ИД: ООП - Практикум
Домашна работа

Пояснение:

- Реализирайте задачите спазвайки добрите ООП практики(валидация на данните, подходяща капсулация и т.н.)
- Решение, в които не са спазени ООП принципите ще бъдат оценени с 0 точки.
- Предадените от вас решения трябва да могат да се компилират успешно на Visual C++ или GCC.
- **Не е разрешено** да ползвате библиотеки от STL и STL функции.

Изисквания за предаване:

- Всички задачи ще бъдат проверени автоматично за преписване. Файловете с голямо съвпадение ще бъдат проверени ръчно и при установено плагиатство ще бъдат анулирани.
- Предаване на домашното в указания срок от всеки студент като .zip архив със следното име:

Pract2022_SI_(курс)_(група)_(факултетен_номер)

- (курс) е цяло число, отговарящо на курс (например 1);
- (група) е цяло число, отговарящо на групата Ви (например 1);
- (факултетен_номер) е цяло число, отговарящо на факултетния Ви номер (например 12345);

Пример за .zip архив за домашно: Pract2022_SI_1_1_12345.zip

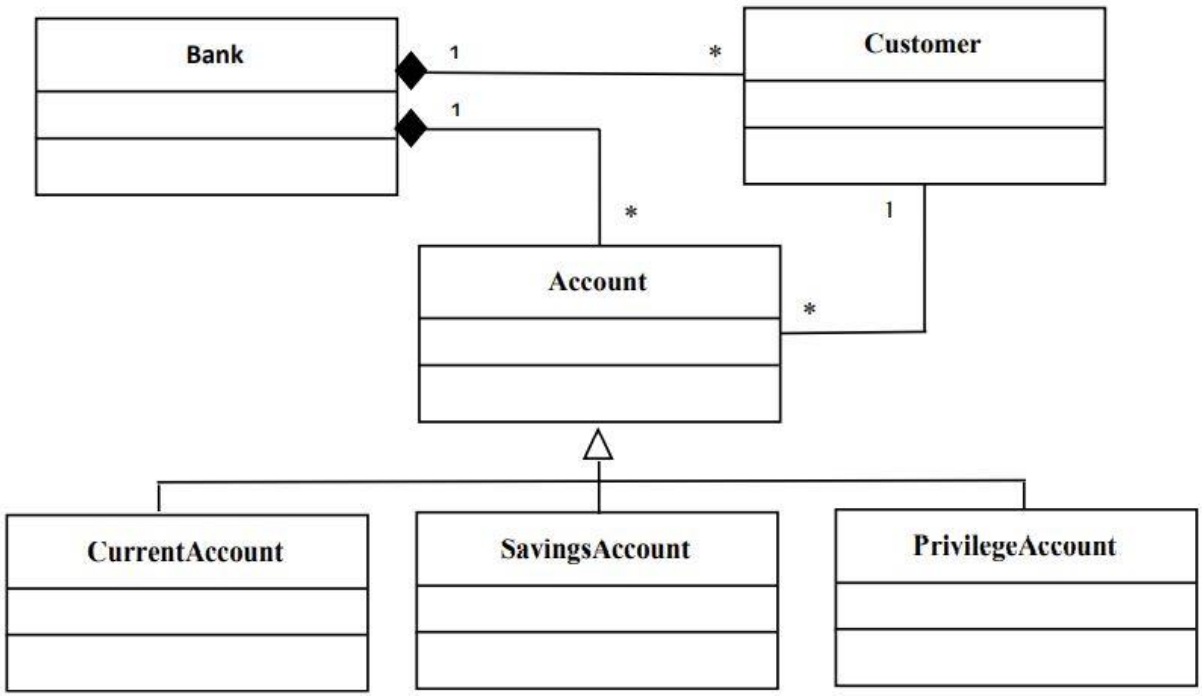
Качване на архива на посоченото място в Moodle;

ИД: ООП - Практикум

Домашна работа

Задача: Bank system

Да се напише програма, която моделира банкова система и операциите, които се извършват в нея. За целта е необходимо да се реализират класовете, показани на следната UML клас-диаграма:



- **Customer (потребител)**
  - id – идентификатор на потребителя
  - name – име на потребителя
  - address – адрес на потребителя
  - Реализирайте подходящи член-функции/методи
- **Account (банкова сметка) - Абстрактен клас(interface)**
  - username – потребителско име
  - password - парола
  - iban – номер на банкова сметка (IBAN)
  - Id – идентификатор на потребителя, собственик на банковата сметка.

ИД: ООП - Практикум

Домашна работа

- amount – налична сума в сметката
- dateOfCreation – дата на създаване (може да използвате ctime)

➢ Минимални функционалности:

- deposit – за добавяне на сума към банковата сметка.
- withdraw – за изтегляне на сума от банковата сметка, ако това е възможно.
- display – за извеждане на информация за сметката.
- getBalance – връща наличната сума в сметката
- Подходящи допълнителни методи.

❖ клас NormalAccount (обикновена сметка)

Класът NormalAccount наследява Account и реализира методите:

- deposit – за добавяне на сума към банковата сметка.
- withdraw – за изтегляне на сума от банковата сметка. Ако в сметката има по-малко пари от исканата сума, връща false, иначе – намалява сумата на сметката с исканата сума и връща true.
- display – за извеждане на информация за сметката – вид на сметка, IBAN, номер на потребител, баланс

❖ Клас SavingsAccount (спестовна сметка)

Класът SavingsAccount наследява Account,като го допълва с:

- interestRate – годишен лихвен процент

и реализира методите:

- deposit – за добавяне на сума към банковата сметка.
- withdraw – Винаги връща false и не позволява изтегляне на сума от банковата сметка.
- display – за извеждане на информация за сметката

❖ клас PrivilegeAccount (привилегирована сметка)

Класът PrivilegeAccount наследява Account, като го допълва с

- overdraft – позволен овърдрафт (сума превишаваща кредита)

и реализира методите:

- deposit – за добавяне на сума към банковата сметка.
- withdraw – за изтегляне на сума от банковата сметка. Ако наличната сума в сметката + позволения овърдрафт е по-малко от исканата сума, връща false, иначе намалява сумата на сметката с исканата сума и връща true (в този случай може да се получи отрицателно число за баланса).
- display – за извеждане на информация за сметката.

ИД: ООП - Практикум

Домашна работа

❖ клас Bank (банка)

- name – име на банката.
- address – адрес на банката.
- customers – списък от потребители. (изберете подходящ начин за представяне)
- accounts – списък от банкови сметки. (изберете подходящ начин за представяне)
- log – списък от всички извършени транзакции на банката(под транзакция разбираме създаване на потребител, създаване на сметка, изтриване на потребител, изтриване на сметка, deposit , withdraw etc)

➢ Минимални функционалности:

- addCustomer(ако не присъства го добавя)
- deleteCustomer - изтрива потребител, както и всички негови сметки
- addAccount – създава нова сметка от съответния тип и я добавя към списъка с банкови сметки.(сметката трябва да си има собственик и да е валидна).
- deleteAccount – изтрива сметка
- listCustomers – извежда списък с потребителите.
- listAccounts – извежда информация за всички сметки в банката.
- listCustomerAccount – извежда информация за банковите сметки на конкретен потребител
- exportLog – запазвала текушия списък от транзакции във файл.
- transfer – извършва банков превод на сума amount от банкова сметка с идентификатор fromIBAN към банкова сметка с идентификатор toIBAN, ако е възможно
- display – извежда информация за банката и колко потребителя и сметки има

ИД: ООП - Практикум

Домашна работа

- Да се напише главна програма
  В главната програма да се създаде банка и да се реализира следното тестово меню:
    1. Edit
        a. Customer
            i. Add new customer
            ii. Delete customer
        b. Account
            i. Add new account
            ii. Delete account
    2. List
        a. List all customers
        b. List all accounts
        c. List customer account
        d. List log
    3. Action
        a. Withdraw from account
        b. Deposit to account
        c. Transfer
    4. Display info for the bank
    5. Quit