Київський національний університет імені Тараса Шевченка

Механіко-математичний факультет

Реферат на тему:

**«Bank account manager»**

Виконав:

студент 3 курсу, 1-ша групи комп’ютерної математики

Чайчено Олег

Київ 2023

**ЗМІСТ**

Вступ

Завдання

Висновки

**Вступ**

Банківська система - це складна інфраструктура, яка взаємодіє з фінансовими ресурсами та забезпечує найважливіші фінансові послуги клієнтам. У цьому проекті ми розглядаємо створення банківського облікового менеджера, який базується на об'єктно-орієнтованому підході та використовує абстракції для реалізації різних видів рахунків.

Клас "Account":

Першим кроком є створення абстрактного класу "Account", який буде базовим класом для конкретних видів рахунків, таких як "CheckingAccount", "SavingsAccount" та "BusinessAccount". Цей клас визначає основні властивості кожного рахунку, такі як номер рахунку, баланс, та кредитний ліміт.

**Завдання**

**Account**

Поля класу:

private String accountNumber: Зберігає номер рахунку.

private double balance: Зберігає поточний баланс рахунку.

public List<Transaction> transactionHistory: Список об'єктів класу Transaction, який використовується для зберігання історії транзакцій рахунку.

private static List<Account> allAccounts: Статичний список, що зберігає всі об'єкти класу Account. Використовується, наприклад, для генерації зведеної виписки для всіх рахунків.

Конструктор:

public Account(String accountNumber, double balance): Ініціалізує об'єкт класу Account з заданим номером рахунку та початковим балансом. Також ініціалізує список транзакцій та додає створений об'єкт до списку всіх рахунків.

Методи роботи з балансом та транзакціями:

public double getBalance(): Повертає поточний баланс рахунку.

public void setBalance(double balance): Встановлює новий баланс рахунку.

public void deposit(double amount): Здійснює поповнення рахунку заданою сумою, записує транзакцію у історію.

public void makePayment(Account toAccount, double amount): Здійснює оплату з поточного рахунку на рахунок toAccount, записує транзакцію у історію.

public abstract void withdraw(double amount): Абстрактний метод для зняття грошей з рахунку. Реалізується в конкретних підкласах.

public void printStatement(): Виводить виписку по рахунку у консоль.

public abstract void generateStatement(): Абстрактний метод для генерації виписки. Реалізується в конкретних підкласах.

public void recordTransaction(String description, double amount): Записує нову транзакцію у історію.

Внутрішній клас Transaction:

Використовується для представлення окремої транзакції. Містить опис та суму транзакції.

Метод generateStatement1():

Генерує виписку у форматі PDF за допомогою бібліотеки iTextPDF. Включає інформацію про рахунок, баланс та історію транзакцій.

Статичний метод generateStatement2(String fileName):

Статичний метод для генерації виписки в форматі Excel (XLS) за допомогою бібліотеки jExcelApi. Створює електронну таблицю з інформацією про всі рахунки та їх транзакції.

Загальною метою цього коду є створення гнучкої системи обліку та взаємодії з банківськими рахунками. Клас Account служить базовим класом для реалізації конкретних видів рахунків та включає функціонал для здійснення транзакцій, генерації виписок та ведення історії операцій.

**BusinessAccount**

Поле класу:

private String businessName: Зберігає ім'я бізнесу для бізнес-рахунку.

Конструктор:

public BusinessAccount(String accountNumber, double balance, String businessName): Ініціалізує об'єкт класу BusinessAccount за допомогою конструктора базового класу Account та додає нове поле businessName.

Метод withdraw:

Перевизначає абстрактний метод withdraw базового класу. Дозволяє здійснювати зняття грошей з бізнес-рахунку, якщо сума зняття не від'ємна та не перевищує поточний баланс. Записує транзакцію у історію та виводить повідомлення про зняття.

Метод generateStatement:

Перевизначає абстрактний метод generateStatement базового класу. Виводить в консоль виписку для бізнес-рахунку, включаючи номер рахунку та ім'я бізнесу.

Цей клас додає специфічність для бізнес-рахунків, реалізуючи методи, що адаптовані до їх особливостей. Крім того, він може використовуватися для створення об'єктів бізнес-рахунків та взаємодії з ними у межах загальної банківської системи.

**CheckingAccount**

Поле класу:

private double overdraftLimit: Зберігає ліміт овердрафту (overdraft limit) для чекового рахунку.

Конструктор:

public CheckingAccount(String accountNumber, double balance, double overdraftLimit): Ініціалізує об'єкт класу CheckingAccount за допомогою конструктора базового класу Account та додає нове поле overdraftLimit.

Метод withdraw:

Перевизначає абстрактний метод withdraw базового класу. Дозволяє здійснювати зняття грошей з чекового рахунку, якщо сума зняття не від'ємна та не перевищує (баланс + ліміт овердрафту). Записує транзакцію у історію та виводить повідомлення про зняття.

Метод generateStatement:

Перевизначає абстрактний метод generateStatement базового класу. Виводить в консоль виписку для чекового рахунку, включаючи номер рахунку, історію транзакцій та поточний баланс.

Цей клас реалізує специфічний функціонал для чекового рахунку, зокрема, враховує ліміт овердрафту при здійсненні зняття грошей. Він може використовуватися для створення об'єктів чекових рахунків та взаємодії з ними у межах загальної банківської системи.

**SavingsAccount**

Поле класу:

private double interestRate: Зберігає процентну ставку для рахунку збережень.

Конструктор:

public SavingsAccount(String accountNumber, double balance, double interestRate): Ініціалізує об'єкт класу SavingsAccount за допомогою конструктора базового класу Account та додає нове поле interestRate.

Метод withdraw:

Перевизначає абстрактний метод withdraw базового класу. Дозволяє здійснювати зняття грошей з рахунку збережень, якщо сума зняття не від'ємна та не перевищує поточний баланс. Записує транзакцію у історію та виводить повідомлення про зняття.

Метод generateStatement:

Перевизначає абстрактний метод generateStatement базового класу. Виводить в консоль виписку для рахунку збережень, включаючи номер рахунку, історію транзакцій та поточний баланс.

Цей клас додає специфічний функціонал для рахунку збережень та може використовуватися для створення об'єктів рахунків збережень у межах загальної банківської системи. Однак, зауважте, що методи withdraw та generateStatement в даному коді тотожні для класів SavingsAccount та CheckingAccount. В реальному проекті ці методи можуть бути реалізовані по-різному, якщо класи відрізняються в їх функціональності.

**Main**

Створення рахунків:

Створюються об'єкти класів CheckingAccount, SavingsAccount та BusinessAccount з відповідними параметрами, такими як номер рахунку, початковий баланс та специфічні властивості (ліміт овердрафту для чекового рахунку, процентна ставка для рахунку збережень, ім'я бізнесу для бізнес-рахунку).

CheckingAccount checkingAccount = new CheckingAccount("CA001", 1000.0, 500.0);

SavingsAccount savingsAccount = new SavingsAccount("SA001", 2000.0, 0.03);

BusinessAccount businessAccount = new BusinessAccount("BA001", 1500.0, "My Business");

Здійснення операцій:

Викликаються методи рахунків для здійснення операцій, таких як депозит, зняття, здійснення платежу між рахунками.

checkingAccount.deposit(500.0);

savingsAccount.withdraw(300.0);

checkingAccount.makePayment(savingsAccount, 200.0);

Вивід виписок та генерація документів:

Викликаються методи генерації виписок та інших документів для кожного рахунку.

checkingAccount.generateStatement();

savingsAccount.generateStatement();

businessAccount.generateStatement();

checkingAccount.generateStatement1();

savingsAccount.generateStatement1();

businessAccount.generateStatement1();

Account.generateStatement2("all\_accounts\_statements.xls");

Цей клас Main відображає взаємодію з різними типами рахунків та демонструє функціонал вашої банківської системи. Виведення в консоль та генерація документів служать для тестування та демонстрації роботи системи.

Виписки я формую у ПДФ, а як базу данних використовую ексель

**Висновки**

У цьому проекті була розроблена банківська система, яка включає в себе абстрактний клас Account та його конкретні підкласи: CheckingAccount, SavingsAccount та BusinessAccount. Кожен рахунок має свої унікальні властивості та функціонал, що відображає різні типи банківських рахунків.

Абстрактний клас Account:

Включає базовий функціонал для всіх видів рахунків, такий як зберігання номера рахунку, балансу та історії транзакцій.

Має методи для здійснення депозитів, зняття грошей, здійснення платежів та генерації виписок.

Підклас CheckingAccount:

Додає можливість встановлення ліміту овердрафту для забезпечення можливості витрат, перевищуючи наявний баланс.

Підклас SavingsAccount:

Додає процентну ставку для збережень.

Підклас BusinessAccount:

Включає ім'я бізнесу для ідентифікації бізнес-рахунку.

Висновок:

Банківська система дозволяє керувати різними типами рахунків з урахуванням їхніх специфічних особливостей.

Кожен рахунок може виконувати операції, такі як депозити, зняття та платежі, а також генерувати виписки в різних форматах (консоль, PDF, Excel).

Система зберігає інформацію відносно всіх створених рахунків та їх транзакцій для подальшого аналізу та відстеження.