Приложение: **“Упрощенная система управления банковскими данными”**

Интерфейс представляет собой консольное окно. При запуске приложения консольное окно предлагает ввести логин и пароль.

Единовременно может быть открыто несколько окон приложения с разными активными пользователями в них.

Виды пользователей перечислены позже, но в общем случае будут доступны работники банка и клиенты банка.

Разным видом пользователей будут доступны разные наборы команд. Приложение будет хранить свои данные в файлах на диске.

**Содержание**

1. Перечень сущностей

2. Описание архитектуры

3. Описание пользовательского интерфейса

4. Сценарии взаимодействия с пользовательским интерфейсом

5. Доп. требования

6. Задачи для студентов

**1. Перечень сущностей**

- Bank

- Client

- BankAccount

- Loan

- LoanDepartment

- ClientDepartment

- BankAccountDepartment

**2. Описание архитектуры**

Сущность Bank содержит по одному объекту сущностей LoanDepartment, ClientDepartment, BankAccountDepartment, константное поле RefinancingRate = 10, метод bool CheckLoanAvailability(), Refill(), CashWithdrawal().

Сущность Client содержит 10 различных полей, которые в совокупности будут характеризовать финансовое положение клиента и его кредитоспособность (Средняя з/п, наличие недвижимости, наличие смертельных заболеваний и т.д.), а также List<int> loan, в котором указаны Id полученных кредитов, и List<int> bankAccount, с указанием в нём Id счетов клиента.

Сущность Loan содержит в себе поля LoanSum, PaidAmount, характеризующие сумму кредита и оплаченную часть соответственно, ClientId, в котором указывается Id клиента получившего кредит.

Сущность BankAccount включает в себя поля Balance, LoanBalance, ClientId, баланс счёта и Id клиента соответственно.

Сущности Client, BankAccount, Loan кроме специфичных для них полей будут иметь поле Id, идентичные значения Id исключены.

Каждая из сущностей LoanDepartment ( ClientDepartment, BankAccountDepartment) содержит в себе Dictionary<int, Loan> loanBase (Dictionary <int, Client> clientBase, Dictionary<string, BankAccount> bankAccountBase), в качестве ключа будет использоваться Id соответствующих сущностей.

Сущность LoanDepartment включает в себя методы CheckLoanBalance(), CheckLoanAvailability(), GetLoan().

Сущность ClientDepartment включает в себя методы RegisterClient(), CheckLoanHistory().

Сущность BankAccountDepartment включает в себя методы CreateBankAccount(), DeleteBankAccount(), Refill(), CashWithdrawal().

**3. Описание пользовательского интерфейса**

Интерфейс представляет собой консольное окно, при запуске которого у пользователя запрашивается логин и пароль или предлагается создать нового пользователя. Далее пользователю предоставляется список возможных для него операций,

Возможные операции соответствуют методам перечисленных сущностей (Оформление кредита, оплата, регистрация счёта и т.д.).

Работа с пользователем осуществляется при помощи считывания введённых им команд.

**4. Сценарии взаимодействия с пользовательским интерфейсом**

LoanDepartment:

- CheckLoanBalance(), позволяет просмотреть остаток по кредиту клиента.

- CheckLoanAvailability(), проверяет возможность клиента на получение кредита (учитывает среднюю з/п, наличие движимого и недвижимого имущества, ..., всего по 10 признакам

(Оставшиеся продумать самостоятельно))

- GetLoan() вызывает метод CheckLoanAvailability() и здесь же вызов метода Bank.CheckLoanAvailability(), если кредит доступен, то выдать клиенту под Bank.RefinancingRate % в год. Занесение кредита

в словарь, с указанием Id клиента, клиенту записывается Id данного кредита.

ClientDepartment:

- RegisterClient(), регистрация нового клиента и занесение его в базу.

- CheckLoanHistory(), вывод списка всех кредитов клиента и остаток по каждому

BankAccountDepartment:

- CreateBankAccount(), создание нового счёта, занесение в базу и добавление клиенту Id данного счёта.

- DeleteBankAccount(), удаление указанного счёта из базы и из списка счетов у клиента.

- Refill(), зачисление денежной суммы на счёт.

- CashWithdrawal(), снятие денег со счёта.

Bank:

- CheckLoanAvailability(), проверка возможности банка выдать кредит (при прочих нормальных условиях банк может выдавать кредиты, в соотношении 1 к 9, где 1 - это баланс банка, 9 - сумма выданных кредитов). Если условия выполняются (учитывая и новый предполагаемо выдаваемый кредит), то возвратить true.

- Refill(), метод должен реагировать на пополнение счетов клиентами и добавлять данную сумму в Balance.

- CashWithdrawal(), метод должен реагировать на снятие денег со счетов клиентами (оплата кредита или снятия в виде наличных) и отнимать данную сумму из Balance.

**5. Доп. требования**

...

**6. Задачи для студентов**

1. Реализовать сервис (статический класс), который генерирует уникальные Id счетов в формате IBAN.

2. Реализовать интерфейс работы с пользователем. Проверка логина и пароля, корректный вызов соответствующих операций. Возможность запуска нескольких окон с различными пользователями.

3. Реализовать хранение баз данных (кредитов, счетов и клиентов) в виде файлов на диске. Возможность корректирования файлов (удаление и добавление различных полей).

4. Реализовать методы проверки возможности получения кредита у клиента и возможности выдачи кредита у банка.

5. Реализовать методы пополнения\снятия со счёта и соответствующее изменение баланса банка (использовать события).

6. Реализовать метод оплаты по кредиту, с возможностью списания средств со счёта или же оплатой наличными, метод вывода кредитной истории по конкретному клиенту и кредитной истории банка.

7. Реализовать регистрацию новых пользователей, занесение в базу, реализовать метод хэширования пароля и метод сравнения введённого пароля и существующего.

8. Реализовать метод регистрации счетов, добавления их в список клиента, метод удаления счетов и метод проверки баланса.

9. Реализовать метод выдачи кредита (после проверки кредитоспособности банка и клиента), занесение в базу и в лист кредитов клиента, возможность проверки остатка по кредиту.

10.