

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
«Основи програмування на Kotlin»
Частина 1
для студентів спеціальності
123 «Комп'ютерна інженерія»
всіх форм навчання

Методичні вказівки до лабораторних робіт з дисципліни «Основи програмування на Kotlin» для студентів спеціальності 123 «Комп'ютерна інженерія» всіх форм навчання. Частина 1. / Укл.: Т.С. Дьячук, Т.В. Голуб, М.Б. Ільяшенко – Запоріжжя: Національний університет «Запорізька політехніка», 2025.– 38 с.

Автори: Т.С. Дьячук, ст.викл.
Т.В. Голуб, к.т.н., доцент
М.Б. Ільяшенко, к.т.н., доцент

Рецензент: Кудерметов Р.К., к.т.н, доцент

Відповідальний
за випуск: Т.С. Дьячук, ст.викл.

Затверджено
на засіданні кафедри КСМ
Протокол № 7 від 28.02.2025

Рекомендовано до видання
на засіданні НМК факультету КНТ
Протокол 7 від 24.03.2025

ЗМІСТ

ВСТУП.....	4
1 ЛАБОРАТОРНА РОБОТА № 1 ВСТАНОВЛЕННЯ СЕРЕДОВИЩА РОЗРОБКИ ТА ПЕРША ПРОГРАМА НА KOTLIN	5
1.1 Загальні відомості.....	5
1.2 Інсталяція IntelliJ IDEA	9
1.3 Реєстрація на GitHub	9
1.4 Створення власної копії репозиторію на GitHub для подальшого виконання завдань.....	11
1.5 Встановлення GitHub Desktop	12
1.6 Створюємо новий проєкт в IntelliJ IDEA	17
1.7 Завдання для самостійної роботи.....	20
1.8 Завантаження нового коду на GitHub	21
1.9 Створення пул-реквесту.....	22
1.10 Оцінка виконання лабораторної роботи.....	24
1.11 Зміст письмового звіту	25
1.12 Контрольні питання.....	25
2 ЛАБОРАТОРНА РОБОТА № 2 ПРОГРАМУВАННЯ НА KOTLIN: ЗМІННІ ТА ТИПИ ДАНИХ, РЯДКИ, УМОВИ ТА ЦИКЛИ, ФУНКЦІЇ	26
2.1 Загальні відомості.....	26
2.2 Оновлення власної копії репозиторію на GitHub	26
2.3 Відкриття багатомодульного Gradle проєкту	27
2.4 Завдання для самостійної роботи.....	35
2.5 Оцінка виконання лабораторної роботи.....	36
2.6 Зміст письмового звіту	36
2.7 Контрольні питання.....	37
РЕКОМЕНДОВАНА ЛІТЕРАТУРА.....	38

ВСТУП

Kotlin – статично типізована, об'єктно-орієнтована мова програмування, яку розробляє компанія JetBrains. Її названо на честь острова Котлін у Фінській затоці, на якому розташоване місто Кронштадт.

Kotlin зазнав впливу багатьох мов, зокрема Java, Scala, Groovy, C#, JavaScript, Swift та дозволяє писати програми як в об'єктно-орієнтованому, так і в функціональному стилі. Він має ясний і зрозумілий синтаксис та досить легкий для навчання. Найпопулярнішим напрямком, де застосовується Kotlin, є перш за все розробка під ОС Android. Але є і інші платформи, для яких можна створювати застосунки на Kotlin: Windows, Linux, macOS, iOS.

Kotlin – прагматична, безпечна, лаконічна та сумісна мова програмування, яка підтримує використання перевірених рішень для поширених завдань. Вона допомагає уникати типових помилок, дозволяє писати компактний і зрозумілий код, а також забезпечує безшовну інтеграцію з Java. Мова Kotlin підходить для розробки програмного забезпечення в різних сферах – від мобільних застосунків до розподілених систем. Завдяки своїй універсальності, вона є зручним інструментом як для індивідуальної, так і для командної розробки. Kotlin активно використовується в промисловій розробці та добре підходить для створення сучасних програмних рішень.

Під час вивчення дисциплін з програмування важливо не лише засвоїти синтаксис мови, а й опанувати сучасні інструменти розробки. Знання мови – це лише частина підготовки. У практиці часто трапляються ситуації, коли програміст не може ефективно працювати над складними командними проєктами через відсутність досвіду роботи з актуальним інструментарієм. Навіть якщо спочатку вивчення нових технологій викликає труднощі, набуті навички стануть важливим фундаментом для подальшої професійної діяльності, особливо при роботі в команді над реальними проєктами.

1 ЛАБОРАТОРНА РОБОТА № 1 ВСТАНОВЛЕННЯ СЕРЕДОВИЩА РОЗРОБКИ ТА ПЕРША ПРОГРАМА НА KOTLIN

Мета роботи – налаштування та опанування інструментів розробки, ознайомлення з мовою програмування Kotlin.

1.1 Загальні відомості

Оволодіння сучасними засобами розробки допомагає студентам вирішувати складні завдання та стати конкурентоспроможними на ринку праці у майбутньому. Сучасні інструменти допомагають створювати програмне забезпечення ефективніше та швидше завдяки високому рівню автоматизації та інтеграції, вони підвищують якість коду через використання сучасних практик та стандартів.

Для успішної роботи програмістом недостатньо знати конкретну мову програмування, необхідно також мати навички роботи із сучасними середовищами розробки, системами контролю за версіями, вміти шукати потрібну інформацію та не боятися труднощів. У даному курсі будемо користуватися такими засобами розробки як: Git, GitHub, GitHub Desktop, IntelliJ IDEA. Первинна перевірка завдань з програмування буде проходити у автоматичному режимі на GitHub, тому потрібно чітко дотримуватися вимог до завдань.

Спочатку розглянемо базову термінологію, необхідну для розуміння процесу роботи з кодом. Для глибшого розуміння викладеного матеріалу або якщо наданої інформації вам буде недостатньо, рекомендуємо звернутися до лекційних матеріалів, додаткової літератури та онлайн-ресурсів.

Акаунт (обліковий запис) – це особистий профіль у системі (онлайн-сервісі, програмі), який ідентифікує вас та надає доступ до певних функцій.

Репозиторій – це сховище, де зберігається весь код проєкту, разом із його історією змін, конфігураційними файлами та документацією.

Гілка (branch) – варіант коду всередині одного репозиторію. Для паралельної роботи над змінами.

Форкована копія – це репозиторій, який був створений на основі іншого репозиторію за допомогою функції Fork на GitHub.

Рефакторинг – це процес покращення внутрішньої структури коду без зміни його зовнішньої поведінки. Мета рефакторингу – зробити код чистішим, зрозумілішим, зручнішим для підтримки та розширення.

Коміт (commit) – це збереження змін у системі контролю версій, наприклад, Git. Коли ви закомічуєте зміни, ви створюєте нову версію файлів у репозиторії, що дозволяє відстежувати історію змін, порівнювати версії та при необхідності повертатися до попередніх станів. Закомітити – означає зафіксувати зміни у репозиторії, додавши їх у його історію. Для цього зазвичай використовують команду в консолі `git commit -m "Commit message"`, де "Commit message" – це короткий коментар про внесені правки. Ми це будемо робити за допомогою GitHub Desktop.

Пуш (push) – відправлення змін з локального репозиторію на віддалений (наприклад, GitHub).

Пул-реквест (PR – pull request) – це пропозиція зміни коду у репозиторії. По суті це запит на злиття змін з однієї гілки в іншу в системах контролю версій, таких як GitHub. Розробник створює гілку, вносить зміни, пушить їх у репозиторій і відкриває PR. Інші розробники переглядають код, коментують і, якщо все добре, схвалюють злиття. Це потрібно для код-рев'ю, збереження історії змін і контролю якості коду.

Код-рев'ю (code review) – це процес перевірки коду іншими розробниками перед його злиттям у основну гілку. Мета – знайти помилки, покращити якість коду, дотримання стандартів і оптимізацію. Зазвичай проводиться через PR на GitHub.

Open-source (відкритий код) – це програмне забезпечення (ПЗ) з відкритим вихідним кодом, який кожен може переглядати, змінювати та розповсюджувати. Такі проекти розвиваються спільнотою розробників у відкритому доступі.

На рисунку 1.1 представлено стек технологій для сучасної розробки на мові Kotlin. Розглянемо їх більш детально.

Отже, Kotlin – статично типізована, об'єктно-орієнтована мова програмування, яка працює поверх JVM (Java Virtual Machine), може застосовуватися для створення застосунків у багатьох проблемних областях та є сучасною та високопродуктивною альтернативою Java.

Gradle – потужний інструмент автоматизації збірки програмного забезпечення, який дозволяє встановлювати та керувати залежностями

проєкту, виконувати збірку та розгортання, управляти бібліотеками. Gradle є популярним інструментом у сучасній розробці програмного забезпечення, особливо в області Java та Android.

IntelliJ IDEA – інтегроване середовище розробки, що підтримує багато мов програмування та забезпечує широкі можливості для комфортної розробки. Вона відома своєю високою продуктивністю та широким спектром функцій, що допомагають розробникам писати код швидше та ефективніше. Має розширену підтримку рефакторингу, автоматичного завершення коду, аналізу коду на льоту, інтеграцію з системами контролю версій, тестування та іншими корисними функціями. Ця IDE підтримує роботу з різними фреймворками і технологіями, що робить її популярним вибором серед розробників усього світу.

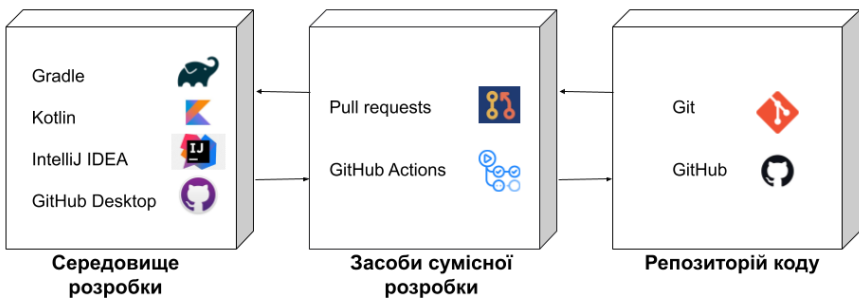


Рисунок 1.1 – Стек технологій для розробки на мові Kotlin

GitHub – це веб-платформа для спільної розробки програмного забезпечення, яка базується на системі контролю версій Git. Вона забезпечує можливості для зберігання, спільної роботи над кодом, відстеження змін, обговорень та управління проєктами. GitHub є однією з найпопулярніших платформ для розробки програмного забезпечення у світі та використовується мільйонами програмістів. GitHub надає механізм PR, який дозволяє розробникам запропонувати зміни до коду в репозиторії. Це спосіб ініціювати обговорення та перевірку внесених змін перед їх об'єднанням з основною гілкою проєкту. GitHub також є великою соціальною мережею для розробників. Тут можна ділитися своїми проєктами, знаходити нові

рішення в роботах інших, брати участь у проєктах з відкритим кодом та будувати своє професійне портфоліо.

GitHub Actions – інструмент автоматизації робочих процесів, який дозволяє створювати та розгортати сценарії для автоматичного виконання дій у вашому репозиторії GitHub. Наприклад, ви можете налаштувати автоматичну збірку проєкту або запуск автоматичних тестів при відправці коду.

GitHub Desktop – графічний клієнт, який має зручний інтерфейс для роботи з Git та GitHub. Він спрощує керування репозиторіями, комітами, гілками та злиттям змін без використання терміналу. Дозволяє легко клонувати репозиторії, візуально керувати комітами та історією змін, створювати й перемикатися між гілками, а також синхронізувати зміни з GitHub. Ця програма ідеально підходить для початківців або тих, хто хоче спростити роботу з Git. Вона підтримує інтеграцію з редакторами коду, що робить розробку ще зручнішою.

Отже, для проходження курсу з вивчення мови Kotlin вам знадобляться наступні інструменти:

- середовище розробки IntelliJ IDEA;
- GitHub акаунт;
- застосунок GitHub Desktop.

Насамперед ознайомимось із процесом налаштування робочого середовища для виконання лабораторних робіт у межах цього курсу. Перша лабораторна робота спрямована на налаштування та опанування інструментів розробки. Ця підготовка допоможе вам комфортно працювати з Kotlin і зберігати код у GitHub, що є важливою частиною командної роботи та ефективного керування проєктами. Після налаштування середовища ви зможете створювати та запускати Kotlin-програми у IntelliJ IDEA, працювати з Git – відстежувати зміни в коді, створювати коміти та при необхідності гілки, а також завантажувати свої проєкти в GitHub, що спрощує їх резервне копіювання та співпрацю з іншими розробниками. Крім того, ви отримаєте базові навички DevOps, необхідні для сучасної розробки. Опанування цих інструментів зробить вашу роботу більш ефективною, дозволить легко повертатися до попередніх версій коду та сприятиме професійному зростанню як розробника.

Увага!! Оскільки програмне забезпечення (ПЗ) постійно вдосконалюється, розробники регулярно вносять зміни не лише у функціональність, але й у зовнішній вигляд програм. Це може

призводити до того, що інтерфейс користувача, розташування елементів керування або візуальне оформлення можуть мати певні відмінності між різними версіями однієї й тієї ж програми, адже це є частиною процесу постійного розвитку та покращення ПЗ.

1.2 Інсталяція IntelliJ IDEA

Для інсталяції IntelliJ IDEA перейдіть за посиланням <https://www.jetbrains.com/idea/download> – на цій сторінці можливо обрати який варіант IDE ви хочете завантажити.

Повна версія – **Ultimate**, НУ «Запорізька політехніка» бере участь у програмі, по котрій компанія IntelliJ надає безкоштовну академічну ліцензію у разі реєстрації акаунту на e-mail з доменним ім'ям університету (**@zp.edu.ua**). Або безкоштовна версія – **Community**, її цілком достатньо для вивчення цього курсу.

У разі відсутності необхідної для реєстрації на зовнішніх освітніх ресурсах електронної пошти в домені **zp.edu.ua** центр інновацій пропонує такий алгоритм дій: студент в довільній формі направляє e-mail на **student_reg@zp.edu.ua**, в якому вказує ПІБ, форму навчання, групу і додає до листа вкладення зі скан-копією (або скріншотом з застосунка «Дія») студентського квитка. У відповідь студенту направляється логін та пароль для нової поштової скриньки.

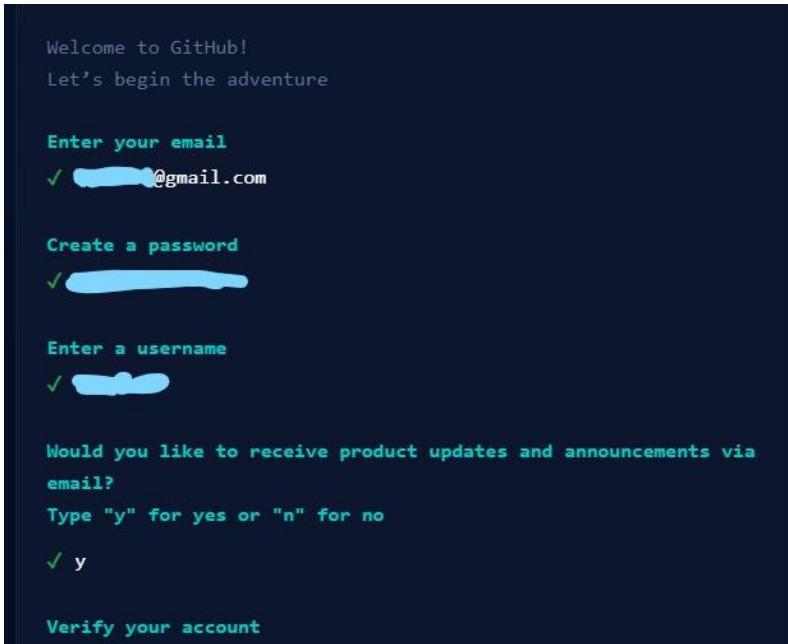
1.3 Реєстрація на GitHub

Якщо ви вже маєте акаунт на цьому сервісі, то цей пункт можна пропустити.

Для реєстрації зайдіть на <https://github.com/>. Натисніть на кнопку **Sign up**, введіть ваш e-mail, пароль для ресурсу GitHub, ім'я користувача. Далі ви обираєте дати чи не дати згоду на “Бажаєте отримувати оновлення продукту та оголошення електронною поштою?” Проходите верифікацію, що ви справжня людина (рисунок 1.2). Після чого потрібно підтвердити акаунт кодом, який прийде на електронну пошту, яку ви вказали при реєстрації.

Далі можна буде пройти налаштування персоналізації, хоча це опціонально і можна пропустити (рисунок 1.3). Якщо ви вкажете, що ви студент, – вам буде запропоноване “Отримайте додаткові

студентські переваги”, що є не обов'язковим для вивчення даного курсу.



Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ [redacted]@gmail.com

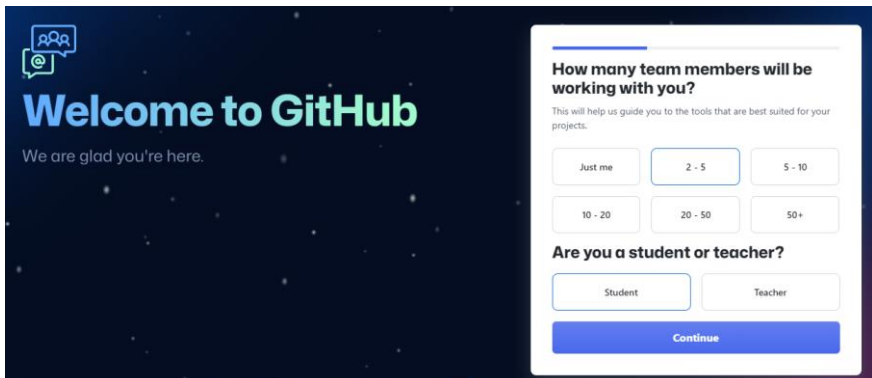
Create a password
✓ [redacted]

Enter a username
✓ [redacted]

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
✓ y

Verify your account

Рисунок 1.2 – Реєстрація на GitHub



 **Welcome to GitHub**
We are glad you're here.

How many team members will be working with you?
This will help us guide you to the tools that are best suited for your projects.

Are you a student or teacher?

Рисунок 1.3 – Налаштування персоналізації на GitHub

1.4 Створення власної копії репозиторію на GitHub для подальшого виконання завдань

Для того щоб можна було внести зміни в репозиторій потрібно створити власну копію (форк) репозиторію з правами на запис. Для цього заходимо на сторінку <https://github.com/DiachT/KotlinLabsNUZP> та тиснемо на кнопку “Fork” (рисунок 1.4).

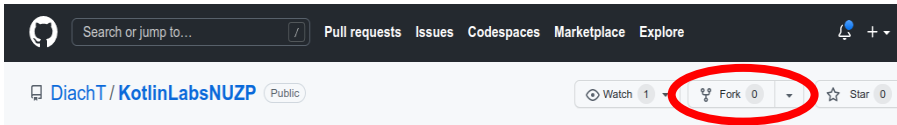


Рисунок 1.4 – Створення власної копії репозиторію

Створення копії репозиторію є надзвичайно важливим кроком у роботі з відкритим кодом. Це дозволяє працювати над проектом незалежно від основної версії, не впливаючи на оригінальний код. Особливо це актуально, коли ви не маєте прав на пряме редагування репозиторію або працюєте над розробкою у команді з розподіленими ролями. Форк надає вам повну свободу дій: ви можете експериментувати, виправляти помилки, додавати нові функції, оновлювати документацію або модифікувати структуру проекту – все це у вашій особистій копії. Після завершення змін ви маєте можливість запропонувати їх авторам основного репозиторію через PR (це стандартна практика у світі open-source). Крім того, форк виконує роль резервної копії, яку ви контролюєте. Це зручно, якщо оригінальний репозиторій буде видалений або змінений. Також це – безпечне середовище для навчання, у якому можна спокійно практикуватися без ризику зіпсувати основний код. Таким чином, копіювання репозиторію на GitHub — це ключовий інструмент для самостійної розробки, внеску в спільні проекти та безпечного вивчення нових технологій.

Далі потрібно перейти на сторінку налаштування копії репозиторію (рисунок 1.5). На цій сторінці потрібно лише обрати власника (поле Owner) нової копії, всі інші поля можна не змінювати. Після обрання тиснемо на зелену кнопку Create Fork. Після цього можна перейти на наступний пункт 1.5.

DiachT / KotlinLabsNUZP Public

Watch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Single sign-on to see more options within the Logitech organization.

Owner * **Repository name ***

Select an owner / KotlinLabsNUZP

Default forks inherit the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

☒ **Copy the main branch only**
Contribute back to DiachT/KotlinLabsNUZP by adding your own branch. [Learn more.](#)

You are creating a fork.

Create fork

Рисунок 1.5 – Налаштування копії репозиторію

1.5 Встановлення GitHub Desktop

На цьому кроці потрібно встановити застосунок GitHub Desktop та завантажити (клонувати) репозиторій на власний комп'ютер. Закачати програму можна за посиланням <https://desktop.github.com/> (рисунок 1.6).

Далі запускаємо установку GitHubDesktopSetup-x64.exe (рисунок 1.7).



Рисунок 1.6 – Домашня сторінка GitHub Desktop

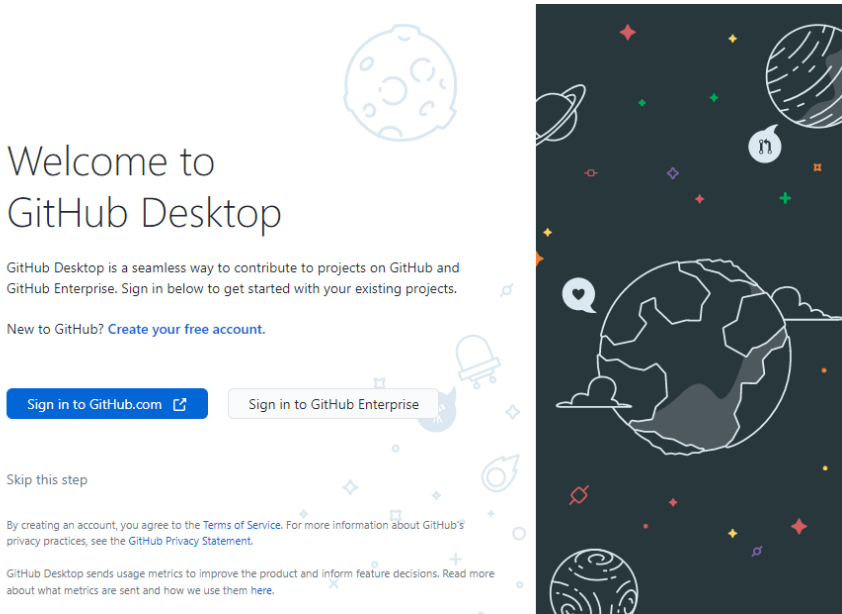


Рисунок 1.7 – Встановлення GitHub Desktop

На наступному кроці потрібно підтвердити авторизацію GitHub Desktop на GitHub. Переходимо у браузер та тиснемо “Authorize desktop” (рисунок 1.8). Після чого побачимо напис (рисунок 1.9).

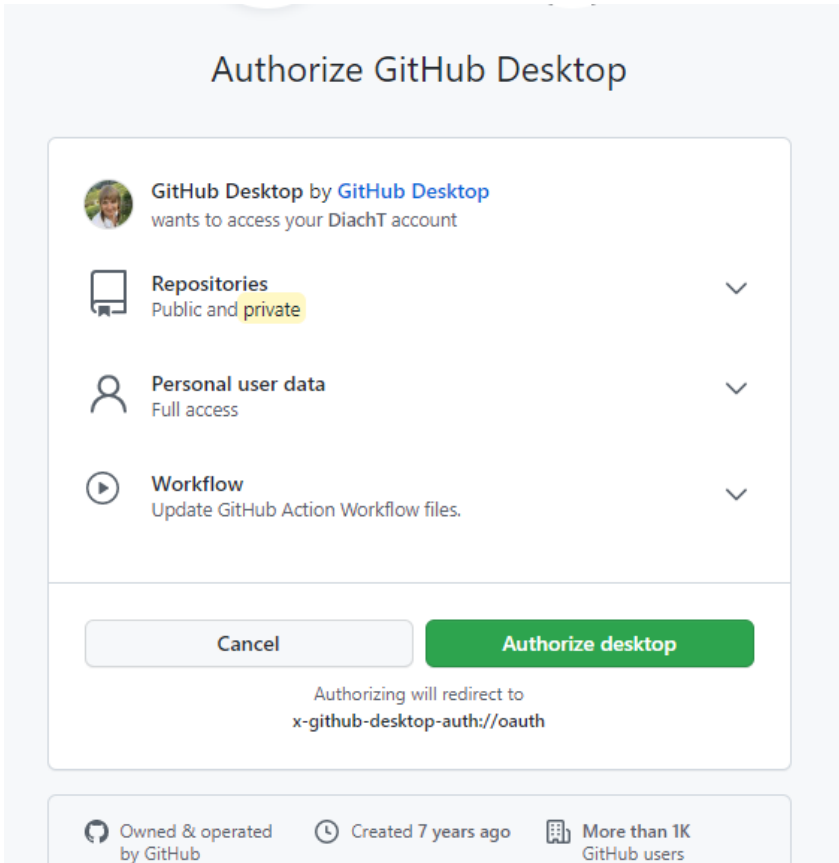


Рисунок 1.8 – Авторизація GitHub Desktop на GitHub

GitHub Desktop дозволяє взаємодіяти з GitHub, спрощуючи управління репозиторіями та командну роботу. Користувач може швидко завантажувати зміни, відправляти оновлення в хмару, створювати Pull requests та переглядати історію комітів. GitHub Desktop також забезпечує зручну синхронізацію між локальним і

віддаленим репозиторієм, що полегшує співпрацю з іншими розробниками.

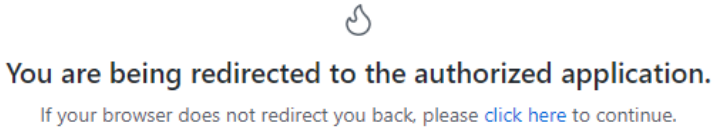


Рисунок 1.9 – Підтвердження авторизації GitHub Desktop на GitHub

Далі потрібно повернутися у GitHub Desktop та перевірити ваші дані (рисунок 1.10). Після перевірки тиснемо Finish.

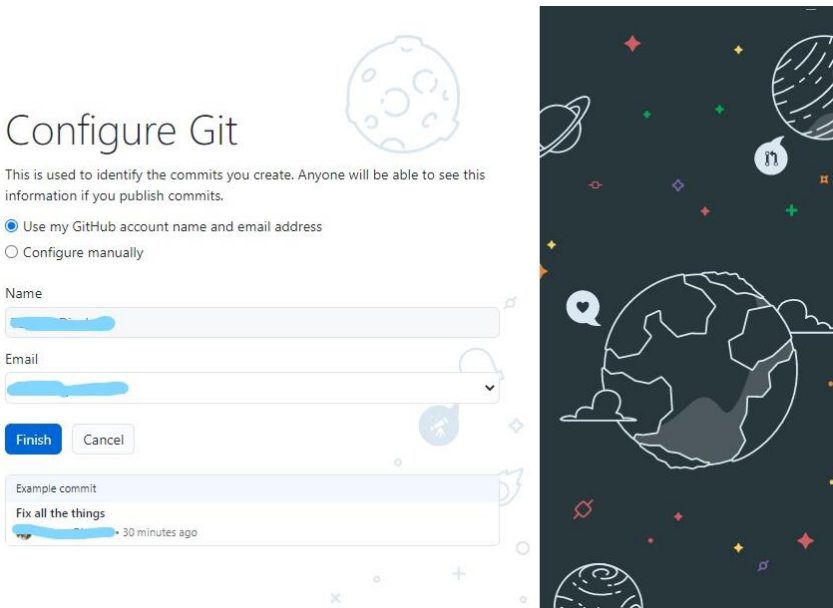


Рисунок 1.10 – Перевірка даних у GitHub Desktop

На наступному екрані вибираємо "Clone a repository from the Internet...". Вам потрібно вказати адресу репозиторію власної копії з

пункта 1.4, наприклад <https://github.com/UserName/KotlinLabsNUZP> та вибрати папку на вашому комп'ютері, де ви плануєте працювати з проєктом (рисунок 1.11).

Увага!! Уникайте використання файлових шляхів із кирилицею, оскільки це може призвести до проблем із сумісністю, зчитуванням файлів та виконанням коду. Рекомендується використовувати лише латиницю та стандартні символи, щоб уникнути непередбачених помилок у тестуванні та роботі програми.

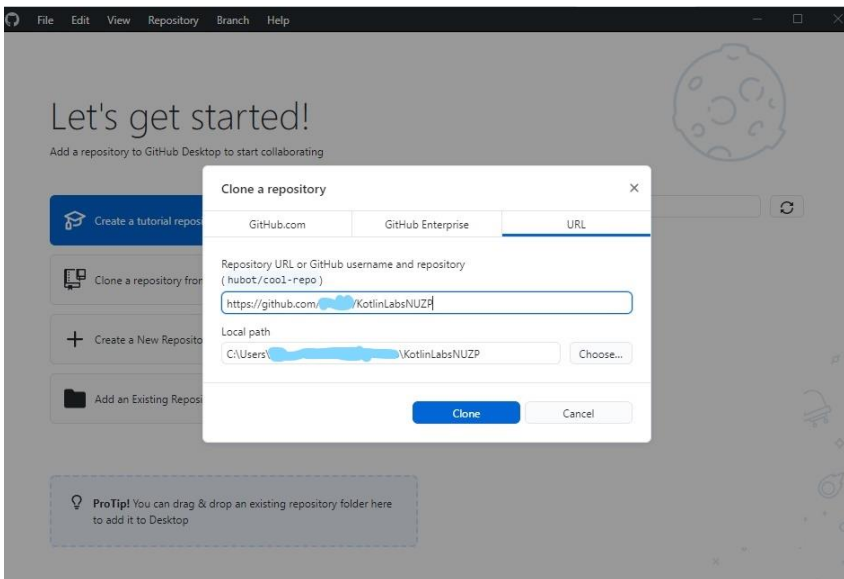


Рисунок 1.11 – Клонування репозиторію

Клонування репозиторію – це процес створення локальної копії віддаленого репозиторію Git на вашому комп'ютері. Це дозволяє працювати з кодом, змінювати його, створювати нові гілки та вносити зміни, які потім можна відправити назад у віддалений репозиторій. Клонування репозиторію дозволяє швидко почати роботу з проєктом, маючи всі його файли та історію змін.

Після клонування з'явиться екран локального репозиторію (рисунок 1.12). А у вибраній папці з'явиться папка KotlinLabsNUZP з проєктом.

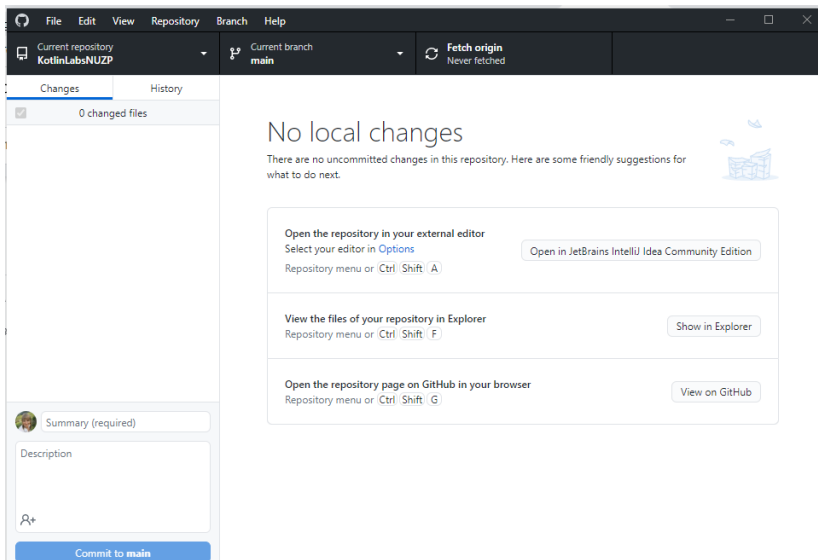


Рисунок 1.12 – Екран локального репозиторію

1.6 Створюємо новий проєкт в IntelliJ IDEA

Після першого запуску IntelliJ IDEA ви побачите вікно де можна вибрати проєкт. В нашому випадку, ще жодного проєкту немає, тому потрібно його створити – натисніть “New Project” (рисунок 1.13).

У вікні створення проєкту потрібно (рисунок 1.14):

- 1) обрати мову Kotlin;
- 2) обрати шлях до репозиторію який ви завантажили при виконанні попереднього розділу;
- 3) ім'я проєкту – введіть helloworld;
- 4) тип проєкту – консольний застосунок;
- 5) система сборки – Gradle Kotlin;
- 6) версія JDK – оберіть 17.x.x.

JDK (Java Development Kit) необхідний для розробки на Kotlin, оскільки Kotlin працює на JVM (Java Virtual Machine). Код, написаний на Kotlin, компілюється у байт-код Java, який виконується у JVM.

Тому для компіляції та запуску Kotlin-застосунків потрібен встановлений JDK. Найчастіше розробники обирають JDK 11 або JDK 17, оскільки вони є довготривало підтримуваними (LTS).

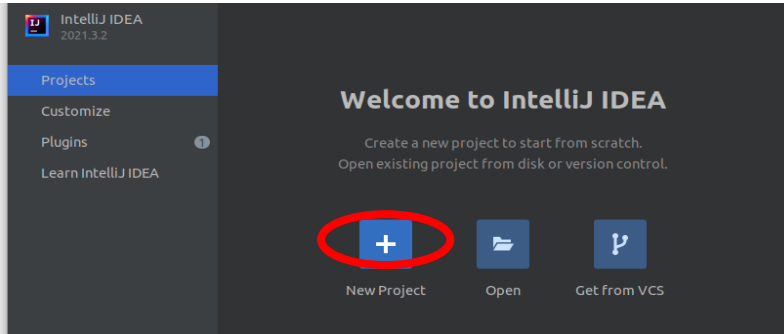


Рисунок 1.13 – Створення нового проєкту в IntelliJ IDEA

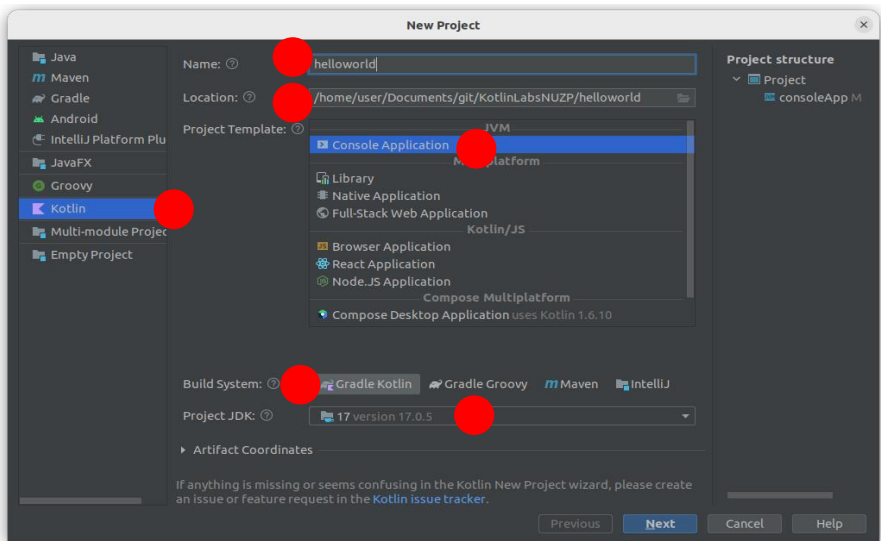


Рисунок 1.14 – Налаштування проєкту в IntelliJ IDEA

Новий проєкт в IntelliJ IDEA має лише один файл на Kotlin – Main.kt зі змістом як на рисунку 1.15.

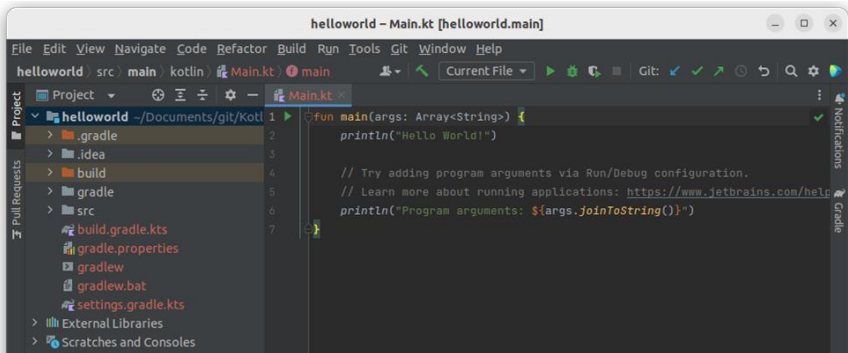


Рисунок 1.15 – Новий проєкт в IntelliJ IDEA

Проект “Hello World” на Kotlin – це базовий приклад, який допомагає зрозуміти основи синтаксису мови та роботу середовища розробки. Функція `main()` є точкою входу в програму. Команда `println("...")` виводить текст у консоль. Цей простий приклад дозволяє перевірити, чи коректно налаштоване середовище для розробки на Kotlin.

Перевірте версію Kotlin в `helloworld/build.gradle.kts` файлі, вона повинна бути "1.8.0":

```
1: plugins {
2:     kotlin("jvm") version "1.8.0"
3:     application
4: }
```

Далі давайте модифікуємо код на Kotlin у файлі `Main.kt` наступним чином:

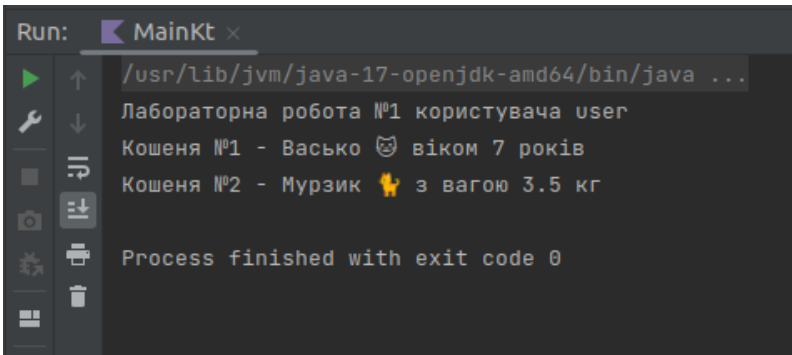
```
1: fun seed(): String = "user"
2:
3: fun labNumber() : Int = 1
4:
5: fun main(args: Array<String>) {
6:     println("Лабораторна робота №${labNumber()}
   користувача ${seed()}")
}
```

```

7:
8:     var kitty = "Васько"
9:     kitty += " \uD83D\uDC31"
10:    val age = 7
11:    println("Кошеня №1 - $kitty віком $age років")
12:
13:    val catName: String = "Мурзик \uD83D\uDC08"
14:    val weight: Float = 3.5f
15:    println("Кошеня №2 - $catName з вагою $weight
    кг")
16: }

```

У програмі є декілька функцій, використовуються різні типи змінних та показані прості приклади роботи з ними. Запустимо цей код, у вікні виводу застосунку побачимо наступний вивід (рисуюнок 1.16).



Рисуюнок 1.16 – Результат роботи застосунку

1.7 Завдання для самостійної роботи

Необхідно внести зміни до вихідної програми, яку ви створили під час налаштування середовища розробки. Завдання передбачає модифікацію файлу Main.kt.

1. Впишіть ім'я свого GitHub-користувача у функцію `seed()`.
2. Вам потрібно додати ще один рядок “Кошеня №3”, якого звать “Рудий”, “віком 6 років та вагою 8.2 кг”, для чого потрібно створити три змінні.

Увага!! Пул-реквест буде завантажено на GitHub, де викладач перевірить виконання завдань. Важливо чітко дотримуватися всіх вимог, оскільки навіть незначні відхилення у тексті можуть призвести до помилок і тестування не пройде успішно. Будьте уважні до деталей, щоб забезпечити коректну роботу вашого коду.

1.8 Завантаження нового коду на GitHub

Далі потрібно закомітити зміни у `git`. Для цього відкриваємо GitHub Desktop, де бачимо всі зміни у коді. Потрібно ввести ім'я коміту, наприклад “лабораторна робота 1” та натиснути на кнопку “Commit to main” (рисунк 1.17).

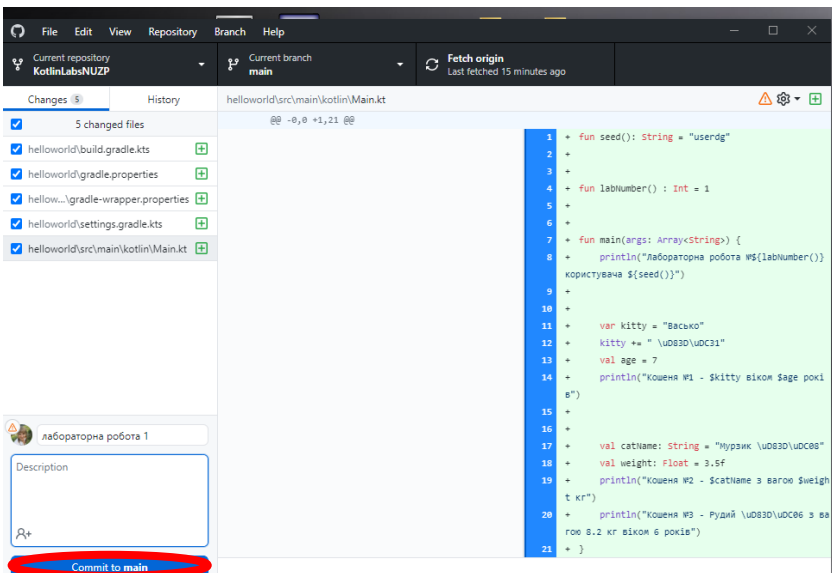


Рисунок 1.17 – Фіксування змін у репозиторії

Коміт зберігає стан файлів лише в локальній копії репозиторію, тобто на вашому комп'ютері. Це означає, що всі внесені зміни залишаються доступними тільки для вас і ще не потрапляють у віддалений репозиторій, доступний для інших учасників проєкту. Отже, для того щоб хтось інший мав можливість працювати з тим самим кодом (або перевірити код), потрібно його відправити на GitHub, для цього тиснемо на “Push origin” (рисунк 1.18). Після цього буде відправлено набір змін на сервер, і вони стануть доступними для всіх членів команди.

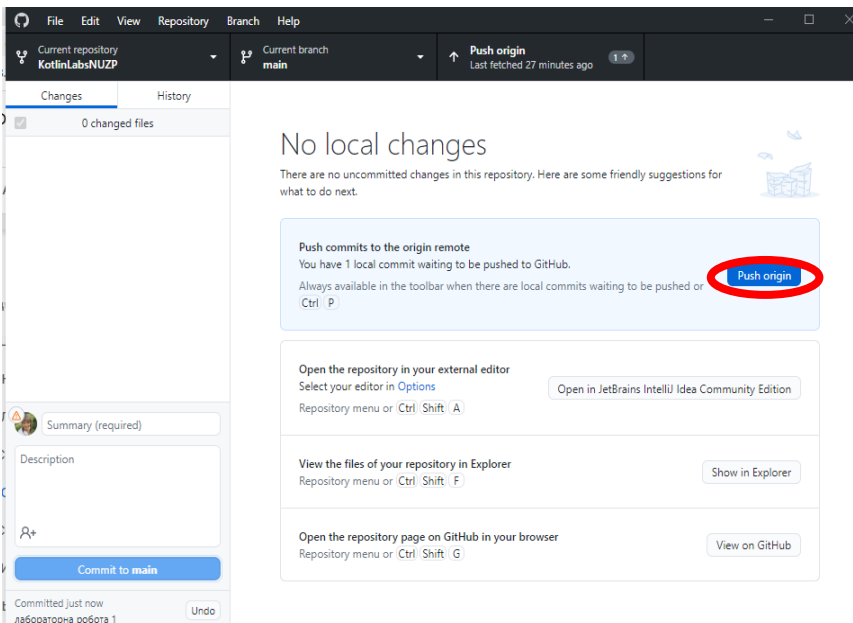


Рисунок 1.18 – Відправлення змін у віддалений репозиторій на GitHub

1.9 Створення пул-реквесту

Після вдалого коміту та пушу (пункт 1.8) потрібно перейти в браузер на сторінку вашої копії репозиторію (рисунк 1.19). Натисніть “Contribute” і потім “Open pull request” (рисунк 1.20).

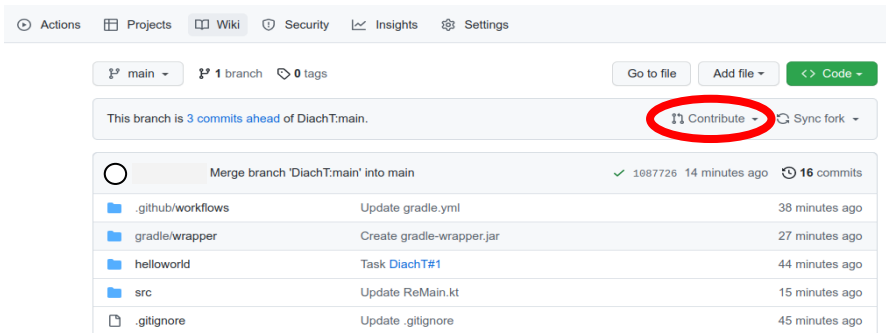


Рисунок 1.19 – Сторінка власної копії репозиторію

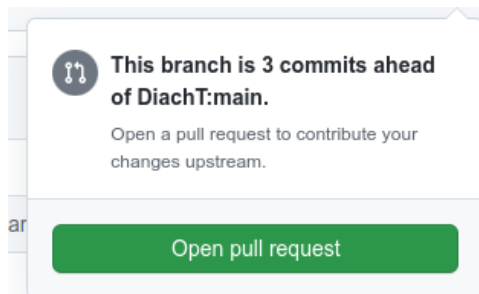


Рисунок 1.20 – Повідомлення про відкриття PR

Опинимось на сторінці PR (рисунок 1.21). Потрібно заповнити поле Title – напишіть тут **“Лабораторна робота #1 варіант XX”**. Номер варіанту вкажіть зі списку студентів у Moodle розділу курсу “Основи програмування на Kotlin”. Варіант потрібен для ідентифікації студентів. В кінці тиснемо на кнопку “Create pull request”. Пул-реквест буде завантажено на GitHub, де викладач перевірить виконання завдань (пункт 1.10).

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: DiachT/KotlinLabsNUZP base: main head repository: /KotlinLabsNUZP compare: main

✓ Able to merge. These branches can be automatically merged.

Title

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

☒ Allow edits and access to secrets by maintainers

Create pull request

Helpful resources
[GitHub Community Guidelines](#)

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Рисунок 1.21 – Сторінка створення PR

На цьому етапі ви успішно виконали електронну частину першої лабораторної роботи.

1.10 Оцінка виконання лабораторної роботи

В цій лабораторній роботі за електронну частину роботи ви отримуєте:

- якщо у вас вийшло зробити pull request і ви коректно вказали варіант та логін на GitHub, то +5 балів;
- якщо ваш pull request пройшов автоматичний тест, то +3 бали;
- якщо ваш код задовольняє всім умовам при перевірці викладачем, то +2 бали.

Також потрібно зробити звіт та загрузити його у Moodle на сторінку курсу “Основи програмування на Kotlin”. За звіт ви можете отримати максимально 5 балів.

Таким чином, за виконання першої лабораторної роботи, максимально нараховуються 15 балів, які будуть зараховані до загальної оцінки з поточної дисципліни.

1.11 Зміст письмового звіту

1. Титульна сторінка.
2. Мета та хід роботи.
3. Код та скріншот роботи модифікованої програми.
4. Скріншот пул-реквеста.
5. Відповіді на контрольні питання.

1.12 Контрольні питання

1. Напишіть ваш логін на GitHub, який ви будете використовувати під час виконання лабораторних робіт з цього курсу.
2. З якими труднощами ви зіткнулися при виконанні цієї роботи?
3. Що таке git, для чого використовується?
4. Що таке GitHub, для чого використовується?
5. Що таке GitHub Desktop, для чого використовується?
6. Що таке IntelliJ IDEA, для чого використовується?
7. Для чого використовуються val та var у Kotlin?
8. Що таке коміт?
9. Що таке пул-реквест?
10. Що таке GitHub Actions та його можливості?
11. Як вивести повідомлення у Kotlin в консолі?
12. Поясніть терміни код-рев'ю та рефакторинг.

2 ЛАБОРАТОРНА РОБОТА № 2

ПРОГРАМУВАННЯ НА KOTLIN: ЗМІННІ ТА ТИПИ ДАНИХ, РЯДКИ, УМОВИ ТА ЦИКЛИ, ФУНКЦІЇ

Мета роботи – ознайомитися з основами програмування мовою Kotlin: вивчити змінні та типи даних, опрацювати роботу з рядками, умовними конструкціями та циклами, а також засвоїти принципи створення та використання функцій.

2.1 Загальні відомості

У процесі виконання завдань студенти набудуть практичних навичок написання коду, структуризації програмної логіки та ефективного використання мовних конструкцій Kotlin.

Первинна перевірка завдань з програмування буде проходити у автоматичному режимі на GitHub, тому потрібно чітко дотримуватися вимог до завдань. Якщо будуть виникати труднощі під час проходження лабораторної роботи, необхідно продивитися лекційний матеріал, звернутися до рекомендованої літератури або самостійно знайти потрібну інформацію в Інтернеті.

Отже, для виконання лабораторної роботи вам знадобляться наступні інструменти (див. лаб.роб. № 1):

- середовище розробки IntelliJ IDEA;
- GitHub акаунт;
- застосунок GitHub Desktop.

2.2 Оновлення власної копії репозиторію на GitHub

Оновлення власної копії репозиторію на GitHub є важливим кроком при командній роботі над реальними проєктами. Це дозволяє отримати найактуальніші зміни від інших розробників, уникнути конфліктів у коді та забезпечити узгоджену спільну роботу. Регулярне оновлення гарантує, що ви працюєте з найсвіжішою версією проєкту, що особливо важливо для ефективної командної роботи та збереження цілісності коду.

На першому кроці потрібно завантажити останні зміни з головного репозиторію, для цього відкрийте сторінку вашої копії репозиторію у GitHub та натисніть Sync fork (рисунк 2.1).

Sync fork – це процес оновлення вашої форкованої копії репозиторію на GitHub, щоб вона відповідала останнім змінам з оригінального репозиторію.

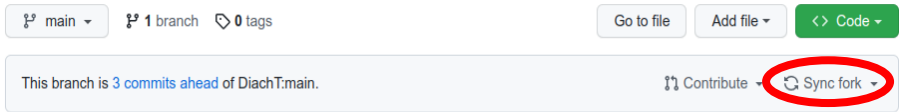


Рисунок 2.1 – Завантаження останніх змін з головного репозиторію

Далі потрібно завантажити зміни з GitHub у вашу локальну копію репозиторію. Для цього у застосунку GitHub Desktop натисніть Pull origin або зверху вікна, або в центрі вікна (рисунок 2.2).

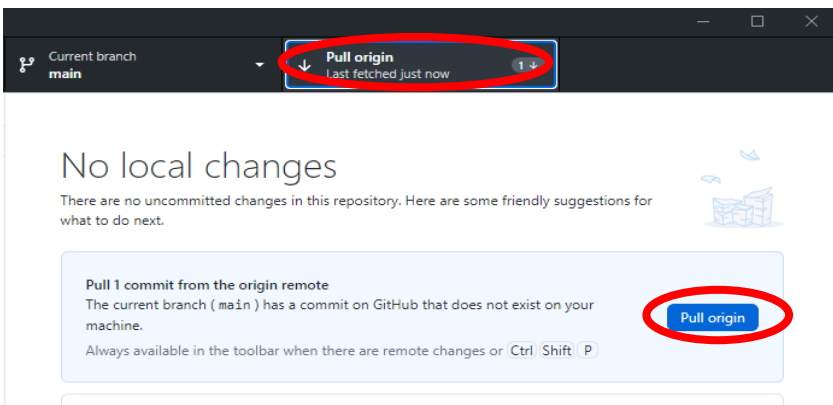


Рисунок 2.2 – Завантаження змін у локальну копію репозиторію

2.3 Відкриття багатомодульного Gradle проєкту

В цій лабораторній роботі ми познайомимось з багатомодульністю в Gradle проєктах. Багатомодульні Gradle проєкти дозволяють створювати більш складні та масштабні програми, що складаються з декількох модулів або компонентів, які можуть мати власний життєвий цикл та залежності.

Основні переваги багатомодульних Gradle проєктів:

1. Модульність: Багатомодульні проєкти дозволяють легко розділяти код на окремі модулі, що можуть мати власну логіку, залежності, тести та конфігурацію. Це дозволяє розвивати проєкт більш швидко та ефективно, оскільки різні модулі можуть бути розроблені паралельно, а також зменшує ймовірність помилок при редагуванні та рефакторингу коду.

2. Перевикористання: Код, написаний в одному модулі, може бути легко перевикористаний в іншому модулі. Наприклад, модуль, що містить інтерфейс користувача, може бути використаний в декількох інших модулях, що надають різну функціональність.

3. Залежності: Модулі можуть мати власні залежності від зовнішніх бібліотек, які можуть бути легко оновлені чи замінені при необхідності. Також можливо використовувати внутрішні залежності між модулями проєкту.

4. Тестування: Багатомодульні проєкти дозволяють проводити більш ефективне тестування, оскільки можна тестувати окремі модулі незалежно від інших. Це дозволяє швидше виявляти та виправляти помилки в коді.

5. Масштабованість: Багатомодульні проєкти можуть бути легко масштабовані, оскільки нові модулі можуть бути додані або вилучені без впливу на інші частини проєкту. Це дозволяє проєкту легко адаптуватися до змінних вимог та розширюватися з часом.

Отже, багатомодульні Gradle проєкти дозволяють створювати більш складні та масштабні програми з більш ефективним розвитком, тестуванням та масштабуванням.

Для початку роботи відкриємо IntelliJ IDEA та натиснемо File->Open, обираємо папку локального репозиторію (цього разу без helloworld), див. рисунок 2.3.

Після цього ви побачите проєкт в якому є головний модуль “KTests” та один вкладений модуль – “helloworld”, який ви створили у попередній лабораторній роботі (рисунок 2.4).

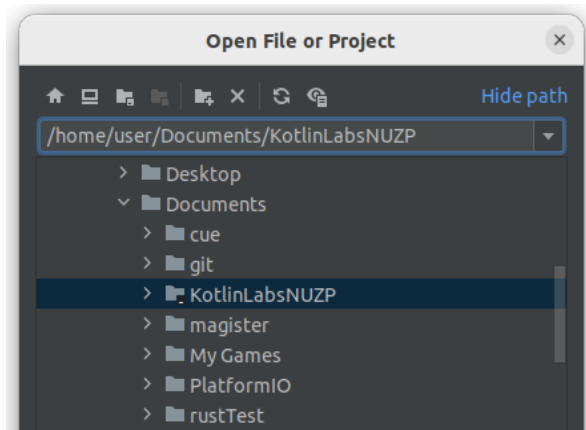


Рисунок 2.3 – Відкриття локального репозиторію в IntelliJ IDEA

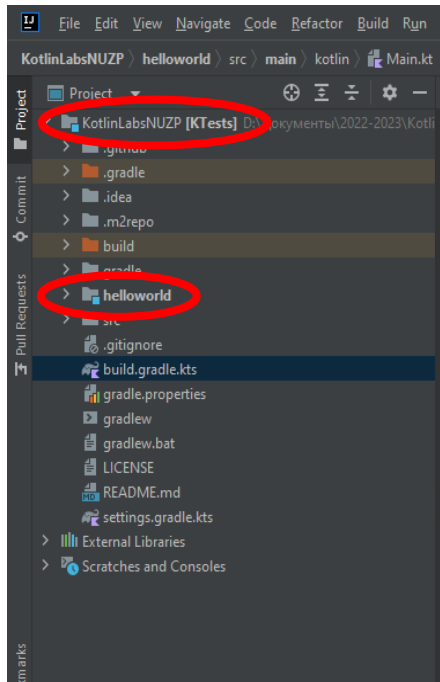


Рисунок 2.4 – Локальний репозиторій в IntelliJ IDEA

Перевіримо версію JDK, яка буде використовуватись системою збірки Gradle – її потрібно виставити в 17 або вищу версію. Відкриваємо налаштування IntelliJ IDEA (рисунок 2.5).

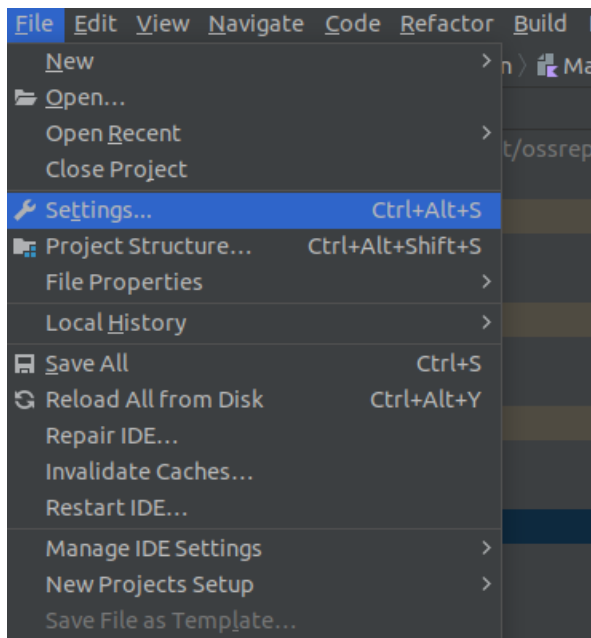


Рисунок 2.5 – Відкриття налаштувань проєкту в IntelliJ IDEA

Заходимо в розділ “Build,Execution, Deployment” -> Gradle та дивимось Gradle JVM (рисунок 2.6). Виставляємо в 17 або вищу версію та тиснемо OK.

Gradle JVM – це Java Virtual Machine (JVM), яка використовується для виконання завдань у Gradle, інструменті автоматизації побудови проєктів. Gradle підтримує виконання на Java і дозволяє налаштувати конкретну версію JVM для запуску завдань.

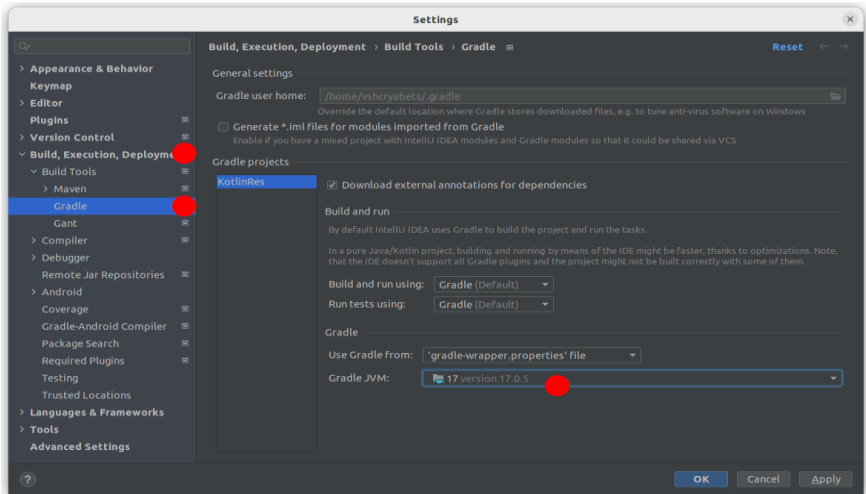


Рисунок 2.6 – Вибір версії Gradle JVM

Далі відредагуємо кореневий `build.gradle.kts` файл. Знайдіть в ньому рядок

```
val labNumber = 1
```

та виправте номер роботи на 2.

Розглянемо файл `helloworld/build.gradle.kts` та внесемо в нього зміни:

```
1: plugins {
2:     kotlin("jvm") version Versions.kotlin
3:     id("org.jetbrains.compose") version Versions.compose
4:     application
5: }
6:
7: group = "org.example"
8: version = "1.0-SNAPSHOT"
9:
10: repositories {
```

```

11:     mavenCentral()
12:     maven("https://maven.pkg.jetbrains.space/public/p/compose/dev")
13:     maven("file://${rootDir}/.m2repo/")
14: }
15:
16: dependencies {
17:     implementation(Versions.library)
18:     implementation(compose.desktop.currentOs)
19:     testImplementation(kotlin("test"))
20: }
21:
22: tasks.test {
23:     useJUnitPlatform()
24: }
25:
26: kotlin {
27:     jvmToolchain(Versions.jvmLevel)
28: }
29:
30: application {
31:     mainClass.set("MainKt")
32: }

```

Рядки 3,12,18 – під’єднаємо бібліотеку Jetpack Compose для побудови UI.

Рядки 26-28 – вмикаємо підтримку JDK 17.

В рядках 13, 17 – під’єднаємо бібліотеку для автоматичних тестів.

Після будь яких змін Gradle файлів проєкту потрібно його перевантажити. Зробити це можна двома шляхами:

- 1) може з’явиться плаваюча панель з кнопкою оновлення (рисунок 2.7);
- 2) якщо нічого такого немає, то можна відкрити панель Gradle (зазвичай вона вгорі правої панелі інструментів IDEA) та натиснути кнопку примусового оновлення в ній (рисунок 2.8).

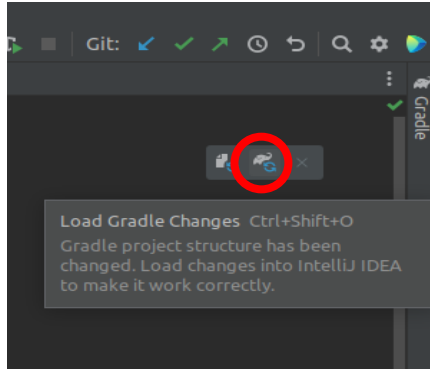


Рисунок 2.7 – Плаваюча панель з кнопкою оновлення Gradle

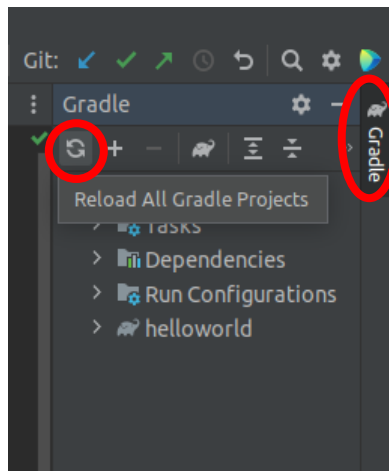


Рисунок 2.8 – Панель Gradle

Для виконання завдання потрібно відредагувати Main.kt файл вашого helloworld проекту наступним чином:

```
1: import com.diacht.ktest.compose.startTestUi
2: import me.tetiana.helloworld.BuildConfig
3:
```

```

4: fun seed(): String = "user"
5:
6:
7: fun labNumber() : Int = BuildConfig.LAB_NUMBER
8:
9:
10: fun main(args: Array<String>) {
11:     println("Лабораторна робота №${labNumber()}
        користувача ${seed()}")
12:
13:     startTestUi(seed(), labNumber())
14: }

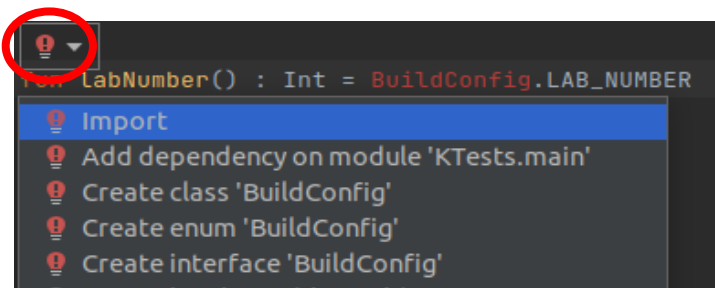
```

Рядок 4 – впишіть ім'я свого GitHub користувача у функцію seed();

Рядок 7 – використаємо номер лабораторної роботи, який ми задали на попередніх кроках в головному build.gradle.kts;

Рядок 13 – виклик функції для генерації завдання на лабораторну роботу;

Рядки 1 та 2 – з'являться автоматично при імпорті класів, якщо цього не трапилось то встановіть текстовий курсор на рядок з помилкою, після чого має з'явитись “червона лампа”. Якщо на неї натиснути то з'являється варіанти вирішення проблеми, нам потрібно обрати Import (рисуюнок 2.9).



Рисуюнок 2.9 – Варіанти вирішення проблем в IntelliJ IDEA

Якщо ви все зробили правильно, то при запуску Main.kt на екрані з'явиться вікно з вашим варіантом завдання на лабораторну роботу (рисунок 2.10).

2.4 Завдання для самостійної роботи

Потрібно зробити завдання згідно вашого варіанта на лабораторну роботу. Приклад такого завдання наведено на рисунку 2.10.

Завдання складається з трьох частин і відповідно має три вкладки. Для першої частини завдання число змінних, значення за замовчанням та вигляд функції є варіативними та залежать від ідентифікатора студента. Базовою є умова створити функцію з певним типом та назвою. Друга та третя частини завдання також мають базові та варіативні умови.

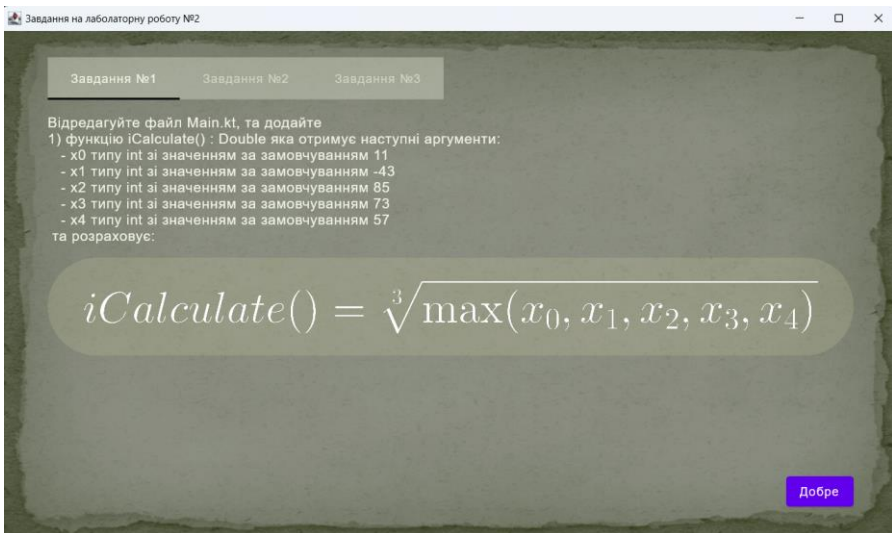


Рисунок 2.10 – Вікно з варіантом завдання на лабораторну роботу № 2

Для написання математичних формул вам буде в нагоді пакет математичних функцій та констант <https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.math/>.

Далі необхідно додати 3 функції у Main.kt згідно вашого завдання, відладити та перевірити ваші функції. Ці функції повинні бути у глобальному просторі імен.

Зауважимо, що для завдання з рядками, нумерація у рядках починається з 0, тобто парні елементи це 0, 2, 4 ...

Після реалізації та перевірки функцій, необхідно завантажити новий код на GitHub та зробити пул-реквест (як це зробити див. лаб.роб. № 1 розділи 1.8 та 1.9). На даному етапі електронну частину другої лабораторної роботи завершено.

2.5 Оцінка виконання лабораторної роботи

В цій лабораторній роботі за електронну частину роботи ви отримуєте:

- якщо у вас вийшло зробити pull request і ви коректно вказали варіант та логін на GitHub, у Main.kt є 3 функції з потрібними назвами, то +1 бал;
- якщо перша функція пройшла автоматичний тест, то +2 бали;
- якщо перша функція задовольняє всім умовам при перевірці викладачем, то +1 бал;
- якщо друга функція пройшла автоматичний тест, то +2 бали;
- якщо друга функція задовольняє всім умовам при перевірці викладачем, то +1 бал;
- якщо третя функція пройшла автоматичний тест, то +3 бали;
- якщо третя функція задовольняє всім умовам при перевірці викладачем, то +1 бал.

Також потрібно зробити звіт та загрузити його у Moodle на сторінку курсу “Основи програмування на Kotlin”.

За звіт ви можете отримати максимально 6 балів.

Таким чином, за виконання другої лабораторної роботи, ви можете отримати максимально 17 балів, які будуть зараховані до загальної оцінки з поточної дисципліни.

2.6 Зміст письмового звіту

1. Титульна сторінка.
2. Мета роботи.

3. Скріншоти всіх завдань згідно з вашим варіантом (приклад на рис. 2.10).
4. Програмний код з текстовим описом ваших трьох функцій.
5. Блок-схема алгоритму роботи функції з рядками.
6. Пояснення як ви перевіряли роботу ваших функцій, та на яких наборах даних.
7. Скріншот пул-реквеста.
8. Відповіді на контрольні питання.

2.7 Контрольні питання

1. З якими труднощами ви зіткнулися при виконанні цієї лабораторної роботи?
2. Як у Kotlin реалізуються умови та цикли?
3. Що таке значення за замовчуванням у функції?
4. Напишіть про рядкові літерали у Kotlin з прикладами.
5. Шаблони рядків, що це, приклади?
6. Що таке статична типізація?
7. Що таке автоматичне виведення типів?
8. Які типи даних є у Kotlin?
9. Що таке Gradle та для чого потрібен?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Iyanu Adelekan – Kotlin Programming By Example: Packt Publishing, 2018. – 500с.
2. Dmitry Jemerov, Svetlana Isakova – Kotlin in Action: Manning Publications, 2017 – 360с.
3. Pierre-Yves Saumont – The Joy of Kotlin: Manning Publications, 2019 – 480с.
4. Офіційний сайт мови Kotlin (англ.) [Електронний ресурс]/ Режим доступа: <https://kotlinlang.org/>
5. KOTLIN REFERENCE MANUAL – Переклад укр. – Арсеній Чеботарьов [Електронний ресурс]/ Режим доступа: <http://ac2epsilon.github.io/TRANS/KOTLIN/KotlinReference.html> – Київ 2018
6. GitHub Docs [Електронний ресурс]/ Режим доступа: <https://docs.github.com/en>
7. Git [Електронний ресурс]/ Режим доступа: <https://git-scm.com>