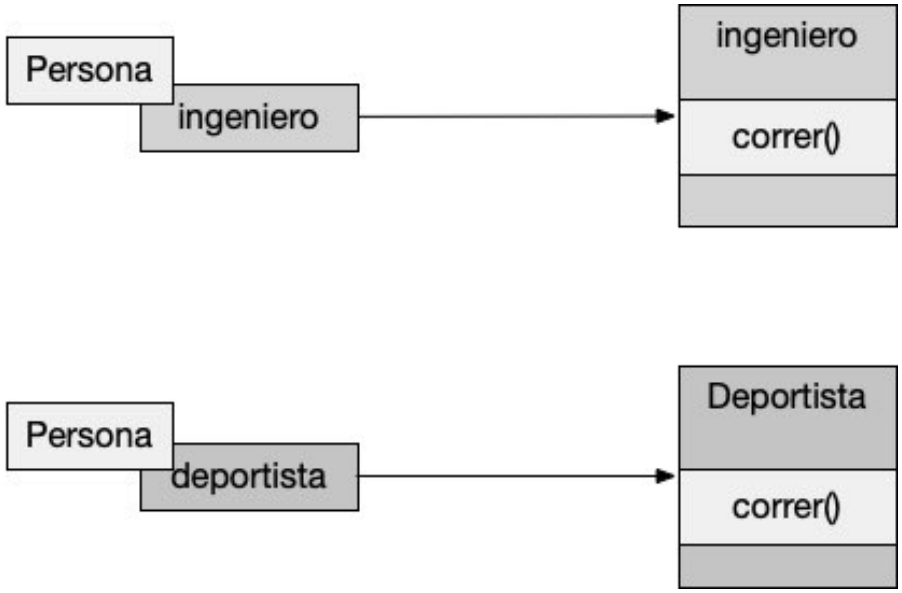
Unidad 4

Herencia y Polimorfismo

Herencia	Polimorfismo
Las clases principales extienden atributos y comportamientos a las clases secundarias. A través de la definición en una clase de los atributos y comportamientos básicos, se pueden crear clases secundarias, ampliando así la funcionalidad de la clase principal y agregando atributos y comportamientos adicionales.	El polimorfismo consiste en diseñar objetos para compartir comportamientos, lo que nos permite procesar objetos de diferentes maneras. Es la capacidad de presentar la misma interfaz para diferentes formas subyacentes o tipos de datos.

Unidad 4





El método correr de la clase Persona es un método abstracto y no tiene implementación . Por el contrario los métodos de la clases hijas tienen sobrecargado el método correr. El deportista correrá a 7 hm/hora y el informático a 2km/h .

Unidad 4

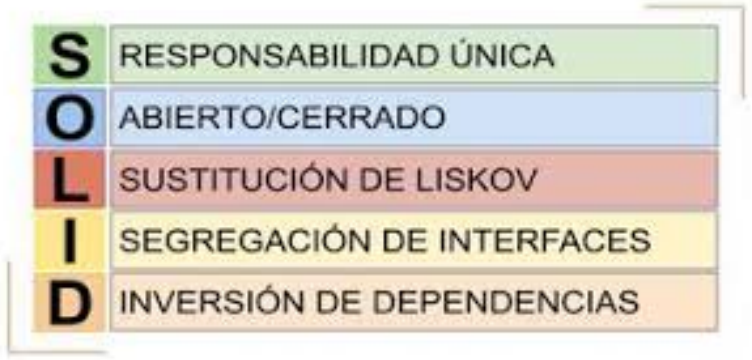
Principios básicos de diseño Orientado a Objetos

Introducción a los principios SOLID

SOLID es una palabra que debes conocer como uno de los fundamentos de la arquitectura y desarrollo de software.

SOLID es el acrónimo que acuñó Michael Feathers, basándose en los principios de la programación orientada a objetos que Robert C. Martin había recopilado en el año 2000 en su paper "Design Principles and Design Patterns". Ocho años más tarde, el tío Bob siguió compendiando consejos y buenas prácticas de desarrollo y se convirtió en el padre del código limpio con su célebre libro Clean Code.

Los 5 principios SOLID de diseño de aplicaciones de software son:
S – Single Responsibility Principle (SRP)
O – Open/Closed Principle (OCP)
L – Liskov Substitution Principle (LSP)
I – Interface Segregation Principle (ISP)
D – Dependency Inversion Principle (DIP)



Entre los objetivos de tener en cuenta estos 5 principios a la hora de escribir código encontramos:

- Crear un software eficaz: que cumpla con su cometido y que sea robusto y estable.
- Escribir un código limpio y flexible ante los cambios: que se pueda modificar fácilmente según necesidad, que sea reutilizable y mantenible.
- Permitir escalabilidad: que acepte ser ampliado con nuevas funcionalidades de manera ágil.

Material complementario de la unidad

Link a video relacionado

Principios SOLID de Programación Orientada a Objetos

https://youtu.be/5flyAn9IvAU

Link a lectura complementaria

Principios fundamentales de la Programación Orientada a Objetos

https://desarrolloweb.com/manuales/programacion-orientada-objetos-dotnet.html