**УНІВЕРСИТЕТ ІМЕНІ АЛЬФРЕДА НОБЕЛЯ**

**КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

|  |
| --- |
| Представлено на кафедру\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| *(дата, підпис секретаря кафедри)* |
| Рецензування:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| *(кількість балів, "до захисту" ("на доопрацювання"),*  *дата, підпис керівника курсової роботи)* |
| Захист:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| *(кількість балів, дата, підпис зав. кафедрою)* |
| Підсумкова оцінка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| *(кількість балів, оцінка за 4-х бальною системою, дата, підпис зав. кафедрою)* |

**Курсова РОБОТА**

з дисципліни

«Основи програмування (Об'єктно-орієнтоване програмування)»

на тему «Розробка додатку ,,логування,,»

Здобувача Долинський Даниїла

Групи КН-22-2 (2)

Керівник курсової роботи:   
Ю.М. Барташевська, к.е.н., доц.

Дніпро

2023

**Зміст**

Оглавление

[**ВСТУП** 3](#_Toc153516869)

[**Мета** 3](#_Toc153516870)

[**Логування в інформаційних технологіях** 3](#_Toc153516871)

[**Роль логування** 3](#_Toc153516872)

[**Перша Згадка Логування** 4](#_Toc153516873)

[**Стан Логування Зараз** 4](#_Toc153516874)

[**Види Логів** 6](#_Toc153516875)

[**Логи Помилок (Error Logs)** 6](#_Toc153516876)

[**Логи Відлагодження (Debug Logs)** 6](#_Toc153516877)

[**Логи Інформації (Info Logs)** 6](#_Toc153516878)

[**Логи Попереджень (Warning Logs)** 6](#_Toc153516879)

[**Логи Аудиту (Audit Logs)** 6](#_Toc153516880)

[**Логи Активності Користувачів (User Activity Logs)** 7](#_Toc153516881)

[**Логи Продуктивності (Performance Logs)** 7](#_Toc153516882)

[**Логи Запитів (Query Logs)** 7](#_Toc153516883)

[**Логи Безпеки (Security Logs)** 7](#_Toc153516884)

[**Логи Системи (System Logs)** 7](#_Toc153516885)

[**РОЗДІЛ 2 Підготовча частина** 8](#_Toc153516886)

[**ВИСНОВКИ** 26](#_Toc153516887)

# **ВСТУП**

## **Мета**

### **Логування в інформаційних технологіях**

Логування (англ. logging) в інформаційних технологіях — це процес реєстрації подій, виникших під час роботи програми чи системи. Логи, які отримуються під час цього процесу, називаються логами. Логи містять інформацію, яка може бути корисною для аналізу та вирішення проблем.

### **Роль логування**

Логування відіграє критичну роль в розробці, тестуванні та підтримці програм. Основні функції логування включають:

1. **Відстеження Подій:** Запис подій відображає важливі моменти у виконанні програми, такі як помилки, операції користувачів, або зміни стану системи.
2. **Аналіз та Діагностика:** Логи дозволяють розробникам виявляти та аналізувати проблеми, виникаючі в процесі роботи програми.
3. **Підтримка та Відладка:** Інформація у логах є корисною для відлагодження (debugging) та виправлення програмних помилок.
4. **Моніторинг та Аудит:** Логи використовуються для моніторингу роботи системи та аудиту важливих дій користувачів.

### **Перша Згадка Логування**

Не існує конкретної інформації щодо перших осіб чи часів, коли було введено поняття логування в інформаційній технології. Концепція логування розвивалася органічно, і виникла від потреб розробників у відстеженні та реєстрації подій у своїх програмах та системах.

Зазвичай, розробники виявляли потребу в логуванні, коли їм було необхідно вирішувати проблеми в їх програмах або системах. Введення логів спрощувало процес відлагодження (debugging) та аналізу роботи програми в реальному часі. Логування також виявилося корисним для моніторингу, аудиту та відслідковування подій для покращення продуктивності та надійності систем.

Спочатку, логи, можливо, були реалізовані простими методами виводу інформації на екран консолі чи в файли. З розвитком програмування та з'явленням різних мов програмування, бібліотек та інструментів для логування стали більш стандартизованими.

Важливо відзначити, що логування виникло не як ізольована інновація від конкретних людей чи у конкретний період часу, а більше як еволюція практик та потреб розробників у процесі розвитку інформаційних технологій

### **Стан Логування Зараз**

На сьогоднішній день, із зростанням складності програмних систем та вимог до їх надійності, логування стало невід'ємною частиною розробки програмного забезпечення. Сучасні системи логування пропонують розширені можлиності, спрощуючи не лише процес відлагодження, а й підтримку та моніторинг системи в реальному часі.

Однією з ключових характеристик сучасних систем логування є можливість конфігурування рівнів деталізації. Розробники можуть визначати, які типи повідомлень (DEBUG, INFO, WARNING, ERROR, CRITICAL) вони хочуть включати в логи. Це дозволяє відключати або включати різні рівні в залежності від обставин та етапу розробки.

Сучасні системи логування підтримують різноманітні обробники, які визначають, куди і які дані будуть виводитися. Наприклад, логи можуть виводитися на консоль для відладки, у файли для архівування та аналізу, або навіть у спеціалізовані системи моніторингу.

Логери стали більш конфігуруємими та можуть використовувати різні форматувальники для легшого аналізу інформації. Розробники можуть налаштовувати формат виводу логів, додаючи додаткові дані, такі як час, рівень, інформація про модуль чи клас, та інше.

Використання об'єктно-орієнтованого підходу у системах логування дозволяє використовувати класи для представлення різних компонентів, що робить їх більш гнучкими та легко розширюваними. Це спрощує використання логування в різних частинах програми та сприяє підтримці коду.

Багато сучасних систем логування можуть інтегруватися з іншими інструментами для моніторингу та аналізу, такими як ELK (Elasticsearch, Logstash, Kibana), Splunk, або Prometheus з Grafana. Це надає розробникам зручні інструменти для візуалізації та аналізу логів.

Узагальнюючи, сучасні системи логування стали невід'ємною частиною розробки програмного забезпечення, надаючи розробникам потужні інструменти для відстеження, аналізу та вдосконалення продуктивності та надійності систем. Об'єктно-орієнтований підхід та розширені можливості конфігурації роблять сучасні системи логування ефективним та потужним інструментом у розробці програмного забезпечення.

# **РОЗДІЛ 1 Теоретична частина**

## **Види Логів**

Види логів представляють різні категорії інформації, які фіксуються для подальшого аналізу та моніторингу в програмуванні та системному адмініструванні. Логи включають логи помилок для реєстрації невдалих подій, логи відлагодження для деталізації виконання коду, інформаційні логи для запису загальних подій, а також логи безпеки, аудиту та інші, що забезпечують повноту та різноманіття інформації для ефективного виявлення та вирішення проблем.

### **Логи Помилок (Error Logs)**

* Опис: Логи помилок використовуються для фіксації помилок та винятків, які виникають під час виконання програми або системи. Це допомагає розробникам швидко виявляти та виправляти проблеми для забезпечення надійності та стабільності програмного забезпечення.
* Приклад: Запис про помилку бази даних: "Error: Unable to establish a connection to the database."

### **Логи Відлагодження (Debug Logs)**

* Опис: Використовуються для деталізації роботи програми під час відлагодження. Вони містять важливі дані, які допомагають розробникам визначити точне місце та хід виникнення проблеми.
* Приклад: "DEBUG: Entering function calculate\_total() with parameters x=10, y=5."

### **Логи Інформації (Info Logs)**

* Опис: Логи інформації містять загальну корисну інформацію про стан системи або програми, яка може бути корисною для моніторингу та аналізу.
* Приклад: "INFO: Module 'auth' successfully loaded."

### **Логи Попереджень (Warning Logs)**

* Опис: Використовуються для фіксації потенційно проблемних ситуацій, які не є критичними, але можуть призвести до проблем в майбутньому.
* Приклад: "WARNING: Deprecated method used. This method will be removed in the next version."

### **Логи Аудиту (Audit Logs)**

* Опис: Використовуються для фіксації дій користувачів або системних подій для забезпечення безпеки та відслідковування змін.
* Приклад: "AUDIT: User John changed access rights to file 'document.txt'."

### **Логи Активності Користувачів (User Activity Logs)**

* Опис: Записують дії користувачів у системі для відстеження їх активності та входження.
* Приклад: "USER ACTIVITY: User Sarah logged into the system."

### **Логи Продуктивності (Performance Logs)**

* Опис: Логують дані про продуктивність системи, такі як час виконання операцій, що допомагає виявляти та виправляти бутланеки.
* Приклад: "PERFORMANCE: Operation completed in 50 ms."

### **Логи Запитів (Query Logs)**

* Опис: Використовуються для фіксації SQL-запитів до бази даних, допомагаючи аналізувати та вдосконалювати роботу з базою даних.
* Приклад: "QUERY: SELECT \* FROM users WHERE username='john\_doe'."

### **Логи Безпеки (Security Logs)**

* Опис: Логи безпеки фіксують події, пов'язані з безпекою системи, такі як спроби вторгнення чи невдалий доступ.
* Приклад: "SECURITY: Failed login attempt by user Admin from IP address 192.168.0.1."

### **Логи Системи (System Logs)**

* Опис: Фіксують інформацію про стан системи та роботу апаратного забезпечення.
* Приклад: "SYSTEM: Disk C: remaining space - 20%, RAM usage - 75%."

# **РОЗДІЛ 2 Підготовча частина**

**Постановка задачі**

Метою цього проекту є створення власного додатку логування, який буде взаємодіяти з програмами та системами для фіксації та аналізу подій. Цей додаток буде розширювати можливості стандартного логування та надавати гнучкість у керуванні та обробці журналів.

## **Основні Кроки та Компоненти Проекту:**

### **Визначення Функціональності:**

Розгляд можливостей логування та обрання функціональності, яку ми хочемо підтримувати.

Розробка плану роботи та основних функцій додатку логування.

### **Створення Проекту:**

Створення нового проекту в обраному середовищі розробки (наприклад, IntelliJ IDEA).

Встановлення залежностей та бібліотек для розробки.

### **Реалізація Логування:**

Використання мови програмування (Java) для створення основної логіки додатку логування.

Вибір структури та формату виводу логів.

### **Створення Власних Апендерів:**

Розробка власних апендерів для розширення можливостей логування.

Вивчення принципів додавання власних обробників подій.

### **Конфігурування та Налаштування:**

Реалізація конфігураційних опцій для налаштування різних аспектів логування.

Визначення рівнів деталізації, виводу та форматування логів.

### **Інтеграція з Проектами:**

Впровадження додатку логування в реальні проекти чи системи.

Тестування та аналіз виводу логів в різних сценаріях.

### **Документація та Звіт:**

Підготовка документації з визначенням основних аспектів архітектури та функціональності додатку логування.

Складання звіту з результатами роботи.

## **Середовище розробки**

Середовище розробки є важливим кроком при створенні програмного забезпечення. Для цього проекту обрано VS Code, як інтегроване середовище розробки.

Visual Studio Code (VS Code) — це легкий та потужний текстовий редактор, який надає розширені можливості для розробки різноманітних мов програмування, включаючи Java. Ось детальніше, чому вибір VS Code може бути обгрунтований для вашого Java-проекту:

1. **Легкість та Швидкодія:**
   * VS Code є легким та швидкодією інтегрованим середовищем розробки. Він завантажується швидко та не вимагає великої кількості ресурсів, що забезпечує ефективну роботу навіть на менш потужних системах.
2. **Розширюваність:**
   * VS Code має багато розширень, що дозволяє розробникам вибирати саме ті інструменти та функції, які їм потрібні. Для розробки на Java ви можете встановити розширення, такі як "Language Support for Java" та "Debugger for Java", що полегшують роботу з мовою та відлагодження коду.
3. **Інтеграція з Maven та Gradle:**
   * VS Code добре інтегрується з інструментами для автоматизації збирання проектів, такими як Maven та Gradle. Ви можете легко налаштовувати та виконувати ваші проекти з цими інструментами прямо з VS Code.
4. **Підтримка Git та Інших Інструментів Контролю Версій:**
   * VS Code надає вбудовану підтримку для Git та інших систем контролю версій. Це полегшує вам ведення історії проекту, виявлення змін та співпрацю з іншими розробниками.
5. **Багатофункціональність Розширень:**
   * За допомогою розширень, ви можете використовувати VS Code не тільки для розробки Java, а й для інших завдань, таких як робота з базами даних, розробка веб-сайтів, робота з Docker, тощо.
6. **Активна Спільнота та Підтримка:**
   * VS Code має велику та активну спільноту розробників. Це означає, що ви можете легко знайти рішення для своїх питань, а також користуватися регулярними оновленнями та виправленнями помилок.
7. **Крос-платформенність:**
   * VS Code є крос-платформеним, тобто ви можете використовувати його на операційних системах Windows, macOS та Linux. Це дозволяє розробникам працювати на тій платформі, яка їм найбільше вигідна.

**Мова програмування**

Мова програмування Java версії 21 обрана для розробки проекту. Java - це високорівнева мова програмування, відома своєю переносимістю та безпекою. Використання останньої версії Java надає доступ до нововведень та вдосконалень, що поліпшують якість та функціональність коду.

## **Парадигма Логування**

У проекті використовується парадигма логування, реалізована за допомогою бібліотеки Log4j. Log4j є потужним інструментом для логування в Java, що дозволяє ефективно вести журнали подій та моніторити роботу програми.

### **Деталізація та Рівні Логування:**

* + Використання різних рівнів логування (ERROR, WARN, INFO, DEBUG, TRACE) дозволяє налаштовувати рівень деталізації в залежності від потреб розробки чи експлуатації. Це забезпечує баланс між достатньою інформацією для виявлення помилок та ефективністю роботи системи.

### **Категорії та Ізоляція Логів:**

* + Використання категорій логування дозволяє ізолювати різні компоненти системи та виводити логи тільки для конкретних аспектів. Це полегшує аналіз та вирішення проблем у конкретних модулях чи функціональних частинах системи.

### **Конфігураційна Гнучкість:**

* + Можливість динамічно змінювати конфігурацію логування без перезапуску системи робить Log4j гнучким інструментом. Це особливо важливо в умовах реального часу, коли необхідно відслідковувати та адаптувати вивід логів на льоту.

### **Масштабованість та Витрати Ресурсів:**

* + Log4j забезпечує ефективність витрат ресурсів, що особливо важливо в великих та високонавантажених системах. Його масштабованість дозволяє використовувати логування в різноманітних середовищах без значного впливу на продуктивність.

### **Адаптабельність до Виводу Логів:**

* + Бібліотека Log4j підтримує апендери, що дозволяє направляти логи у різноманітні вихідні пункти, такі як консоль, файли, бази даних та інші. Це дозволяє розробникам адаптувати вивід логів під конкретні потреби та вимоги проекту.

### **Підтримка Спільноти та Розвиток:**

* + Log4j є широко використовуваною та підтримуваною бібліотекою в спільноті розробників Java. Його активний розвиток та велика спільнота гарантують надійність та актуальність інструменту.

Загалом, парадигма логування, яку надає Log4j, забезпечує інструменти для зручного та ефективного контролю за подіями та виводом інформації у великих, складних системах.

## **Додаткові бібліотеки**

Використання бібліотек Log4j та власного апендера (MyAppender) в проекті має кілька важливих переваг:

### **Розширення Функціональності Log4j:**

* + Бібліотека Log4j надає потужний інструментарій для логування в Java. Використання версії 1.2.17 дозволяє користуватися вже визнаним та стабільним рішенням з багатофункціональними можливостями.

### **Можливість Налаштування Різноманітних Апендерів:**

* + Бібліотека Log4j дозволяє визначити різні апендери для виведення логів у різні вихідні пункти. Використання власного апендера (MyAppender) розширює ці можливості, надаючи вивід повідомлень на консоль. Це особливо корисно для швидкого виводу інформації під час відладки чи тестування.

### **Контроль за Виведенням Логів:**

* + Власний апендер дозволяє розробникам контролювати та змінювати механізм виведення логів. Наприклад, виведення повідомлень на консоль може бути корисним при розробці та тестуванні, але для продуктивного середовища може використовуватися інший апендер (наприклад, для виведення в файли чи бази даних).

### **Легкість Налаштування та Використання:**

* + Власний апендер може бути легко доданий та налаштований у конфігураційному файлі Log4j. Це забезпечує зручність та простоту у використанні, а розширення можливостей логування може бути використано за потреби проекту.

### **Підвищення Читабельності та Розуміння Коду:**

* + Використання власного апендера може полегшити розуміння коду та структури логів. Це особливо корисно для розробників, які працюють над проектом, спрощуючи їм взаємодію з механізмами логування.

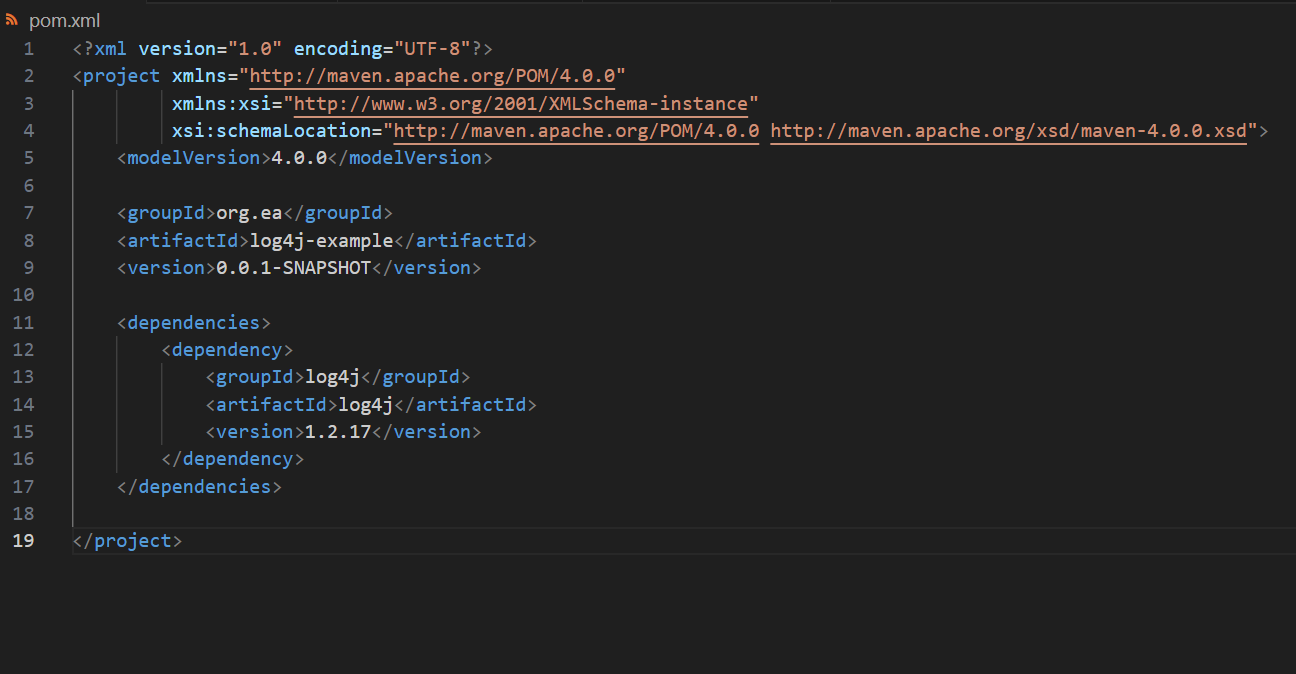
Всі ці аспекти дозволяють більш гнучко та ефективно використовувати інструменти логування в контексті конкретного проекту, забезпечуючи при цьому зручність, стабільність та можливість вдосконалення.

# **РОЗДІЛ 3 Практична частина**

## **Додаємо Залежність у `pom.xml`**

У файлі **pom.xml** ви вказуєте залежності вашого проекту, тобто бібліотеки, які використовуються у вашому коді. Додаючи залежність, ви дозволяєте Maven автоматично завантажувати та включати ці бібліотеки у ваш проект.

У вашому конкретному випадку ви додали залежність для бібліотеки Log4j версії 1.2.17.



* **<groupId>:** Це ідентифікатор групи артефакта (бібліотеки). У випадку Log4j, група - це **log4j**. Група - це організація чи команда, яка випускає бібліотеку.
* **<artifactId>:** Це ідентифікатор артефакта (конкретної бібліотеки) всередині групи. У вас **log4j** - це основна бібліотека Log4j.
* **<version>:** Це версія бібліотеки. У вас вказана версія **1.2.17**.

Коли ви вказуєте ці три параметри, Maven автоматично завантажує бібліотеку Log4j версії 1.2.17 з репозиторію Maven та додає її у ваш проект.

Це дозволяє вам користуватися функціональністю Log4j у вашому коді без необхідності завантаження бібліотеки вручну чи включення її в репозиторій проекту.

## **Використання Log4j**

**Log4JExample** - це основний клас вашого проекту, який демонструє використання бібліотеки Log4j для логування подій у вашому Java-додатку. Давайте докладніше розглянемо цей клас:



**Клас Log4JExample має такі елементи:**

1. **Логер (Logger):**

Створюємо глобальний об'єкт логера, пов'язаний із класом **Log4JExample**. Цей логер буде використовуватися для логування подій у цьому класі.

1. **Макет для логування (Pattern for Logging):**

Визначаємо макет для форматування логів. У вашому випадку **%d** відображає дату та час, **%p** - рівень логування, **%c** - категорія логу, **%C{1}** - ім'я класу (перше слово), **%m** - повідомлення, **%n** - новий рядок.

1. **Консольний апендер (ConsoleAppender):**

Створюємо консольний апендер, налаштовуємо його макет та рівень логування, активуємо опції.

1. **Додавання апендерів до кореневого логера:**

Додаємо консольний апендер та власний апендер (**MyAppender**) до кореневого логера. Таким чином, обидва апендери будуть використовуватися для логування.

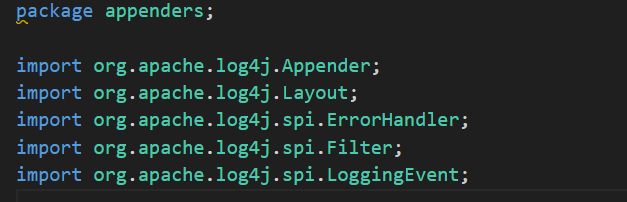
1. **Логування подій:**

Викликаємо метод **info** для логування повідомлення рівня INFO. Це повідомлення буде оброблено всіма доданими апендерами.

У цьому коді ви встановлюєте конфігурацію логування для вашого проекту, додаєте апендери та логуєте події. Вивід логів буде відображено згідно налаштувань, які ви визначаєте у **log4j.properties** файлі або коді (у вашому випадку, у коді).

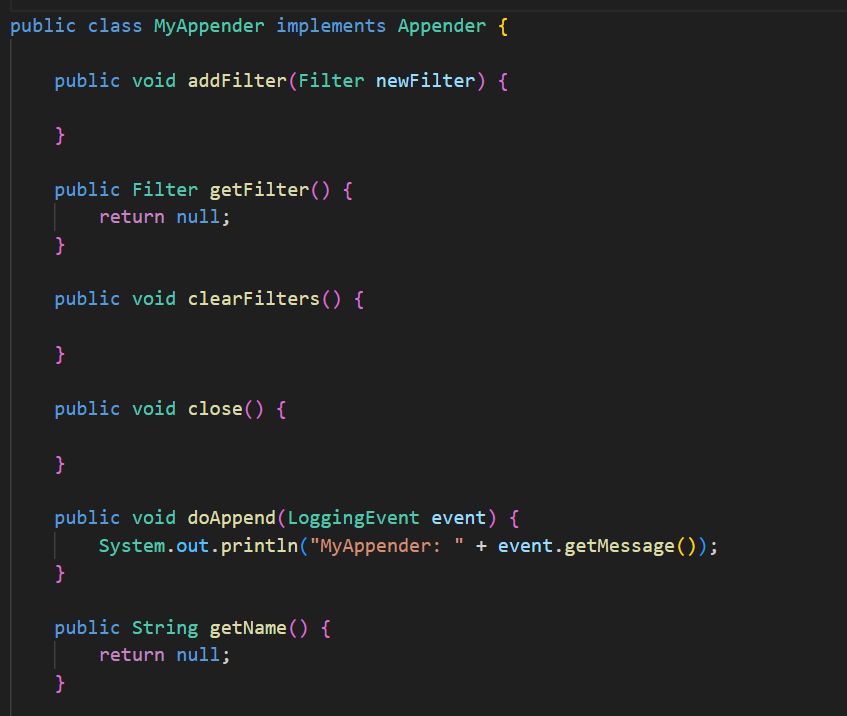
## **Створюємо власне доповнення**

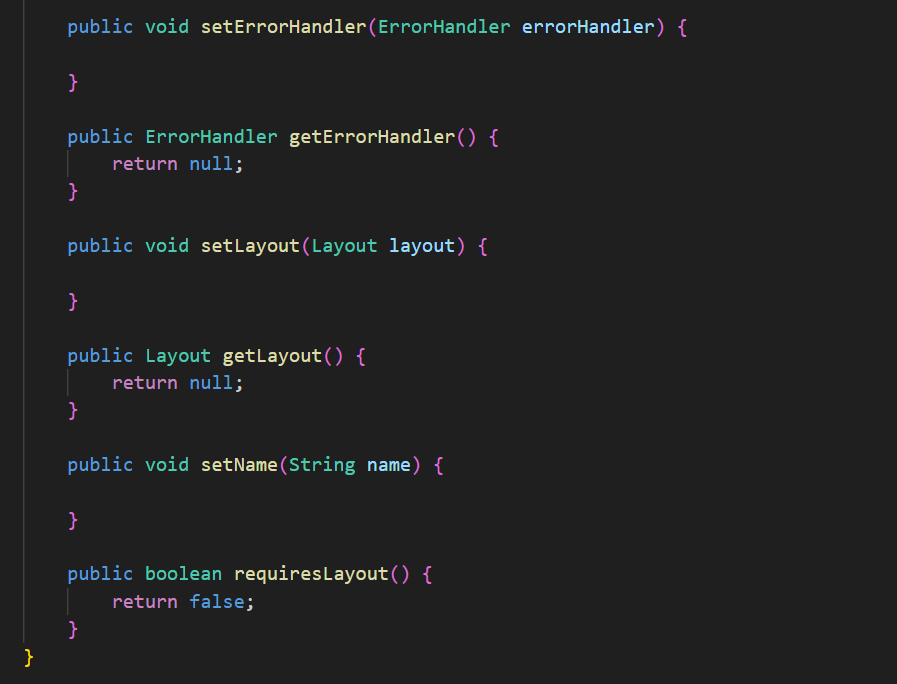
**MyAppender.java** - це клас, який ви визначили як власний апендер для бібліотеки Log4j.



1. **Імплементація інтерфейсу Appender:**

**MyAppender** реалізує інтерфейс **Appender**. Це означає, що ми повинні надати реалізацію методів, які вимагає цей інтерфейс.





1. **Метод doAppend:**

Цей метод викликається при кожному логуванні події. У вашому випадку, він виводить повідомлення на консоль разом з текстом "MyAppender:" та текстом логування.

## **ВИСНОВКИ**

У цій курсовій роботі я розібрав тему «Розробка додатку ,,Текстовий квест,,». У першому розділі я описав та розібрав види текстового квесту та надав їх прикалади та найближчі сьогоденні аналоги. Далі ,після постановки задачі, я розповів про основні засади та принципи дії парадигми MVC, про основні плюси мови Java, та її графічну бібліотеку JavaFX, тобто розповів про все, що потрібно для створення програми такого типу. У практичній частині я завдяки скріншотам свого проєкту показав як написана та влаштована програма. Розроблений додаток муже слугувати макетом або шаблоном для тих, хто тільки починає свій шлях у створенні ігор, оскільки її функціонал простий та не має в собі якісь поглибленні знання.

**ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Документація Java 21 <https://docs.oracle.com/en/java/javase/21/>
2. Стаття JavaFX <https://javarush.com/groups/posts/2560-vvedenie-v-java-fx>
3. Сайт зі статтями Metanit <https://metanit.com/>
4. Стаття JavaFX https://docs.oracle.com/javafx/2/get\_started/hello\_world.htm