Київський національний університет імені Тараса Шевченка

Факультет комп'ютерних наук та кібернетики

Звіт

до лабораторної роботи

з курсу

«Інструментальні засоби розробки програмного забезпечення»

на тему

# «Практичне застосування системи контролю версій Git та юніт-тестування у процесі розробки програмного забезпечення. Формування повного робочого циклу з GitHub — від створення репозиторію до Pull Request.»

Виконала студентка групи ІПС-21

факультету комп'ютерних наук та кібернетики

Яницька Марія Андріївна

Київ-2025

**Мета роботи:**

Отримати практичні навички використання сучасних інструментів розробки ПЗ, а саме:

- роботи з системою контролю версій Git та сервісом GitHub;
- побудови історії комітів і гілок проєкту;
- написання юніт-тестів до існуючого коду;
- формування правильного процесу створення Pull Request та командної взаємодії у GitHub.

**Короткий теоретичний вступ:**

- **Система контролю версій Git** — це розподілена система, що дозволяє розробникам відстежувати зміни у файлах проєкту, повертатися до попередніх версій, створювати ізольовані гілки для розробки нових функцій та ефективно співпрацювати у команді.
- **GitHub** — це веб-платформа для хостингу Git-репозиторіїв, яка надає інструменти для управління проєктом, відстеження завдань, проведення код-рев'ю за допомогою Pull Request та автоматизації процесів.
- **Юніт-тестування** — це процес написання коду, який перевіряє коректність роботи окремих компонентів програми (функцій, методів, класів) в ізоляції від решти системи. Для написання тестів у середовищі .NET було використано фреймворк **NUnit**.

## 2. Посилання на репозиторій GitHub

Робочий репозиторій з вихідним кодом, історією розробки та фінальним Pull Request доступний за посиланням: https://github.com/Mariia11212/IZR

## 3. Опис основних етапів роботи

### 3.1. Створення гілки для розробки

Робота над додаванням юніт-тестів розпочалася у спеціально створеній для цього гілці feature/unit-tests. Це було зроблено для ізоляції нових змін від стабільної основної гілки main, що є стандартною та рекомендованою практикою при роботі з Git. Такий підхід дозволяє безпечно вести розробку, не впливаючи на основний код проєкту.

### 3.2.Ведення історії комітів

Історія комітів у гілці feature/unit-tests велася послідовно та логічно. Кожен коміт представляв завершений етап роботи: додавання тестів для окремого модуля, рефакторинг коду, виправлення помилок або вдосконалення тестів згідно з рекомендаціями. Повідомлення до комітів були інформативними та стислими, наприклад: *“Add initial tests for Vector and Point classes”*, *“Refactor Rectangle tests using SetUp method”*, *“Add edge case tests for MyDynamicArrayBasedList”*.

### 3.3. Створення та обговорення Pull Request

Після завершення розробки тестів було створено Pull Request (PR) для злиття гілки feature/unit-tests до main. В описі PR було детально зазначено, які модулі були покриті тестами та які вдосконалення було внесено. В рамках код-рев'ю були надані зауваження, які стосувалися дублювання коду, відсутності перевірок на граничні випадки та стилю іменування. Усі зауваження були виправлені у наступних комітах, які автоматично додалися до цього ж Pull Request, після чого він був успішно схвалений та злитий.

## 4. Опис процесу написання юніт-тестів

В рамках роботи було створено комплексний набір юніт-тестів для перевірки коректності реалізованих модулів: MyDynamicArrayBasedList, Vector, Point, Rectangle та Circle.

Початковий етап включав написання базових тестів для перевірки основної функціональності: коректність роботи конструкторів, обчислення площі та периметра, а також базових операцій зі списком.

Після отримання зворотного зв'язку в рамках Pull Request, тестовий код було значно покращено:

- **Усунення дублювання коду**: Повторювана ініціалізація об'єктів була винесена у метод [SetUp].
- **Оптимізація тестів**: Схожі сценарії перевірки винятків об'єднано за допомогою атрибута [TestCase].
- **Розширення тестового покриття**: Було додано тести для важливих граничних випадків, таких як вставка/видалення елементів на початку/кінці списку та коректна поведінка ітератора (foreach) на порожньому списку.
- **Гарантія незмінності (Immutability)**: Для методів Scale та Translate додано перевірку, що вони повертають новий екземпляр об'єкта.
- **Надійність обробки даних**: Додано тести, що перевіряють коректну обробку невалідних (NaN, Infinity) або null трансформацій.
- **Стандартизація коду**: Точність порівняння чисел з плаваючою комою була стандартизована за допомогою константи (Eps = 1e-3).
-

## 5. Висновки

Під час виконання лабораторної роботи я отримала практичні навички повного циклу розробки з використанням сучасних інструментальних засобів.

**Набуті навички:**

1. **Робота з Git та GitHub:** Я навчилася впевнено працювати з гілками, створювати логічні коміти та проходити повний цикл командної

взаємодії через Pull Request, включаючи обробку зауважень та внесення правок.

2. **Юніт-тестування з NUnit:** Я освоїла принципи написання юніт-тестів, навчилася покривати тестами не лише базову логіку, а й складні граничні випадки, що значно підвищує якість та надійність коду.
3. **Культура розробки:** Робота продемонструвала важливість чистого, читабельного коду та структурованого підходу до розробки. Процес код-рев'ю виявився ефективним інструментом для покращення якості коду.

Ця робота дозволила на практиці поєднати теоретичні знання з контролю версій та тестування, сформувавши цілісне розуміння сучасного процесу розробки програмного забезпечення.