УДК 004.415.532

Козлик С.О., здобувач,
Концидайло А.М., ст. викладач
Державний університет «Житомирська політехніка»

ЗАСТОСУВАННЯ ПРИНЦИПІВ F.I.R.S.T. У НАПИСАННІ UNIT-ТЕСТІВ

Тестування є критичним етапом при розробці програмного забезпечення, проходження якого гарантує, що продукт відповідає бажаним вимогам та функціонує належним чином. Під час тестування використовуються різні методи та стратегії, кожна з яких має свої особливості при застосуванні. Однією з таких стратегій є піраміда тестування, яка наголошує на побудові міцного фундаменту з unit-тестів, або ж модульних тестів, які доповнюються меншою кількістю інтеграційних та E2E тестів.

Unit-тести – це автоматичні тести, які перевіряють невеликі, ізольовані частини коду, так звані модулі: методи, функції тощо. Вони дозволяють виявляти дефекти на ранніх етапах розробки, а тому спрощують процес рефакторингу. Для їх якісного і легкого написання та ефективної перевірки програмного забезпечення необхідно дотримуватися визначених чітких правил, принципів. Одними з найбільш відомих є принципи F.I.R.S.T. [2], запропоновані Робертом Мартіном. Вони включають п’ять правил, які утворюють вищезгадану абревіатуру: Fast, Independent, Repeatable, Self-Validating, Timely.

Перший принцип Fast стверджує, що тести повинні виконуватися швидко, щоб не сповільнювати розробку. Тести, які виконуються занадто довго, перешкоджають частому запусканню в процесі розробки, що зменшує їх практичну цінність. Unit-тести використовуються для швидкої перевірки написаного функціоналу, раннього виявлення можливих помилок та їх подальшого виправлення. Ізоляція тестованого коду від зовнішніх ресурсів пришвидшує виконання тестів, для досягнення цього використовуються mock-об’єкти замість реальних сервісів, in-memory бази даних, мінімізуються I/O операції.

Independent – тести не повинні залежати один від одного, тобто вони мусять виконуватися самостійно, не створювати умови для наступних тестів та запускатися у будь-якому порядку. Це дозволить виявити усі дефекти у тестованих модулях без приховувань помилок за першим же непройденим тестом. На практиці Independent реалізується завдяки ініціалізації тестових даних, відсутності спільного змінюваного стану та використанню патерну Arrange–Act–Assert.

Repeatable означає, що тести повинні давати однаковий результат при будь-якому запуску та будь-якому середовищі. Кожен тест має самостійно ініціалізовувати необхідні дані та не залежати від жодних зовнішніх факторів для його проведення. Наприклад, якщо в методі зустрічається генерація псевдовипадкових чисел, значення яких є динамічними, то для коректного тестування потрібно фіксувати параметр seed для генератора, щоб послідовність чисел була завжди однаковою при кожному запуску. Це полегшує виявлення і виправлення помилок, оскільки їх легко можна відтворити завдяки повторюваності результатів.

Четвертий принцип Self-Validating вказує на те, що тести повинні мати логічний вивід, тобто самостійно перевіряти правильність роботи коду без необхідності ручного втручання для додаткового аналізу результатів. Завдяки цьому можна зекономити час та зусилля, витрачені на тестування, та уникнути людської помилки при інтерпретації результатів. Найкращим практиками в даному випадку є уникання кількох стверджень (asserts) в одному тесті та використання інформативних назв тестових сценаріїв.

Timely – тести повинні бути написані своєчасно, приблизно в той самий час, що й тестований код, а з підходом TDD – навіть до розробки функціональності, щоб мінімізувати появу помилок та полегшити рефакторинг. Цей принцип реалізується через раннє написання тестів, їх регулярне виконання та рефакторинг коду. Варто зазначити, що в джерелах, відмінних від праць Роберта Мартіна, може зустрічатися альтернативне пояснення літери T: Thorough [1]. У такому трактуванні підкреслюється необхідність повного покриття логіки функціональності з урахуванням усіх «щасливих шляхів» виконання, граничних випадків, можливих помилок та негативних сценаріїв. Хоча лише Timely відповідає первинному формулюванню концепції F.I.R.S.T., при написанні тестів можна дотримуватися обох інтерпретацій цього принципу, оскільки вони спрямовані, перш за все, на підвищення якості тестування та не суперечать одна одній.

Принципи F.I.R.S.T. формують стандарти написання ефективних unit-тестів та є рекомендованою практикою для команд, які прагнуть високої якості розроблюваного продукту.

Список використаних джерел:

1. FIRST Principles as Solid Rules for Tests. DZone. URL: https://dzone.com/articles/first-principles-solid-rules-for-tests (дата звернення: 10.03.2026).

2. Martin R. Clean Code: A Handbook of Agile Software Craftsmanship. Pearson Education, Limited, 2008. 464 с.