

# Лабораторна робота 6

Проектування інформаційних систем

lakub.muravlov@gmail.com [Switch accounts](#)



Draft saved

The name and photo associated with your Google Account will be recorded when you upload files and submit this form. Your email address is not part of your response.

**\*Required**

Ім'я \*

Андрій

Прізвище \*

Муравльов

Група \*



ДА-81



ДА-82

Я не списую з будь який джерел, не користуюся лабораторними роботами своїх колег, не роблю копії відповідей з інших лабораторних робіт і т.д. \*



Так



Мета роботи: оволодіти навичками створення програмного забезпечення за методологією TDD та ознайомитися з процедурами рефакторинга.

Задача: 1. Використовувати методологію Test Driven Development для створення класів архітектурної програмної моделі. 2. Скласти тестові сценарії, які продемонструють функціонування всіх методів проекрованої моделі. 3. Виконати юніт-тестування складових частин (внутрішніх класів), що реалізують об'єктмоделювання. 4. Виконати "зовнішнє" юніт-тестування для API. 5. Провести рефакторинг коду програми, для поліпшення реалізації.

Короткі теоретичні відомості. Розробка через тестування (англ. test-Driven Development, TDD) - техніка розробки програмного забезпечення, яка ґрунтується на повторенні дуже коротких циклів розробки: спочатку пишеться тест, що покриває бажану зміну, потім пишеться код, який дозволить пройти тест, і під кінець проводиться рефакторинг нового коду до відповідних стандартів. Рефакторинг (англ. refactoring) - процес зміни внутрішньої структури програми, що не зачіпає її зовнішньої поведінки і має на меті полегшити розуміння її роботи. В основі рефакторинга лежить послідовність невеликих еквівалентних (тобто таких, що зберігають поведінку) перетворень. Оскільки кожне перетворення маленьке, програмісту легше простежити за його правильністю, і в той же час вся послідовність може привести до істотної перебудови програми і поліпшенню її узгодженості і чіткості. Модульне тестування, або юніт-тестування (англ. unit testing) - процес у програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми. Ідея полягає в тому, щоб писати тести для кожної нетривіальної функції або методу. Це дозволяє досить швидко перевірити, чи не призвело чергову зміну коду до регресії, тобто до появи помилок увже відтестованих місцях програми, а також полегшує виявлення та усунення таких помилок.



## Test Driven Development

Test driven development (TD...



## Code Refactoring

Code Refactoring



## Refactoring

Code Refactoring 2



Опишіть, у чому полягає методика модульного тестування ? \*

Модульное тестирование - процесс в программировании, который позволяет проверить корректность работы отдельных модулей проекта. Основная идея состоит в том чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить не привело ли очередное изменение кода к появлению ошибок в уже оттестированных участках программы и также позволяет упростить обнаружение и устранение таких ошибок

Опишіть, у чому полягає методика тестування програмного продукту відповідно до ISO/IEC/IEEE 29119 або IEEE\_829 \*

IEEE829 - стандарт для документации по тестированию ПО, является стандартом IEEE, определяющим форму набора документов на определенных этапах тестирования ПО, каждый этап потенциально создает отдельный тип документа.

По IEEE829 присутствуют следующие этапы:

- идентификатор плана тестирования
- введение
- элементы тестирования
- Тестируемые функции
- функции которые не нужно или невозможно оттестировать
- подход
- критерии прохождения задания
- критерии приостановки и возобновления
- результаты тестирования
- задачи тестирования
- требования к среде разработки
- обязанности
- потребности в персонале и обучении
- график
- риски и непредвиденные обстоятельства
- утверждения



Опишіть основні види асертів для використання у модульних тестах. \*

Асерт - конструкция позволяющая проверять предположения о значениях произвольных данных в произвольном месте программы. Эта конструкция автоматически сигнализирует об обнаружении некорректных данных

Асерты можно разбить на следующие классы:



- те которые проверяют входящие аргументы в начале функции
- те которые проверяют данные с которыми работает функция внутри кода функции
- те которые проверяют что функция возвращает

Все эти классы асеров могут быть разными, например:

- проверка на значение
  - проверка на правдивость
  - проверка на равенство
  - проверка на строку
  - проверка на содержание элемента в итерируемых типах
  - проверка на exception
  - проверка колбека
  - проверка респонс/реджекта
- и тд.

Додайте файл з кодом програми для вашої інформаційної системи відповідно до методики TDD. Опишіть 3 ітерації (тест-код, тест-код, тест-код) розробки коду відповідно до TDD.

Приєднати файл з кодом програми та юніт тестами по TDD. \*

 unit-test.txt 



Які основні види рефакторингу існують? \*

- изменение сигнатуры метода
- инкапсуляция поля
- выделение класса
- выделение интерфейса
- выделение локальной переменной
- выделение метода
- генерализация типа
- встраивание
- введение фабрики
- введение параметра
- подъем метода
- спуск метода
- переименование метода
- перемещение метода
- замена оператора на полиморфизм
- замена наследования делегированием
- замена кода типа подклассами

Додайте файл з описом коду до та після рефакторингу. Застосуйте 5-6 видів рефакторингу. \*

 refactored.txt ✕

### Контрольні питання

Чому методологія TDD є більш ефективною та економить час при тестуванні? \*

TDD сразу берет в учет граничные случаи модулей и функций, и уже исходя от него пишется код, т.е. не будет тратиться время на отлаживание и починку ошибки на этапах разработки/продакшна



Опишіть цикл розробки програмного забезпечення за допомогою методології TDD? \*

- 1) Добавление теста
- 2) Убедиться что новые тесты не проходят
- 3) Написать код который покрывает граничные случаи теста
- 4) Запуск всех тестов, убедиться что все тесты проходят
- 5) Рефакторинг
- 6) Повтор цикла

Яким чином забезпечується покриття коду та покриття тестами в методології TDD? \*

- подготовка тестовых данных. Это могут быть входящие параметры, глобальное состояние (например, записи в БД).
- вызов метод с нужными параметрами
- проверка, что произошло то, что ожидается в этом сценарии. Тут речь идет как о результате метода, так и о побочных эффектах, например, изменении состояния в БД.

Чи є розробка нового функціоналу програми рефакторингом? \*

По определению, рефакторинг это процесс изменения внутренней структуры программы, не затрагивающий ее внешнего поведения и имеющий цель облегчить понимание ее работы и скейлинг проекта. Само внедрение нового функционала не является рефакторингом, однако рефакторинг проводится зачастую именно с целью внедрения нового функционала и облегчения работы с ним.



Чим відрізняється рефакторинг від реінжинірингу? \*

Рефакторинг - улчшает имеющееся ПО для внедрения нового функционала путем небольших преобразований, смысл которых идентичен предыдущей реализации однако имеет более логичную и организованную структуру.

Реинжиниринг - создание нового функционала путем большого изменения, например изменения архитектуры программы

Висновки \*

В процессе выполнения лабораторной работы ознакомился с принципами тест дривен разработки и ее спецификациями, также ознакомился с принципами рефакторинга и провел рефакторинг участка кода покрытого тестами.

Submit

Clear form

Never submit passwords through Google Forms.

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#).

Google Forms

