Лабораторная работа № 3

«Объектно-ориентированное программирование в C++»

Общее задание

Требуется разработать программу с применением объектно-ориентированного программирования (ООП), предназначенную для ведения реестра банковских счетов. Разработку нужно выполнить в три этапа; результаты каждого этапа следует сохранить в отдельный проект (сдаются 3 проекта).

1 Инкапсуляция и жизненный цикл объекта

Необходимо разработать класс `DebitAccount` дебитного банковского счета.

Атрибутами счета является имя вкладчика (строка C, обязательный), номер счета (целое число, обязательный), сумма средств на счете. К перечисленным атрибутам нужно обеспечить доступ для чтения. Обязательные атрибуты должны заполняться в конструкторе.

На счет можно начислять неотрицательную сумму, а также снимать средства в неотрицательном объеме, если их на счете достаточно. Для этих действий нужно реализовать методы `deposit()` и `withdraw()` соответственно. При невозможности выполнить указанные действия методы возвращают **`false`**, иначе — **`true`**.

Счет может быть заморожен и разморожен (состояние можно проверить извне объекта), после заморозки методы `deposit()` и `withdraw()` не изменяют баланс. Атрибуты и методы для этой функциональности предложите и реализуйте самостоятельно.

Метод `close()` закрывает счет, уничтожая текущий объект.

Объекты владеют строками с именами вкладчиков. Должно быть обеспечено корректное создание, копирование, перемещение и удаление объектов, а также правильное присваивание (по крайней мере, копирующее).

Также требуется реализовать функцию для печати данных счета (не метод).

При запуске программа должна предлагать пользователю совершать действия:

а) Открыть счет, заполнив имя вкладчика и номер. Созданный счет запоминается в реестре на время работы программы.

б) Вывести на печать отчет о счетах в реестре.

в) Выполнить операции над счетом (выбрав его по номеру): напечатать информацию, заморозить, разморозить, внести или снять средства.

г) Закрыть счет. Данный счет удаляется из реестра.

Программа должна выводить понятные сообщения на английском языке.

2 Наследование и полиморфизм

Требуется расширить функциональность программы работой с кредитными счетами, создав для их представления класс `CreditAccount`, унаследованный от `DebitAccount`. Изменять реализацию класса `DebitAccount` недопустимо (интерфейс — можно и нужно).

Кредитный счет имеет дополнительные атрибуты — долю комиссии (fee) и максимальный долг (maximum debt). Они могут изменяться после создания счета, при этом объект должен печатать уведомление. Предложите реализацию для этой функциональности по своему усмотрению.

С кредитного счета можно снять больше средств, чем имеется, если при этом не превышается размер максимального долга. Если снимается бóльшая сумма, чем имеется на счете, разность между балансом и снимаемой суммой, увеличенная на долю комиссии, записывается как долг (отрицательная сумма на счете). Это новое поведение должно реализовываться методом `withdraw()`.

У кредитного счета могут быть примечания (строка С), которые оператор может вводить или запрашивать. Кредитный счет владеет строкой примечаний так же, как именем вкладчика, а семантика копирования, перемещения и присваивания кредитных счетов должна быть такой же, как у дебитных.

Функциональность программы должна быть расширена возможностью работать с кредитными счетами, то есть, часть счетов в реестре может быть сберегательными, а часть — кредитными. С точки зрения пользователя:

1) При открытии счета у пользователя должен запрашиваться тип счета: сберегательный или кредитный, и создаваться объект соответствующего класса.
2) Должно предлагаться совершать действия, характерные только для кредитных счетов: смена процентной ставки и т. п. При попытке осуществить такое действие над сберегательным счетом требуется печатать сообщение об ошибке.

Все объекты-счета должны храниться в едином реестре. Печать данных счетов следует внести в состав классов как метод (для печати данных текущего объекта).

3 Абстрактные классы

Предлагается создать абстрактный класс `Account`, содержащий общий интерфейс для любых счетов (методы `deposit()` и `withdraw()` — чисто виртуальные). Старые классы следует переписать, сделав их наследниками `Account`. Такая модель лучше отражает предметную область: действительно, кредитный счет не является частным

случаем дебитного, с другой стороны, можно говорить о «счета вообще», вариантами которого являются дебитный и кредитный счета.

Функциональность новой программы должна быть такой же, как у предыдущей. Если в программах использовалась `<iostream>`, методы вывода на печать данных счетов следует заменить дружественными операторами вывода в поток `std::ostream`. Все объекты-счета должны храниться в едином реестре. На этом этапе следует задействовать, где это уместно, расширенные средства ООП в C++ (согласно авторскому конспекту).