Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп’ютерних наук та інформаційних технологій
Кафедра автоматизованих систем управління

Курсова робота

з дисципліни

“Прикладне програмування”

на тему:

КРЕДИТИ

Виконав:

студент ОІ-25

Басала Валентин

Керівник:

Скрибайло-Леськів Д.Ю.

Львів – 2024

Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра автоматизованих систем управління

Завдання до курсової роботи

з дисципліни **“Прикладне програмування”**

Прізвище, ім'я студента	Басала Валентин
Група	ОІ-25
Тема курсової роботи	Кредити

Спеціальна частина завдання:

1. Використати мову програмування Java та набір інструментів JavaFX для розробки програмного забезпечення для управління кредитами, враховуючи можливість взяти нові кредити, достроково закрити кредит, збільшити кредитну лінію, частково виплатити кредит, знаходити кредити за певними параметрами. Також розглянути можливість об’єднання програмного продукту з базою даних для зберігання інформації про кредити.
2. Розробити інтерфейс, який містить сторінки для пошуку кредитів за параметрами та взяття кредитів, додавання кредитних пропозицій, перегляду власних кредитів та виконання певних операцій над ними.
3. Використати базу даних SQLite для зберігання даних про кредити, забезпечуючи надійне та ефективне управління інформацією.
4. Середовище функціонування програми – IntelliJ IDEA.
5. Термін завершення роботи – 26 травня 2024р.

Завдання видано: 5 березня 2024 р.

Керівник (_____) Скрибайло-Леськів Д.Ю.

Студент (_____) Басала В.Р.

Зміст

Вступ

В сучасному світі доступ до кредитних послух відіграє важливу роль в плануванні життя, бізнесу для багатьох осіб та підприємств. Одночасно банки адаптують свої кредитні послуги під споживачів, які прагнуть більшої участі в управлінні власними фінансами.

З метою оптимізації процесу кредитування та забезпечення клієнтам можливості отримати найвигідніші умови кредитування, у даній роботі ставиться завдання розробки програмного продукту на мові програмування Java з використанням програмного засобу JavaFX для графічного інтерфейсу, який дозволяє створювати інтерактивні та ефективні застосунки з зрозумілим інтерфейсом, які можуть надати користувачам широкий вибір функціональності.

Актуальність даної теми полягає в тому, що клієнти мають все більше різних можливостей для вибору кредитних пропозицій, оскільки постійно зростає кількість банків, фінансових установ, які пропонують ці послуги, проте не завжди в банківських програмних додатках є можливість ефективно порівняти умови кредитування та вибрати для себе оптимальний варіант.

В майбутньому ці результати можуть бути застосовані з метою підвищення зручності та доступності фінансових послуг для клієнтів. Клієнти зможуть легко порівнювати та вибирати кредитні пропозиції та виконувати необхідні операції, сприяючи підвищенню фінансової грамотності та ефективному управлінню фінансами.

1. Постановка задачі

Основною задачею є розробка настільного (desktop) додатку з метою надання клієнту можливості порівняти та вибрати найбільш вигідні цільові кредити від різних банків.

Функціональні можливості програми повинні включати в себе перегляд кредитних пропозицій, фільтрацію пропозицій за різними параметрами, такими як максимальна процентна ставка, мінімальний термін кредиту, максимальна сума, можливість дострокового погашення кредиту, можливість збільшення кредитної лінії, фільтрація за зростанням та спаданням суми, терміну, процентної ставки кредиту. Також має бути присутня можливість взяття кредиту, перегляд власних кредитів, їх сортування, можливість збільшити кредитну лінію чи достроково закрити кредит, можливість частково виплатити кредит. Важливим є також можливість додавання кредитних пропозицій та видалення їх.

Крім перелічених функціональних можливостей, важливим є забезпечення інтуїтивно зрозумілого та зручного інтерфейсу користувача, що дозволить легко орієнтуватися у програмі та здійснювати всі необхідні операції без зайвих зусиль. Також важливо створити логічну та зручну структуру програми, розмістивши всі функції на кількох сторінках.

2. Огляд літератури

2.1 Роль кредитних послуг та сучасні вимоги до програмного забезпечення

Банківський кредит відіграє важливу роль у підтримці як споживчих витрат, так і ділових операцій. Споживачам доступ до кредитів дозволяє робити покупки та інвестиції, які вони не можуть собі дозволити наперед. У сфері бізнесу банківський кредит надає компаніям необхідні кошти для роботи, розширення та інвестування в свою діяльність [1].

Розробка програмного забезпечення для кредитування стає все більш актуальною, оскільки зростає кількість фінансових установ, які надають кредитні пропозиції. Розроблені програми повинні забезпечувати зручний та зрозумілий інтерфейс, який дозволить користувачам досить легко орієнтуватися в кредитних пропозиціях та виконувати потрібні операції.

Надання більшої кількості інструментів для управління особистими фінансами сприяє підвищенню фінансової грамотності населення та ефективності управління фінансовими ресурсами. Зокрема, використання програмного забезпечення для порівняння кредитних пропозицій дозволяє користувачам приймати обґрунтовані рішення, що сприяє їх фінансовій стабільності [2].

Існуючі програми для вибору кредитів мають різні функціональні можливості, включаючи перегляд кредитних пропозицій, фільтрацію за різними параметрами параметрами, можливість дострокового погашення та інші.

2.2 Порівняльний аналіз програм для кредитування

Існують численні програмні рішення, які допомагають користувачам вибрати кредитні пропозиції. Наприклад, додатки на зразок Credit Karma та NerdWallet дозволяють порівнювати різні кредитні пропозиції за такими параметрами, як процентні ставки, терміни кредитування та умови погашення. Вони також надають користувачам персоналізовані рекомендації на основі їх кредитного рейтингу та фінансової історії.

Платформа "Finance.ua" для порівняння кредитів пропонує функціонал для фільтрації кредитних пропозицій за різними критеріями, включаючи суму кредиту, строк кредиту, працевлаштування клієнта. Така платформа також забезпечує зручний інтерфейс для користувачів, що дозволяє легко орієнтуватися в різноманітті кредитних продуктів та робити правильні фінансові рішення.

На платформі "Banki.ua" фільтрація кредитів відбувається за сумою кредиту, терміном кредиту, також присутня можливість вибору кредитів під 0 %, кредитів без відмов.

Отже, популярні програмні рішення для кредитування надають можливість користувачам порівнювати різні пропозиції за параметрами, такими як процентні ставки, терміни кредитування, сума кредиту, можливість дострокового погашення кредиту. Вони мають зрозумілий інтерфейс користувача, що забезпечує зручність для вибору кредитної пропозиції.

3. Опис етапу проєктування

3.1 Формування підзадач проєктування

На етапі розробки необхідно спроєктувати та реалізувати функціонал на мові програмування Java для виконання операцій з кредитами. Це включає в себе пошук кредитів за певними параметрами, створення можливостей для додавання нових кредитних пропозицій, взяття нових кредитів, дострокового закриття кредитів, збільшення кредитних ліній та часткової виплати кредитів.

Після цього слід налаштувати зв'язок з базою даних SQLite для ефективного зберігання та управління даними про кредити. Це включає створення та налаштування таблиць для зберігання інформації про кредитні пропозиції та кредити користувача, а також реалізацію механізмів взаємодії з базою даних для зчитування, запису та оновлення даних.

Також необхідно розробити дизайн трьох основних сторінок інтерфейсу користувача з використанням JavaFX. Ці сторінки включатимуть в себе можливість пошуку кредитів за параметрами, взяття нового кредиту, можливість перегляду власних кредитів та виконання операцій над ними.

Для забезпечення надійності та стабільності програмного забезпечення, необхідно покрити програму unit тестами. Це дозволить виявити та усунути можливі помилки та недоліки. Крім того, необхідно додайте логгер для запису основних дій та виняткових ситуацій. Критичні помилки можна додатково надсилатися на електронну пошту. Також необхідно написати автоматизовані тести для графічного інтерфейсу, що може бути корисним для забезпечення якості програмного забезпечення.

Кожна з цих задач допоможе створити зручний та ефективний інструмент для управління кредитами з інтегрованою системою зберігання даних.

3.2 Функціонал програми та схема бази даних

Use Case діаграма

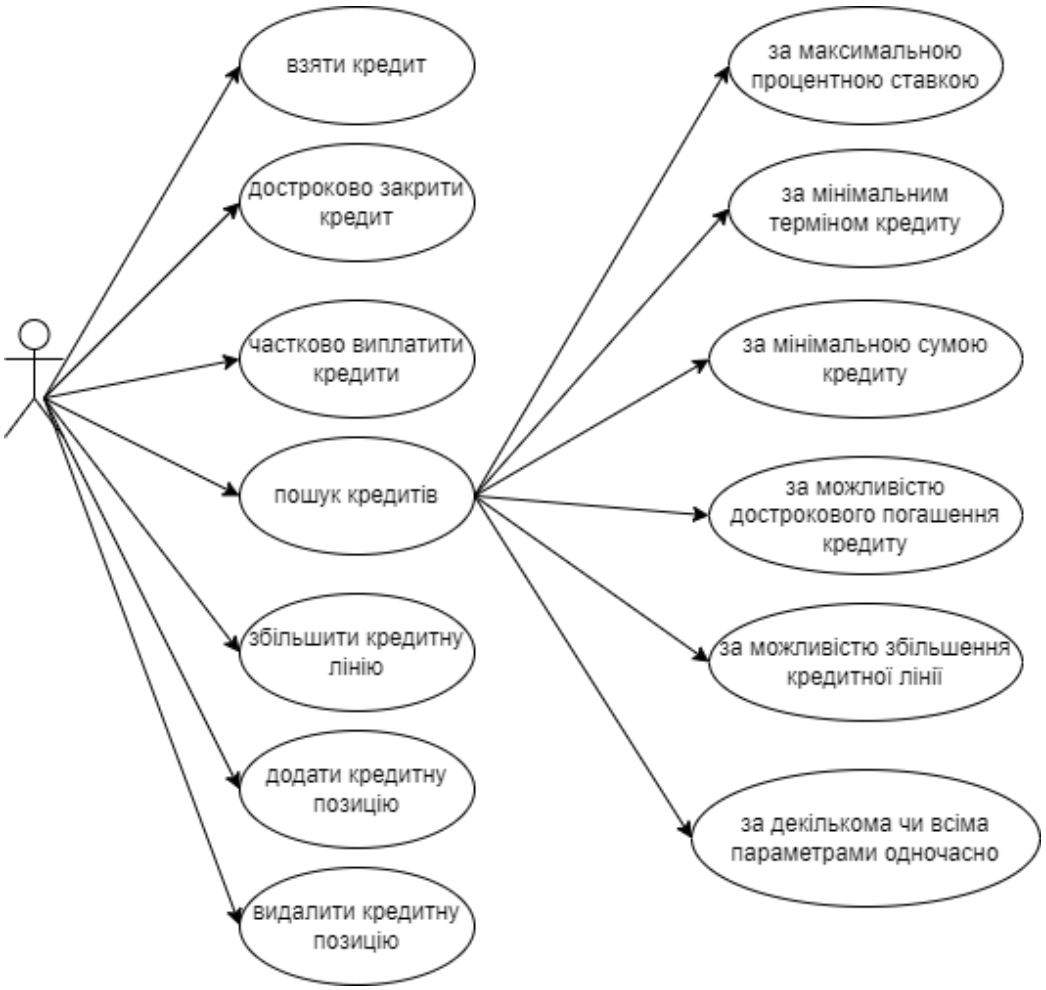

Рис. 1 – Use Case діаграма

Користувач програмного застосунку матиме можливість:

Взяти кредит

- Взяти кредит із списку кредитних пропозицій.

Достроково закрити кредит

- Достроково закрити кредит, якщо це доступно у вибраній кредитній пропозиції.

Збільшити кредитну пропозицію

- Збільшити кредитну лінію, якщо це доступно у вибраній кредитній пропозиції.

Частково виплатити кредит

- Заплатити частину кредиту, вказавши через скільки місяців вноситься сума для нарахування відсотків, та відповідну суму.

Пошук кредитів

- Знайти кредит за одним, декількома чи всіма наступними параметрами: максимальна процентна ставка, мінімальний термін кредиту, мінімальна сума кредиту, можливість дострокового погашення кредиту, можливість збільшення кредитної лінії.

Додати кредитну пропозицію

- Додати власну кредитну пропозицію до загального списку пропозицій.

Видалити кредитну пропозицію

- Видали кредитну пропозицію із загального списку пропозицій.

Схема бази даних credits (SQLite)

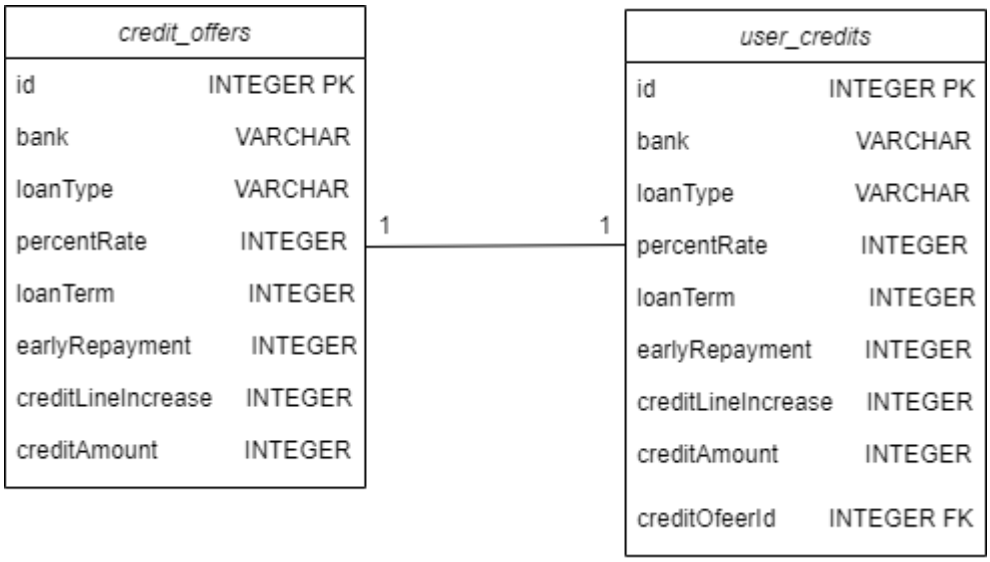

Рис. 2 – Схема бази даних credits (SQLite)

База даних credits містить дві таблиці: «credit_offers» та «user_credits».

Таблиця «credit_offers» містить інформацію про кредитні пропозиції, що надаються різними банками. Кожен запис у цій таблиці має унікальний ідентифікатор (id) і поля, що відображають назву банка, тип кредиту, процентну ставку, термін кредиту, можливість дострокового погашення, можливість збільшення кредитної лінії та суму кредиту.

Таблиця «user_credits» призначена для зберігання інформації про кредити, взяті користувачами. Кожен запис у цій таблиці також має унікальний ідентифікатор (id) і поля, що відображають назву банка, тип кредиту, процентну ставку, термін кредиту, який користувач може змінити, можливість дострокового погашення, можливість збільшення кредитної лінії та суму кредиту, яку користувач може частково виплачувати. Також таблиця містить поле, що посилається на ідентифікатор кредитної пропозиції у таблиці «credit_offers» (creditOfeerId). Цей зовнішній ключ (FOREIGN KEY) забезпечує зв'язок між таблицями, дозволяючи зв'язати кредити, взяті користувачами, з відповідними кредитними пропозиціями в таблиці «credit_offers».

Ця база даних дозволяє вести облік різноманітних кредитних пропозицій від різних банків, а також кредитів, взятих користувачами, і забезпечує зв'язок між ними для зручного управління кредитами.

Була вибрана саме база даних SQLite, оскільки вона відносно проста у використанні та не вимагає складних налаштувань, не потребує окремого сервера. Також вся база даних SQLite зберігається в одному файлі на диску [3].

4. Програмне рішення

4.1 Опис класів та їхніх методів

Діаграма класів

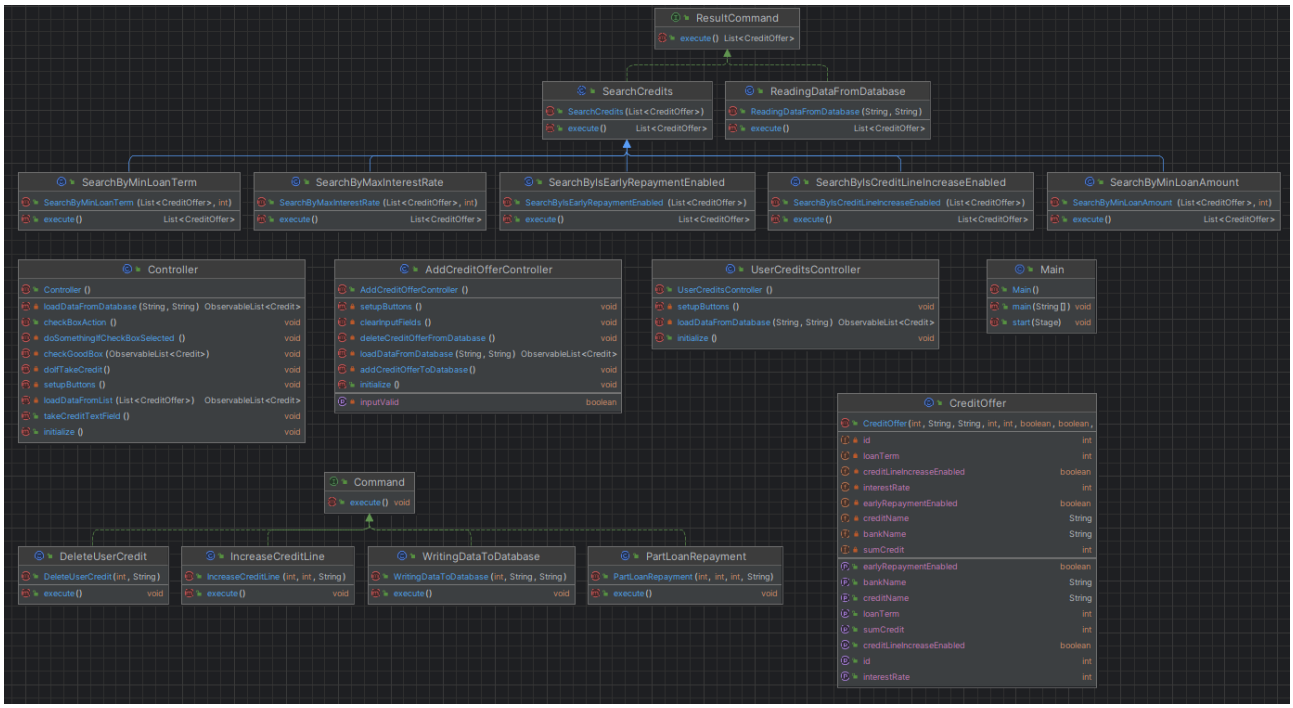

Рис. 3 – Діаграма класів

Клас Main

Клас Main відповідає за запуск та ініціалізацію головного вікна додатку та містить наступні методи:

Метод *static void main(String[] args)* – цей метод є точко входу, яка запускає додаток.

Метод *void start(Stage stage) викликається при запуску додатку і ініціалізує головне вікно.*

Клас Controller

Клас Controller відповідає за управління вікном з кредитнимим пропозиціями та містить наступні методи:

Метод *void initialize()* – цей метод ініціалізує вікно з кредитними пропозиціями та відображає кредитні пропозиції.

Метод *void setupButtons() відповідає за налаштування обробника подій для кнопок (кнопок переключення на інші сторінки, пошук кредитів, взяття кредиту) вікна з кредитними пропозиціями.*

Метод *void checkBoxAction() перевіряє, які параметри пошуку вибрано.*

Метод void takeCreditTextField перевіряє, чи введно номер кредиту для його взяття.

Метод void takeCreditTextField перевіряє, чи введно номер кредиту для його взяття.

Метод void doIfTakeCredit() додає вибраний кредит до бази даних.

Метод void doSomethingIfCheckBoxSelected() використовується для пошуку кредитів за параметрами.

Метод void doSomethingIfCheckBoxSelected() використовується для пошуку кредитів за параметрами.

ObservableList<Credit> loadDataFromDatabase(String databaseUrl, String tableName) виводить дані з бази даних на сторінку.

Клас UserCreditsController

Клас UserCreditsController відповідає за управління вікном з додаванням кредитних пропозицій та містить наступні методи:

Метод *void initialize()* – цей метод ініціалізує вікно з додаванням кредитних пропозицій та відображає наявні кредитні пропозиції.

Метод *void setupButtons() відповідає за налаштування обробника подій для кнопок вікна (кнопок переключення на інші сторінки, дострокове закриття*

кредиту, збільшення кредитної лінії, часткова виплата кредиту)з взятими кредитами.

ObservableList<Credit> loadDataFromDatabase(String databaseUrl, String tableName) виводить дані з бази даних на сторінку.

Клас AddCreditOfferController

Клас AddCreditOfferController відповідає за управління вікном для додавання кредитної пропозиції та містить наступні методи:

Метод *void initialize()* – цей метод ініціалізує вікно з кредитами користувача та відображає взяті кредити.

Метод *void setupButtons() відповідає за налаштування обробника подій для кнопок вікна (кнопок переключення на інші сторінки, додавання кредитної пропозиції, видалення кредитної пропозиції)з додаванням кредитної позиції.*

Метод *void addCreditOfferToDatabase() додає кредитну прозицію до бази даних.*

Метод *void deleteCreditOfferFromDatabase() видаляє кредитну прозицію із бази даних.*

Метод *boolean isInputValid() перевіряє чи всі поля заповнені для додавання кредитної пропозиції.*

Метод *void clearInputFields() очищає дані з текстових полів.*

Пакет database

Цей пакет містить 2 класи-команди: ReadingDataFromDatabase та WritingDataToDatabase.

Клас-команда ReadingDataFromDatabase відповідає за зчитування даних з бази даних.

Клас-команда WritingDataToDatabase відповідає за запис даних до бази даних.

Пакет searchCredits

Цей пакет містить 6 класів-команд: SearchCredits, SearchByMinLoanTerm, SearchByMinLoanAmount, SearchByMaxInterestRate, SearchByIsEarlyRepaymentEnabled, SearchByIsCreditLineIncreaseEnabled.

Клас SearchCredits – це абстрактний клас для пошуку кредитів.

Клас-команда SearchByMinLoanTerm відповідає за пошук кредитів за мінімальним терміном кредиту.

Клас-команда SearchByMinLoanAmount відповідає за пошук кредитів за мінімальною сумою кредиту.

Клас-команда SearchByMaxInterestRate відповідає за пошук кредитів за максимальною процентною ставкою.

Клас-команда SearchByIsEarlyRepaymentEnabled відповідає за пошук кредитів за можливістю дострокового закриття кредитів.

Клас-команда SearchByIsCreditLineIncreaseEnabled відповідає за пошук кредитів за можливістю збільшення кредитної лінії.

Пакет creditOffer

Цей пакет містить клас CreditOffer, який містить гетери, сетери змінних для даних про кредити.

Пакет userCredits

Цей пакет містить 3 класи-команди: DeleteUserCredit, IncreaseCreditLine, PartLoanRepayment.

Клас-команда DeleteUserCredit відповідає за видалення кредиту користувача.

Клас-команда IncreaseCreditLine відповідає за збільшення крединої лінії, якщо кредитна пропозиція це дозволяє.

Клас-команда PartLoanRepayment відповідає за часткову виплату кредиту.

4.2 Опис unit тестів, автоматизації тестування графічного інтерфейсу, логування основних подій

Unit тести

Unit тест — це невелика програма, яка тестує роботу окремого відрізка коду. Завдання тесту — переконатися, що саме ця ділянка коду функціонує нормально, виконує своє завдання в різних умовах, і не заважає роботі інших ділянок коду та всього продукту [4].

У програмі використовується бібліотека для написання тестів у Java під назвою JUnit, а саме версія JUnit 5.

JUnit 5 — це поточне покоління інфраструктури тестування JUnit, яка забезпечує сучасну основу для тестування на стороні розробника на JVM. Це включає зосередження на Java 8 і вище, а також уможливлення багатьох різних стилів тестування [5].

Програма містить наступні тести:

IncreaseCreditLineTest - цей тест перевіряє, чи правильно працює функціонал збільшення ліміту кредитної лінії користувача. Спочатку він створює тестовий запис у базі даних з встановленим початковим значенням loanTerm. Після цього викликається метод для збільшення кредитної лінії. Далі проводиться запит до бази даних, щоб отримати оновлене значення loanTerm для перевірки, чи відбулося збільшення. Якщо нове значення loanTerm більше за початкове, тест вважається успішним.

PartLoanRepaymentTest - цей тест перевіряє, чи правильно працює функціонал часткового погашення кредиту користувача. Спочатку він створює тестовий запис у базі даних з встановленим початковим значенням creditAmount. Після цього викликається метод для часткового погашення кредиту. Далі проводиться запит до бази даних, щоб отримати оновлене значення creditAmount для перевірки, чи відбулося зменшення суми кредиту. Якщо нове значення creditAmount менше за початкове, тест вважається успішним.

DeleteUserCreditTest - цей тест перевіряє, чи відбувається видалення кредитного запису користувача з бази даних працює належним чином. Спочатку відбувається отримання початкової кількості записів у базі даних. Далі відбувається виклик методу для видалення кредитного запису. Потім відбувається отримання кількості записів у базі після видалення. Потім тест порівнює початкову кількість записів з кількістю записів після видалення. Тест успішний, якщо кількість записів після видалення менша за початкову.

ReadingDataFromDatabaseTest - цей тест перевіряє, чи правильно працює функціонал зчитування даних з бази даних про кредитні пропозиції. Спочатку встановлюються URL та назва таблиці бази даних. Після цього викликається метод execute() для отримання списку кредитних пропозицій з бази даних. У випадку успішного виконання методу, перевіряється, чи список кредитних пропозицій не є пустим. Якщо список не є пустим, тест вважається успішним.

Також присутні тестування пошуку кредитних пропозицій за різними параметрами: SearchByMinLoanTermTest, SearchByMinLoanAmountTest, SearchByMaxInterestRateTest, SearchByIsEarlyRepaymentEnabledTest, SearchByIsCreditLineIncreaseEnabledTest. Кожен тест спочатку набір кредитних пропозицій. Потім викликається метод execute() для виконання пошуку за певним параметром. Після цього тест порівнює очікувані та отримані результати пошуку, чи вони містять правильні кредитні пропозиції.

Автоматизації тестування графічного інтерфейсу

У програмі для графічного тестування використовується бібліотека TestFX. Вона має такі переваги, як гнучке налаштування та очищення тестових інструментів JavaFX, прості роботи для імітації взаємодії користувача, багата колекція відповідників для перевірки очікуваних станів [6].

Програма містить наступні тестування графічного інтерфейсу:

ControllerTest – цей клас тестує сторінку з кредитними пропозиціями та мітить декілька тестів:

- testOpenCreditButton() перевіряє, що після кліку на кнопку «Взяти кредит» відображається вікно з відповідним заголовком.
- testOpenMyCreditsButton() перевіряє, що після кліку на кнопку «Мої кредити» відображається вікно з відповідним заголовком.
- testAddCreditOfferButton() перевіряє, що після кліку на кнопку «Додати кредит» відображається вікно з відповідним заголовком.
- testSearchByCreditLineIncrease() клікає на прапорець «Збільшення кредитного ліміту» та кнопку «Пошук», а потім перевіряє, що таблиця містить певну кількість рядків, що відповідає кредитам зі збільшеним кредитним лімітом.

- testSearchByEarlyRepayment() аналогічно шукає кредити з можливістю дострокового погашення та перевіряє кількість рядків у таблиці.
- testSearchByMaxPercentRate() шукає кредити з максимальною відсотковою ставкою та перевіряє кількість рядків.
- testSearchByMinLoanTerm() шукає кредити з мінімальним терміном кредиту та перевіряє кількість рядків.
- testSearchByMinLoanAmount() шукає кредити з мінімальною сумою кредиту та перевіряє кількість рядків у таблиці.

UserCreditsControllerTest - цей клас тестує функціональність сторінки з кредитами користувача і містить кілька тестів:

У методі start класу UserCreditsControllerTest створюється нове вікно із сценою з кредитами користувача для подальшого тестування.

- testOpenCreditButton() перевіряє, що після кліку на кнопку «Взяти кредит» відображається вікно з відповідним заголовком.
- testOpenMyCreditsButton() перевіряє, що після кліку на кнопку «Мої кредити» відображається вікно з відповідним заголовком.
- testAddCreditOfferButton() перевіряє, що після кліку на кнопку «Додати кредит» відображається вікно з відповідним заголовком.
- testCreditIncreaseButton() виконує збільшення кредитної лінії для конкретного користувача при натисненні кнопки «Збільшити» та перевіряє, чи збільшилася кредитна лінія.

Логування основних подій

Логування – це запис будь-яких подій, які відбуваються під час роботи програми. Будь-яку помилку буде усунено в рази швидше, якщо у вас буде вся інформація про неї та всі передісторії викликів [7].

У програмі успішно реалізовано логування, яке фіксує основні дії та виняткові ситуації. Логи зберігаються у файлі.

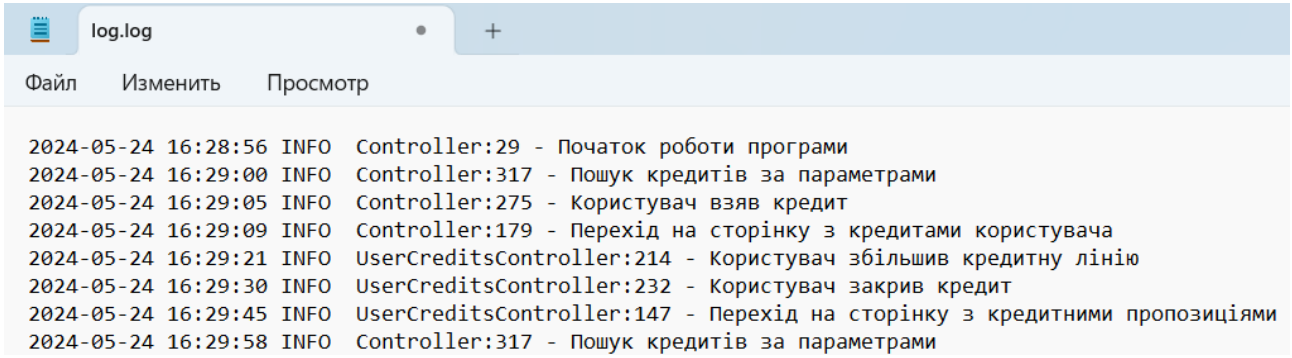

Рис. 4 – Логування основних подій

5. Контрольні приклади та результати їх реалізації

Користувач під час запуску програми потрапляє на сторінку, яка містить набір кредитних пропозицій.

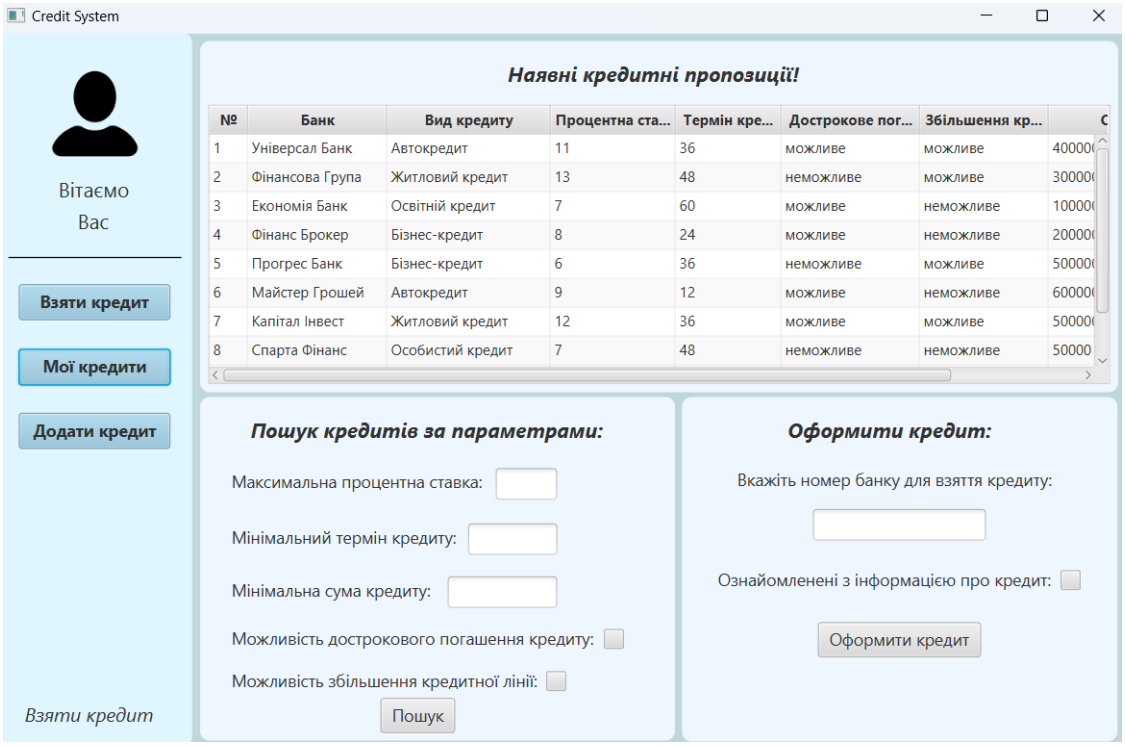

Рис. 5 – Сторінка з кредитними пропозиціями

Він має можливість відсортувати кредитні пропозиції вказавши наступні параметрами: максимальна процентна ставка, мінімальний термін кредиту, мінімальна сума кредиту, можливість дострокового погашення кредиту, можливість збільшення кредитної лінії.

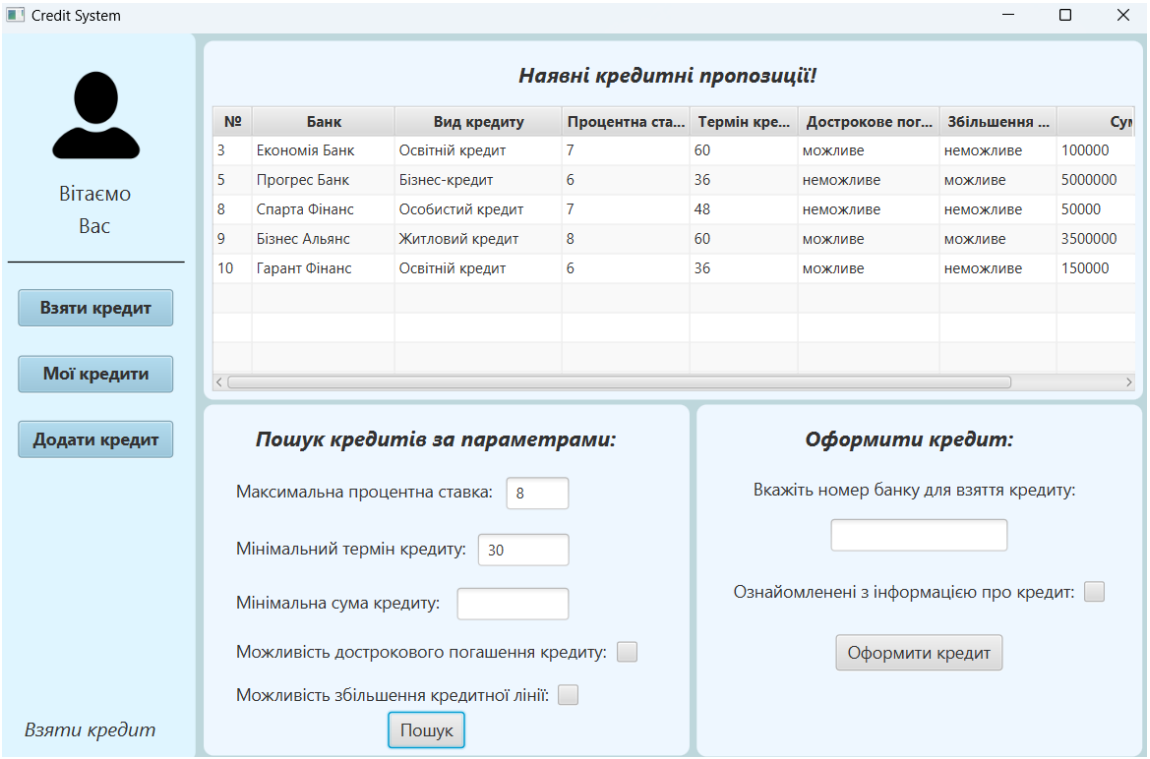

Рис. 6.1 – Пошук кредитів за параметрами

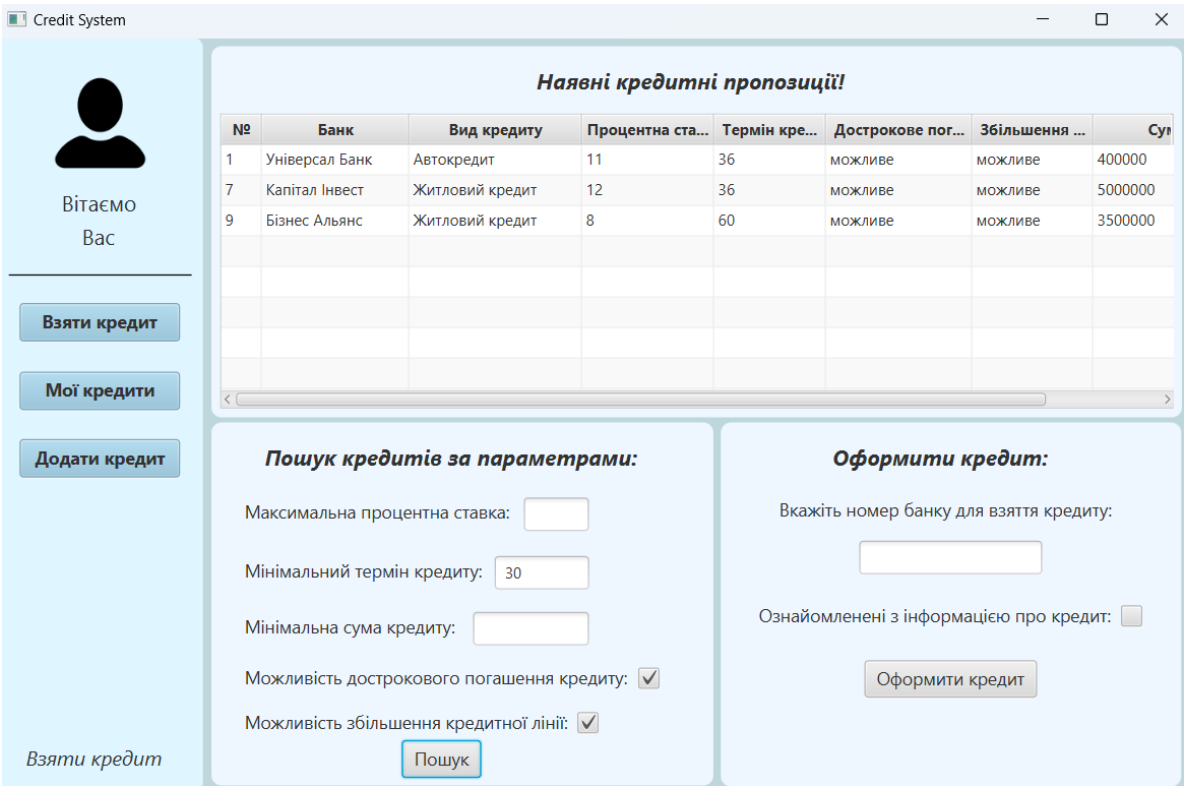

Рис. 6.2 – Пошук кредитів за параметрами

Також користувач має оформити кредит, вказавши номер банку, ознайомившись з інформацією про кредит та натиснувши кнопку «Оформити кредит».

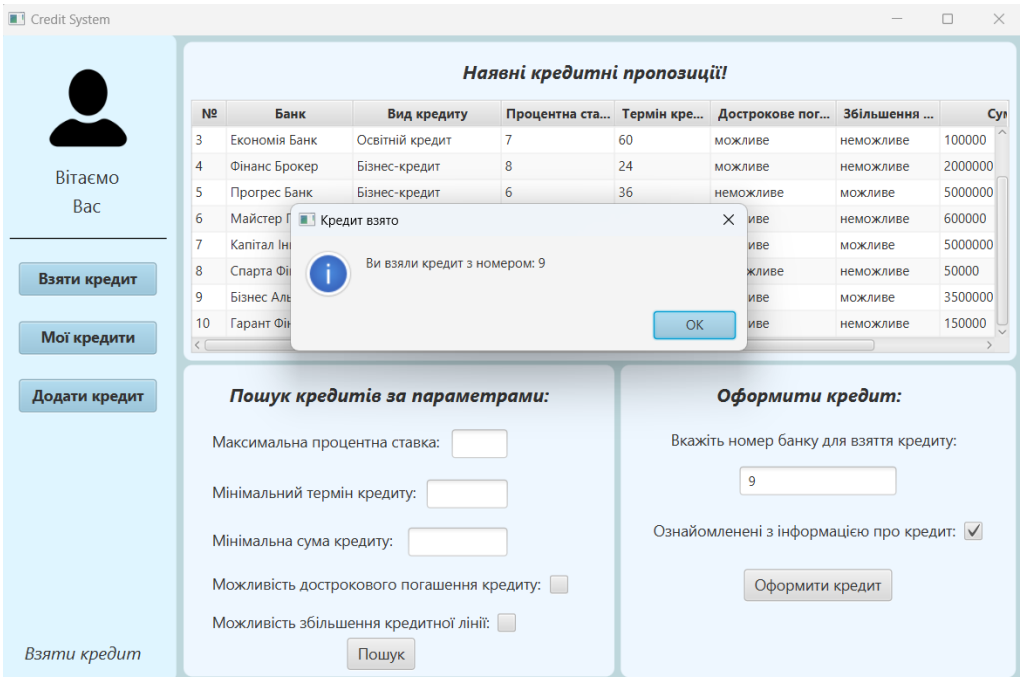

Рис. 7 – Оформлення кредиту

На боковій панелі користувач може перейти на сторінку з його кредитами, натиснувши кнопку «Мої кредити».

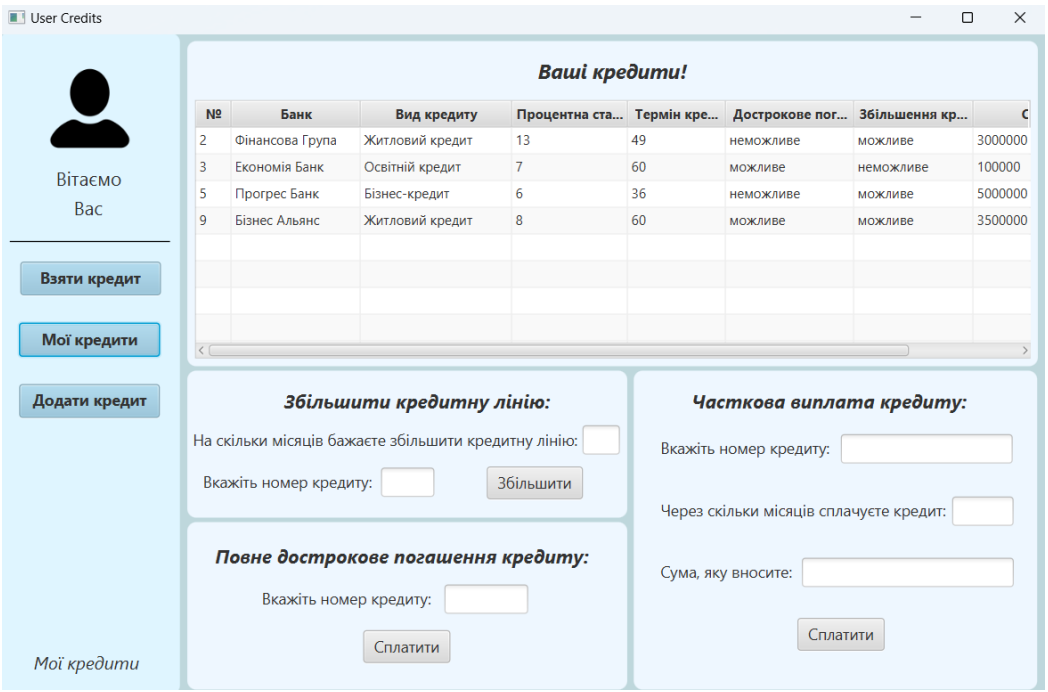

Рис. 8 – Сторінка з кредитами користувача

Користувач може збільшити кредитну лінію, якщо це можливо для певного кредиту. Має можливість достроково закрити кредит та частково виплатити кредит.

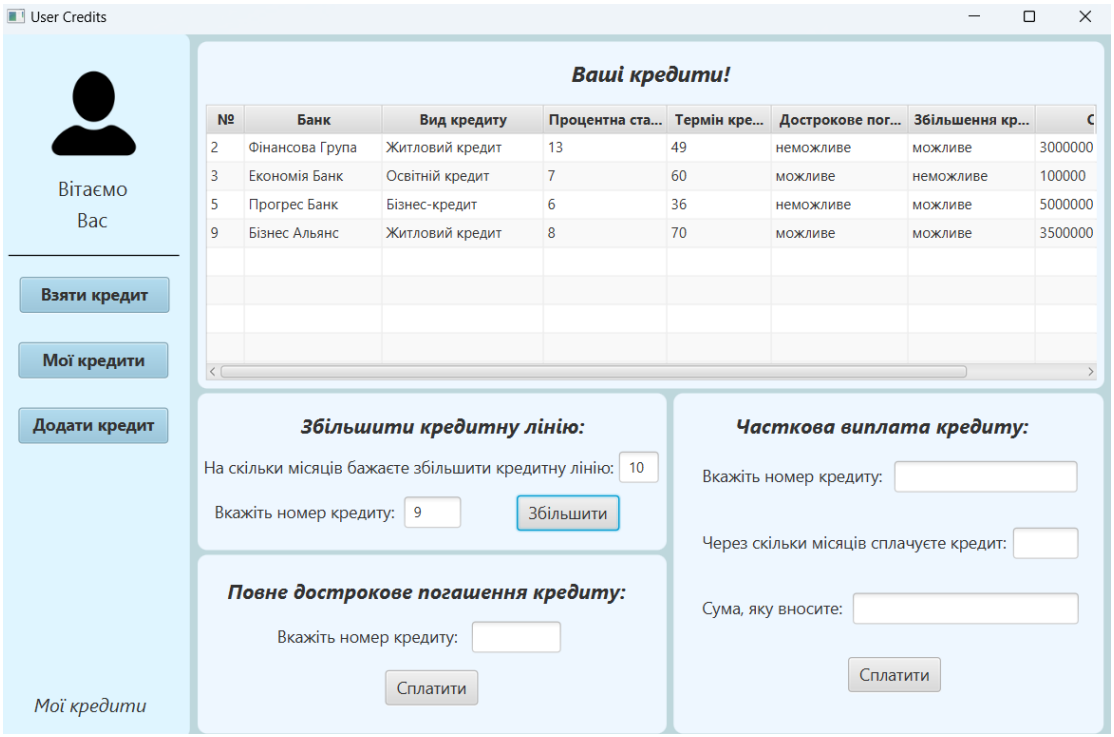

Рис. 9 – Збільшення кредитної лінії

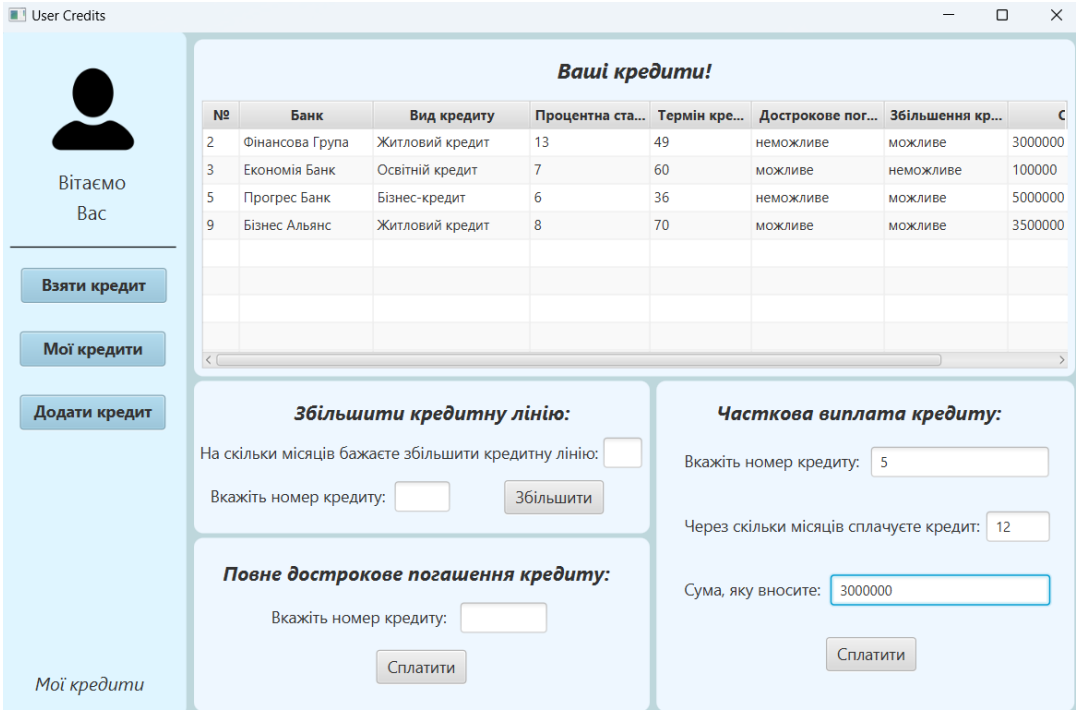

Рис. 10.1 – Часткова виплата кредиту

Рис. 10.2 – Часткова виплата кредиту

На боковій панелі користувач може перейти на сторінку з додаванням кредитної пропозиції, натиснувши кнопку «Додати кредит».

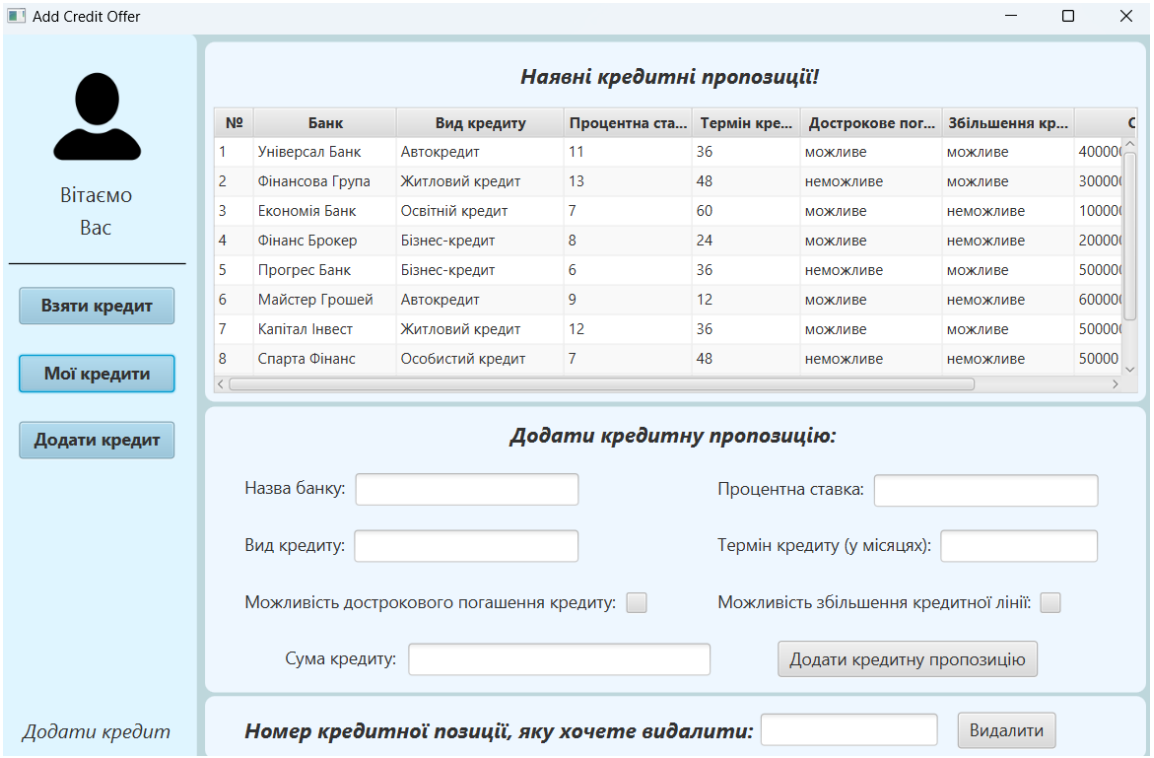

Рис. 11 – Сторінка з додаванням кредитної пропозиції

Користувач може додавати кредитні пропозиції та видаляти їх.

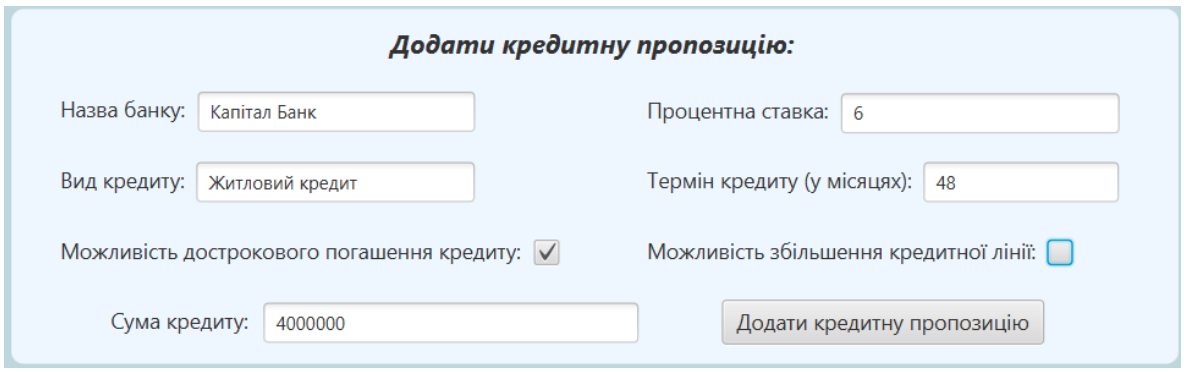

Рис. 12.1 – Додавання кредитної пропозиції

Рис. 12.2 – Додавання кредитної пропозиції

Висновки

В результаті було розроблено настільний додаток на мові програмування Java з використанням JavaFX для створення зрозумілого та зручного інтерфейсу, який надає користувачу можливість пошуку за різними параметрами кредитні пропозиції, можлиівсть брати кредити, збільшувати кредитні лінії, достроково закривати кредит, частково виплачувати кредит, можливість додавання та видалення кредитних пропозицій.

Результати даної роботи можуть бути використані для підвищення зручності та доступності кредитування для користувачів, дозволяючи їм легко порівнювати та вибирати найвигідніші кредитні пропозиції. У майбутньому можливе розширення функціональності додатку, додаючи більшу кількість параметрів пошуку кредитних пропозицій, також можливе покращення графічного інтерфейсу.

Під час розробки цієї системи було отримано цінний досвід у використанні Java та JavaFX для створення графічних інтерфейсів, а також у проєктуванні та реалізації функціональних можливостей, які відповідають потребам користувачів. Крім того, було здобуто практичний досвід роботи з базою даних SQLite, що дозволило ефективно зберігати та маніпулювати даними кредитних пропозицій.

Список використаної літератури

1. Understanding the Importance of Banking Credit and Its Impact on the Economy. Режим доступу: https://enrichest.com/en/blog/understanding-importance-of-banking-credit-and-its-impact-on-economy

2. The influence of financial literacy on smes performance through access to finance and financial risk attitude as mediation variables. Режим доступу: https://www.abacademies.org/articles/The-Influence-of-Financial-Literacy-on-Smes-Performance-Through-Access-1528-2635-24-5-595.pdf

3. Що таке sqlite? Режим доступу: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-sqlite/

4. Unit тестування в Java. Режим доступу: https://foxminded.ua/unit-testuvannya-v-java/

5. The 5th major version of the programmer-friendly testing framework for Java and the JVM. Режим доступу: https://junit.org/junit5/

6. TestFX. Режим доступу: https://testfx.github.io/TestFX/

7. Рівні подій. Режим доступу: https://javarush.com/ua/quests/lectures/ua.questservlets.level05.lecture02

8. Що таке Unit тести і як їх писати. Режим доступу: https://foxminded.ua/yunit-testy/

9. Java Логування. Режим доступу: https://javarush.com/ua/groups/posts/uk.2200.java-loguvannja-rozmotati-klubok-stektreysu

10. Логування: що, як, де і чим? Режим доступу: https://javarush.com/ua/groups/posts/4052-logirovanie-chto-kak-gde-i-chem

Додатки

Додаток 1. Текст всіх програмних файлів

Файл Main.java

```
package main;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;
import org.apache.log4j.Logger;

import java.io.IOException;

/**
 * Клас, що відповідає за запуск та ініціалізацію головного вікна додатку Credit System.
 * Використовує JavaFX для створення графічного інтерфейсу користувача.
 */
public class Main extends Application {

    private static final Logger logger = Logger.getLogger(Controller.class);
    /**
     * Метод start викликається при запуску додатку і ініціалізує головне вікно.
     * @param stage Об'єкт Stage для головного вікна додатку.
     * @throws IOException Якщо виникає помилка при завантаженні файлу FXML.
     */
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("credit-offers.fxml"));
        Scene scene = new Scene(fxmlLoader.load());
        stage.setTitle("Credit System");
        stage.setScene(scene);
        logger.info("Початок роботи програми");
        stage.show();
    }

    /**
     * Вхідна точка програми. Запускає додаток.
     */
    public static void main(String[] args) {
        launch();
    }
}
```

Файл Controller.java

```
package main;

import creditOffer.Credit;
import database.WritingDataToDatabase;
import creditOffer.CreditOffer;
import database.ReadingDataFromDatabase;
```

```
import command.Command;
import command.ResultCommand;
import searchCredits.*;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;
import javafx.scene.control.CheckBox;
import javafx.scene.control.TextField;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import org.apache.log4j.Logger;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;

/**
 * Клас відповідає за управління головним вікном додатку та взаємодію з користувачем.
 */
public class Controller {

    @FXML
    private TableView<Credit> table;
    @FXML
    private TableColumn<Credit,Integer> id;
    @FXML
    private TableColumn<Credit,String> bank;
    @FXML
    private TableColumn<Credit,String> loanType;
    @FXML
    private TableColumn<Credit,Integer> percentRate;
    @FXML
    private TableColumn<Credit,Integer> loanTerm;
    @FXML
    private TableColumn<Credit,Boolean> earlyRepayment;
    @FXML
    private TableColumn<Credit,Boolean> creditLineIncrease;
    @FXML
    private TableColumn<Credit,Integer> creditAmount;

    @FXML
    private Button openCreditButton; // Оголошення змінної для кнопки
    @FXML
    private Button openMyCreditsButton;
    @FXML
    private Button addCreditOfferButton;
```

```
    @FXML
    private Button searchCredits;
    @FXML
    private Button takeCredits;

    @FXML
    private TextField maxPercentRate;
    @FXML
    private TextField minLoanTerm;
    @FXML
    private TextField minLoanAmount;
    @FXML
    private TextField takeCredit;

    private int eventButton;

    @FXML
    private CheckBox creditLineIncreaseCheckBox;

    @FXML
    private CheckBox isEarlyRepayment;

    @FXML
    private CheckBox checkTakeCredit;

    private static List<CreditOffer> creditOffers = new ArrayList<>();

    public ObservableList<Credit> list;

    public static int globalInt = 0;
    Connection co;

    private static final Logger logger = Logger.getLogger(Controller.class);

    /**
     * Ініціалізує головне вікно та завантажує дані при запуску програми.
     */
    public void initialize() {
        if(globalInt == 0) {

            // Ініціалізуємо таблицю з даними зчитаними з файлу
            String database;
            database = "jdbc:sqlite:c:\\sqlite\\credits.db";
            String tableName = "credit_offers";
            list = loadDataFromDatabase(database, tableName);

            id.setCellValueFactory(new PropertyValueFactory<Credit, Integer>("id"));
            bank.setCellValueFactory(new PropertyValueFactory<Credit, String>("bank"));
            loanType.setCellValueFactory(new PropertyValueFactory<Credit, String>("loanType"));
            percentRate.setCellValueFactory(new PropertyValueFactory<Credit, Integer>("percentRate"));
            loanTerm.setCellValueFactory(new PropertyValueFactory<Credit, Integer>("loanTerm"));
            earlyRepayment.setCellValueFactory(new                    PropertyValueFactory<Credit,
Boolean>("earlyRepayment"));
            creditLineIncrease.setCellValueFactory(new                    PropertyValueFactory<Credit,
Boolean>("creditLineIncrease"));
            creditAmount.setCellValueFactory(new PropertyValueFactory<Credit, Integer>("creditAmount"));
```

```
        // Зміна відображення значень тільки при виведенні
        earlyRepayment.setCellFactory(column -> new TableCell<Credit, Boolean>() {
            @Override
            protected void updateItem(Boolean item, boolean empty) {
                super.updateItem(item, empty);
                if (empty || item == null) {
                    setText(null);
                } else {
                    setText(item ? "можливе" : "неможливе");
                }
            }
        });

        creditLineIncrease.setCellFactory(column -> new TableCell<Credit, Boolean>() {
            @Override
            protected void updateItem(Boolean item, boolean empty) {
                super.updateItem(item, empty);
                if (empty || item == null) {
                    setText(null);
                } else {
                    setText(item ? "можливе" : "неможливе");
                }
            }
        });

        table.setItems(list);
    }
    // Викликати метод для налаштування кнопок
    setupButtons();
}

/**
 * Налаштовує обробник подій для кнопок у головному вікні.
 */
private void setupButtons() {
    openCreditButton.setOnAction(event -> {
        openCreditButton.getScene().getWindow().hide();

        FXMLLoader loader  = new FXMLLoader();
        loader.setLocation(getClass().getResource("credit-offers.fxml"));

        try {
            loader.load();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        Parent root = loader.getRoot();
        Stage stage = new Stage();
        stage.setScene(new Scene(root));
        stage.setTitle("Credit Offers");
        stage.show();
    });

    openMyCreditsButton.setOnAction(event -> {
        openMyCreditsButton.getScene().getWindow().hide();
```

```
        FXMLLoader loader_2  = new FXMLLoader();
        loader_2.setLocation(getClass().getResource("user-credits.fxml"));

        try {
            loader_2.load();
            logger.info("Перехід на сторінку з кредитами користувача");
        } catch (IOException e) {
            logger.error("Не вдалося відкрити сторінку з кредитами користувача");
            throw new RuntimeException(e);
        }
        Parent root = loader_2.getRoot();
        Stage stage = new Stage();
        stage.setScene(new Scene(root));
        stage.setTitle("User Credits");
        stage.show();
    });

    addCreditOfferButton.setOnAction(event -> {
        addCreditOfferButton.getScene().getWindow().hide();

        FXMLLoader loader_3  = new FXMLLoader();
        loader_3.setLocation(getClass().getResource("add-credit-offer.fxml"));

        try {
            loader_3.load();
            logger.info("Перехід на сторінку з додаванням кредитної пропозиції");
        } catch (IOException e) {
            logger.error("Не вдалося відкрити сторінку з додаванням кредитної пропозиції");
            throw new RuntimeException(e);
        }
        Parent root = loader_3.getRoot();
        Stage stage = new Stage();
        stage.setScene(new Scene(root));
        stage.setTitle("User Credits");
        stage.show();
    });

    searchCredits.setOnAction(event -> {
        checkBoxAction();
    });

    takeCredits.setOnAction(event -> {

        takeCreditTextField();
    });
}

/**
 * Перевіряє чи введно номер кредиту.
 */
public void takeCreditTextField() {
    if (!takeCredit.getText().isEmpty() && checkTakeCredit.isSelected()) {
        doIfTakeCredit();
        takeCredit.clear();
        checkTakeCredit.setSelected(false); // Зняття прапорця з `checkTakeCredit`
```

```
        } else {
            globalInt = 0;
            initialize();
        }

    }

    /**
     * Обробляє вибір чекбоксів та викликає пошук кредитів за заданими параметрами.
     */
    public void checkBoxAction() {
        if (creditLineIncreaseCheckBox.isSelected()
                || isEarlyRepayment.isSelected()
                || !maxPercentRate.getText().isEmpty()
                || !minLoanTerm.getText().isEmpty()
                || !minLoanAmount.getText().isEmpty()) {
            doSomethingIfCheckBoxSelected();
        } else {
            globalInt = 0;
            initialize();
        }

    }

    /**
     * Додає кредиту до бази даних
     */
    private void doIfTakeCredit() {
        ResultCommand readingFileCommand = new ReadingDataFromDatabase("jdbc:sqlite: d:\\JavaFXDemo
\\sqlite\\credits.db", "credit_offers");
        creditOffers = readingFileCommand.execute();
        if (!takeCredit.getText().isEmpty()) {
            String takeCredittext = takeCredit.getText();
            int takeCredit = Integer.parseInt(takeCredittext);
            Command writingToFileCommand = new WritingDataToDatabase(takeCredit, "credit_offers",
                    "user_credits");
            writingToFileCommand.execute();
            System.out.println("Вітаю ви взяли кредит!");
            loadDataFromList(creditOffers);

            // Відображення спливаючого вікна
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Кредит взято");
            alert.setHeaderText(null);
            alert.setContentText("Ви взяли кредит з номером: " + takeCredit);
            alert.showAndWait();

            logger.info("Користувач взяв кредит");
        }
    }

    /**
     * Пошук кредитів
     */
    private void doSomethingIfCheckBoxSelected() {
```

```
        ResultCommand readingFileCommand = new ReadingDataFromDatabase("jdbc:sqlite:d:\\JavaFXDemo
\\sqlite\\credits.db", "credit_offers");
        creditOffers = readingFileCommand.execute();
        if (creditLineIncreaseCheckBox.isSelected()) {
            ResultCommand searchByIsCreditLineIncreaseEnabledCommand =
                    new SearchByIsCreditLineIncreaseEnabled(creditOffers);
            creditOffers = searchByIsCreditLineIncreaseEnabledCommand.execute();
        }
        if (isEarlyRepayment.isSelected()) {
            ResultCommand searchByIsEarlyRepaymentEnabledCommand =
                    new SearchByIsEarlyRepaymentEnabled(creditOffers);
            creditOffers = searchByIsEarlyRepaymentEnabledCommand.execute();
        }
        if (!maxPercentRate.getText().isEmpty()) {
            String maxPercentRateText = maxPercentRate.getText();
            int maxPercentRateValue = Integer.parseInt(maxPercentRateText);
            ResultCommand searchByMaxInterestRateCommand =
                    new SearchByMaxInterestRate(creditOffers, maxPercentRateValue);
            creditOffers = searchByMaxInterestRateCommand.execute();
        }
        if (!minLoanTerm.getText().isEmpty()) {
            String minLoanTermText = minLoanTerm.getText();
            int minLoanTerm = Integer.parseInt(minLoanTermText);
            ResultCommand searchByMinLoanTermCommand =
                    new SearchByMinLoanTerm(creditOffers, minLoanTerm);
            creditOffers = searchByMinLoanTermCommand.execute();
        }
        if (!minLoanAmount.getText().isEmpty()) {
            String minLoanAmountText = minLoanAmount.getText();
            int minLoanAmount = Integer.parseInt(minLoanAmountText);
            ResultCommand searchByMinLoanAmountCommand =
                    new SearchByMinLoanAmount(creditOffers, minLoanAmount);
            creditOffers = searchByMinLoanAmountCommand.execute();
        }

        logger.info("Пошук кредитів за параметрами");

        loadDataFromList(creditOffers);
        // Тут виконайте необхідні дії, коли чекбокс вибраний
        System.out.println("Checkbox is selected");
    }

    /**
     * Завантажує дані зі списку кредитів до таблиці.
     * @param creditList Список кредитів.
     * @return ObservableList<Credit> Завантажені дані у вигляді ObservableList.
     */
    private ObservableList<Credit> loadDataFromList(List<CreditOffer> creditList) {
        ObservableList<Credit> dataList = FXCollections.observableArrayList();
        for (CreditOffer credit : creditList) {
            int id = credit.getId();
            String bank = credit.getBankName();
            String loanType = credit.getCreditName();
            int percentRate = credit.getInterestRate();
            int loanTerm = credit.getLoanTerm();
            boolean earlyRepayment = credit.getEarlyRepaymentEnabled();
```

```
            boolean creditLineIncrease = credit.getCreditLineIncreaseEnabled();
            int creditAmount = credit.getSumCredit();
            Credit newCredit = new Credit(id, bank, loanType, percentRate, loanTerm, earlyRepayment,
creditLineIncrease, creditAmount);
            dataList.add(newCredit);
        }
        checkGoodBox(dataList);
        return dataList;
    }

    /**
     * Завантажує дані із бази даних до таблиці.
     * @param databaseUrl - посилання на базу даних.
     * @param tableName - назва таблиці бази даних.
     * @return dataList Завантажені дані у вигляді dataList.
     */
    private ObservableList<Credit> loadDataFromDatabase(String databaseUrl, String tableName) {
        ObservableList<Credit> dataList = FXCollections.observableArrayList();
        String sql = "SELECT id, bank, loanType, percentRate, loanTerm, earlyRepayment, creditLineIncrease,
creditAmount FROM " + tableName;

        try (Connection conn = DriverManager.getConnection(databaseUrl);
             PreparedStatement pstmt = conn.prepareStatement(sql);
             ResultSet rs = pstmt.executeQuery()) {

            while (rs.next()) {
                int id = rs.getInt("id");
                String bank = rs.getString("bank");
                String loanType = rs.getString("loanType");
                int percentRate = rs.getInt("percentRate");
                int loanTerm = rs.getInt("loanTerm");
                boolean earlyRepayment = rs.getBoolean("earlyRepayment");
                boolean creditLineIncrease = rs.getBoolean("creditLineIncrease");
                int creditAmount = rs.getInt("creditAmount");

                Credit credit = new Credit(id, bank, loanType, percentRate, loanTerm, earlyRepayment,
creditLineIncrease, creditAmount);
                dataList.add(credit);
            }
        } catch (SQLException e) {
            logger.error("Не вдалося вивести дані з бази даних!");
            e.printStackTrace();
        }

        return dataList;
    }

    /**
     * Оновлює дані.
     */
    private void checkGoodBox(ObservableList<Credit> dataList){
        list = dataList;
        id.setCellValueFactory(new PropertyValueFactory<Credit,Integer>("id"));
        bank.setCellValueFactory(new PropertyValueFactory<Credit,String>("bank"));
        loanType.setCellValueFactory(new PropertyValueFactory<Credit,String>("loanType"));
        percentRate.setCellValueFactory(new PropertyValueFactory<Credit,Integer>("percentRate"));
```

```
        loanTerm.setCellValueFactory(new PropertyValueFactory<Credit,Integer>("loanTerm"));
        earlyRepayment.setCellValueFactory(new
PropertyValueFactory<Credit,Boolean>("earlyRepayment"));
        creditLineIncrease.setCellValueFactory(new
PropertyValueFactory<Credit,Boolean>("creditLineIncrease"));
        creditAmount.setCellValueFactory(new PropertyValueFactory<Credit,Integer>("creditAmount"));
        table.setItems(list);
    }
}
```

Файл UserCreditsController.java

```
package main;

import creditOffer.Credit;
import creditOffer.CreditOffer;
import database.ReadingDataFromDatabase;
import command.Command;
import command.ResultCommand;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;
import org.apache.log4j.Logger;
import userCredits.*;
import java.io.IOException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 * Клас відповідає за управління вікном з кредитами користувача та взаємодію з користувачем.
 */
public class UserCreditsController {

    @FXML
    private TableView<Credit> table;

    @FXML
```

```
private TableColumn<Credit,Integer> id;
@FXML
private TableColumn<Credit,String> bank;
@FXML
private TableColumn<Credit,String> loanType;
@FXML
private TableColumn<Credit,Integer> percentRate;
@FXML
private TableColumn<Credit,Integer> loanTerm;
@FXML
private TableColumn<Credit,Boolean> earlyRepayment;
@FXML
private TableColumn<Credit,Boolean> creditLineIncrease;
@FXML
private TableColumn<Credit,Integer> creditAmount;

@FXML
private Button openCreditButton; // Оголошення змінної для кнопки
@FXML
private Button openMyCreditsButton;
@FXML
private Button addCreditOfferButton;
@FXML
private Button creditIncreaseButton;
@FXML
private Button partLoanRepaymentButton;

@FXML
private Button creditLineDeleteButton;

@FXML
private TextField creditIncreaseLine;
@FXML
private TextField numberCreditIncreaseLine;
```

```
    @FXML
    private TextField numberCreditLineDelete;
    @FXML
    private TextField numberCreditPartRepayment;
    @FXML
    private TextField monthCreditPartRepayment;
    @FXML
    private TextField amountCreditPartRepayment;
    private static List<CreditOffer> creditOffers = new ArrayList<>();

    private int eventButton;

    ObservableList<Credit> list;

    private static final Logger logger = Logger.getLogger(UserCreditsController.class);

    /**
     * Ініціалізує вікно з кредитами користувача та завантажує дані.
     */
    public void initialize() {
        // Ініціалізуємо таблицю з даними зчитаними з файлу
        String database;
        database = "jdbc:sqlite:d:\\JavaFXDemo\\sqlite\\credits.db";
        String tableName = "user_credits";
        list = loadDataFromDatabase(database, tableName);

        id.setCellValueFactory(new PropertyValueFactory<Credit,Integer>("id"));
        bank.setCellValueFactory(new PropertyValueFactory<Credit,String>("bank"));
        loanType.setCellValueFactory(new PropertyValueFactory<Credit,String>("loanType"));
        percentRate.setCellValueFactory(new PropertyValueFactory<Credit,Integer>("percentRate"));
        loanTerm.setCellValueFactory(new PropertyValueFactory<Credit,Integer>("loanTerm"));
        earlyRepayment.setCellValueFactory(new
PropertyValueFactory<Credit,Boolean>("earlyRepayment"));
        creditLineIncrease.setCellValueFactory(new
PropertyValueFactory<Credit,Boolean>("creditLineIncrease"));
```

```
    creditAmount.setCellValueFactory(new PropertyValueFactory<Credit,Integer>("creditAmount"));

    // Зміна відображення значень тільки при виведенні
    earlyRepayment.setCellFactory(column -> new TableCell<Credit, Boolean>() {
        @Override
        protected void updateItem(Boolean item, boolean empty) {
            super.updateItem(item, empty);
            if (empty || item == null) {
                setText(null);
            } else {
                setText(item ? "можливе" : "неможливе");
            }
        }
    });

    creditLineIncrease.setCellFactory(column -> new TableCell<Credit, Boolean>() {
        @Override
        protected void updateItem(Boolean item, boolean empty) {
            super.updateItem(item, empty);
            if (empty || item == null) {
                setText(null);
            } else {
                setText(item ? "можливе" : "неможливе");
            }
        }
    });

    table.setItems(list);

    // Викликати метод для налаштування кнопок
    setupButtons();
}

/**
```

```
 * Налаштовує обробник подій для кнопок у вікні.
 */
private void setupButtons() {
    // Додати обробник події для кнопки відкриття вікна кредиту
    openCreditButton.setOnAction(event -> {
        openCreditButton.getScene().getWindow().hide();

        FXMLLoader loader  = new FXMLLoader();
        loader.setLocation(getClass().getResource("credit-offers.fxml"));

        try {
            loader.load();
            logger.info("Перехід на сторінку з кредитними пропозиціями");
        } catch (IOException e) {
            logger.info("Не вдалося вікрити сторінку з кредитними пропозиціями");
            throw new RuntimeException(e);
        }

        eventButton = 1;
        Parent root = loader.getRoot();
        Stage stage = new Stage();
        stage.setScene(new Scene(root));
        stage.setTitle("Credit Offers");

        stage.show();
    });

    openMyCreditsButton.setOnAction(event -> {
        openMyCreditsButton.getScene().getWindow().hide();

        FXMLLoader loader_2  = new FXMLLoader();
        loader_2.setLocation(getClass().getResource("user-credits.fxml"));

        try {
```

```
            loader_2.load();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        eventButton = 2;
        Parent root = loader_2.getRoot();
        Stage stage = new Stage();
        stage.setScene(new Scene(root));
        stage.setTitle("User Credits");
        stage.show();
    });

    addCreditOfferButton.setOnAction(event -> {
        addCreditOfferButton.getScene().getWindow().hide();

        FXMLLoader loader_3  = new FXMLLoader();
        loader_3.setLocation(getClass().getResource("add-credit-offer.fxml"));

        try {
            loader_3.load();
            logger.info("Перехід на сторінку з додаванням кредитної пропозиції");
        } catch (IOException e) {
            logger.info("Не вдалося відкрити сторінку з додаванням кредитної пропозиції");
            throw new RuntimeException(e);
        }
        Parent root = loader_3.getRoot();
        Stage stage = new Stage();
        stage.setScene(new Scene(root));
        stage.setTitle("Add Credit Offer");
        stage.show();
    });

    creditIncreaseButton.setOnAction(event -> {
```

```
        if (!numberCreditIncreaseLine.getText().isEmpty() && !creditIncreaseLine.getText().isEmpty()) {
            String creditIncreaseLineText = creditIncreaseLine.getText();
            int creditIncreaseLine = Integer.parseInt(creditIncreaseLineText);

            String numberCreditIncreaseLineText = numberCreditIncreaseLine.getText();
            int numberCreditIncreaseLine = Integer.parseInt(numberCreditIncreaseLineText);

            ResultCommand readingFileCommand2 = new ReadingDataFromDatabase("jdbc:sqlite:
d:\\JavaFXDemo\\sqlite\\credits.db", "user_credits");
            creditOffers = readingFileCommand2.execute();
            Command increaseCreditLineCommand = new IncreaseCreditLine(numberCreditIncreaseLine,
creditIncreaseLine, "jdbc:sqlite:c:\\sqlite\\credits.db");
            increaseCreditLineCommand.execute();
            logger.info("Користувач збільшив кредитну лінію");
            initialize();
        } else {
            initialize();
        }
    });


    creditLineDeleteButton.setOnAction(event -> {
        if (!numberCreditLineDelete.getText().isEmpty()) {
            String creditDeleteLineText = numberCreditLineDelete.getText();
            int creditDelete = Integer.parseInt(creditDeleteLineText);

            ResultCommand readingFileCommand2 = new
ReadingDataFromDatabase("jdbc:sqlite:c:\\sqlite\\credits.db", "user_credits");
            creditOffers = readingFileCommand2.execute();
            Command deleteCreditLineCommand = new DeleteUserCredit(creditDelete,
"jdbc:sqlite:c:\\sqlite\\credits.db");
            deleteCreditLineCommand.execute();
            numberCreditLineDelete.clear();
            logger.info("Користувач закрив кредит");
            initialize();
        } else {
```

```
            initialize();
        }
    });

    partLoanRepaymentButton.setOnAction(event -> {
        if (!numberCreditPartRepayment.getText().isEmpty()
                && !monthCreditPartRepayment.getText().isEmpty()
                && !amountCreditPartRepayment.getText().isEmpty()) {
            String creditPartRepaymentLineText = numberCreditPartRepayment.getText();
            int numberCredit = Integer.parseInt(creditPartRepaymentLineText);

            String monthPartRepaymentLineText = monthCreditPartRepayment.getText();
            int monthCredit = Integer.parseInt(monthPartRepaymentLineText);

            String amountPartRepaymentLineText = amountCreditPartRepayment.getText();
            int amountCredit = Integer.parseInt(amountPartRepaymentLineText);

            ResultCommand readingFileCommand2 = new ReadingDataFromDatabase("jdbc:sqlite:
d:\\JavaFXDemo \\sqlite\\credits.db", "user_credits");
            creditOffers = readingFileCommand2.execute();

            Command partRepaymentCreditLineCommand = new PartLoanRepayment(numberCredit,
monthCredit, amountCredit, "jdbc:sqlite:d:\\JavaFXDemo\\sqlite\\credits.db");
            partRepaymentCreditLineCommand.execute();

            numberCreditPartRepayment.clear();
            monthCreditPartRepayment.clear();
            amountCreditPartRepayment.clear();

            logger.info("Користувач вніс часткову виплату за кредит");

            initialize();
        } else {
            initialize();
        }
```

```
        });
    }



    /**
     * Завантажує дані із бази даних до таблиці.
     * @param databaseUrl - посилання на базу даних.
     * @param tableName - назва таблиці бази даних.
     * @return dataList Завантажені дані у вигляді dataList.
     */
    private ObservableList<Credit> loadDataFromDatabase(String databaseUrl, String tableName) {
        ObservableList<Credit> dataList = FXCollections.observableArrayList();
        String sql = "SELECT id, bank, loanType, percentRate, loanTerm, earlyRepayment, creditLineIncrease,
creditAmount FROM " + tableName;


        try (Connection conn = DriverManager.getConnection(databaseUrl);
             PreparedStatement pstmt = conn.prepareStatement(sql);
             ResultSet rs = pstmt.executeQuery()) {


            while (rs.next()) {
                int id = rs.getInt("id");
                String bank = rs.getString("bank");
                String loanType = rs.getString("loanType");
                int percentRate = rs.getInt("percentRate");
                int loanTerm = rs.getInt("loanTerm");
                boolean earlyRepayment = rs.getBoolean("earlyRepayment");
                boolean creditLineIncrease = rs.getBoolean("creditLineIncrease");
                int creditAmount = rs.getInt("creditAmount");


                Credit credit = new Credit(id, bank, loanType, percentRate, loanTerm, earlyRepayment,
creditLineIncrease, creditAmount);
                dataList.add(credit);
            }
        } catch (SQLException e) {
            logger.error("Не вдалося вивести дані з бази даних!");
```

```
            e.printStackTrace();
        }

        return dataList;
    }
}
```

Файл AddCreditOfferController

```
package main;

import creditOffer.Credit;
import creditOffer.CreditOffer;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;
import javafx.scene.control.CheckBox;
import javafx.scene.control.TextField;
import org.apache.log4j.Logger;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
```

```
 * Клас відповідає за управління вікном для додавання кредитної пропозиції та взаємодію з користувачем.
 */
public class AddCreditOfferController {


    @FXML
    private TableView<Credit> table;
    @FXML
    private TableColumn<Credit,Integer> id;
    @FXML
    private TableColumn<Credit,String> bank;
    @FXML
    private TableColumn<Credit,String> loanType;
    @FXML
    private TableColumn<Credit,Integer> percentRate;
    @FXML
    private TableColumn<Credit,Integer> loanTerm;
    @FXML
    private TableColumn<Credit,Boolean> earlyRepayment;
    @FXML
    private TableColumn<Credit,Boolean> creditLineIncrease;
    @FXML
    private TableColumn<Credit,Integer> creditAmount;


    @FXML
    private Button openCreditButton; // Оголошення змінної для кнопки
    @FXML
    private Button openMyCreditsButton;
    @FXML
    private Button addCreditOfferButton;
    @FXML
    private Button addCreditButton;
    @FXML
```

```
private Button deleteCreditButton;



@FXML
private TextField bankNameTextField;
@FXML
private TextField loanTypeTextField;
@FXML
private TextField percentRateTextField;
@FXML
private CheckBox isEarlyRepaymentCheckBox;
@FXML
private TextField creditSumTextField;
@FXML
private TextField loanTermTextField;
@FXML
private TextField numberCreditToDeleteTextField;

@FXML
private CheckBox creditLineIncreaseCheckBox;



private static List<CreditOffer> creditOffers = new ArrayList<>();

ObservableList<Credit> list;

public static int globalInt = 0;
Connection co;

private static final Logger logger = Logger.getLogger(AddCreditOfferController.class);

/**
```

```
    * Ініціалізує вікно з додаванням кредитної пропозиції та завантажує дані про наявні кредитні пропозиції.
    */
   public void initialize() {
      if(globalInt == 0) {

         // Ініціалізуємо таблицю з даними зчитаними з файлу
         String database;
         database = "jdbc:sqlite:c:\\sqlite\\credits.db";
         String tableName = "credit_offers";
         list = loadDataFromDatabase(database, tableName);

         id.setCellValueFactory(new PropertyValueFactory<Credit, Integer>("id"));
         bank.setCellValueFactory(new PropertyValueFactory<Credit, String>("bank"));
         loanType.setCellValueFactory(new PropertyValueFactory<Credit, String>("loanType"));
         percentRate.setCellValueFactory(new PropertyValueFactory<Credit, Integer>("percentRate"));
         loanTerm.setCellValueFactory(new PropertyValueFactory<Credit, Integer>("loanTerm"));
         earlyRepayment.setCellValueFactory(new PropertyValueFactory<Credit,
Boolean>("earlyRepayment"));
         creditLineIncrease.setCellValueFactory(new PropertyValueFactory<Credit,
Boolean>("creditLineIncrease"));
         creditAmount.setCellValueFactory(new PropertyValueFactory<Credit, Integer>("creditAmount"));

         // Зміна відображення значень тільки при виведенні
         earlyRepayment.setCellFactory(column -> new TableCell<Credit, Boolean>() {
            @Override
            protected void updateItem(Boolean item, boolean empty) {
               super.updateItem(item, empty);
               if (empty || item == null) {
                  setText(null);
               } else {
                  setText(item ? "можливе" : "неможливе");
               }
            }
```

```
        });

        creditLineIncrease.setCellFactory(column -> new TableCell<Credit, Boolean>() {
            @Override
            protected void updateItem(Boolean item, boolean empty) {
                super.updateItem(item, empty);
                if (empty || item == null) {
                    setText(null);
                } else {
                    setText(item ? "можливе" : "неможливе");
                }
            }
        });

        table.setItems(list);
    }

    setupButtons();
}

/**
 * Налаштовує обробник подій для кнопок у вікні.
 */
private void setupButtons() {
    openCreditButton.setOnAction(event -> {
        openCreditButton.getScene().getWindow().hide();

        FXMLLoader loader  = new FXMLLoader();
        loader.setLocation(getClass().getResource("credit-offers.fxml"));

        try {
            loader.load();
            logger.info("Перехід на сторінку з кредитними пропозиціями");
```

```
    } catch (IOException e) {
        logger.info("Не вдалося вікрити сторінку з кредитними пропозиціями");
        throw new RuntimeException(e);
    }
    Parent root = loader.getRoot();
    Stage stage = new Stage();
    stage.setScene(new Scene(root));
    stage.setTitle("Credit Offers");
    stage.show();
});

openMyCreditsButton.setOnAction(event -> {
    openMyCreditsButton.getScene().getWindow().hide();

    FXMLLoader loader_2  = new FXMLLoader();
    loader_2.setLocation(getClass().getResource("user-credits.fxml"));

    try {
        loader_2.load();
        logger.info("Перехід на сторінку з кредитами користувача");
    } catch (IOException e) {
        logger.error("Не вдалося відкрити сторінку з кредитами користувача");
        throw new RuntimeException(e);
    }
    Parent root = loader_2.getRoot();
    Stage stage = new Stage();
    stage.setScene(new Scene(root));
    stage.setTitle("User Credits");
    stage.show();
});

addCreditOfferButton.setOnAction(event -> {
    addCreditOfferButton.getScene().getWindow().hide();
```

```
        FXMLLoader loader_3  = new FXMLLoader();
        loader_3.setLocation(getClass().getResource("add-credit-offer.fxml"));

        try {
            loader_3.load();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        Parent root = loader_3.getRoot();
        Stage stage = new Stage();
        stage.setScene(new Scene(root));
        stage.setTitle("Add Credit Offer");
        stage.show();
    });

    addCreditButton.setOnAction(event -> {
        addCreditOfferToDatabase();

    });

    deleteCreditButton.setOnAction(event -> {
        deleteCreditOfferFromDatabase();
    });

}

/**
 * Завантажує дані із бази даних до таблиці.
 * @param databaseUrl - посилання на базу даних.
 * @param tableName - назва таблиці бази даних.
 * @return dataList Завантажені дані у вигляді dataList.
 */
```

```
    private ObservableList<Credit> loadDataFromDatabase(String databaseUrl, String tableName) {
        ObservableList<Credit> dataList = FXCollections.observableArrayList();
        String sql = "SELECT id, bank, loanType, percentRate, loanTerm, earlyRepayment, creditLineIncrease, creditAmount FROM " + tableName;

        try (Connection conn = DriverManager.getConnection(databaseUrl);
             PreparedStatement pstmt = conn.prepareStatement(sql);
             ResultSet rs = pstmt.executeQuery()) {

            while (rs.next()) {
                int id = rs.getInt("id");
                String bank = rs.getString("bank");
                String loanType = rs.getString("loanType");
                int percentRate = rs.getInt("percentRate");
                int loanTerm = rs.getInt("loanTerm");
                boolean earlyRepayment = rs.getBoolean("earlyRepayment");
                boolean creditLineIncrease = rs.getBoolean("creditLineIncrease");
                int creditAmount = rs.getInt("creditAmount");

                Credit credit = new Credit(id, bank, loanType, percentRate, loanTerm, earlyRepayment, creditLineIncrease, creditAmount);
                dataList.add(credit);
            }
        } catch (SQLException e) {
            logger.error("Не вдалося вивести дані з бази даних!");
            e.printStackTrace();
        }

        return dataList;
    }

    /**
     * Додає кредитну прозицію до бази даних.
     */
```

```
    @FXML
    private void addCreditOfferToDatabase() {
        // Перевірка, чи всі поля заповнені
        if (isInputValid()) {
            // Отримуємо дані з текстових полів та чекбоксів
            String bankName = bankNameTextField.getText();
            String loanType = loanTypeTextField.getText();
            int percentRate = Integer.parseInt(percentRateTextField.getText());
            int loanTerm = Integer.parseInt(loanTermTextField.getText());
            boolean earlyRepayment = isEarlyRepaymentCheckBox.isSelected();
            boolean creditLineIncrease = creditLineIncreaseCheckBox.isSelected();
            int creditAmount = Integer.parseInt(creditSumTextField.getText());

            // Підготовка запиту SQL для вставки даних в базу даних
            String sql = "INSERT INTO credit_offers (bank, loanType, percentRate, loanTerm, earlyRepayment,
creditLineIncrease, creditAmount) VALUES (?, ?, ?, ?, ?, ?, ?)";

            try (Connection conn = DriverManager.getConnection("jdbc:sqlite: d:\\JavaFXDemo
\\sqlite\\credits.db");
                 PreparedStatement pstmt = conn.prepareStatement(sql)) {

                // Встановлюємо значення параметрів запиту SQL
                pstmt.setString(1, bankName);
                pstmt.setString(2, loanType);
                pstmt.setInt(3, percentRate);
                pstmt.setInt(4, loanTerm);
                pstmt.setBoolean(5, earlyRepayment);
                pstmt.setBoolean(6, creditLineIncrease);
                pstmt.setInt(7, creditAmount);

                // Виконуємо запит SQL
                pstmt.executeUpdate();

                // Вивід повідомлення про успішне додавання
```

```
            System.out.println("Дані успішно додані в базу даних!");

            // Опціонально: очищаємо поля введення
            clearInputFields();
            initialize();
        } catch (SQLException e) {
            logger.error("Не вдалося додати кредит до бази даних!");
            System.out.println("Помилка при додаванні даних в базу даних: " + e.getMessage());
        }
    } else {
        System.out.println("Будь ласка, заповніть всі поля.");
    }
}

/**
 * Видаляє кредитну прозицію із бази даних.
 */
@FXML
private void deleteCreditOfferFromDatabase() {
    // Перевірка, чи поле ID кредиту заповнене
    if (!numberCreditToDeleteTextField.getText().isEmpty()) {
        int id = Integer.parseInt(numberCreditToDeleteTextField.getText());

        String sql = "DELETE FROM credit_offers WHERE id = ?";

        try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:d:\\JavaFXDemo\\sqlite\\credits.db");
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setInt(1, id);

            pstmt.executeUpdate();
```

```
            System.out.println("Кредитна пропозиція з ID " + id + " була видалена з бази даних.");

            numberCreditToDeleteTextField.clear();

            logger.error("Видалено кредитну пропозицію з бази даних");

            initialize();
        } catch (SQLException e) {
            logger.error("Не вдалося видалити кредит з бази даних!");
            System.out.println("Помилка видалення з бази даних: " + e.getMessage());
        }
    } else {
        System.out.println("Введіть ID для видалення!");
    }
}


/**
 * Перевіряє чи всі поля заповнені для додавання кредитної пропозиції
 */
private boolean isInputValid() {
    return !bankNameTextField.getText().isEmpty() &&
            !loanTypeTextField.getText().isEmpty() &&
            !percentRateTextField.getText().isEmpty() &&
            !loanTermTextField.getText().isEmpty() &&
            !creditSumTextField.getText().isEmpty();
}

/**
 * Очищає дані з текстових полів
 */
private void clearInputFields() {
    bankNameTextField.clear();
```

```
        loanTypeTextField.clear();
        percentRateTextField.clear();
        loanTermTextField.clear();
        creditSumTextField.clear();
    }
}
```

Файл Command.java

```
package command;

public interface Command {
    void execute();
}
```

Файл ResultCommand.java

```
package command;

import java.util.List;
import creditOffer.CreditOffer;

public interface ResultCommand {
    List<CreditOffer> execute();
}
```

Файл CreditOffer.java

```
package creditOffer;

/**
 * Клас, який містить гетери, сетери змінних та метод toString для даних про кредити
 */
public class CreditOffer {
    private int id;
    private final String bankName;
```

```
private final String creditName;
private final int interestRate;
private int loanTerm;
private int sumCredit;
private final boolean earlyRepaymentEnabled;
private final boolean creditLineIncreaseEnabled;

public CreditOffer(int id, String bankName, String creditName, int interestRate, int loanTerm,
            boolean earlyRepaymentEnabled, boolean creditLineIncreaseEnabled, int sumCredit) {
   this.id = id;
   this.bankName = bankName;
   this.creditName = creditName;
   this.interestRate = interestRate;
   this.loanTerm = loanTerm;
   this.earlyRepaymentEnabled = earlyRepaymentEnabled;
   this.creditLineIncreaseEnabled = creditLineIncreaseEnabled;
   this.sumCredit = sumCredit;
}

public int getId() {
   return id;
}
public String getBankName() {
   return bankName;
}
public String getCreditName() {
   return creditName;
}

public int getInterestRate() {
   return interestRate;
}

public int getLoanTerm() {
```

```
        return loanTerm;
    }
    public int getSumCredit() {
        return sumCredit;
    }


    public boolean getEarlyRepaymentEnabled() {
        return earlyRepaymentEnabled;
    }

    public boolean getCreditLineIncreaseEnabled() {
        return creditLineIncreaseEnabled;
    }
}
```

Файл ReadingDataFromDatabase.java

```
package database;

import command.ResultCommand;
import creditOffer.CreditOffer;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

/**
 * Клас для зчитування даних з бази даних
 */
public class ReadingDataFromDatabase implements ResultCommand {
```

```
private final String databaseUrl;
private final String tableName;

/**
 * Конструктор
 *
 * @param databaseUrl - URL бази даних
 * @param tableName   - назва таблиці
 */
public ReadingDataFromDatabase(String databaseUrl, String tableName) {
    this.databaseUrl = databaseUrl;
    this.tableName = tableName;
}

/**
 * Метод для зчитування даних з бази даних
 *
 * @return - список зі зчитаних даних про кредити
 */
public List<CreditOffer> execute() {
    List<CreditOffer> creditOffers = new ArrayList<>();
    String selectSql = "SELECT * FROM " + tableName;

    try (Connection conn = DriverManager.getConnection(databaseUrl);
         PreparedStatement pstmt = conn.prepareStatement(selectSql)) {

        ResultSet rs = pstmt.executeQuery();
        while (rs.next()) {
            int id = rs.getInt("id");
            String bankName = rs.getString("bank");
            String creditName = rs.getString("loanType");
            int interestRate = rs.getInt("percentRate");
            int loanTerm = rs.getInt("loanTerm");
            boolean earlyRepaymentEnabled = rs.getBoolean("earlyRepayment");
```

```
            boolean creditLineIncreaseEnabled = rs.getBoolean("creditLineIncrease");
            int sumCredit = rs.getInt("creditAmount");

            CreditOffer creditOffer = new CreditOffer(id, bankName, creditName, interestRate,
                    loanTerm, earlyRepaymentEnabled, creditLineIncreaseEnabled, sumCredit);
            creditOffers.add(creditOffer);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return creditOffers;
  }
}
```

Файл WritingDataToDatabase.java

```
package database;

import command.Command;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

/**
 * Клас для запису даних про кредити користувача в іншу таблицю бази даних
 */
public class WritingDataToDatabase implements Command {
    private final int numberCredit;
    private final String sourceTable;
    private final String destinationTable;

    /**
```

```
    * Конструктор
    *
    * @param numberCredit      - номер кредиту, дані якого ви хочете записати
    * @param sourceTable      - назва таблиці, з якої беруться дані
    * @param destinationTable - назва таблиці, у яку вставляються дані
    */
   public WritingDataToDatabase(int numberCredit, String sourceTable, String destinationTable) {
      this.numberCredit = numberCredit;
      this.sourceTable = sourceTable;
      this.destinationTable = destinationTable;
   }


   public void execute() {
      String selectSql = "SELECT * FROM " + sourceTable + " WHERE id = ?";
      String insertSql = "INSERT INTO " + destinationTable + " (id, bank, loanType, percentRate, loanTerm,
earlyRepayment, creditLineIncrease, creditAmount) VALUES (?, ?, ?, ?, ?, ?, ?, ?)"; // SQL-запит для
вставки даних у призначення


      try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:d:\\JavaFXDemo\\sqlite\\credits.db"); // Встановлення з'єднання
з базою даних
           PreparedStatement selectStmt = conn.prepareStatement(selectSql);
           PreparedStatement insertStmt = conn.prepareStatement(insertSql)) {


         selectStmt.setInt(1, numberCredit); // Встановлення значення параметра у SQL-запиті для вибору
даних
         ResultSet rs = selectStmt.executeQuery(); // Виконання SQL-запиту для вибору даних
         if (rs.next()) { // Якщо є результат вибору
            // Встановлення значень параметрів у SQL-запиті для вставки даних
            insertStmt.setInt(1, rs.getInt("id"));
            insertStmt.setString(2, rs.getString("bank"));
            insertStmt.setString(3, rs.getString("loanType"));
            insertStmt.setInt(4, rs.getInt("percentRate"));
            insertStmt.setInt(5, rs.getInt("loanTerm"));
            insertStmt.setInt(6, rs.getInt("earlyRepayment"));
            insertStmt.setInt(7, rs.getInt("creditLineIncrease"));
```

```
            insertStmt.setInt(8, rs.getInt("creditAmount"));
            insertStmt.executeUpdate(); // Виконання SQL-запиту для вставки даних
            System.out.println("Дані кредиту з id " + numberCredit + " успішно вставлено в таблицю " +
destinationTable);
        } else {
            System.out.println("Кредит з id " + numberCredit + " не знайдено в таблиці " + sourceTable);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

Файл SearchCredits.java

```
package searchCredits;

import creditOffer.CreditOffer;
import command.*;

import java.util.List;

/**
 * Абстрактний клас для пошуку кредитів
 */
public abstract class SearchCredits implements ResultCommand {
    protected List<CreditOffer> creditOffers;

    public SearchCredits(List<CreditOffer> creditOffers) {
        this.creditOffers = creditOffers;
    }

    public abstract List<CreditOffer> execute();
}
```

Файл SearchByMinLoanTerm.java

```
package searchCredits;

import creditOffer.CreditOffer;

import java.util.ArrayList;
import java.util.List;

/**
 * Клас для пошуку кредитів за мінімальним терміном кредиту
 */
public class SearchByMinLoanTerm extends SearchCredits {
    private int minLoanTerm;

    /**
     * Конструктор
     * @param creditOffers - список кредитних пропозицій
     * @param minLoanTerm - мінімальний термін для кредиту
     */
    public SearchByMinLoanTerm(List<CreditOffer> creditOffers, int minLoanTerm) {
        super(creditOffers);
        this.minLoanTerm = minLoanTerm;
    }


    public List<CreditOffer> execute() {
        List<CreditOffer> filteredOffers = new ArrayList<>();
        for (CreditOffer offer : creditOffers) {
            if (offer.getLoanTerm() >= minLoanTerm) {
                filteredOffers.add(offer);
            }
        }
        return filteredOffers;
    }
```

```
}
```

Файл SearchByMinLoanAmount.java

```
package searchCredits;

import creditOffer.CreditOffer;
import java.util.ArrayList;
import java.util.List;

/**
 * Клас для пошуку кредитних пропозицій за мінімальною сумою
 */
public class SearchByMinLoanAmount extends SearchCredits {
    private int minLoanAmount;

    /**
     * Конструктор
     * @param creditOffers - список кредитних пропозицій
     * @param minLoanAmount - мінімальна сума
     */
    public SearchByMinLoanAmount(List<CreditOffer> creditOffers, int minLoanAmount) {
        super(creditOffers);
        this.minLoanAmount = minLoanAmount;
    }


    public List<CreditOffer> execute() {
        List<CreditOffer> filteredOffers = new ArrayList<>();
        for (CreditOffer offer : creditOffers) {
            if (offer.getSumCredit() >= minLoanAmount) {
                filteredOffers.add(offer);
            }
        }
        return filteredOffers;
```

```
    }
}
```

Файл SearchByMaxInterestRate.java

```
package searchCredits;
import creditOffer.CreditOffer;

import java.util.ArrayList;
import java.util.List;

/**
 * Клас для пошуку кредитів за максимальним відсотковою ставкою
 */
public class SearchByMaxInterestRate extends SearchCredits {
    private int maxInterestRate;

    /**
     * Конструктор
     * @param creditOffers - список кредитних пропозицій
     * @param maxInterestRate - максимальна відсоткова ставка
     */
    public SearchByMaxInterestRate(List<CreditOffer> creditOffers, int maxInterestRate) {
        super(creditOffers);
        this.maxInterestRate = maxInterestRate;
    }



    public List<CreditOffer> execute() {
        List<CreditOffer> filteredOffers = new ArrayList<>();
        for (CreditOffer offer : creditOffers) {
            if (offer.getInterestRate() <= maxInterestRate) {
                filteredOffers.add(offer);
            }
        }
        return filteredOffers;
```

```
    }
}
```

Файл SearchByIsEarlyRepaymentEnabled.java

```
package searchCredits;

import creditOffer.CreditOffer;

import java.util.ArrayList;
import java.util.List;

/**
 * Клас для пошуку кредитів, які мають можливість дострокового закриття кредиту
 */
public class SearchByIsEarlyRepaymentEnabled extends SearchCredits{
    public SearchByIsEarlyRepaymentEnabled(List<CreditOffer> creditOffers) {
        super(creditOffers);
    }

    public List<CreditOffer> execute() {
        List<CreditOffer> filteredOffers = new ArrayList<>();
        for (CreditOffer offer : creditOffers) {
            if (offer.getEarlyRepaymentEnabled()) {
                filteredOffers.add(offer);
            }
        }
        return filteredOffers;
    }
}
```

Файл SearchByIsCreditLineIncreaseEnabled.java

```
package searchCredits;

import creditOffer.CreditOffer;

import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Клас для пошуку кредитів, які мають можливість збільшення кредитної лінії
 */
public class SearchByIsCreditLineIncreaseEnabled extends SearchCredits{

    /**
     * Конструктор
     * @param creditOffers - список кредитних пропозицій
     */
    public SearchByIsCreditLineIncreaseEnabled(List<CreditOffer> creditOffers) {
        super(creditOffers);
    }

    /**
     * Метод для пошуку кредитів, які мають можливість збільшення кредитної лінії
     * @return фідвільтрований список
     */
    public List<CreditOffer> execute() {
        List<CreditOffer> filteredOffers = new ArrayList<>();
        for (CreditOffer offer : creditOffers) {
            if (offer.getCreditLineIncreaseEnabled()) {
                filteredOffers.add(offer);
            }
        }
        return filteredOffers;
    }
}
```

Файл DeleteUserCredit.java

```
package userCredits;

import command.Command;
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 * Клас для видалення кредиту користувача
 */
public class DeleteUserCredit implements Command {
    private final int creditId;
    private final String databaseUrl;

    public DeleteUserCredit(int creditId, String databaseUrl) {
        this.creditId = creditId;
        this.databaseUrl = databaseUrl;
    }

    public void execute() {
        // SQL-запит для видалення запису з бази даних, якщо earlyRepayment = true
        String deleteSql = "DELETE FROM user_credits WHERE id = ? AND earlyRepayment = 1";

        try (Connection conn = DriverManager.getConnection(databaseUrl);
             PreparedStatement pstmt = conn.prepareStatement(deleteSql)) {

            // Встановлюємо значення параметра у SQL-запиті
            pstmt.setInt(1, creditId);

            // Виконуємо SQL-запит для видалення запису
            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Кредит з id " + creditId + " успішно видалено з бази даних.");
            } else {
                System.out.println("Кредит з id " + creditId + " не було знайдено в базі даних або умова
earlyRepayment = true не виконана.");
            }
```

```
        } catch (SQLException e) {
            System.out.println("Помилка видалення кредиту з бази даних: " + e.getMessage());
        }
    }
}
```

Файл IncreaseCreditLine.java

```
package userCredits;

import command.Command;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

/**
 * Клас для збільшення кредитної лінії користувача
 */
public class IncreaseCreditLine implements Command {
    private final int creditId;
    private final int monthsToIncrease;
    private final String databaseUrl;

    public IncreaseCreditLine(int creditId, int monthsToIncrease, String databaseUrl) {
        this.creditId = creditId;
        this.monthsToIncrease = monthsToIncrease;
        this.databaseUrl = databaseUrl;
    }

    public void execute() {
        // SQL-запит для збільшення кредитної лінії у базі даних
        String updateSql = "UPDATE user_credits SET loanTerm = loanTerm + ? WHERE id = ? AND
creditLineIncrease = 1";
```

```
        try (Connection conn = DriverManager.getConnection(databaseUrl);
             PreparedStatement pstmt = conn.prepareStatement(updateSql)) {

            // Встановлюємо значення параметрів у SQL-запиті
            pstmt.setInt(1, monthsToIncrease);
            pstmt.setInt(2, creditId);

            // Виконуємо SQL-запит для збільшення кредитної лінії
            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {
                System.out.println("Кредитна лінія збільшена на " + monthsToIncrease + " місяців.");
            } else {
                System.out.println("Кредит з id " + creditId + " не було знайдено в базі даних або умова creditLineIncrease = true не виконана..");
            }

        } catch (SQLException e) {
            System.out.println("Помилка збільшення кредитної лінії: " + e.getMessage());
        }
    }
}
```

Файл PartLoanRepayment.java

```
package userCredits;

import command.Command;

import java.sql.*;

/**
 * Клас для часткової виплати за кредит
 */
```

```
public class PartLoanRepayment implements Command {
    private final int creditId;
    private final int sum;
    private final int months;
    private final String databaseUrl;

    public PartLoanRepayment(int creditId, int months, int sum, String databaseUrl) {
        this.creditId = creditId;
        this.sum = sum;
        this.months = months;
        this.databaseUrl = databaseUrl;
    }

    public void execute() {
        // SQL-запит для отримання суми кредиту та процентної ставки за вказаним id
        String selectSql = "SELECT creditAmount, percentRate FROM user_credits WHERE id = ?";

        try (Connection conn = DriverManager.getConnection(databaseUrl);
             PreparedStatement selectStmt = conn.prepareStatement(selectSql)) {

            // Встановлюємо значення параметра у SQL-запиті
            selectStmt.setInt(1, creditId);

            try (ResultSet rs = selectStmt.executeQuery()) {
                if (rs.next()) {
                    int sumCredit = rs.getInt("creditAmount");
                    int interestRate = rs.getInt("percentRate");
                    // Розраховуємо bonusSumCredit, використовуючи отриману суму кредиту sumCredit та процентну ставку interestRate
                    int bonusSumCredit = (int) (sumCredit * (interestRate / 100.0) * (months / 12.0));

                    // SQL-запит для оновлення суми кредиту з врахуванням часткової виплати
                    String updateSql = "UPDATE user_credits SET creditAmount = creditAmount - ? WHERE id = ?";
```

```
            try (PreparedStatement updateStmt = conn.prepareStatement(updateSql)) {
                updateStmt.setInt(1, sum - bonusSumCredit);
                updateStmt.setInt(2, creditId);

                int rowsAffected = updateStmt.executeUpdate();

                if (rowsAffected > 0) {
                    // Повторно виконуємо SQL-запит для отримання оновленої суми кредиту
                    try (ResultSet updatedRs = selectStmt.executeQuery()) {
                        if (updatedRs.next()) {
                            int updatedCreditAmount = updatedRs.getInt("creditAmount");
                            System.out.println("Сума вашого кредиту після внесення суми: " +
updatedCreditAmount);

                            // Якщо сума кредиту стала 0, видаляємо запис
                            if (updatedCreditAmount <= 0) {
                                String deleteSql = "DELETE FROM user_credits WHERE id = ?";
                                try (PreparedStatement deleteStmt = conn.prepareStatement(deleteSql)) {
                                    deleteStmt.setInt(1, creditId);
                                    int deleteRowsAffected = deleteStmt.executeUpdate();
                                    if (deleteRowsAffected > 0) {
                                        System.out.println("Кредит з id " + creditId + " було видалено, оскільки
сума кредиту стала 0.");
                                    } else {
                                        System.out.println("Не вдалося видалити кредит з id " + creditId + ".");
                                    }
                                }
                            }
                        }
                    }
                } else {
                    System.out.println("Кредит з id " + creditId + " не було знайдено в базі даних.");
                }
            } catch (SQLException e) {
                System.out.println("Помилка часткової виплати за кредит: " + e.getMessage());
```

```
                }

            } else {
                System.out.println("Кредит з id " + creditId + " не знайдено в базі даних.");
            }
        }

    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
  }
}
```

Додаток 2. Текст файлів fxml для розмітки інтерфейсу

Файл credit-offers.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.CheckBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.shape.Line?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="500.0" prefWidth="940.0" style="-fx-background-color: #BED7DC;"
xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="main.Controller">
  <children>
```

```
    <AnchorPane layoutX="-1.0" layoutY="-1.0" prefHeight="593.0" prefWidth="162.0" style="-fx-background-color: #DFF5FF; -fx-background-radius: 8px;">
      <children>
        <Label layoutX="51.0" layoutY="118.0" text="Вітаємо">
          <font>
            <Font size="17.0" />
          </font>
        </Label>
        <Label layoutX="67.0" layoutY="144.0" text="Вас">
          <font>
            <Font size="17.0" />
          </font>
        </Label>
        <Line endX="100.0" layoutX="53.0" layoutY="187.0" startX="-42.90000534057617" />
        <ImageView fitHeight="85.0" fitWidth="128.0" layoutX="39.0" layoutY="25.0" pickOnBounds="true" preserveRatio="true">
          <image>
            <Image url="@profile.png" />
          </image>
        </ImageView>
        <Button fx:id="openMyCreditsButton" defaultButton="true" layoutX="18.0" layoutY="263.0" mnemonicParsing="false" prefHeight="30.0" prefWidth="126.0" text="Мої кредити">
          <font>
            <Font name="System Bold" size="14.0" />
          </font>
        </Button>
        <Button fx:id="openCreditButton" defaultButton="true" layoutX="18.0" layoutY="209.0" mnemonicParsing="false" prefHeight="30.0" prefWidth="126.0" text="Взяти кредит">
          <font>
            <Font name="System Bold" size="14.0" />
          </font>
        </Button>
        <Button fx:id="addCreditOfferButton" defaultButton="true" layoutX="18.0" layoutY="316.0" mnemonicParsing="false" prefHeight="30.0" prefWidth="126.0" text="Додати кредит">
          <font>
            <Font name="System Bold" size="14.0" />
```

```
            </font>
          </Button>
          <Label layoutX="23.0" layoutY="555.0" text="Взяти кредит">
            <font>
              <Font name="System Italic" size="16.0" />
            </font>
          </Label>
        </children>
      </AnchorPane>
      <AnchorPane layoutX="168.0" layoutY="6.0" prefHeight="292.0" prefWidth="764.0" style="-fx-background-color: #EEF7FF; -fx-background-radius: 8px;">
        <children>
          <TableView fx:id="table" layoutX="7.0" layoutY="53.0" prefHeight="232.0" prefWidth="750.0">
            <columns>
              <TableColumn fx:id="id" prefWidth="32.0" text="№" />
              <TableColumn fx:id="bank" prefWidth="115.99998474121094" text="Банк" />
              <TableColumn fx:id="loanType" prefWidth="135.20001220703125" text="Вид кредиту" />
              <TableColumn fx:id="percentRate" prefWidth="103.20001220703125" text="Процентна ставка" />
              <TableColumn fx:id="loanTerm" prefWidth="87.20001220703125" text="Термін кредиту (у місяцях)" />
              <TableColumn fx:id="earlyRepayment" prefWidth="114.4000244140625" text="Дострокове погашення" />
              <TableColumn fx:id="creditLineIncrease" prefWidth="107.199951171875" text="Збільшення кредитної лінії" />
              <TableColumn fx:id="creditAmount" prefWidth="167.19989013671875" text="Сума кредиту" />
            </columns>
          </TableView>
          <Label layoutX="256.0" layoutY="14.0" prefHeight="27.0" prefWidth="251.0" text="Наявні кредитні пропозиції!">
            <font>
              <Font name="System Bold Italic" size="17.0" />
            </font>
          </Label>
        </children></AnchorPane>
      <AnchorPane layoutX="168.0" layoutY="302.0" prefHeight="285.0" prefWidth="395.0" style="-fx-background-color: #EEF7FF; -fx-background-radius: 8px;">
```

```
    <children>
      <Label layoutX="42.0" layoutY="14.0" prefHeight="27.0" prefWidth="309.0" text="Пошук кредитів за параметрами:">
        <font>
          <Font name="System Bold Italic" size="17.0" />
        </font>
      </Label>
      <Label layoutX="26.0" layoutY="56.0" prefHeight="30.0" prefWidth="214.0" text="Максимальна процентна ставка:">
        <font>
          <Font size="14.0" />
        </font>
      </Label>
      <TextField fx:id="maxPercentRate" layoutX="246.0" layoutY="59.0" prefHeight="26.0" prefWidth="51.0" />
      <Label layoutX="26.0" layoutY="102.0" prefHeight="30.0" prefWidth="214.0" text="Мінімальний термін кредиту:" textAlignment="CENTER">
        <font>
          <Font size="14.0" />
        </font>
      </Label>
      <TextField fx:id="minLoanTerm" layoutX="223.0" layoutY="105.0" prefHeight="26.0" prefWidth="74.0" />
      <Label layoutX="26.0" layoutY="146.0" prefHeight="30.0" prefWidth="214.0" text="Мінімальна сума кредиту:">
        <font>
          <Font size="14.0" />
        </font>
      </Label>
      <TextField fx:id="minLoanAmount" layoutX="206.0" layoutY="149.0" prefHeight="26.0" prefWidth="91.0" />
      <Label layoutX="26.0" layoutY="186.0" prefHeight="30.0" prefWidth="310.0" text="Можливість дострокового погашення кредиту:">
        <font>
          <Font size="14.0" />
        </font>
      </Label>
```

```
        <CheckBox fx:id="isEarlyRepayment" layoutX="336.0" layoutY="192.0" mnemonicParsing="false" prefHeight="18.0" prefWidth="18.0" />

        <Label layoutX="26.0" layoutY="221.0" prefHeight="30.0" prefWidth="310.0" text="Можливість збільшення кредитної лінії:">

          <font>

            <Font size="14.0" />

          </font>

        </Label>

        <CheckBox fx:id="creditLineIncreaseCheckBox" layoutX="288.0" layoutY="227.0" mnemonicParsing="false" prefHeight="18.0" prefWidth="18.0" />

        <Button fx:id="searchCredits" layoutX="150.0" layoutY="250.0" mnemonicParsing="false" text="Пошук">

          <font>

            <Font size="14.0" />

          </font>

        </Button>

      </children></AnchorPane>

    <AnchorPane layoutX="569.0" layoutY="302.0" prefHeight="286.0" prefWidth="362.0" style="-fx-background-color: #EEF7FF; -fx-background-radius: 8px;">

      <children>

        <Label layoutX="88.0" layoutY="14.0" prefHeight="27.0" prefWidth="186.0" text="Оформити кредит:">

          <font>

            <Font name="System Bold Italic" size="17.0" />

          </font>

        </Label>

        <Label layoutX="46.0" layoutY="54.0" prefHeight="30.0" prefWidth="270.0" text="Вкажіть номер банку для взяття кредиту:">

          <font>

            <Font size="14.0" />

          </font>

        </Label>

        <TextField fx:id="takeCredit" layoutX="109.0" layoutY="93.0" prefHeight="26.0" prefWidth="144.0" />

        <Button fx:id="takeCredits" layoutX="113.0" layoutY="187.0" mnemonicParsing="false" text="Оформити кредит">

          <font>

            <Font size="14.0" />
```

```
            </font>
          </Button>
          <Label layoutX="30.0" layoutY="137.0" prefHeight="30.0" prefWidth="282.0"
text="Ознайомленені з інформацією про кредит:">
            <font>
              <Font size="14.0" />
            </font>
          </Label>
          <CheckBox fx:id="checkTakeCredit" layoutX="315.0" layoutY="143.0" mnemonicParsing="false"
prefHeight="18.0" prefWidth="18.0" />
        </children>
      </AnchorPane>
    </children>
</AnchorPane>
```

Файл user-credits.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.shape.Line?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="500.0" prefWidth="940.0" style="-fx-background-color: #BED7DC;"
xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="main.UserCreditsController">
   <children>
      <AnchorPane layoutX="-1.0" layoutY="-1.0" prefHeight="593.0" prefWidth="162.0" style="-fx-
background-color: #DFF5FF; -fx-background-radius: 8px;">
```

```
        <children>
           <Label layoutX="51.0" layoutY="118.0" text="Вітаємо">
              <font>
                 <Font size="17.0" />
              </font>
           </Label>
           <Label layoutX="67.0" layoutY="144.0" text="Вас">
              <font>
                 <Font size="17.0" />
              </font>
           </Label>
           <Line endX="100.0" layoutX="53.0" layoutY="187.0" startX="-42.90000534057617" />
           <ImageView fitHeight="85.0" fitWidth="128.0" layoutX="39.0" layoutY="25.0"
pickOnBounds="true" preserveRatio="true">
              <image>
                 <Image url="@profile.png" />
              </image>
           </ImageView>
           <Button fx:id="openMyCreditsButton" defaultButton="true" layoutX="18.0" layoutY="260.0"
mnemonicParsing="false" prefHeight="30.0" prefWidth="126.0" text="Мої кредити">
              <font>
                 <Font name="System Bold" size="14.0" />
              </font>
           </Button>
           <Button fx:id="openCreditButton" defaultButton="true" layoutX="19.0" layoutY="205.0"
mnemonicParsing="false" prefHeight="30.0" prefWidth="126.0" text="Взяти кредит">
              <font>
                 <Font name="System Bold" size="14.0" />
              </font>
           </Button>
        <Button fx:id="addCreditOfferButton" defaultButton="true" layoutX="18.0" layoutY="315.0"
mnemonicParsing="false" prefHeight="30.0" prefWidth="126.0" text="Додати кредит">
           <font>
             <Font name="System Bold" size="14.0" />
           </font>
        </Button>
```

```
        <Label layoutX="31.0" layoutY="553.0" prefHeight="26.0" prefWidth="102.0" text="Мої кредити">
          <font>
            <Font name="System Italic" size="16.0" />
          </font>
        </Label>
        </children>
      </AnchorPane>
      <AnchorPane layoutX="168.0" layoutY="6.0" prefHeight="292.0" prefWidth="764.0" style="-fx-background-color: #EEF7FF; -fx-background-radius: 8px;">
        <children>
          <TableView fx:id="table" layoutX="7.0" layoutY="53.0" prefHeight="232.0" prefWidth="750.0">
            <columns>
              <TableColumn fx:id="id" prefWidth="32.0" text="№" />
              <TableColumn fx:id="bank" prefWidth="115.99998474121094" text="Банк" />
              <TableColumn fx:id="loanType" prefWidth="135.20001220703125" text="Вид кредиту" />
              <TableColumn fx:id="percentRate" prefWidth="103.20001220703125" text="Процентна ставка" />
              <TableColumn fx:id="loanTerm" prefWidth="87.20001220703125" text="Термін кредиту (у місяцях)" />
              <TableColumn fx:id="earlyRepayment" prefWidth="114.4000244140625" text="Дострокове погашення" />
              <TableColumn fx:id="creditLineIncrease" prefWidth="107.199951171875" text="Збільшення кредитної лінії" />
              <TableColumn fx:id="creditAmount" prefWidth="167.19989013671875" text="Сума кредиту" />
            </columns>
          </TableView>
        <Label layoutX="315.0" layoutY="14.0" text="Ваші кредити!">
          <font>
            <Font name="System Bold Italic" size="18.0" />
          </font>
        </Label>
        </children></AnchorPane>
      <AnchorPane layoutX="168.0" layoutY="302.0" prefHeight="132.0" prefWidth="395.0" style="-fx-background-color: #EEF7FF; -fx-background-radius: 8px;">
        <children>
```

```
        <Label layoutX="87.0" layoutY="14.0" prefHeight="27.0" prefWidth="250.0" text="Збільшити кредитну лінію:">
          <font>
            <Font name="System Bold Italic" size="17.0" />
          </font>
        </Label>
        <Label layoutX="5.0" layoutY="48.0" prefHeight="30.0" prefWidth="346.0" text="На скільки місяців бажаєте збільшити кредитну лінію:">
          <font>
            <Font size="14.0" />
          </font>
        </Label>
      <TextField fx:id="creditIncreaseLine" layoutX="355.0" layoutY="49.0" prefHeight="26.0" prefWidth="33.0" />
      <Label layoutX="14.0" layoutY="85.0" prefHeight="30.0" prefWidth="162.0" text="Вкажіть номер кредиту:">
        <font>
          <Font size="14.0" />
        </font>
      </Label>
      <TextField fx:id="numberCreditIncreaseLine" layoutX="174.0" layoutY="87.0" prefHeight="26.0" prefWidth="48.0" />
      <Button fx:id="creditIncreaseButton" layoutX="269.0" layoutY="86.0" mnemonicParsing="false" text="Збільшити">
        <font>
          <Font size="14.0" />
        </font>
      </Button>
      </children></AnchorPane>
    <AnchorPane layoutX="569.0" layoutY="302.0" prefHeight="286.0" prefWidth="362.0" style="-fx-background-color: #EEF7FF; -fx-background-radius: 8px;">
      <children>
        <Label layoutX="52.0" layoutY="14.0" prefHeight="27.0" prefWidth="258.0" text="Часткова виплата кредиту:">
          <font>
            <Font name="System Bold Italic" size="17.0" />
          </font>
        </Label>
```

```
            <Label layoutX="24.0" layoutY="55.0" prefHeight="30.0" prefWidth="162.0" text="Вкажіть номер кредиту:">
               <font>
                  <Font size="14.0" />
               </font>
            </Label>
            <TextField fx:id="numberCreditPartRepayment" layoutX="186.0" layoutY="57.0" prefHeight="26.0" prefWidth="154.0" />
            <Button fx:id="partLoanRepaymentButton" layoutX="147.0" layoutY="222.0" mnemonicParsing="false" text="Сплатити">
               <font>
                  <Font size="14.0" />
               </font>
            </Button>
            <Label layoutX="24.0" layoutY="111.0" prefHeight="30.0" prefWidth="267.0" text="Через скільки місяців сплачуєте кредит:">
               <font>
                  <Font size="14.0" />
               </font>
            </Label>
         <TextField fx:id="monthCreditPartRepayment" layoutX="286.0" layoutY="113.0" prefHeight="26.0" prefWidth="55.0" />
         <Label layoutX="24.0" layoutY="166.0" prefHeight="30.0" prefWidth="123.0" text="Сума, яку вносите:">
            <font>
              <Font size="14.0" />
            </font>
         </Label>
         <TextField fx:id="amountCreditPartRepayment" layoutX="151.0" layoutY="168.0" prefHeight="26.0" prefWidth="190.0" />
         </children>
      </AnchorPane>
    <AnchorPane layoutX="168.0" layoutY="438.0" prefHeight="150.0" prefWidth="395.0" style="-fx-background-color: #EEF7FF; -fx-background-radius: 8px;">
       <children>
         <Label layoutX="25.0" layoutY="17.0" prefHeight="27.0" prefWidth="345.0" text="Повне дострокове погашення кредиту:">
            <font>
```

```
            <Font name="System Bold Italic" size="17.0" />
          </font>
        </Label>
        <Label layoutX="67.0" layoutY="54.0" prefHeight="30.0" prefWidth="154.0" text="Вкажіть номер кредиту:">
          <font>
            <Font size="14.0" />
          </font>
        </Label>
        <Button fx:id="creditLineDeleteButton" layoutX="158.0" layoutY="97.0" mnemonicParsing="false" text="Сплатити">
          <font>
            <Font size="14.0" />
          </font>
        </Button>
        <TextField fx:id="numberCreditLineDelete" layoutX="231.0" layoutY="56.0" prefHeight="26.0" prefWidth="75.0" />
      </children>
    </AnchorPane>
  </children>
</AnchorPane>
```

Файл add-credit-offer.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.control.Button?>
<?import javafx.scene.control.CheckBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.shape.Line?>
```

```
<?import javafx.scene.text.Font?>


<AnchorPane prefHeight="500.0" prefWidth="940.0" style="-fx-background-color: #BED7DC;"
xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="main.AddCreditOfferController">

   <children>

      <AnchorPane layoutX="-1.0" layoutY="-1.0" prefHeight="593.0" prefWidth="162.0" style="-fx-
background-color: #DFF5FF; -fx-background-radius: 8px;">

         <children>

            <Label layoutX="51.0" layoutY="118.0" text="Вітаємо">

               <font>

                  <Font size="17.0" />

               </font>

            </Label>

            <Label layoutX="67.0" layoutY="144.0" text="Вас">

               <font>

                  <Font size="17.0" />

               </font>

            </Label>

            <Line endX="100.0" layoutX="53.0" layoutY="187.0" startX="-42.90000534057617" />

            <ImageView fitHeight="85.0" fitWidth="128.0" layoutX="39.0" layoutY="25.0"
pickOnBounds="true" preserveRatio="true">

               <image>

                  <Image url="@profile.png" />

               </image>

            </ImageView>

            <Button fx:id="openMyCreditsButton" defaultButton="true" layoutX="18.0" layoutY="261.0"
mnemonicParsing="false" prefHeight="30.0" prefWidth="126.0" text="Мої кредити">

               <font>

                  <Font name="System Bold" size="14.0" />

               </font>

            </Button>

            <Button fx:id="openCreditButton" defaultButton="true" layoutX="18.0" layoutY="206.0"
mnemonicParsing="false" prefHeight="30.0" prefWidth="126.0" text="Взяти кредит">

               <font>

                  <Font name="System Bold" size="14.0" />

               </font>
```

```
            </Button>
            <Button fx:id="addCreditOfferButton" defaultButton="true" layoutX="18.0" layoutY="315.0"
mnemonicParsing="false" prefHeight="30.0" prefWidth="126.0" text="Додати кредит">
               <font>
                  <Font name="System Bold" size="14.0" />
               </font>
            </Button>
            <Label fx:id="titleLabel" layoutX="21.0" layoutY="554.0" text="Додати кредит">
               <font>
                  <Font name="System Italic" size="16.0" />
               </font>
            </Label>
         </children>
      </AnchorPane>
      <AnchorPane layoutX="168.0" layoutY="6.0" prefHeight="292.0" prefWidth="764.0" style="-fx-
background-color: #EEF7FF; -fx-background-radius: 8px;">
         <children>
            <TableView fx:id="table" layoutX="7.0" layoutY="53.0" prefHeight="232.0"
prefWidth="750.0">
               <columns>
                  <TableColumn fx:id="id" prefWidth="32.0" text="№" />
                  <TableColumn fx:id="bank" prefWidth="115.99998474121094" text="Банк" />
                  <TableColumn fx:id="loanType" prefWidth="135.20001220703125" text="Вид кредиту" />
                  <TableColumn fx:id="percentRate" prefWidth="103.20001220703125" text="Процентна
ставка" />
                  <TableColumn fx:id="loanTerm" prefWidth="87.20001220703125" text="Термін кредиту
(у місяцях)" />
                  <TableColumn fx:id="earlyRepayment" prefWidth="114.4000244140625"
text="Дострокове погашення" />
                  <TableColumn fx:id="creditLineIncrease" prefWidth="107.199951171875"
text="Збільшення кредитної лінії" />
                  <TableColumn fx:id="creditAmount" prefWidth="167.19989013671875" text="Сума
кредиту" />
               </columns>
            </TableView>
            <Label layoutX="256.0" layoutY="14.0" prefHeight="27.0" prefWidth="251.0" text="Наявні
кредитні пропозиції!">
               <font>
```

```
                <Font name="System Bold Italic" size="17.0" />
              </font>
            </Label>
          </children></AnchorPane>
      <AnchorPane layoutX="168.0" layoutY="302.0" prefHeight="232.0" prefWidth="764.0" style="-fx-background-color: #EEF7FF; -fx-background-radius: 8px;">
          <children>
            <Label layoutX="248.0" layoutY="9.0" prefHeight="27.0" prefWidth="276.0" text="Додати кредитну пропозицію:">
              <font>
                <Font name="System Bold Italic" size="17.0" />
              </font>
            </Label>
            <Label layoutX="32.0" layoutY="51.0" prefHeight="30.0" prefWidth="102.0" text="Назва банку:">
              <font>
                <Font size="14.0" />
              </font>
            </Label>
            <TextField fx:id="bankNameTextField" layoutX="122.0" layoutY="54.0" prefHeight="26.0" prefWidth="181.0" />
            <Label layoutX="32.0" layoutY="97.0" prefHeight="30.0" prefWidth="91.0" text="Вид кредиту:" textAlignment="CENTER">
              <font>
                <Font size="14.0" />
              </font>
            </Label>
            <TextField fx:id="loanTypeTextField" layoutX="121.0" layoutY="100.0" prefHeight="26.0" prefWidth="182.0" />
            <Label layoutX="416.0" layoutY="52.0" prefHeight="30.0" prefWidth="128.0" text="Процентна ставка:">
              <font>
                <Font size="14.0" />
              </font>
            </Label>
            <TextField fx:id="percentRateTextField" layoutX="543.0" layoutY="55.0" prefHeight="26.0" prefWidth="182.0" />
```

```
        <Label layoutX="32.0" layoutY="144.0" prefHeight="30.0" prefWidth="310.0"
text="Можливість дострокового погашення кредиту:">
            <font>
               <Font size="14.0" />
            </font>
         </Label>
         <CheckBox fx:id="isEarlyRepaymentCheckBox" layoutX="342.0" layoutY="150.0"
mnemonicParsing="false" prefHeight="18.0" prefWidth="18.0" />
         <Label layoutX="416.0" layoutY="144.0" prefHeight="30.0" prefWidth="310.0"
text="Можливість збільшення кредитної лінії:">
            <font>
               <Font size="14.0" />
            </font>
         </Label>
         <CheckBox fx:id="creditLineIncreaseCheckBox" layoutX="678.0" layoutY="150.0"
mnemonicParsing="false" prefHeight="18.0" prefWidth="18.0" />
         <Button fx:id="addCreditButton" layoutX="465.0" layoutY="190.0" mnemonicParsing="false"
text="Додати кредитну пропозицію">
            <font>
               <Font size="14.0" />
            </font>
         </Button>
         <Label layoutX="416.0" layoutY="97.0" prefHeight="30.0" prefWidth="182.0" text="Термін
кредиту (у місяцях):">
            <font>
               <Font size="14.0" />
            </font>
         </Label>
         <TextField fx:id="loanTermTextField" layoutX="597.0" layoutY="100.0" prefHeight="26.0"
prefWidth="128.0" />
         <Label layoutX="65.0" layoutY="189.0" prefHeight="30.0" prefWidth="91.0" text="Сума
кредиту:" textAlignment="CENTER">
            <font>
               <Font size="14.0" />
            </font>
         </Label>
         <TextField fx:id="creditSumTextField" layoutX="165.0" layoutY="192.0" prefHeight="26.0"
prefWidth="245.0" />
```

```
        </children></AnchorPane>
    <AnchorPane layoutX="168.0" layoutY="537.0" prefHeight="51.0" prefWidth="764.0" style="-fx-background-color: #EEF7FF; -fx-background-radius: 8px;">
      <children>
        <Label layoutX="32.0" layoutY="14.0" prefHeight="27.0" prefWidth="424.0" text="Номер кредитної позиції, яку хочете видалити:">
          <font>
            <Font name="System Bold Italic" size="17.0" />
          </font>
        </Label>
        <Button fx:id="deleteCreditButton" layoutX="611.0" layoutY="13.0" mnemonicParsing="false" text="Видалити">
          <font>
            <Font size="14.0" />
          </font>
        </Button>
        <TextField fx:id="numberCreditToDeleteTextField" layoutX="451.0" layoutY="14.0" prefHeight="26.0" prefWidth="144.0" />
      </children>
    </AnchorPane>
  </children>
</AnchorPane>
```

Додаток 3. Текст всіх файлів для тестування роботи програми

Файл WritingDataToDatabaseTest.java

```
import creditOffer.Credit;
import database.WritingDataToDatabase;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.sql.*;
```

```
import static org.junit.jupiter.api.Assertions.assertNotEquals;

public class WritingDataToDatabaseTest {

    private String destinationTable = "user_credits";
    private Credit firstCredit;

    @BeforeEach
    void setUp() throws SQLException {
        try (Connection conn = DriverManager.getConnection("jdbc:sqlite:d\\JavaFXDemo\\sqlite\\credits.db");
             Statement stmt = conn.createStatement()) {
            // Отримуємо перший запис
            ResultSet rs = stmt.executeQuery("SELECT * FROM " + destinationTable + " LIMIT 1");
            if (rs.next()) {
                firstCredit = new Credit(
                        rs.getInt("id"),
                        rs.getString("bank"),
                        rs.getString("loanType"),
                        rs.getInt("percentRate"),
                        rs.getInt("loanTerm"),
                        rs.getBoolean("earlyRepayment"),
                        rs.getBoolean("creditLineIncrease"),
                        rs.getInt("creditAmount")
                );
                // Видаляємо перший запис
                stmt.executeUpdate("DELETE FROM " + destinationTable + " WHERE id = " +
firstCredit.getId());
            }
        }
    }

    @AfterEach
    void tearDown() throws SQLException {
        // Оновлюємо запис в таблиці після кожного тесту
        if (firstCredit != null) {
```

```
        try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:d\\JavaFXDemo\\sqlite\\credits.db");
             PreparedStatement pstmt = conn.prepareStatement(
                 "INSERT OR REPLACE INTO " + destinationTable + " (id, bank, loanType, percentRate,
loanTerm, earlyRepayment, creditLineIncrease, creditAmount) VALUES (?, ?, ?, ?, ?, ?, ?, ?)")) {
            pstmt.setInt(1, firstCredit.getId());
            pstmt.setString(2, firstCredit.getBank());
            pstmt.setString(3, firstCredit.getLoanType());
            pstmt.setInt(4, firstCredit.getPercentRate());
            pstmt.setInt(5, firstCredit.getLoanTerm());
            pstmt.setBoolean(6, firstCredit.getEarlyRepayment());
            pstmt.setBoolean(7, firstCredit.getCreditLineIncrease());
            pstmt.setInt(8, firstCredit.getCreditAmount());
            pstmt.executeUpdate();
        }
    }
}

@Test
public void testExecute_SuccessfullyInserted() throws SQLException {
    // Arrange
    int numberCredit = 1;
    String sourceTable = "credit_offers";

    // Act
    WritingDataToDatabase writingDataToDatabase = new WritingDataToDatabase(numberCredit,
sourceTable, destinationTable);
    writingDataToDatabase.execute();

    // Assert
    int finalRowCount;
    // Отримання кількості записів у таблиці призначення після вставки
    try (Connection conn =
DriverManager.getConnection("jdbc:sqlite:d:\\JavaFXDemo\\sqlite\\credits.db");
         Statement stmt = conn.createStatement();
         ResultSet resultSet = stmt.executeQuery("SELECT COUNT(*) FROM " + destinationTable)) {
```

```
            finalRowCount = resultSet.getInt(1);
        }

        // Assert
        assertNotEquals(0, finalRowCount);
    }
}
```

Файл ReadingDataFromDatabaseTest.java

```
import creditOffer.CreditOffer;
import database.ReadingDataFromDatabase;
import org.junit.jupiter.api.Test;

import java.sql.SQLException;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class ReadingDataFromDatabaseTest {

    @Test
    public void testExecute_ReturnsCorrectListOfCreditOffers() throws SQLException {
        // Arrange
        String databaseUrl = "jdbc:sqlite:d:\\JavaFXDemo\\sqlite\\credits.db";
        String tableName = "credit_offers";
        // Виклик методу execute() для отримання списку кредитних пропозицій
        ReadingDataFromDatabase readingDataFromDatabase = new ReadingDataFromDatabase(databaseUrl,
tableName);
        List<CreditOffer> actualCreditOffers = readingDataFromDatabase.execute();
        // Assert
        // Перевірка, чи список не є пустим
        assertEquals(false, actualCreditOffers.isEmpty());
    }
}
```

Файл DeleteUserCreditTest.java

```
import org.junit.jupiter.api.Test;
import userCredits.DeleteUserCredit;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import static org.junit.jupiter.api.Assertions.*;

class DeleteUserCreditTest {

    @Test
    void testExecute_SuccessfullyDeleted() throws SQLException {
        // Arrange
        int creditId = 123;
        String databaseUrl = "jdbc:sqlite:d\\JavaFXDemo\\sqlite\\credits.db";
        int initialRowCount;
        int finalRowCount;

        // Отримання початкової кількості записів у базі даних
        try (Connection conn = DriverManager.getConnection(databaseUrl);
             Statement stmt = conn.createStatement();
             ResultSet resultSet = stmt.executeQuery("SELECT COUNT(*) FROM user_credits")) {
            initialRowCount = resultSet.getInt(1);
        }

        // Act
        DeleteUserCredit deleteUserCredit = new DeleteUserCredit(creditId, databaseUrl);
        deleteUserCredit.execute();

        // Отримання кількості записів у базі даних після видалення
```

```
        try (Connection conn = DriverManager.getConnection(databaseUrl);
             Statement stmt = conn.createStatement();
             ResultSet resultSet = stmt.executeQuery("SELECT COUNT(*) FROM user_credits")) {
            finalRowCount = resultSet.getInt(1);
        }

        // Assert
        assertEquals(initialRowCount, finalRowCount);
    }
}
```

Файл IncreaseCreditLineTest.java

```
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import userCredits.IncreaseCreditLine;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import static org.junit.jupiter.api.Assertions.assertTrue;

public class IncreaseCreditLineTest {
    private final String databaseUrl = "jdbc:sqlite:d\\JavaFXDemo\\sqlite\\credits.db";
    private final String tableName = "user_credits";
    private int initialLoanTerm;

    @BeforeEach
    void setUp() throws SQLException {
```

```
        try (Connection conn = DriverManager.getConnection(databaseUrl);
             Statement stmt = conn.createStatement()) {
            // Створюємо тестовий запис
            stmt.executeUpdate("INSERT INTO " + tableName + " (id, loanTerm, creditLineIncrease) VALUES (123, 12, 1)");

            // Зчитуємо поточне значення loanTerm
            ResultSet rs = stmt.executeQuery("SELECT loanTerm FROM " + tableName + " WHERE id = 123");
            if (rs.next()) {
                initialLoanTerm = rs.getInt("loanTerm");
            }
        }
    }

    @AfterEach
    void tearDown() throws SQLException {
        try (Connection conn = DriverManager.getConnection(databaseUrl);
             Statement stmt = conn.createStatement()) {
            // Видаляємо тестовий запис
            stmt.executeUpdate("DELETE FROM " + tableName + " WHERE id = 123");
        }
    }

    @Test
    public void testExecute_SuccessfullyIncreased() throws SQLException {
        // Arrange
        int creditId = 123;
        int monthsToIncrease = 6;

        // Act
        IncreaseCreditLine increaseCreditLine = new IncreaseCreditLine(creditId, monthsToIncrease, databaseUrl);
        increaseCreditLine.execute();

        // Assert
```

```
        int updatedLoanTerm;
        try (Connection conn = DriverManager.getConnection(databaseUrl);
             PreparedStatement pstmt = conn.prepareStatement("SELECT loanTerm FROM " + tableName + " WHERE id = ?")) {
            pstmt.setInt(1, creditId);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                updatedLoanTerm = rs.getInt("loanTerm");
                assertTrue(updatedLoanTerm > initialLoanTerm, "Loan term should have been increased");
            }
        }
    }
}
```

Файл PartLoanRepaymentTest.java

```
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import userCredits.PartLoanRepayment;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import static org.junit.jupiter.api.Assertions.assertTrue;

public class PartLoanRepaymentTest {
    private final String databaseUrl = "jdbc:sqlite:d:\\JavaFXDemo\\sqlite\\credits.db";
    private final String tableName = "user_credits";
    private int initialCreditAmount;
```

```
    @BeforeEach
    void setUp() throws SQLException {
        try (Connection conn = DriverManager.getConnection(databaseUrl);
             Statement stmt = conn.createStatement()) {
            // Видаляємо запис з id = 1, якщо він існує, щоб уникнути конфлікту
            stmt.executeUpdate("DELETE FROM " + tableName + " WHERE id = 1");

            // Створюємо тестовий запис
            stmt.executeUpdate("INSERT INTO " + tableName + " (id, creditAmount, percentRate) VALUES
(1, 1000, 10)");

            // Зчитуємо поточне значення creditAmount
            ResultSet rs = stmt.executeQuery("SELECT creditAmount FROM " + tableName + " WHERE id =
1");
            if (rs.next()) {
                initialCreditAmount = rs.getInt("creditAmount");
            }
        }
    }

    @AfterEach
    void tearDown() throws SQLException {
        try (Connection conn = DriverManager.getConnection(databaseUrl);
             Statement stmt = conn.createStatement()) {
            // Видаляємо тестовий запис
            stmt.executeUpdate("DELETE FROM " + tableName + " WHERE id = 1");
        }
    }

    @Test
    public void testExecute_SuccessfullyReducedCreditAmount() throws SQLException {
        // Arrange
        int creditId = 1;
        int months = 6;
```

```
        int sum = 200;


        // Act
        PartLoanRepayment partLoanRepayment = new PartLoanRepayment(creditId, months, sum,
databaseUrl);
        partLoanRepayment.execute();


        // Assert
        int updatedCreditAmount;
        try (Connection conn = DriverManager.getConnection(databaseUrl);
             PreparedStatement pstmt = conn.prepareStatement("SELECT creditAmount FROM " + tableName +
" WHERE id = ?")) {
            pstmt.setInt(1, creditId);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                updatedCreditAmount = rs.getInt("creditAmount");
                assertTrue(updatedCreditAmount < initialCreditAmount, "Сума кредиту повинна була
зменшитися");
            }
        }
    }
}
```

Файл SearchByMinLoanTermTest.java

```
import creditOffer.CreditOffer;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import searchCredits.SearchByMinLoanTerm;


import java.util.ArrayList;
import java.util.List;


import static org.junit.jupiter.api.Assertions.assertEquals;


public class SearchByMinLoanTermTest {
```

```
    private List<CreditOffer> creditOffers;
    private SearchByMinLoanTerm searchByMinLoanTerm;

    @BeforeEach
    public void setUp() {
        creditOffers = new ArrayList<>();
        searchByMinLoanTerm = new SearchByMinLoanTerm(creditOffers, 40);
    }

    @Test
    public void testExecute_ReturnsFilteredOffers() {
        List<CreditOffer> creditOfferList = new ArrayList<>();
        creditOfferList.add(new CreditOffer(1, "Універсал Банк", "Автокредит", 11, 36, true, true, 400000));
        creditOfferList.add(new CreditOffer(2,"Фінансова Група", "Житловий кредит", 13, 48, false, true,
3000000));
        creditOfferList.add(new CreditOffer(3,"Економія Банк", "Освітній кредит", 7, 60, true, false,
100000));
        creditOfferList.add(new CreditOffer(4,"Фінанс Брокер", "Бізнес-кредит", 8, 24, true, false,
2000000));

        creditOffers.addAll(creditOfferList);

        List<CreditOffer> expectedFilteredOffers = new ArrayList<>();
        expectedFilteredOffers.add(creditOfferList.get(1));
        expectedFilteredOffers.add(creditOfferList.get(2));

        List<CreditOffer> filteredOffers = searchByMinLoanTerm.execute();

        assertEquals(expectedFilteredOffers.size(), filteredOffers.size());
        assertEquals(expectedFilteredOffers, filteredOffers);
    }
}
```

Файл SearchByMinLoanAmountTest.java

```
import creditOffer.CreditOffer;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import searchCredits.SearchByMinLoanAmount;

import java.util.ArrayList;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class SearchByMinLoanAmountTest {
    private List<CreditOffer> creditOffers;
    protected SearchByMinLoanAmount searchByMinLoanAmount;

    @BeforeEach
    public void setUp() {
        creditOffers = new ArrayList<>();
        searchByMinLoanAmount = new SearchByMinLoanAmount(creditOffers, 1000000);
    }

    @Test
    public void testExecute_ReturnsFilteredOffers() {
        List<CreditOffer> creditOfferList = new ArrayList<>();
        creditOfferList.add(new CreditOffer(1, "Універсал Банк", "Автокредит", 11, 36, true, true, 400000));
        creditOfferList.add(new CreditOffer(2, "Фінансова Група", "Житловий кредит", 13, 48, false, true,
3000000));
        creditOfferList.add(new CreditOffer(3,"Економія Банк", "Освітній кредит", 7, 60, true, false,
100000));
        creditOfferList.add(new CreditOffer(4,"Фінанс Брокер", "Бізнес-кредит", 8, 24, true, false,
2000000));

        creditOffers.addAll(creditOfferList);

        List<CreditOffer> expectedFilteredOffers = new ArrayList<>();
        expectedFilteredOffers.add(creditOfferList.get(1));
        expectedFilteredOffers.add(creditOfferList.get(3));
```

```
        List<CreditOffer> filteredOffers = searchByMinLoanAmount.execute();

        assertEquals(expectedFilteredOffers.size(), filteredOffers.size());
        assertEquals(expectedFilteredOffers, filteredOffers);
    }
}
```

Файл SearchByMaxInterestRateTest.java

```
import creditOffer.CreditOffer;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import searchCredits.SearchByMaxInterestRate;

import java.util.ArrayList;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class SearchByMaxInterestRateTest {

    private List<CreditOffer> creditOffers;
    private SearchByMaxInterestRate searchByMaxInterestRate;

    @BeforeEach
    public void setUp() {
        creditOffers = new ArrayList<>();
        searchByMaxInterestRate = new SearchByMaxInterestRate(creditOffers, 10);
    }

    @Test
    public void testExecute_ReturnsFilteredOffers() {
        List<CreditOffer> creditOfferList = new ArrayList<>();
```

```
        creditOfferList.add(new CreditOffer(1,"Універсал Банк", "Автокредит", 11, 36, true, true, 400000));
        creditOfferList.add(new CreditOffer(2,"Фінансова Група", "Житловий кредит", 13, 48, false, true,
3000000));
        creditOfferList.add(new CreditOffer(3,"Економія Банк", "Освітній кредит", 7, 60, true, false,
100000));
        creditOfferList.add(new CreditOffer(4,"Фінанс Брокер", "Бізнес-кредит", 8, 24, true, false,
2000000));
        creditOfferList.add(new CreditOffer(5,"Прогрес Банк", "Бізнес-кредит", 6, 36, false, true, 5000000));


        creditOffers.addAll(creditOfferList);


        List<CreditOffer> expectedFilteredOffers = new ArrayList<>();
        expectedFilteredOffers.add(creditOfferList.get(2));
        expectedFilteredOffers.add(creditOfferList.get(3));
        expectedFilteredOffers.add(creditOfferList.get(4));


        List<CreditOffer> filteredOffers = searchByMaxInterestRate.execute();


        assertEquals(expectedFilteredOffers.size(), filteredOffers.size());
        assertEquals(expectedFilteredOffers, filteredOffers);
    }
}
```

Файл SearchByIsEarlyRepaymentEnabledTest.java

```
import creditOffer.CreditOffer;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import searchCredits.SearchByIsEarlyRepaymentEnabled;


import java.util.ArrayList;
import java.util.List;


import static org.junit.jupiter.api.Assertions.assertEquals;
```

```
public class SearchByIsEarlyRepaymentEnabledTest {
    private List<CreditOffer> creditOffers;
    private SearchByIsEarlyRepaymentEnabled searchByIsEarlyRepaymentEnabled;

    @BeforeEach
    public void setUp() {
        creditOffers = new ArrayList<>();
        searchByIsEarlyRepaymentEnabled = new SearchByIsEarlyRepaymentEnabled(creditOffers);
    }

    @Test
    public void testExecute_ReturnsFilteredOffers() {
        List<CreditOffer> creditOfferList = new ArrayList<>();
        creditOfferList.add(new CreditOffer(1,"Універсал Банк", "Автокредит", 11, 36, true, true, 400000));
        creditOfferList.add(new CreditOffer(2,"Фінансова Група", "Житловий кредит", 13, 48, false, true,
3000000));
        creditOfferList.add(new CreditOffer(3,"Економія Банк", "Освітній кредит", 7, 60, true, false,
100000));
        creditOfferList.add(new CreditOffer(4,"Фінанс Брокер", "Бізнес-кредит", 8, 24, true, false,
2000000));
        creditOfferList.add(new CreditOffer(5,"Прогрес Банк", "Бізнес-кредит", 6, 36, false, true, 5000000));

        creditOffers.addAll(creditOfferList);

        List<CreditOffer> expectedFilteredOffers = new ArrayList<>();
        expectedFilteredOffers.add(creditOfferList.get(0));
        expectedFilteredOffers.add(creditOfferList.get(2));
        expectedFilteredOffers.add(creditOfferList.get(3));

        List<CreditOffer> filteredOffers = searchByIsEarlyRepaymentEnabled.execute();

        assertEquals(expectedFilteredOffers.size(), filteredOffers.size());
        assertEquals(expectedFilteredOffers, filteredOffers);
    }
}
```

Файл SearchByIsCreditLineIncreaseEnabledTest.java

```
import creditOffer.CreditOffer;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import searchCredits.SearchByIsCreditLineIncreaseEnabled;

import java.util.ArrayList;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class SearchByIsCreditLineIncreaseEnabledTest {

    private List<CreditOffer> creditOffers;
    private SearchByIsCreditLineIncreaseEnabled searchByIsCreditLineIncreaseEnabled;

    @BeforeEach
    public void setUp() {
        creditOffers = new ArrayList<>();
        searchByIsCreditLineIncreaseEnabled = new SearchByIsCreditLineIncreaseEnabled(creditOffers);
    }

    @Test
    public void testExecute_ReturnsFilteredOffers() {
        List<CreditOffer> creditOfferList = new ArrayList<>();
        creditOfferList.add(new CreditOffer(1,"Універсал Банк", "Автокредит", 11, 36, true, true, 400000));
        creditOfferList.add(new CreditOffer(2,"Фінансова Група", "Житловий кредит", 13, 48, false, true,
3000000));
        creditOfferList.add(new CreditOffer(3,"Економія Банк", "Освітній кредит", 7, 60, true, false,
100000));
        creditOfferList.add(new CreditOffer(4,"Фінанс Брокер", "Бізнес-кредит", 8, 24, true, false,
2000000));
        creditOfferList.add(new CreditOffer(5,"Прогрес Банк", "Бізнес-кредит", 6, 36, false, true, 5000000));

        creditOffers.addAll(creditOfferList);
```

```
        List<CreditOffer> expectedFilteredOffers = new ArrayList<>();
        expectedFilteredOffers.add(creditOfferList.get(0));
        expectedFilteredOffers.add(creditOfferList.get(1));
        expectedFilteredOffers.add(creditOfferList.get(4));

        List<CreditOffer> filteredOffers = searchByIsCreditLineIncreaseEnabled.execute();

        assertEquals(expectedFilteredOffers.size(), filteredOffers.size());
        assertEquals(expectedFilteredOffers, filteredOffers);
    }
}
```

Файл ControllerTest.java

```
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import org.junit.jupiter.api.Test;
import org.testfx.framework.junit5.ApplicationTest;
import org.testfx.matcher.control.TableViewMatchers;
import org.testfx.matcher.control.LabeledMatchers;
import org.testfx.matcher.control.TextInputControlMatchers;

import static org.testfx.api.FxAssert.verifyThat;
import static org.testfx.matcher.control.LabeledMatchers.hasText;

public class ControllerTest extends ApplicationTest {

    @Override
    public void start(Stage stage) throws Exception {
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/main/credit-offers.fxml"));
        Parent root = loader.load();
```

```
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
}

@Test
public void testOpenCreditButton() {
    clickOn("#openCreditButton");
    // Перевірка, що вікно "Credit Offers" відкрито
    verifyThat("#titleLabel", hasText("Взяти кредит"));
}

@Test
public void testOpenMyCreditsButton() {
    clickOn("#openMyCreditsButton");
    verifyThat("#titleLabel", hasText("Мої кредити"));
}

@Test
public void testAddCreditOfferButton() {
    clickOn("#addCreditOfferButton");
    verifyThat("#titleLabel", hasText("Додати кредит"));
}

@Test
public void testSearchByCreditLineIncrease() {
    clickOn("#creditLineIncreaseCheckBox");
    clickOn("#searchCredits");
    // Перевірка, що таблиця містить кредити зі збільшенням кредитного ліміту
    verifyThat("#table", TableViewMatchers.hasNumRows(5));
}

@Test
```

```
    public void testSearchByEarlyRepayment() {
        clickOn("#isEarlyRepayment");
        clickOn("#searchCredits");
        verifyThat("#table", TableViewMatchers.hasNumRows(7));
    }

    @Test
    public void testSearchByMaxPercentRate() {
        clickOn("#maxPercentRate").write("8");
        clickOn("#searchCredits");
        verifyThat("#table", TableViewMatchers.hasNumRows(6));
    }

    @Test
    public void testSearchByMinLoanTerm() {
        clickOn("#minLoanTerm").write("50");
        clickOn("#searchCredits");
        // Перевірка, що таблиця містить кредити з мінімальним терміном кредиту 50
        verifyThat("#table", TableViewMatchers.hasNumRows(2));
    }

    @Test
    public void testSearchByMinLoanAmount() {
        clickOn("#minLoanAmount").write("2000000");
        clickOn("#searchCredits");
        verifyThat("#table", TableViewMatchers.hasNumRows(5));
    }
}
```

Файл UserCreditsControllerTest.java

```
import creditOffer.Credit;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
```

```
import javafx.scene.Scene;
import javafx.scene.control.TableView;
import javafx.stage.Stage;
import org.junit.jupiter.api.Test;
import org.testfx.framework.junit5.ApplicationTest;

import static org.junit.jupiter.api.Assertions.assertTrue;
import static org.testfx.matcher.control.LabeledMatchers.hasText;
import org.testfx.matcher.control.TextInputControlMatchers;

import static org.testfx.api.FxAssert.verifyThat;
import static org.testfx.util.WaitForAsyncUtils.waitForFxEvents;

public class UserCreditsControllerTest extends ApplicationTest {

    @Override
    public void start(Stage stage) throws Exception {
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/main/user-credits.fxml"));
        Parent root = loader.load();
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }

    @Test
    public void testOpenCreditButton() {
        clickOn("#openCreditButton");
        verifyThat("#titleLabel", hasText("Взяти кредит"));
    }

    @Test
    public void testOpenMyCreditsButton() {
        clickOn("#openMyCreditsButton");
        verifyThat("#titleLabel", hasText("Мої кредити"));
```

```
}

@Test
public void testAddCreditOfferButton() {
    clickOn("#addCreditOfferButton");
    verifyThat("#titleLabel", hasText("Додати кредит"));
}

@Test
public void testCreditIncreaseButton() {
    TableView<Credit> tableView = lookup("#table").query();
    int initialCount = tableView.getItems().size();

    // Зберігаємо початкове значення кредитної лінії
    int initialCreditAmount = tableView.getItems().stream()
            .filter(credit -> credit.getId() == 3)
            .mapToInt(Credit::getLoanTerm)
            .findFirst().orElse(0);

    clickOn("#numberCreditIncreaseLine").write("3");
    clickOn("#creditIncreaseLine").write("2");

    clickOn("#creditIncreaseButton");

    waitForFxEvents();

    assertTrue(isCreditLineIncreased(initialCreditAmount, tableView));
}

private boolean isCreditLineIncreased(int initialCreditAmount, TableView<Credit> tableView) {
    int newCreditAmount = tableView.getItems().stream()
            .filter(credit -> credit.getId() == 3)
            .mapToInt(Credit::getCreditAmount)
            .findFirst().orElse(0);
```

```
        return newCreditAmount > initialCreditAmount;
    }
}
```

Файл AddCreditOfferControllerTest.java

```
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import org.junit.jupiter.api.Test;
import org.testfx.framework.junit5.ApplicationTest;

import static org.testfx.matcher.control.LabeledMatchers.hasText;

import static org.testfx.api.FxAssert.verifyThat;

public class AddCreditOfferControllerTest extends ApplicationTest {

    @Override
    public void start(Stage stage) throws Exception {
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/main/add-credit-offer.fxml"));
        Parent root = loader.load();
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }

    @Test
    public void testOpenCreditButton() {
        clickOn("#openCreditButton");
        verifyThat("#titleLabel", hasText("Взяти кредит"));
    }
```

```
    @Test
    public void testOpenMyCreditsButton() {
        clickOn("#openMyCreditsButton");
        verifyThat("#titleLabel", hasText("Мої кредити"));
    }

    @Test
    public void testAddCreditOfferButton() {
        clickOn("#addCreditOfferButton");
        verifyThat("#titleLabel", hasText("Додати кредит"));
    }
}
```