Київський національний університет імені Тараса Шевченка

КРЕНЕВИЧ А.П.

Методичні вказівки  
до лабораторних занять із дисципліни

"Системи контролю версій"

для студентів механіко-математичного факультету

Київ – 2024

УДК 519.942+550

Рецензенти:

доктор фіз.-мат. наук, професор

доктор фіз.-мат. наук,

Рекомендовано до друку вченою радою механіко-математичного факультету

(протокол № \_\_ від \_\_ \_\_\_\_\_\_\_\_\_201\_\_ року)

Креневич А.П.

Методичні вказівки до лабораторних занять із дисципліни "Системи контролю версій" для студентів механіко-математичного факультету – К.: ВПЦ "Київський Університет", 20. – \_\_\_ с.

Посібник містить перелік завдань для аудиторної та самостійної роботи з дисципліни «Системи контролю версій», що викладається студентам механіко-математичного факультету Київського національного університету імені Тараса Шевченка.

Для студентів механіко-математичного факультету та викладачів, які проводять заняття з курсу "Об’єктно-орієнтоване програмування".

**ЗМІСТ**

[ВСТУП 4](#_Toc157438274)

[ЛАБОРАТОРНА РОБОТА 1. Поняття про системи контролю версій. 5](#_Toc157438275)

[ЛАБОРАТОРНА РОБОТА 2. Робота з гілками та віддаленими репозиторіями. 7](#_Toc157438276)

# ВСТУП

# ЛАБОРАТОРНА РОБОТА 1. Поняття про системи контролю версій.

## 1.1. Контрольні запитання

1.1.1.

## 1.2. Завдання

1.3.1. Встановіть на свій персональний комп’ютер систему контролю версій Git з офіційного сайту Git. Проведіть глобальні налаштування (користувач, електронна адреса) Git.

1.2.2. Запустіть командний рядок (Git Bash, термінал, тощо). Використовуючи командний рядок

1. Створіть папку Repository (наприклад у кореневій папці поточного користувача) для зберігання репозиторіїв.
2. У папці Repository створіть папку Test, перейдіть у неї та створіть в ній репозиторій.
3. Додайте кілька текстових файлів з кількома рядками коду в кожному з них.
4. Перегляньте стан репозиторія командою git status.
5. Додайте створені файли у індекс.
6. Відредагуйте один з доданих файлів та перегляньте стан репозиторія.
7. Додайте здійснені зміни у індекс та перегляньте стан репозиторія.
8. Зробіть коміт з повідомленням «Initial commit» та перегляньте стан репозиторія.
9. Відредагуйте один або кілька з доданих файлів, перегляньте стан репозиторія та здійсніть коміт з автоматичним додаванням змін у індекс використовуючи параметр -a команди git commit.
10. Створіть новий репозиторій на GitHub та, користуючись інструкціями GitHub, надішліть у нього створений локальний репозиторій.
11. Перегляньте стан репозиторія на сторінці GitHub.

1.3.2. Встановіть на свій персональний комп’ютер одну з графічних оболонок для роботи з системою контролю версій Git (GitHub Desktop, Tortice Git, SourceTree тощо).

1.3.3. Створіть локальний репозиторій DevSW за допомогою командного рядка. У цьому репозиторії (всі операції проробіть з командного рядка)

1. Перейдіть у папку з новим репозиторієм та, використовуючи команду git status, перегляньте статус репозиторія.
2. Створіть перший коміт у репозиторій git commit з параметром -m та задайте повідомлення для коміту «Initial commit»
3. Додайте файл ignored.txt з текстом «Якийсь текст» у репозиторій.
4. Перегляньте статус репозиторія командою git status.
5. Додайте цей файл у список gitignore.
6. Перегляньте статус репозиторія.
7. Зробіть коміт у репозиторій з текстом «Added gitignore».
8. Додайте файл main.kt в якому опишіть програму для виведення на екран повідомлення “Hello, World”.
9. Додайте новий файл main.kt у індекс та перегляньте статус репозиторія командою git status.
10. Відмініть додавання у репозиторій зазначеного вище файлу (див відповідну команду як результат виведення команди git status).
11. Додайте знову файл main.kt у індекс.
12. Відредагуйте програму, створену вище, додавши функцію, що здійснює виведення повідомлення “Hello, World” та здійсніть виклик цієї функції 10 разів (наприклад, за допомогою циклу).
13. Перегляньте статус репозиторія командою git status.
14. Додайте нові зміни у індекс.
15. Перегляньте статус репозиторія.
16. Зробіть коміт у репозиторій командою git commit без параметрів та задайте у текстовому редакторі vim повідомлення для коміту «Printing message “hello world” 10 times».
17. Перегляньте історію комітів репозиторія командою git log.
18. Додайте у файл main.kt нову функцію vivaMechMat(), що виводить повідомлення «Viva Mech-mat faculty» та здійсніть виклик цієї функції.
19. Зробіть коміт у репозиторій з одночасним додаванням всіх здійснених змін у індекс.
20. Перегляньте історію комітів обмеживши виведення останніми двома комітами.
21. Поправте функцію vivaMechMat(), так щоб виведення повідомлення закінчувалося трьома знаками оклику.
22. Виправте попередній коміт командою git commit –amend.
23. Перегляньте історію комітів обмеживши виведення останніми двома комітами.

1.3.4. Створіть віддалений репозиторій DevelopSW на сайті GitHub.

1. Склонуйте цей репозиторій собі локально на комп’ютер використовуючи командний рядок.
2. Проробіть всі операції наведені у попередній вправі для цього склонованого репозиторія.

1.3.5. Проробіть всі операції (або такі що їм відповідають) з попередніх двох вправ за допомогою будь-якої з графічних оболонок.

# ЛАБОРАТОРНА РОБОТА 2. Робота з гілками та віддаленими репозиторіями.

## 2.1. Контрольні запитання

2.1.1.

## 2.2. Завдання

2.2.1. Базові операції для роботи з гілками. Оберіть один з раніше створених тестових репозиторіїв. У випадку, якщо такі репозиторії відсутні, створіть новий репозиторій та створіть у ньому принаймні три коміти. Проробіть всі операції наведені нижче за допомогою командного рядка (Git Bash, terminal, тощо).

1. Перегляньте для цього репозиторія список гілок.
2. Перейменуйте головну гілку у гілку main. Якщо ваша головна гілка вже називається main, то перейменуйте її в feature\_main, а потім назад на main.
3. Створіть гілку develop на поточному коміті.
4. Перейдіть на гілку develop.
5. Додайте на цій гілці кілька комітів.
6. Перейдіть на головну гілку main.
7. Проведіть зливання гілки develop у гілку main. Зверніть увагу на інформацію яку при цьому буде виведено в консоль.
8. Перегляньте список гілок та історію комітів.
9. Видаліть гілку develop.
10. Перегляньте поточний список гілок та історію комітів репозиторія.

2.2.2. Злиття гілок. Оберіть один з раніше створених тестових репозиторіїв.

1. Перегляньте для цього репозиторія список гілок. Перейменуйте головну гілку репозиторія у main.
2. Створіть гілку develop на поточному коміті гілки main.
3. Створіть файл main\_test.txt на поточній гілці main та додайте в нього кілька рядків коду. Зробіть коміт, додавши створений файл у історію.
4. Перейдіть на гілку develop та додайте там файл dev\_test.txt. Додайте в ньому кілька рядків тексту. Зробіть коміт, на цій гілці.
5. Відредагуйте файл dev\_test.txt додавши до нього ще кілька рядків будь-якого тексту. Зробіть коміт, щоб додати ці зміни в історію.
6. Поверніться на гілку main.
7. Відредагуйте файл main\_test.txt додавши в нього кілька рядків тексту. Зробіть коміт.
8. Проведіть злиття гілок main та develop – у гілку main влийте зміни з гілки develop. Перегляньте історію комітів. Видаліть гілку develop.
9. Додайте файл common.txt та додайте в нього такий текст.

Hello World!

Hello Mech-Mat!

1. Зробіть коміт.
2. Створіть гілку dev на поточному коміті гілки main.
3. Відредагуйте файл common.txt замінивши в ньому рядок

Hello Mech-Mat!

рядком

Привіт Мех-мат!

та зробіть коміт з проведеними змінами.

1. Поверніться на гілку main та відредагуйте файл common.txt замінивши в ньому рядок

Hello Mech-Mat!

рядком

Добрий день механіко-математичний!

та зробіть коміт з проведеними змінами.

1. Виконайте команду злиття гілки dev у головну гілку main. Розв’яжіть конфлікт, який при цьому виникне.
2. Після успішного злиття перейдіть на гілку dev, відредагуйте файл common.txt (довільним чином) зробіть коміт та поверніться назад на гілку main.
3. Видаліть гілку dev.

2.2.3. Робота з віддаленими репозиторіями.

1. Створіть локально новий репозиторій та додайте у нього принаймні три коміти.
2. Перейменуйте головну гілку репозиторія у main.
3. Перегляньте список віддалених репозиторіїв (він має бути порожнім).
4. Створіть новий репозиторій на GitHub та, користуючись інструкціями GitHub надішліть у нього створений локальний репозиторій.
5. Перегляньте список віддалених репозиторіїв. Перегляньте свій репозиторій та його гілки на сторінці GitHub.
6. Створіть у поточній гілці main принаймні один будь-який коміт. Надішліть зміни на гілку main\_for\_review віддаленого репозиторія. Знайдіть та перегляньте зміни на сторінці GitHub для цієї гілки.
7. Використовуючи сторінку GitHub, додайте у гілки main та main\_for\_review принаймні по одному коміту.
8. Синхронізуйте локальний репозиторій з віддаленим репозиторієм та перегляньте історію комітів, список віддалених та локальних гілок.
9. Видаліть гілку main\_for\_review з віддаленого репозиторія користуючись командним рядком.
10. Створіть у локальному репозиторії гілку develop. Зробіть у ній принаймні один коміт та надішліть зміни на віддалений репозиторій.
11. Зробіть гілку develop відслідковуваною, так щоб вона автоматично синхронізувалася з гілку develop віддаленого репозиторія.
12. Перегляньте список відслідковуваних гілок.
13. Зробіть кілька комітів на гілці develop та надішліть їх у віддалений репозиторій.
14. Злийте гілку develop у головну гілку main. Видаліть локальну та віддалені гілки develop.

## СПИСОК ЛІТЕРАТУРИ ТА ДОДАТКОВИХ ДЖЕРЕЛ

1.