**Практичне заняття №1**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Реалізація класів, конструктори та деструктори»

Варіант №1

**Виконав:**

студент гр. БС-51

Солярик А. С.

**Перевірив:**

ст. викладач каф. БМК

Алхімова С.М.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2017

**Завдання:**

1. Вивчити теоретичні відомості щодо створення класів та об’єктів мовою програмування C++.
2. Відповідно до свого варіанту розробити клас та відповідну діаграму класів в нотації UML для описаних в завданні характерних даних, визначити конструктор користувача, за допомогою якого можна ініціалізувати всі поля визначеного класу, а також методи для зміни значень кожного із полів:

***Створити клас, що характеризує абонента телефонно станці: прізвище, ім’я, по батькові, адреса, телефон, час розмови, тариф.***

1. Розробити програмний додаток, в якому необхідно створити об’єкт, що є екземпляром визначеного в завданні класу, та ініціалізувати його поля за допомогою створеного конструктора користувача, вивести інформацію про створений об’єкт на екран, змінити значення полів об’єкта за допомогою викликів відповідних методів класу, після чого знову вивести інформацію про модифікований об’єкт на екран, видалити створений об’єкт.
2. Скласти і захистити звіт.

**🞏 Практичне заняття без зауважень**

**🞏 Практичне заняття має зауваження:**

**🞏 несвоєчасний захист**

**🞏 присутні зауваження до UML діаграми:**

**🞏 діаграма класу не відповідає коду**

**🞏 виконані не за стандартом:**

**🞏 атрибути**

**🞏 відношення**

**🞏 багатостатність**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 присутні зауваження до коду:**

**🞏 задача завдання вирішена хибно**

**🞏 код програми не компілюється**

**🞏 хибно задані специфікатори доступу**

**🞏 помилки у визначенні конструкторів / деструкторів**

**🞏 використано глобальні змінні**

**🞏 статичні змінні при роботі з масивами**

**🞏 оформлення коду**

**🞏 присутні зайві символи «{» та «}»**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 невірні відповіді на запитання:**

**🞏 №1 🞏 №2 🞏 №3 🞏 №4 🞏 №5**

**🞏 №6 🞏 №7 🞏 №8 🞏 №9 🞏 №10**

**🞏 маються інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Лiстинг файлу main.cpp:

using namespace std;

#include "Abonent.hpp"

int main(int argc, const char \* argv[]) {

    abonent \*ab  = new abonent("Anton","Solyarik","Sergeevich","Kiev","0635814472", 15, 50);

    char dataIn[] = "";

    ab -> getInfo();

    ab -> setAdress("Dagestan");

    ab -> setTelephoneNumber("0980840458");

    cout<<"Got this name from class: "<<ab -> getName()<<endl;

    ab -> getInfo();

    ab -> setAdress("K");

    cin>>dataIn;

    ab->setSurname(dataIn);

    ab -> getInfo();

    return 0;

}

Лiстинг файлу abonent.hpp:

#ifndef Abonent\_hpp

#define Abonent\_hpp

#include <iostream>

#include <string>

#include <stdio.h>

class abonent

{

private:

    char

    \*name,

    \*surname,

    \*thirdname,

    \*adress,

    \*telephN;

    int

    timetalking,

    tariff;

public:

    abonent(char \*nm, char \*sr, char \*th, char \*ad, char \*tlpn, int tm, int tr): name(nm), surname(sr), thirdname(th),adress(ad),telephN(tlpn),timetalking(tm),tariff(tr)

    {

        name = new char[strlen((nm))];

        strcpy((name), (nm));

        surname = new char[strlen((sr))];

        strcpy((surname), (sr));

        thirdname = new char[strlen((th))];

        strcpy((thirdname), (th));

        adress = new char[strlen((ad))];

        strcpy((adress), (ad));

        telephN = new char[strlen((tlpn))];

        strcpy((telephN), (tlpn));

    }

    ~abonent()

    {

        delete[] name;

        delete[] surname;

        delete[] thirdname;

        delete[] adress;

        delete[] telephN;

        tariff = NULL;

        timetalking = NULL;

    }

    void setName(char \*nm);

    void setSurname(char \*nm);

    void setThirdname(char \*nm);

    void setAdress(char \*nm);

    void setTelephoneNumber(char \*nm);

    void setTariff(int nm);

    void setTimeTalking(int nm);

    void getInfo();

    char \*getName() {return name;};

    char \*getSurname() {return surname;};

    char \*getThirdname() {return thirdname;};

    char \*getAdress() {return adress;};

    char \*gettelephN() {return telephN;};

    int getTariff(){return tariff;};

    int getTimetalking(){return timetalking;};

};

#endif

Лiстинг файлу abonent.cpp

#include "Abonent.hpp"

using namespace std;

#include <iostream>

#include <string>

void abonent::setName(char \*nm)

    {

        if (name!=NULL)

        {

            delete[] name;

            name = new char[strlen((nm))];

            strcpy((name), (nm));

        }

        else

        {

            name = new char[strlen((nm))];

            strcpy((name), (nm));

        }

    }

void abonent::setSurname(char \*nm)

    {

        if (surname!=NULL)

        {

            delete[] surname;

            surname = new char[strlen((nm))];

            strcpy((surname), (nm));

        }

        else

        {

            surname = new char[strlen((nm))];

            strcpy((surname), (nm));

        }

    }

void abonent::setThirdname(char \*nm)

    {

        if (thirdname!=NULL)

        {

            delete[] thirdname;

            thirdname = new char[strlen((nm))];

            strcpy((thirdname), (nm));

        }

        else

        {

            thirdname = new char[strlen((nm))];

            strcpy((thirdname), (nm));

        }

    }

    void abonent::setAdress(char \*nm)

    {

        if (adress!=NULL)

        {

            delete[] adress;

            adress = new char[strlen((nm))];

            strcpy((adress), (nm));

        }

        else

        {

            adress = new char[strlen((nm))];

            strcpy((adress), (nm));

        }

    }

    void abonent::setTelephoneNumber(char \*nm)

    {

        if (telephN!=NULL)

        {

            delete[] telephN;

            telephN = new char[strlen((nm))];

            strcpy((telephN), (nm));

        }

        else

        {

            telephN = new char[strlen((nm))];

            strcpy((telephN), (nm));

        }

    }

    void abonent::setTariff(int nm)

    {

        tariff = nm;

    }

    void abonent::setTimeTalking(int nm)

    {

        timetalking = nm;

    }

void abonent::getInfo()

    {

        cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

        cout <<name<<" "<<surname<<" "<<thirdname<<" "<<endl;

        cout<<telephN<<" "<<adress<<" "<<endl;

        cout<<"Tariff: "<<tariff<<endl;

        cout<<"TimeTalked: "<<timetalking<<endl;

        cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    }

**Дiаграмма UML:**



**Контрольнi питання:**

1. **Що таке клас, навіщо він потрібен?**

Класс – основной элемент в ООП; базовый объект, описывающий абстрактный тип данных, а также реализацию работы с ними.

1. **Що таке об’єкт?**

Объект – основной элемент в ООП, от которого унаследованы все остальные объекты.

1. **Як пов’язані між собою класи і об’єкти в програмі.**

Класс – частный случай абстрактного объекта, который может содержать другие объекты (например, данные).

1. **Що таке інкапсуляція?**

Инкапсуляция – подход, согласно которому данные внутри класса защищены от внешнего воздействия, и могут быть изменены только методами, реализованными в самом классе.

1. **За рахунок чого реалузіється захист від несанкіонованного доступу до даних?**

За счет инкапсуляции.

1. **Чим відрізняються поля від змінних?**

Полями называют переменные внутри класса.

1. **Що визначають методи класу?**

Они определяют то, как объекты будут себя вести.

1. **Перелічити, чим можуть бути поля класу.**

Ограничений нет: это любые переменные (bool, char, string, int, short,long или любой пользовательский тип данных).

1. **Що таке конструктор, навіщо він потрібен?**

Конструктор – специальный тип метода в классе, который можно использовать чтобы заполнить класс данными. Конструктор присутствует всегда, даже если его явно не реализовывать.

**10. Що таке деструктор, навіщо він потрібен?**

Деструктор – специальный тип метода в классе, который вызывается при удалении класса (например, выходе его из зон видимости). В деструкторе явно описываются действия, которые нужно совершить для удаления объекта.