The **SOLID Principles** are five principles of Object-Oriented class design. They are a set of rules and best practices to follow while designing a class structure.

These five principles help us understand the need for certain design patterns and software architecture in general. So I believe that it is a topic that every developer should learn.

This article will teach you everything you need to know to apply SOLID principles to your projects.

We will start by taking a look into the history of this term. Then we are going to get into the nitty-gritty details – the why's and how's of each principle – by creating a class design and improving it step by step.

So grab a cup of coffee or tea and let's jump right in!

Background

The SOLID principles were first introduced by the famous Computer Scientist Robert J. Martin (a.k.a Uncle Bob) in his paper in 2000. But the SOLID acronym was introduced later by Michael Feathers.
Uncle Bob is also the author of bestselling books *Clean Code* and *Clean Architecture*, and is one of the participants of the "Agile Alliance".
Therefore, it is not a surprise that all these concepts of clean coding, object-oriented architecture, and design patterns are somehow connected and complementary to each other.

They all serve the same purpose:

"To create understandable, readable, and testable code that many developers can collaboratively work on."

Let's look at each principle one by one. Following the SOLID acronym, they are:

- The **S**ingle Responsibility Principle
- The **O**pen-Closed Principle
- The **L**iskov Substitution Principle
- The **I**nterface Segregation Principle
- The **D**ependency Inversion Principle