Звіт

Автор: Чугунов В.Ю

КІТ-119а

Лабораторна робота №6

**КЛАСИ. СПАДКУВАННЯ**

**Мета:** Отримати знання про парадигму ООП – спадкування. Навчитися застосовувати отримані знання на практиці.

1 ЗАВДАННЯ ДО РОБОТИ

**Загальне завдання.** Модернізувати попередню роботу шляхом:  
 - додавання класу-спадкоємця, котрий буде поширювати функціонал «базового класу» у відповідності до індивідуального завдання;  
 - додавання ще одного класу-списку, що буде керувати лише елементами класу-спадкоємця;  
 - в функціях базового класу та класу-спадкоємця обов’язкове використання ключових слів *final* та *override*.

**Індивідуальне завдання.** Прикладна галузь: роботи студентів. Додаткові поля в класі-спадкоємцеві: кількість завдань для домашньої роботи.

2 РОЗРОБКА АЛГОРИТМІВ ФУНКЦІЙ

Клас **Home** – спадкоємець класу **Lessons**

**private поля:**

int numOfHWs – кількість завдань у домашній роботі.

Методи:

**void generate( ) override** – перевизначення методу **virtual void Lessons::generate( )**

Призначення: створення об’єкта класу Home.

Опис роботи: функція викликає базовий метод, після чого пропонує ввести кількість завдань для домашньої роботи.

Схема алгоритму подана на рис. 1.

Клас **HomeList** – клас-список класу **Home**

**private поля:**

Home \*hws – показник на перший елемент масиву об’єктів класу Home

int size – відображає кількість елементів у списку

int sumHW – призначена для зберігання загальної кількості домашніх завдань

Методи:  
 **void addHW( Home& hw )**  
 Призначення: додавання готового об’єкта класу Home у список.

Опис роботи: функція перевіряє чи є у списку якісь записи. Якщо ні, то вона створює динамічний масив та заносить у нього об’єкт. А якщо у ньому вже щось є, то вона створює новий масив, що на 1 більший за попередній, переносить у нього усі об’єкти попереднього та додає у кінець новий об’єкт.  
 Схема алгоритму подана на рис. 2.

**void removeHW( int pos )**  
 Призначення: видалення об’єкта зі списку за заданим індексом.  
 Опис роботи: функція перевіряє чи є у списку якісь записи. Якщо ні, то функція виводить помилку. В іншому випадку функція створює новий масив, що на 1 менший за попередній та за допомогою цикла переносить у нього усі об’єкти попереднього, крім того, що видаляється.  
 Схема алгоритму подана на рис. 3.  
  
 **void showAllHWs( )** Призначення: виведення усіх елементів масиву об’єктів класу Home на екран.

Опис роботи: функція перевіряє чи є у списку якісь записи. Якщо ні, то функція виводить помилку. В іншому випадку функція, використовуючи цикл, виводить усі елементи на екран.

Схема алгоритму подана на рис. 4.

**Схеми алгоритмів функцій**

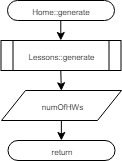


Рисунок 1 – Функція Home::generate

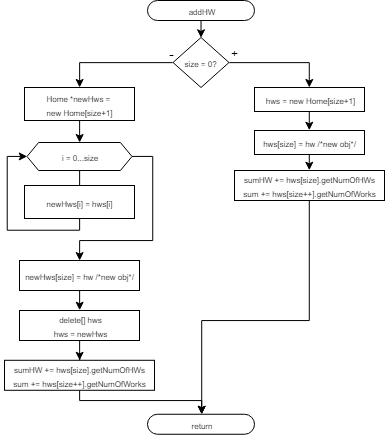


Рисунок 2 – Функція addHW

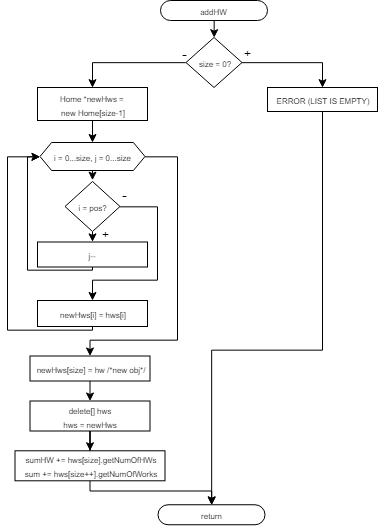


Рисунок 3 – Функція removeHW

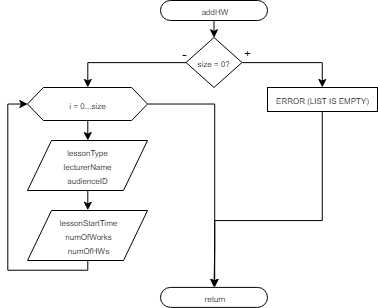


Рисунок 4 – Функція showAllHWs

3 ТЕКСТ ПРОГРАМИ

**Текст файлу Home.h**

#pragma once

#include "Lessons.h"

#include "Works.h"

class Home : public Lessons

{

public:

Home();

Home(const int home, const string t, const string name, const string ID, const int time, const int qty);

Home(const Home &obj);

~Home();

void setNumOfHW(int number) { this->numOfHWs = number; };

inline const int& getNumOfHWs() const { return this->numOfHWs; };

void generate() override;

private:

Works work;

int numOfHWs;

};

**Текст файлу Home.cpp**

#include "Home.h"

Home::Home() : Lessons()

{

this->numOfHWs = 0;

}

Home::Home(const int home, const string t, const string name, const string ID, const int time, const int qty)

: Lessons(t, name, ID, time, qty)

{

this->numOfHWs = home;

cout << "\nConstructor of Home was called. \n";

}

Home::Home(const Home & obj) : Lessons(obj)

{

this->numOfHWs = obj.numOfHWs;

cout << "\nCopying constructor of Home was called. \n";

}

Home::~Home()

{

cout << "\nDestructor of Home was called. \n";

}

void Home::generate()

{

Lessons::generate();

cout << "Enter a number of homework that student must complete per semester: ";

cin >> this->numOfHWs;

}

**Текст файлу HomeList.h**

#pragma once

#include "Home.h"

class HomeList

{

public:

~HomeList();

void addHW(Home& hw);

void removeHW(int pos);

void showAllHWs();

inline const int& get\_size() const { return this->size; };

inline const int& get\_sumHW() const { return this->sumHW; }

inline const int& get\_sum() const { return this->sum; }

private:

Home \*hws;

int size = 0;

int sum = 0;

int sumHW = 0;

};

**Текст файлу HomeList.cpp**

#include "HomeList.h"

HomeList::~HomeList()

{

delete[]hws;

hws = nullptr;

}

void HomeList::addHW(Home & hw)

{

if (!size)

{

hws = new Home[size + 1];

hws[size] = hw;

sumHW += hws[size].getNumOfHWs();

sum += hws[size].get\_numOfWorks();

size++;

}

else {

Home \*newHws = new Home[size + 1];

for (auto i = 0; i < size; i++) {

newHws[i] = hws[i];

}

newHws[size] = hw;

delete[] hws;

hws = newHws;

sumHW += hws[size].getNumOfHWs();

sum += hws[size].get\_numOfWorks();

size++;

}

}

void HomeList::removeHW(int pos)

{

if (!size)

cout << "\nError(list is empty)\n";

else {

Home \*newHws = new Home[size - 1];

for (auto i = 0, j = 0; i < size; i++, j++)

{

if (i == pos) {

j--;

continue;

}

newHws[j] = hws[i];

}

delete[] hws;

hws = newHws;

size--;

}

}

void HomeList::showAllHWs()

{

if (!size)

cout << "\nError (the list is empty)\n";

else {

for (auto i = 0; i < size; i++)

{

cout << "\nInfo about lesson #" << i + 1 << ": " << endl;

cout << "Subject: " << hws[i].get\_lessonType() << endl;

cout << "Lecturer: " << hws[i].get\_lecturerName() << endl;

cout << "Audience ID: " << hws[i].get\_audienceID() << endl;

cout << "Lesson start time: " << hws[i].get\_lessonStartTime() << endl;

cout << "The number of works student wrote for the semester: " << hws[i].get\_numOfWorks() << endl;

cout << "The number of hometasks student wrote per semester: " << hws[i].getNumOfHWs() << endl;

}

}

}

4 РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

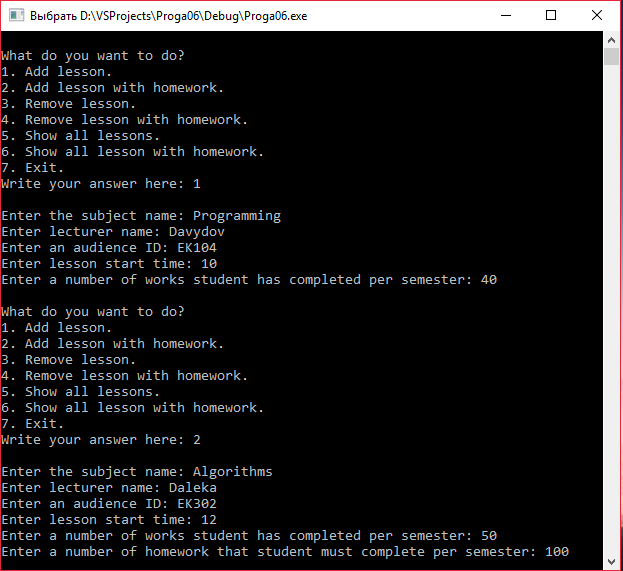


Рисунок 5 – Результат роботи програми (частина 1)

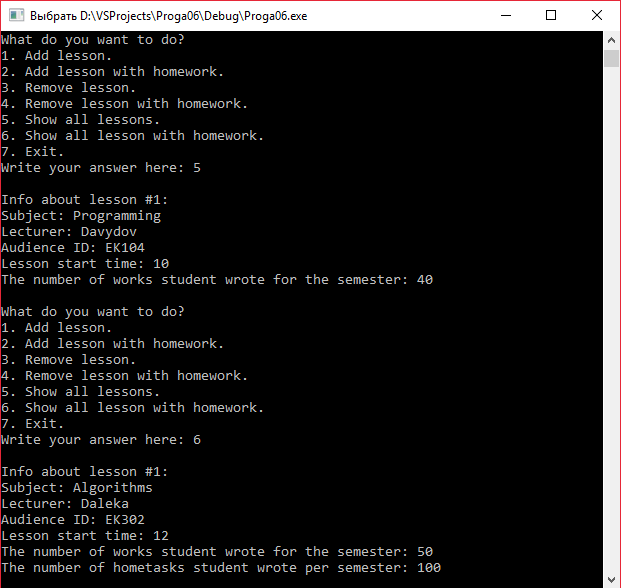


Рисунок 6 – Результат роботи програми (частина 2)

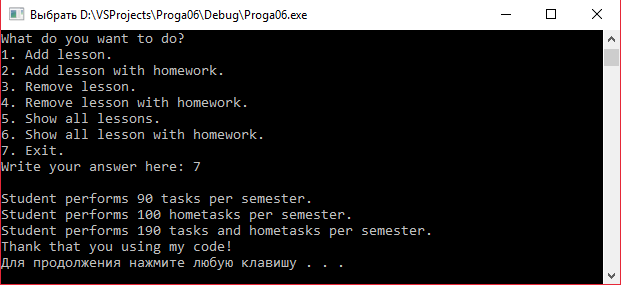


Рисунок 7 – Результат роботи програми (частина 3)

ВИСНОВКИ

При виконанні лабораторної роботи до попередньо розробленої програми (Агрегація та композиція) було додано:  
 - клас Home, як спадкоємець класу Lessons;  
 - клас-список HomeList, який може керувати лише елементами класу Home.

Також досліджені ключові слова *final* та *override*. Слово *override* дозволяє перевизначити будь-який метод базового класу у класі-спадкоємці. А слово *final* навпаки, воно забороняє перевизначати той метод, з яким воно буде використано.