Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: **Web програмування**

**Лабораторна робота №1**

**Тема:** «Основні поняття ООП. Класи та об’єкти. Функції доступу. Вказівник this.»

|  |
| --- |
| Виконав: ст. гр. КН-23 |
| Гончаренко В.В |
| Перевірив:  Казірова Л.В. |
|  |

Кропивницький

2024

**ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ**

**Лабораторна робота №1**

**Основні поняття ООП. Класи та об’єкти. Функції доступу. Вказівник this**

**Мета:** ознайомитись з основними поняттями ООП. Вивчити поняття

клас, об’єкт, сеттер, геттер та навчитись їх програмно реалізовувати мовою

С++.

**Варіант 2**

**Завдання 1**

1. Напишіть клас "BankAccount" для керування банківським рахунком.

Клас повинен мати наступні властивості та функціональність:

Приватні поля класу:

• balance (баланс рахунку, типу double)

• accountNumber (номер рахунку, типу string)

• ownerName (ім'я власника рахунку, типу string)

Публічні методи класу:

• Метод setBalance(), який дозволяє задати початковий баланс рахунку.

• Метод getBalance(), який повертає поточний баланс рахунку.

• Метод setAccountNumber(), який дозволяє задати номер рахунку.

• Метод getAccountNumber(), який повертає номер рахунку.

• Метод setOwnerName(), який дозволяє задати ім'я власника рахунку.

• Метод getOwnerName(), який повертає ім'я власника рахунку.

• Метод deposit(), який дозволяє здійснити поповнення рахунку на певну

суму.

• Метод withdraw(), який дозволяє зняти гроші з рахунку на певну суму.

2. Створіть об'єкт класу "BankAccount".

3. Задайте значення полів об'єкта, включаючи баланс рахунку, номер

рахунку та ім'я власника.

4. Використовуйте методи для отримання та зміни значень полів об'єкта.

5. Викличте методи для поповнення та зняття грошей з рахунку.

6. Виведіть інформацію про рахунок на екран.

7. Реалізувати програму за допомогою роздільної компіляції.

У вашому рішенні можуть бути додаткові методи та поля, якщо ви

вважаєте їх необхідними.

**Завдання 2**

Реалізувати вище наведену задачу за допомогою структурного

програмування. У висновку описати різницю цих методів.

**-- Лістинг –**

**BankAccount.h:**

#ifndef BANKACCOUNT\_H

#define BANKACCOUNT\_H

#include <string>

class BankAccount {

private:

double balance;

std::string accountNumber;

std::string ownerName;

public:

BankAccount();

void setBalance(double initialBalance);

double getBalance() const;

void setAccountNumber(const std::string &accNumber);

std::string getAccountNumber() const;

void setOwnerName(const std::string &owner);

std::string getOwnerName() const;

void deposit(double amount);

void withdraw(double amount);

void displayAccountInfo() const;

};

#endif

**-- Лістинг –**

**BankAccount.cpp:**

#include "BankAccount.h"

#include <iostream>

BankAccount::BankAccount() : balance(0.0), accountNumber(""), ownerName("") {}

void BankAccount::setBalance(double initialBalance) {

balance = initialBalance;

}

void BankAccount::setAccountNumber(const std::string &accNumber) {

accountNumber = accNumber;

}

void BankAccount::setOwnerName(const std::string &owner) {

ownerName = owner;

}

double BankAccount::getBalance() const {

return balance;

}

std::string BankAccount::getAccountNumber() const {

return accountNumber;

}

std::string BankAccount::getOwnerName() const {

return ownerName;

}

void BankAccount::deposit(double amount) {

if (amount > 0) {

balance += amount;

std::cout << "Депозит на " << amount << " успішний.\n";

} else {

std::cout << "Помилка депозиту.\n";

}

}

void BankAccount::withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

std::cout << "Вивід суми на " << amount << " успішний.\n";

} else {

std::cout << "Помилка виводу.\n";

}

}

void BankAccount::displayAccountInfo() const {

std::cout << "Номер власника: " << ownerName << "\n";

std::cout << "Номер рахунку: " << accountNumber << "\n";

std::cout << "Баланс: " << balance << "\n";

}

**-- Лістинг –**

**Main.cpp:**

#include "BankAccount.h"

#include <iostream>

int main() {

system("chcp 1251 & cls");

// Створення об'єкта класу

BankAccount account;

// Налаштування властивостей об'єкта

account.setOwnerName("Vlad");

account.setAccountNumber("555444333222111");

account.setBalance(5000);

account.displayAccountInfo();

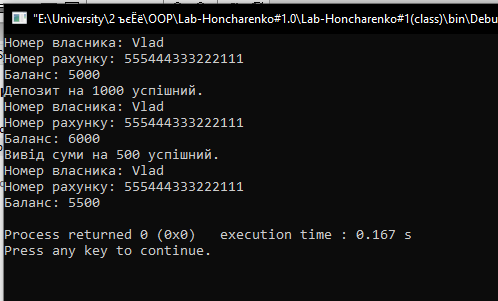
account.deposit(1000);

account.displayAccountInfo();

account.withdraw(500.0);

account.displayAccountInfo();

return 0;}



**-- Лістинг –**

**main.cpp(struct):**

#include <iostream>

#include <string>

struct BankAccount {

double balance;

std::string accountNumber;

std::string ownerName;

};

void setBalance(BankAccount &account, double initialBalance) {

//устанавливает значение balance структуры BankAccount равным значению initialBalance

account.balance = initialBalance;

}

void setAccountNumber(BankAccount &account, const std::string &accNumber) {

account.accountNumber = accNumber;

}

void setOwnerName(BankAccount &account, const std::string &owner) {

//устанавливает значение поля ownerName объекта account равным значению owner

account.ownerName = owner;

}

double getBalance(const BankAccount &account) {

return account.balance;

}

std::string getAccountNumber(const BankAccount &account) {

return account.accountNumber;

}

std::string getOwnerName(const BankAccount &account) {

return account.ownerName;

}

void deposit(BankAccount &account, double amount) {

if (amount > 0) {

account.balance += amount;

std::cout << "Депозит на " << amount << " успішний.\n";

} else {

std::cout << "Помилка депозиту.\n";

}

}

void withdraw(BankAccount &account, double amount) {

if (amount > 0 && amount <= account.balance) {

account.balance -= amount;

std::cout << "Вивід суми на " << amount << " успішний.\n";

} else {

std::cout << "Помилка виводу.\n";

}

}

void displayAccountInfo(const BankAccount &account) {

std::cout << "Номер власника: " << account.ownerName << "\n";

std::cout << "Номер рахунку: " << account.accountNumber << "\n";

std::cout << "Баланс: " << account.balance << "\n";

}

int main() {

system("chcp 1251 & cls");

// Створення об'єкта класу

BankAccount acc;

setBalance(acc, 5000.0);

setAccountNumber(acc, "54321000");

setOwnerName(acc, "Vlad");

displayAccountInfo(acc);

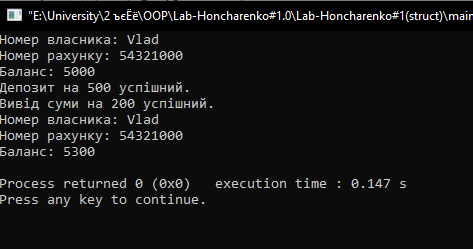
deposit(acc, 500.0);

withdraw(acc, 200.0);

displayAccountInfo(acc);

return 0;

}



**Висновок:**

**Структурне Програмування:**

**Поділ функцій і даних:** Функції та дані працюють окремо. Функції приймають структури як аргументи і виконують операції над їхніми даними.

**Відсутність інкапсуляції:** Дані та функції не об'єднані в одному місці, що може ускладнити управління даними та їх зміни.

**Менша абстракція:** Код менш модульний, що робить його складнішим для підтримки та розширення.

**Об'єктно-Орієнтоване Програмування (ООП):**

**Об'єднання даних і функцій:** Дані (поля) і функції (методи) об'єднані в одному класі, що полегшує управління станом об'єкта.

**Інкапсуляція:** Дані приховані всередині об'єкта і можуть змінюватися тільки через публічні методи, що покращує безпеку і керованість даних.

**Модульність і розширюваність:** Код більш модульний і легше розширюється. Зміни в класі зазвичай не вимагають змін в інших частинах програми.