ВІСНИК ПРИАЗОВСЬКОГО ДЕРЖАВНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
2025р. Серія: Технічні науки Вип. 51
p-ISSN: 2225-6733; e-ISSN: 2519-271X

122 КОМП'ЮТЕРНІ НАУКИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.514+ 004.4'24 **DOI: 10.31498/2225-6733.51.2025.344593**

РОЗРОБКА ШАБЛОНІВ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ НА ОСНОВІ ПАТЕРНІВ ПРОЄКТУВАННЯ

Соколова Н.О. *канд. техн. наук, доцент, Національний технічний університет «Дніпровська політехніка», м. Дніпро, ORCID: https://orcid.org/0000-0003-2493-3553, e-mail: n.olegowna@gmail.com;*

Хара Г.Л. *магістр, Національний технічний університет «Дніпровська політехніка», м. Дніпро, ORCID: https://orcid.org/0009-0000-1165-901X, e-mail: kharaherman@gmail.com;*

Балалаєва О.Ю. *канд. техн. наук, доцент, ДВНЗ «Приазовський державний технічний університет», м. Дніпро, ORCID: https://orcid.org/0000-0003-1461-4399, e-mail: balalaeva_e_u@pstu.edu;*

Олевський В.І. *д-р техн. наук, професор, Національний технічний університет «Дніпровська політехніка», м. Дніпро, ORCID: https://orcid.org/0000-0003-3824-1013, e-mail: olevskyi.v.i@nmu.one*

Дана стаття присвячена дослідженню питання розробки шаблонів інтерфейсів із використанням патернів проєктування. Актуальність дослідження зумовлена постійним зростанням кількості програмних продуктів у різних сферах, що вимагає створення ефективних, гнучких та масштабованих архітектур. Розробка шаблонів користувацького інтерфейсу на основі патернів проєктування є багатогранною темою, яка охоплює адаптивність, спеціалізовані рішення, інтеграцію сучасних технологій і повторне використання існуючих рішень. Розглянуті джерела демонструють, як патерни можуть бути адаптовані до різних контекстів, від гейміфікації до аналізу зображень і метавсесвіту. Це підкреслює їхню універсальність і важливість у сучасному дизайні UI. Наведений опис трьох шаблонів інтерфейсів для різної взаємодії користувача з програмними продуктами, зокрема для кросплатформної програми з надзвичайною інформацією, рекламного застосунку та файлового редактора. Для кожного з них описано вибір патернів, ієрархію та взаємодію класів та особливості реалізації інтерфейсу. Дослідження показало, що використання породжувальних патернів проєктування Фабричний метод та Абстрактна фабрика, а також поведінкового патерну Команда, дозволяє значно підвищити гнучкість, масштабованість і підтримуваність програмних продуктів, а також сприяє скороченню часу розробки та підвищенню якості кінцевого продукту. Дана робота демонструє практичну цінність патернів проєктування для розробки сучасних, ефективних та зручних користувацьких інтерфейсів, а також підтверджує доцільність їх використання у реальних програмних проєктах. Практичне значення роботи полягає у можливості використання отриманих результатів як діючими розробниками, так і студентами для створення ефективних, функціональних і зручних інтерфейсів, а також для поглибленого вивчення дисциплін, пов'язаних із патернами проєктування та архітектурою програмного забезпечення.

Ключові слова: *інтерфейс користувача, патерни проєктування, ієрархія класів, Java, програмний продукт, шаблони інтерфейсу, кросплатформність.*

Постановка проблеми

Щодня з'являється все більше нових розробників, які створюють різноманітне програмне забезпечення для бізнесу, освіти, медицини, спорту чи розваг. Найважливішим і найскладнішим етапом є розробка архітектури додатку. Протягом багатьох років розробники стикалися зі схожими проблемами під час проєктування, і з часом ці труднощі були систематизовані у вигляді патернів проєктування. З подачі «банди чотирьох»: Еріха Гамма, Річарда Хелма, Ральфа Джонсона, Джона Вліссідеса, які написали у 1994 році книгу «Шаблони проєктування: Елементи повторно використовуваного об'єктно-орієнтованого програмного забезпечення» (англ. «Design Patterns: Elements of Reusable Object-Oriented Software»), патерни стали універсальними рішеннями для типових архітектурних задач, допомагають розширювати функціонал додатків, зменшують обсяг коду та дозволяють уникати типових помилок, навчаючись на досвіді інших.

Розробка шаблонів користувацького інтерфейсу (англ. User Interface, UI) на основі патернів проєктування є важливим напрямом у сучасному дизайні програмного забезпечення. Патерни проєктування забезпечують стандартизовані рішення для типових проблем, що виникають у процесі створення інтерефейсів. Вони дозволяють дизайнерам адаптувати готові шаблони під конкретні завдання та контекст проєкту, що

p-ISSN: 2225-6733; e-ISSN: 2519-271X

підвищує якість продукту і зменшує час розробки [1, 2]. Сучасні бібліотеки UI-патернів і дизайн-системи дозволяють швидко знаходити та впроваджувати найкращі практики, забезпечуючи високу якість та уніфікованість інтерфейсів у різних продуктах. Саме тому дослідження можливості впровадження патернів – це важлива складова професійної розробки шаблонів інтерфейсів.

Аналіз останніх досліджень та публікацій

Розробка шаблонів користувацького інтерфейсу на основі патернів проєктування є ключовим підходом для створення ефективних, зручних і послідовних інтерфейсів. Патерни дизайну інтерфейсів – це повторювані рішення, які допомагають вирішувати типові проблеми взаємодії користувача з програмою або сайтом, забезпечуючи стандартизовані, перевірені часом методи [3-5].

Сьогодні разом зі стандартними патернами проєктування розроблені та використовуються специфічні патерни для розробки інтерфейсів [1, 2, 6], які можуть використовуватися в різних контекстах розробки шаблонів UI.

У роботі [7] розглядаються патерни дизайну інтерфейсів для взаємодії між людьми та агентами (англ. Human-Agent Teaming, HAT). Автори виділили 25 патернів, які охоплюють такі аспекти, як планування, просторову орієнтацію, управління завданнями та комунікацію. Ці патерни можуть бути використані для створення мов дизайну HAT, що є корисним для розробників UI. Патерни для аналізу зображень з камерловушок, які допомагають біологам ефективно класифікувати дані [8]. Цей підхід демонструє, як патерни можуть бути адаптовані до вузькоспеціалізованих завдань.

Адаптивні інтерфейси поєднані з сучасними технологіями. Автори [9] розглядають інтеграцію адаптивних інтерфейсів із хмарними мікросервісами та технологіями глибокого навчання. Система автоматично адаптується до змін потреб користувачів, використовуючи алгоритми оптимізації та аналіз поведінки.

У роботі [10] описано гейміфіковані інтерфейси для підвищення енергетичної грамотності користувачів. Використання інтерактивних елементів, таких як місії, досягнення та зворотний зв'язок, сприяє залученню користувачів і зміні їхньої поведінки.

Розробка шаблонів для різних платформ, наприклад, патерни для інформаційно-розважальних систем у транспортних засобах. У дослідженні [11] представлено дев'ять патернів для UI в автомобільних інформаційно-розважальних системах. Вони охоплюють аспекти контенту, стилю, взаємодії, сповіщень та скорочень. Ці патерни були розроблені на основі емпіричних даних, отриманих під час тестування користувачів, і враховують фактори, що впливають на досвід водія. У роботі [12] аналізується дизайн інтерфейсів для додатків цифрових колекцій у контексті метавсесвіту. Автори використовують інноваційні підходи до оптимізації візуальних ефектів, анімацій та інтерактивності.

Автори [13] пропонують методику відновлення патернів дизайну вебінтерфейсів із вихідного коду за допомогою регулярних виразів. Це сприяє повторному використанню та підтримці існуючих рішень.

Процес впровадження патернів у проєкт включає кілька етапів: ідентифікацію проблеми, аналіз ситуації, вибір принципу рішення, опис самого патерну, пояснення його користі, пошук прикладів успішного застосування та технічну інтеграцію у продукт [3]. Важливо також проводити тестування з реальними користувачами, щоб переконатися у зручності та ефективності обраних рішень [1, 6].

Мета статті

Метою даної роботи є аналіз підходів до розробки архітектури програмного забезпечення з використанням патернів проєктування та підтвердження їхньої ефективності при розробці інтерфейсів програмних продуктів.

Виклад основного матеріалу

Дослідження можливості використання патернів проєктування для розробки шаблонів інтерфейсів проводилось для трьох різних сценаріїв взаємодії користувача з програмним продуктом.

Перший сценарій – кросплатформна програма з «надзвичайною інформацією». Кросплатформний застосунок складається з трьох частин та дозволяє користувачу переглядати новини, мапу тривог та мапу бомбосховищ (рисунок 1).

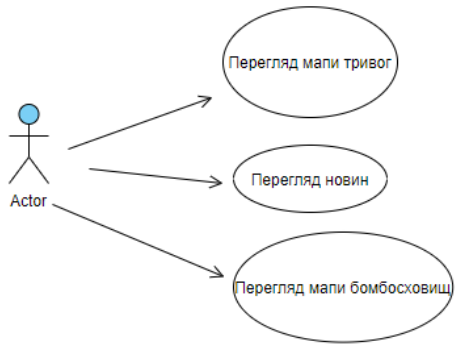


Рис. 1 – Діаграма взаємодії сценарію 1

Під час проєктування системи було застосовано патерн «Фабричний метод» (англ. Factory Method), який визначає загальний інтерфейс створення об’єктів у базовому класі, дозволяючи дочірнім класам мати власну конкретну реалізацію цих об’єктів. Такий підхід дає змогу розробити систему, яка не потребує попередніх знань про те, який саме продукт потрібно створити. Це особливо корисно, коли необхідно

p-ISSN: 2225-6733; e-ISSN: 2519-271X

розширювати систему об’єктами з різною реалізацією, наприклад, для створення програм, що працюють на різних платформах.

Програма має десктопну та вебверсію. Для створення десктопного застосунку була обрана мова програмування Java з бібліотекою Swing. Java була обрана невипадково. По-перше, вона є кросплатформною, що дає можливість перенесення програм без перекомпілювання на інші операційні системи, і вже давно є однією з найвикористовуваніших при розробці програм для інтернет. По-друге, це об'єктно орієнтована мова програмування, і, враховуючи специфіку використання патернів, засновану на взаємодії класів, використання Java дуже доцільне. Це дозволяє створювати модульні програми, вихідний код яких можна використовувати багаторазово. Swing у Java – це легкий інструментарій для створення графічних інтерфейсів, який входить до складу Java Foundation Classes і базується на AWT API. Він повністю написаний на Java, не залежить від платформи та має широкий вибір віджетів для розробки віконних додатків. Swing підтримує зміну дизайну компонентів, використовує архітектуру Model-View-Controller, споживає небагато ресурсів, забезпечує подвійну буферизацію та дозволяє налаштовувати вигляд власних елементів інтерфейсу.

Для створення вебверсії кросплатформного застосунку було використано поєднання HTML, CSS та JavaScript. HTML забезпечує структуру сторінки, є простим для редагування та легко інтегрується з іншими мовами, а також підтримується всіма браузерами. CSS відповідає за оформлення й дозволяє швидко змінювати дизайн на всіх сторінках, забезпечує послідовність вигляду та ефективне завантаження сайту. JavaScript забезпечує динамічну поведінку сторінки у взаємодії з користувачем.

Ієрархія та взаємодія класів наведена на рис. 2.

Клас GUI містить абстрактний фабричний метод createInterface(), який повертає загальний інтерфейс для конкретних продуктів і залишається абстрактним, щоб кожен конкретний фабричний клас міг реалізувати його відповідно до своїх завдань. Конкретні фабричні класи WindowsGUI та HtmlGUI створюють і об’єднують реалізації продуктів, представлені у класах WindowsVersion та HtmlVersion. Ці класи відповідають за різні варіанти інтерфейсу, але метод їх створення спільний, оскільки реалізує один і той же інтерфейс. Клієнтський код визначає операційну систему користувача і за допомогою умовного оператора if створює відповідний екземпляр класу з реалізацією інтерфейсу на Java або HTML-CSS-JS.

Для інтерфейсу був розроблений шаблон з трьома вкладками (рис. 3). Перша вкладка повинна показувати мапу тривог України, друга вкладка відображає стрічку новин, на останній вкладці – карта бомбосховищ.

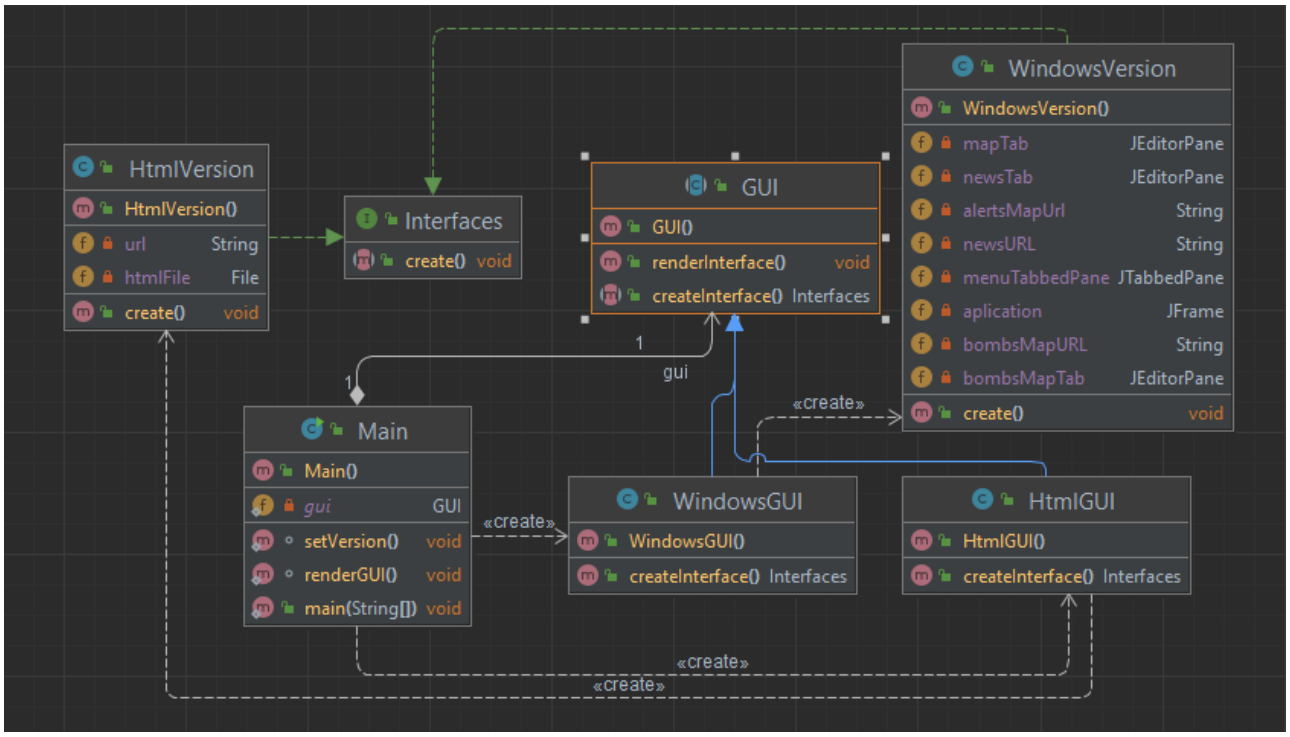


Рис. 2 – Діаграма класів сценарію 1

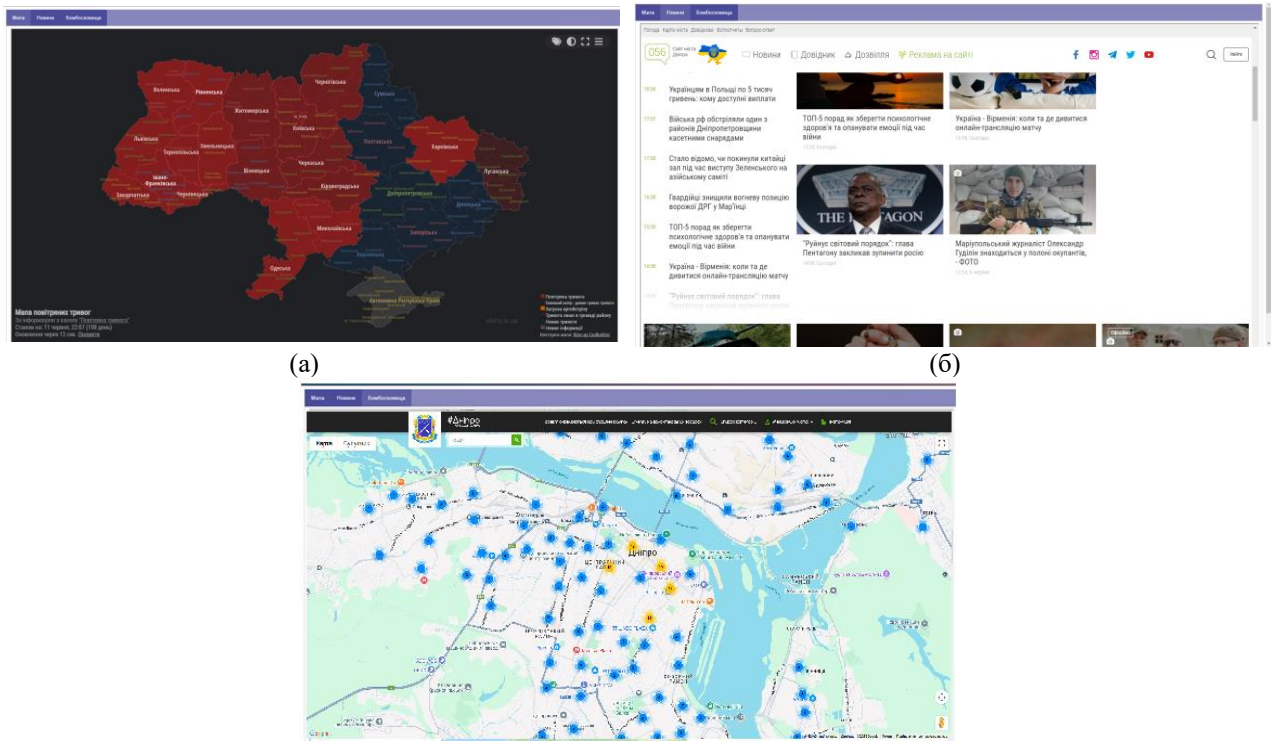
p-ISSN: 2225-6733; e-ISSN: 2519-271X

Рис. 3 – Шаблон вкладок застосунку

На основі патерну «Фабричний метод» та розроблених шаблонів був створений застосунок, який дозволяє переглядати інформацію як у Windows 11, так і через вебверсію. Для створення десктопного інтерфейсу використовувались елементи Java Swing: для створення вкладок JTabbedPane, для відображення контенту з сайтів JEditorPane, JFrame як основа всього застосунку. Реалізація вебінтерфейсу базується на контейнерах , які переключаються за допомогою <button> з використання JS та CSS. На рисунках 4 та 5 наведені реалізовані інтерфейси.

Було проведено ручне тестування розробленого застосунку, яке підтвердило працездатність обох розроблених інтерфейсів. Розроблений застосунок можна легко розширювати, наприклад, для мобільних гаджетів, як вебверсію, так і десктопну.

Другий сценарій взаємодії користувача з програмним продуктом – це десктопний рекламний застосунок для різних операційних систем, який демонструє рекламу та дозволяє обрати користувачу продукт з переліку та перейти на вебсторінку цього продукту. На рисунку 6 наведено діаграму взаємодії застосунку з користувачем.

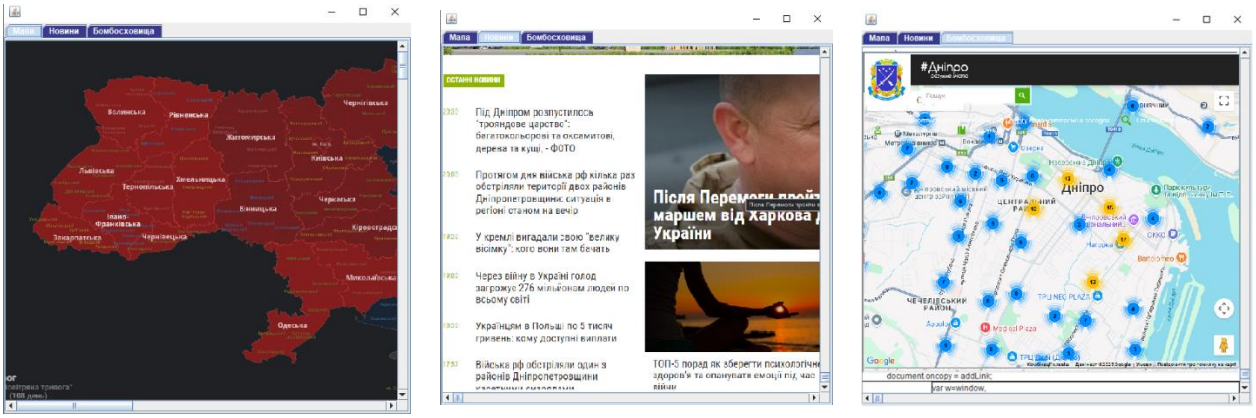


(а) (б)

(в)

Рис. 4 – Реалізація HTML інтерфейсу: вкладка з мапою тривог (а); вкладка з новинами (б); вкладка з бомбосховищами міста Дніпра (в)

p-ISSN: 2225-6733; e-ISSN: 2519-271X



(а) (б) (в)

Рис. 5 – Реалізація десктопного інтерфейсу: мапа тривог (а); вікно з новинами (б); мапа бомбосховищ міста Дніпра (в)

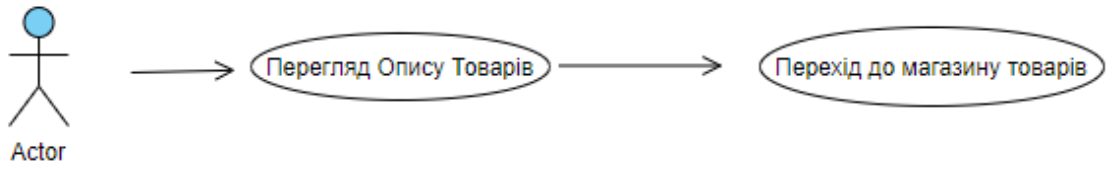


Рис. 6 – Діаграма взаємодії сценарію 2

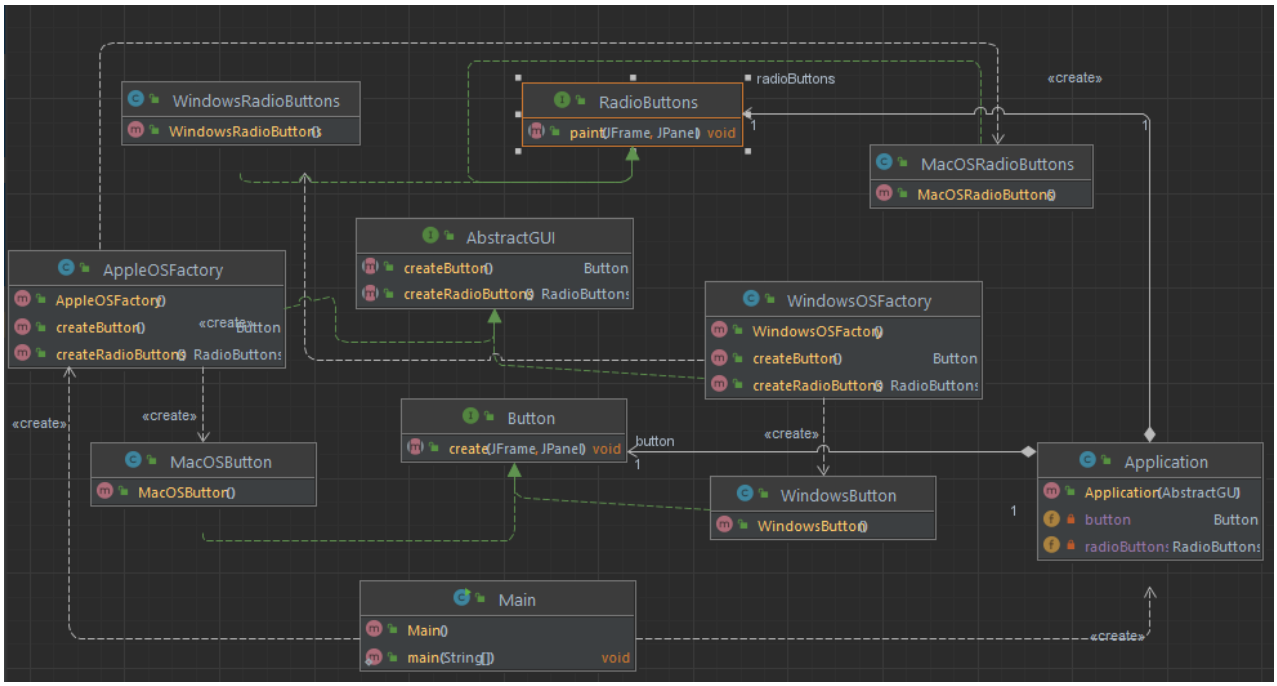


Рис. 7 – Ієрархія та взаємодія класів сценарію 2

Для розробки цього сценарію було обрано патерн «Абстрактна фабрика» (англ. Abstract factory), який дозволяє створювати пов’язані об’єкти, не залежачи від їх конкретної реалізації. Загальні характеристики продуктів виділяються в окремий інтерфейс, який описує спільні властивості, а відмінності переносяться у класи реалізацій. Сам інтерфейс виступає у ролі фабрики. Такий підхід дає змогу створювати різноманітні за реалізацією продукти з подібними ознаками. Використання цього патерну полегшує розробку кросплатформних додатків, які формуються як конструктор із елементів інтерфейсу. Ієрархія класів даного сценарію наведена на рисунку 7.

p-ISSN: 2225-6733; e-ISSN: 2519-271X

Інтерфейс AbstractGUI виконує роль абстрактної фабрики та містить два абстрактні методи – createButton() і createRadioButton(), які дозволяють реалізувати різні фабрики під конкретні операційні системи або вимоги користувача. Фабрики AppleOSFactory та WindowsOSFactory відповідають за створення різних типів продуктів одного сімейства, використовуючи різні елементи як конструктор для формування цілісного інтерфейсу. Методи createButton() та createRadioButton() повертають продукти відповідно до конкретної реалізації, при цьому всі продукти належать до одного типу – Button або RadioButton. Абстрактні класи продуктів Button і RadioButton мають метод paint(), який конкретні класи WindowsButton, MacOSButton, WindowsRadioButton та MacOSRadioButton реалізують з урахуванням різного дизайну та функціональності. Вибір конкретної фабрики здійснює клієнтський код, який залежно від потреб викликає відповідну фабрику.

Для реалізації інтерфейсу був використаний наступний шаблон (рис. 8): рекламне оголошення має містити різні варіанти продуктів, рекламний текст і кнопку, що перенаправляє користувача на сайт із товаром.

Рис. 8 – Шаблон інтерфейсу

Для реалізації цього інтерфейсу використовуються класи: JRadioButton – для вибору продуктів та їх опису, JTextArea – для відображення рекламного банера з інформацією, та JButton – для переходу на сайт продавця, де можна придбати товар.

Реалізація конкретних фабрик була проведена для операційних систем Windows і MacOS, ручне тестування підтвердило працездатність.

Третій сценарій взаємодії користувача з програмним продуктом – це редактор файлів на прикладі умовного текстового редактору. Сценарій взаємодії наведений на рис. 9.

Для розробки редактора файлів було застосовано патерн «Команда» (англ. Command), який дозволяє передавати запити як аргументи при виклику методів, що дає змогу ставити команди в чергу. Основна перевага такого підходу полягає в можливості виконувати відкат дій. Створену команду можна викликати з будь-якої частини коду. Текстовий редактор наочно демонструє це, коли одна й та сама операція може виконуватися у різний спосіб: наприклад, вставка тексту може здійснюватися як за допомогою кнопки «Вставка», так і через певну комбінацію клавіш – обидва способи виконують однаковий запит.

На рис. 10 наведена діаграма класів цього сценарію взаємодії. Клас Command визначає спільний інтерфейс для всіх конкретних команд і містить шаблон для реалізації відміни дій. Основним методом є runCommand(), який запускає відповідну операцію. Класи команд, такі як Copy, Cut, Paste, Save і Open, реалізують свою логіку саме через цей метод. Для команд, що змінюють вміст редактора (наприклад, Cut і Paste), необхідно зберігати стан редактора до виконання дії, щоб у разі відміни можна було відновити попередній стан. Історія виконаних команд зберігається у структурі Stack, що дозволяє додавати команди методом push і скасовувати останню дію через undo, витягуючи команду з кінця – це ідеально підходить для відміни змін у текстовому редакторі. Виклик команди здійснюється через обробник подій графічного інтерфейсу, де створюється об’єкт відповідної команди і викликається метод runCommand(). Такий підхід дозволяє легко реалізувати виконання команд для різних елементів чи комбінацій клавіш, не перевантажуючи основний код і спрощуючи його підтримку.

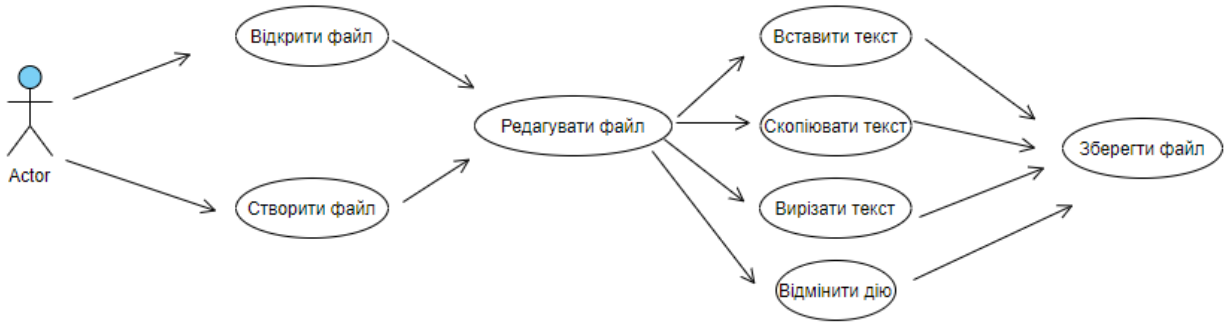


Рис. 9 – Діаграма взаємодії сценарію 3

p-ISSN: 2225-6733; e-ISSN: 2519-271X

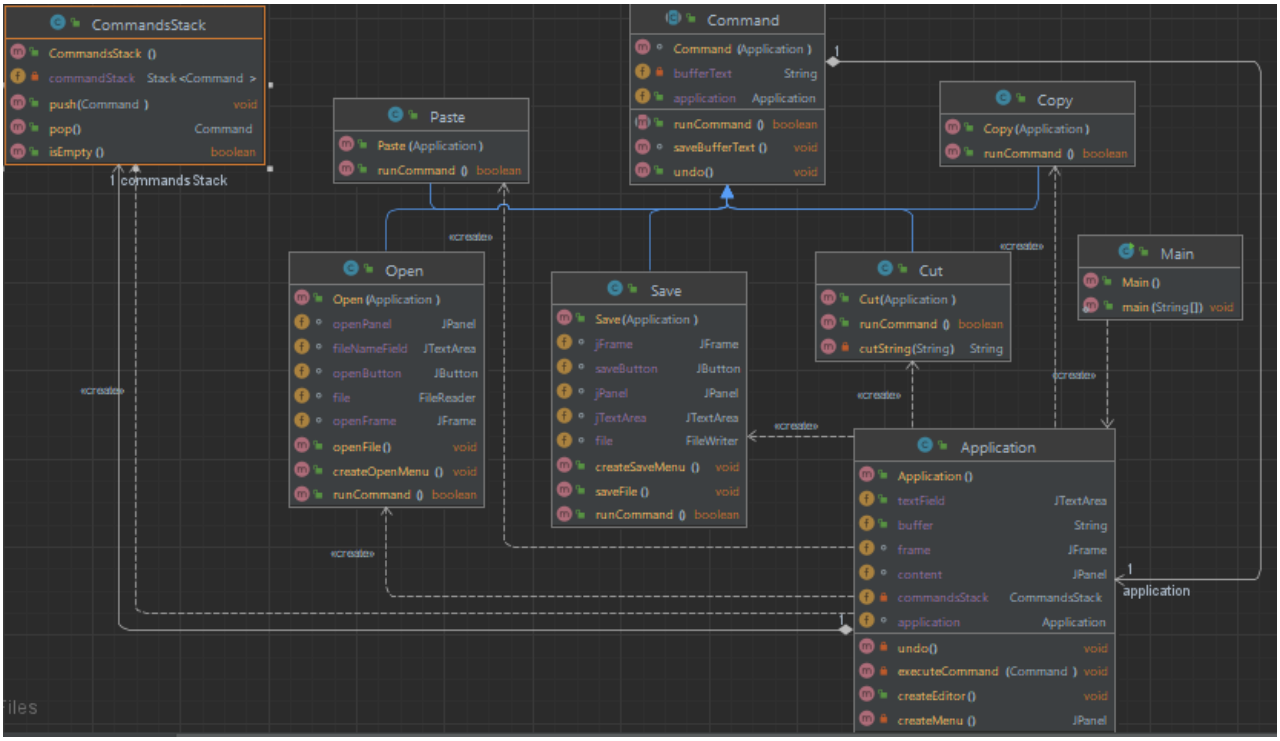


Рис. 10 – Ієрархія та взаємодія класів сценарію 3

Інтерфейс користувача (рис. 11) повинен містити меню з командами для редагування файлу, а також модальне вікно для відкриття та збереження файлів. На першій вкладці слід розмістити меню, яке відкриватиме вікна для вибору файлу або збереження, а також поле для редагування тексту. Друга вкладка має надавати користувачу можливість вставляти, копіювати, вирізати текст і скасовувати виконані дії. Меню відкриття файлу повинно містити поле для введення імені файлу та кнопку для відкриття, а меню збереження – аналогічне поле і кнопку для збереження.

Для реалізації такого інтерфейсу використовуються компоненти JTextArea для редагування тексту та введення імені файлу, JButton для виклику команд відкриття і збереження, а також JMenuBar для створення меню редагування файлів. Реалізований на Java інтерфейс є кросплатформним (рис. 12) і може використовуватися для подальшої розробки редакторів файлів, в тому числі і графічних.

Для даного сценарію також було проведено ручне тестування, яке підтвердило працездатність розробленого інтерфейсу.

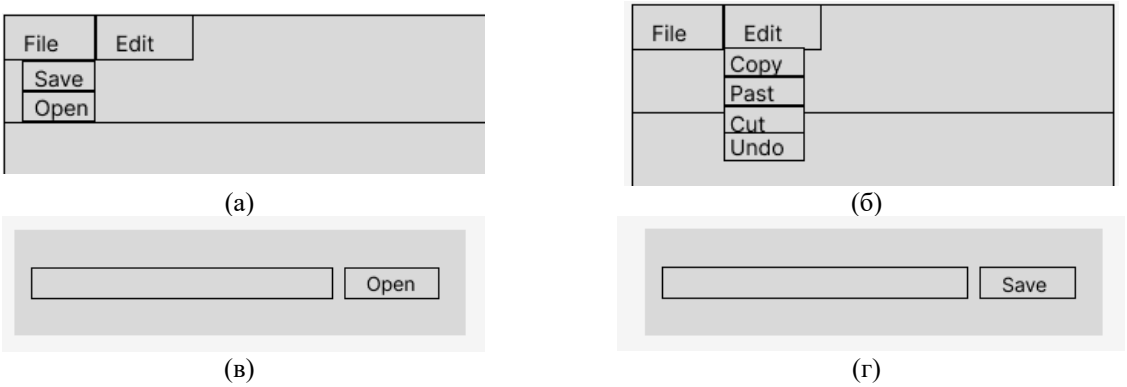


Рис. 11 – Шаблон інтерфейсу сценарію 3: пункт меню файл (а); меню редагування (б); меню відкриття файлу (в); меню збереження файлу (г)

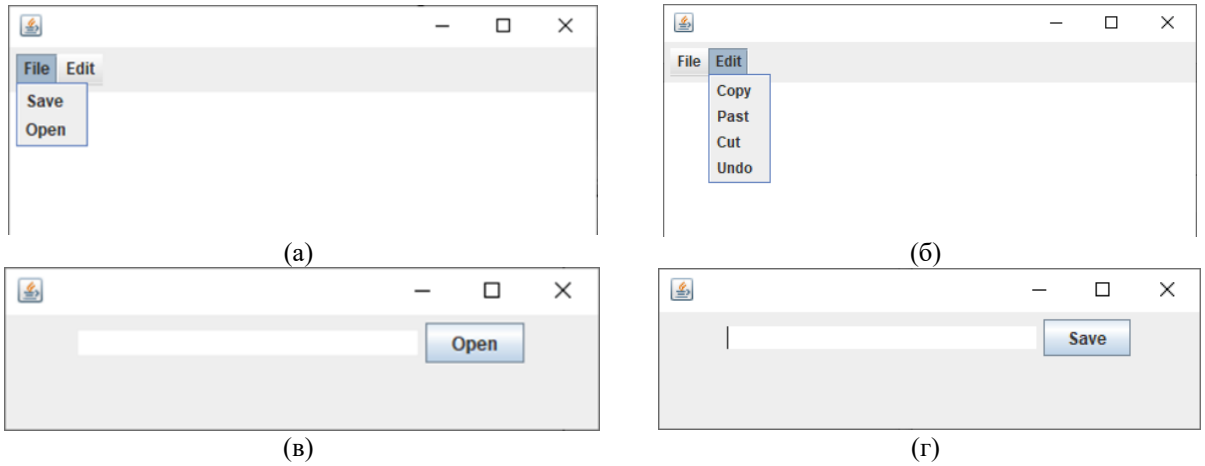


Рис. 12 – Реалізація інтерфейсу сценарію 3: пункт меню файл (а); меню редагування (б); меню відкриття файлу (в); меню збереження файлу (г)

Висновки

Патерни проєктування є чудовим рішенням для типових проблем, які виникають під час створення користувацьких інтерфейсів. Їх використання сприяє підвищенню ефективності розробки, забезпечує повторне використання коду та покращує взаємодію користувачів із системами. Найважливішою якістю будь-якого додатку є його здатність до масштабування та розширення функціоналу. Це дотримання принципу відкритості/закритості, що дозволяє додавати нові можливості без змін у початкових компонентах. Система поділяється на велику кількість простих модулів, які не ускладнюють загальну структуру, відповідаючи принципу KISS. Крім того, розробка інтерфейсів з використанням патернів дотримується одного з принципів SOLID – принципу Барбари Лісков, що гарантує коректну і ефективну заміну базових класів їхніми нащадками в основній логіці програми.

Перелік використаних джерел

[1] Best Practices for UI Design Patterns. URL: https://codener.com/best-practices-for-implementing-ui-design-patterns (дата звернення: 12.01.2025).
[2] UI Patterns. URL: https://www.designrush.com/agency/ui-ux-design/trends/ui-patterns (дата звернення: 12.01.2025).
[3] Patterns in interface design. URL: https://avada-media.ua/en/blog/patterny-v-proyektirovanii-interfeysov/#the-most-common-patterns (дата звернення: 14.01.2025).
[4] Design patterns. URL: https://ui-patterns.com/patterns (дата звернення: 12.01.2025).
[5] Dannaway A. Practical UI. 2nd ed. 2025. 373 p.
[6] Pelegrin J. Key UI and UX Patterns. URL: https://duck.design/key-ui-and-ux-patterns (дата звернення: 17.02.2025).
[7] Video Game Interface Design Patterns to Facilitate Human-Agent Teaming / N.C. Sepich et al. *International Journal of Human–Computer Interaction*. 2023. Vol. 40. Pp. 7270-7285. DOI: https://doi.org/10.1080/10447318.2023.2262824.
[8] Greenberg S., Godin T., Whittington J. User Interface Design Patterns for Wildlife-Related Camera Trap Image Analysis. Report 2019-1114-06, Department of Computer Science, University of Calgary, Calgary, AB., Canada. 26 p. DOI: https://doi.org/10.11575/PRISM%2F36783.
[9] Cherukuri B.R. Development of Design Patterns with Adaptive User Interface for Cloud Native Microservice Architecture Using Deep Learning With IoT. 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT), Greater Noida, India, 9-10 Feb. 2024. Vol. 5. Pp. 1866-1871. DOI: https://doi.org/10.1109/IC2PCT60090.2024.10486720.
[10] Energy gamification: design and development of a user interface tool to upgrade social experience and energy literacy / J. Cravinho et al. *Open Research Europe*. 2023. Vol. 2. 2023. Pp. 1-32. DOI: https://doi.org/10.12688/openreseurope.15158.2.
[11] User Interface Design Patterns for Infotainment Systems Based on Driver Distraction: A Colombian Case Study / J.P. Alarcon et al. *Sustainability*. 2022. Vol. 14(13). Article 8186. DOI: https://doi.org/10.3390/su14138186.
[12] Du Y., Zhang X., Liu Z. User Interface Design for Digital Collections APP in the Metaverse Perspective. *2024 17th International Symposium on Computational Intelligence and Design (ISCID)*, 14-15 Dec. 2024. Pp. 252-255. DOI: https://doi.org/10.1109/ISCID63852.2024.00064.

[13] Mahmood K., Tariq F., Rasool G. Recovery of User Interface Web Design Patterns using Regular Expressions. *International Journal of Advanced Computer Science and Applications*. 2018. Vol. 9, iss. 4. Pp. 405-420. DOI: https://doi.org/10.14569/IJACSA.2018.090457.

DEVELOPMENT OF USER INTERFACE TEMPLATES BASED ON DESIGN PATTERNS

Sokolova N. *PhD (Engineering), associate professor, Dnipro University of Technology, Dnipro, ORCID: https://orcid.org/0000-0003-2493-3553, e-mail: n.olegowna@gmail.com;*

Khara G. *M.Sc., Dnipro University of Technology, Dnipro, ORCID: https://orcid.org/0009-0000-1165-901X, e-mail: kharaherman@gmail.com;*

Balalaieva O. *PhD (Engineering), associate professor, SHEI «Priazovskyi state technical university», Dnipro, ORCID: https://orcid.org/0000-0003-1461-4399, e-mail: balalaeva_e_u@pstu.edu;*

Olevskyi V. *D.Sc. (Engineering), professor, Dnipro University of Technology, Dnipro, ORCID: https://orcid.org/0000-0003-3824-1013, e-mail: olevskyi.v.i@nmu.one*

This article is devoted to the study of the issue of developing interface templates using design patterns. The relevance of the study is due to the constant growth of the number of software products in various fields, which requires the creation of efficient, flexible and scalable architectures. The development of user interface templates based on Design Patterns is a multifaceted topic that covers adaptability, specialized solutions, integration of modern technologies and reuse of existing solutions. The sources reviewed demonstrate how patterns can be adapted to different contexts, from gamification to image analysis and metaverse. This emphasizes their versatility and importance in modern UI design. A description of three interface templates for different user interaction with software products is given, in particular for a cross-platform application with emergency information, an advertising application and a file editor. For each of them, the choice of pattern, the hierarchy and interaction of classes and the features of the interface implementation are described. The study showed that the use of Creational Design Patterns Factory Method and Abstract Factory, as well as the Behavioral Pattern Command, allows to significantly increase the flexibility, scalability and maintainability of software products, and also helps to reduce development time and improve the quality of the final product. This work demonstrates the practical value of design patterns for the development of modern, effective and convenient user interfaces, and also confirms the feasibility of their use in real software projects. The practical significance of the work lies in the possibility of using the obtained results by both active developers and students to create effective, functional and convenient interfaces, as well as for in-depth study of disciplines related to Design Patterns and software architecture.

Keywords: *user interface, design patterns, class hierarchy, Java, software product, interface templates, cross-platform.*

References

[1] Best Practices for UI Design Patterns. [Online]. Available: https://codener.com/best-practices-for-implementing-ui-design-patterns. Accessed on: January 12, 2025.

[2] UI Patterns. [Online]. Available: https://www.designrush.com/agency/ui-ux-design/trends/ui-patterns. Accessed on: January 12, 2025.

[3] Patterns in interface design. [Online]. Available: https://avada-media.ua/en/blog/patterny-v-proyektirovanii-interfeysov/#the-most-common-patterns. Accessed on: January 14, 2025.

[4] Design patterns. [Online]. Available: https://ui-patterns.com/patterns. Accessed on: January 12, 2025.

[5] A. Dannaway, *Practical UI*, 2nd ed. 2025.

[6] J. Pelegrin, Key UI and UX Patterns. [Online]. Available: https://duck.design/key-ui-and-ux-patterns. Accessed on: February 17, 2025.

[7] N. Sepich, M. Talyat, E. Didic, M.C. Dorneich, and S.B. Gilbert, "Video Game Interface Design Patterns to Facilitate Human-Agent Teaming," *International Journal of Human–Computer Interaction*, vol. 40, pp. 7270-7285, 2023. **doi: 10.1080/10447318.2023.2262824**.

[8] S. Greenberg, T. Godin, and J. Whittington, "User Interface Design Patterns for Wildlife-Related Camera Trap Image Analysis," Department of Computer Science, University of Calgary, Calgary, AB., Canada, Report 2019-1114-06, 2019. **doi: 10.11575/PRISM%2F36783**.

[9] B.R. Cherukuri, "Development of Design Patterns with Adaptive User Interface for Cloud Native Microservice Architecture Using Deep Learning With IoT," in Proc. of the 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT), Greater Noida, India, 2024, vol. 5, pp. 1866-1871. **doi: 10.1109/IC2PCT60090.2024.10486720**.

[10] J. Cravinho, R. Lucas, M. Brito, D.P. Albuquerque, U. Mithoowani, and N.M. Mateus, "Energy gamification: design and development of a user interface tool to upgrade social experience and energy literacy,"

Open Research Europe, vol. 2, pp. 1-32, 2023. **doi: 10.12688/openreseurope.15158.2**.

[11] J. Alarcón, I. Balcázar, C.A. Collazos, H. Luna, and F. Moreira, "User Interface Design Patterns for Infotainment Systems Based on Driver Distraction: A Colombian Case Study," *Sustainability*, vol. 14(13), article 8186, 2022. **doi: 10.3390/su14138186**.

[12] Y. Du, X. Zhang, and Z. Liu, "User Interface Design for Digital Collections APP in the Metaverse Perspective," in *Proc. of the 2024 17th International Symposium on Computational Intelligence and Design (ISCID)*, Dec. 14-15, 2024, pp. 252-255. **doi: 10.1109/ISCID63852.2024.00064**.

[13] K. Mahmood, F. Tariq, and G. Rasool, "Recovery of User Interface Web Design Patterns using Regular Expressions," *International Journal of Advanced Computer Science and Applications*, vol. 9, iss. 4, pp. 405-420, 2018. **doi: 10.14569/IJACSA.2018.090457**.

Стаття надійшла 04.07.2025
Стаття прийнята 28.07.2025
Стаття опублікована 30.10.2025

Цитуйте цю статтю як: Розробка шаблонів користувацького інтерфейсу на основі патернів проєктування / Соколова Н. О., Хара Г. Л., Балалаєва О. Ю., Олевський В. І. *Вісник Приазовського державного технічного університету. Серія: Технічні науки*. 2025. Вип. 51. С. 9-18. DOI: https://doi.org/10.31498/2225-6733.51.2025.344593.