УДК 004.41
Ланевич Т. – аспірант
Тернопільський національний технічний університет імені Івана Пулюя

ПРИНЦИПИ SOLID У РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Науковий керівник: канд. техн. наук., доц. Боднарчук І.О.

Lanevych T. – postgraduate
Ternopil Ivan Puluj National Technical University

SOLID PRINCIPLES IN SOFTWARE DEVELOPMENT

Supervisor: Assoc. Prof. Dr. Bodnarchuk I.

Ключові слова: SOLID, клас, розробка, архітектура, дизайн.
Key words: SOLID, class, development, architecture, design.

У розробці програмного забезпечення, об'єктно-орієнтоване проєктування (OOD) відіграє вирішальну роль, коли йдеться про написання гнучкого, масштабованого, підтримуваного та повторно використовуваного коду. Існує багато переваг використання OOD, але кожен розробник повинен також знати принцип SOLID для хорошого об'єктно-орієнтованого проєктування в програмуванні. Принципи SOLID були представлені Робертом К. Мартіном та і є стандартом кодування в програмуванні [4].

Принципи SOLID визначають, як групувати функції і структури даних у класи, а також як ці класи мають взаємодіяти між собою. Використання слова «клас» не означає, що ці принципи застосовуються лише до об'єктно-орієнтованого програмного коду. В цьому випадку «клас» є лише інструментом для об'єднання функцій та даних. Будь-яка програмна система має такі об'єднання, незалежно від того, як вони називаються – «клас» чи якось інакше. До цих об'єднань застосовуються принципи SOLID [1].

Метою принципів є створення програмних структур середнього рівня, які будуть:

1. Толерантними до змін.
2. Простими та зрозумілими.
3. Основою для компонентів, придатних для використання у різних програмних системах.

Термін «середній рівень» відображає той факт, що розробники застосовують ці принципи на рівні модулів, тобто на рівні, який розташовується безпосередньо вище рівня програмного коду. Принципи допомагають визначити програмні структури, що використовуються в модулях і компонентах [1].

SOLID – це абревіатура, яка означає :

1. SRP – принцип єдиної відповідальності. Клас повинен мати одну і тільки одну причину для зміни, тобто клас повинен мати тільки одне завдання.
2. OCP – принцип відкритості/закритості. Об'єкти або сутності повинні бути відкритими для розширення, але закритими для модифікації [2].

3. LSP – принцип підстановки Барбари Лісков. Для створення програмних систем із взаємозамінних частин, ці частини мають відповідати контракту, який дозволяє робити подібну заміну.

4. ISP – Принцип поділу інтерфейсів. Розробники повинні уникати залежності від усього, що не використовується [1].

5. DIP – Принцип інверсії залежності. Сутності повинні залежати від абстракцій, а не від конкретики. Це означає, що високорівневий модуль не повинен залежати від низькорівневого модуля, але вони повинні залежати від абстракцій [2].

Ці принципи допомагають розробникам програмного забезпечення створювати надійні, тестовані, розширювані та підтримувані об'єктно-орієнтовані програмні системи.

Література

1. Мартін Р. Чиста архітектура / Роберт Мартін., 2019. – 368 с.

2. SOLID: The First 5 Principles of Object Oriented Design [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design.

3. SOLID Design Principles in Software Development [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/solid-design-principles-in-software-development/.

4. Why SOLID principles are still the foundation for modern software architecture [Електронний ресурс] – Режим доступу до ресурсу: https://stackoverflow.blog/2021/11/01/why-solid-principles-are-still-the-foundation-for-modern-software-architecture/