МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з навчальної дисципліни

«***Об’єктно-орієнтоване програмування. Частина 1»***

для студентів РВО «бакалавр» 2 року навчання денної та заочної форми   
спеціальності 122 «Комп’ютерні науки»

Затверджено

на засіданні групи забезпечення спеціальності

Протокол №\_6\_ від «\_19\_»\_08\_\_2022 р.

Голова групи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кузніченко С.Д.

Затверджено

на засіданні кафедри інформаційних технологій  
Протокол №\_1\_ від «\_29\_»червня\_2022 р.



Завідувач кафедри \_\_\_\_\_\_\_\_\_\_\_\_ Казакова Н.Ф.

Одеса – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з навчальної дисципліни

«***Об’єктно-орієнтоване програмування. Частина 1»***

для студентів РВО «бакалавр» 2 року навчання денної та заочної форми   
спеціальності 122 «Комп’ютерні науки»

Одеса – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з навчальної дисципліни

«***Об’єктно-орієнтоване програмування. Частина 1»***

для студентів РВО «бакалавр» 2 року навчання денної та заочної форми   
спеціальності 122 «Комп’ютерні науки»

Затверджено

на засіданні групи забезпечення спеціальності

Протокол № 6

від «\_19»\_08 2022р.

Методичні вказівки до лабораторних робіт з дисципліни «***Об’єктно-орієнтоване програмування. Частина 1»***  для студентів 2 року навчання денної та заочної форми за спеціальністю 122 «Комп’ютерні науки», рівень вищої освіти бакалавр / Козловська В. П., Штефан Н.З. – Одеса, ОДЕКУ, 2022.

**ЗМІСТ**

[ВСТУП 5](#_Toc118361875)

[ПРАВИЛА ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНИ ПРАЦІ 6](#_Toc118361876)

[Лабораторна робота № 1 7](#_Toc118361877)

[«КЛАС, СТВОРЕННЯ ОБ'ЄКТУ КЛАСУ. ПОНЯТТЯ ІНКАПСУЛЯЦІЇ» 7](#_Toc118361878)

[Лабораторна робота № 2 17](#_Toc118361879)

[«КОНСТРУКТОРИ. СТАТИЧЕСКИЕ ЧЛЕНЫ КЛАССА. ШАБЛОН ПРОЕКТИРОВАНИЯ SINGLETON» 17](#_Toc118361880)

[ОСНОВНА ЛІТЕРАТУРА 25](#_Toc118361881)

# ВСТУП

Дане видання призначене для вивчення та практичного засвоєння студентами всіх форм навчання з основ об’єктно-орієнтованого програмування. В даних методичних вказівках розміщенні основні, базові теоретичні відомості, необхідні для виконання лабораторних робіт. Таким чином для успішного виконання лабораторної роботи та при підготовці до її захисту відповідно до графіка студенти повинні ознайомитися з конспектом лекцій та рекомендованою літературою.

Для одержання заліку з кожної роботи студент здає викладачу цілком оформлений звіт (студент завантажує в електроному вигляді у систему Moodle), а також демонструє на екрані комп’ютера результати виконання лабораторної роботи (в аудіторії/за допомогою Zoom).

Звіт має містити:

* титульний аркуш;
* тему та мету роботи;
* завдання до роботи;
* знімки екрану, що відображають результати роботи;
* відповіді на контрольні питання до роботи;
* змістовний аналіз отриманих результатів та висновки.

Звіт виконують у електроному форматі A4(210×297 мм). Поля сторінки з усіх боків – 20 мм.

Під час співбесіди при захисті лабораторної роботи студент повинен виявити знання про мету роботи, по теоретичному матеріалу, про методи виконання кожного етапу роботи, по змісту основних розділів оформленого звіту з демонстрацією результатів на конкретних прикладах.

Студент повинен вміти правильно аналізувати отримані результати. Для самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

# ПРАВИЛА ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНИ ПРАЦІ

Кожен студент перед початком учбової практики мусить вивчити правила роботи з комп’ютерною технікою в лабораторіях кафедри інформаційних технологій, пройти співбесіду з інструктором по техніці безпеки та розписатися в журналі по техніці безпеки.

# Лабораторна робота № 1

# «КЛАС, СТВОРЕННЯ ОБ'ЄКТУ КЛАСУ. ПОНЯТТЯ ІНКАПСУЛЯЦІЇ»

**Мета роботи:** здобути навички проектування найпростіших класів. Навчитися створювати об'єкти класу. Освоїти принцип інкапсуляції.

**Завдання до роботи**

1. Вибрати предметну область згідно з варіантом індивідуального завдання.

2. Спроектувати клас для обраної предметної області.

3. Намалювати діаграму спроектованого класу.

4. Передбачити наявність у об'єкта полів, методів та властивостей.

5. Передбачити наявність властивостей лише запису.

**Контрольні питання:**

1. У чому полягає принцип інкапсуляції?
2. За допомогою якого ключового слова створюється об'єкт класу?
3. Як оголосити властивість лише для читання?
4. Чим поля класу відрізняються від властивостей?

**Література:[1], [3], [4]**

**Короткі теоретичні відомості**

Клас – це тип, що визначається програмістом, у якому поєднуються структури даних та функції їх обробки. Змінні типи класу називаються екземплярами класу і створюються за допомогою оператора new.

Класи можуть містити змінні та константи, які називаються полями. У класі можуть бути оголошені функції, що виконують дії над полями, та звані методами.

Проектування класів слід виконувати, дотримуючись стратегії мінімальної пов'язаності та залежності між ними. Це досягається за рахунок використання принципу інкапсуляції.

Інкапсуляція – це обмеження доступу до полів та методів за допомогою модифікаторів доступу. Основні модифікатори доступу до C#:

public – доступ без обмежень;

private – доступ дозволено лише членам класу;

protected – доступ дозволено як членам цього класу, так і похідного (спадкоємцям).

Якщо модифікатор доступу для поля класу явно не визначений, то діє модифікатор доступу private

Приклад оголошення класу Employee (Співробітник) із закритим полем та відкритим методом:

class Employee

{

private string name="Коваль";

public void PrintName()

{

Console.WriteLine("Name = "+name);

}

}

Крім полів та методів у класі можна оголошувати властивості (property – властивість). Властивості дозволяють оголосити два методи, один із яких викликається при встановленні значення властивості (метод set), а другий – при її читанні (метод get). Зазвичай код цих методів містить звернення до поля, що зберігає значення властивості. Можливе оголошення властивості або лише для читання, або лише для запису.

У методі запису set для доступу до значення, що записується, використовують ключове слово value.

Приклад класу Employee доповненого властивістю для читання та запису:

class Employee

{

private string name="Коваль";

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public void PrintName()

{

Console.WriteLine($"Name = {name}");

}

}

Після створення об'єкта класу для доступу до відкритих членів класу (полям, методам та властивостям) використовують оператор точка. В одній програмі можна створити декілька об'єктів одного класу.

**Консольне введення-виведення у програмах на мові C#**

Для консольного введення та виведення у програмі на мові C# використовуються методи класу Console.

Для виведення використовуються методи Write() та WriteLine(). Метод Write() виконує потрібне виведення та залишає курсор на останньому рядку виведеного тексту (даних).

Метод WriteLine() виконує ти самі дії, але наприкінці переводить курсор на початок наступного рядка екрану:

Console.WriteLine("Hello, World!");

Клас міститься у просторі імен (namespace) System. Цей простір імен зазвичай підключають до програм. Підключення простору імен виконується директивою using:

using System;

**Консольне введення**

Окрім виведення інформації на консоль, ми можемо отримувати інформацію з консолі. Для цього призначено метод Console.ReadLine(). Він дозволяє отримати введений рядок.

|  |  |
| --- | --- |
|  | Console.Write("Введите свое имя: ");  string? name = Console.ReadLine();  Console.WriteLine($"Привет {name}"); |

В даному випадку все, що вводить користувач, за допомогою методу Console.ReadLine() передається у змінну name.

Приклад роботи програми:

Введите свое имя: Том

Привет Том

Особливістю методу Console.ReadLine()є те, що він може зчитувати інформацію з консолі лише у вигляді рядка. Крім того, можлива ситуація, коли для методу Console.ReadLine()не виявиться доступних для зчитування рядків, тобто коли йому нема чого зчитувати, він повертає значення null, тобто, грубо кажучи, фактично відсутність значення. І щоб відобразити цю ситуацію ми визначаємо змінну name, у яку отримуємо введення з консолі, як змінну типу string?. Тут string вказує, що змінна може зберігати значення типу string, тобто рядки. А знак питання ? показує, що змінна також може зберігати значення null, тобто насправді не мати ніякого значення. Далі ми детальніше розберемо значення null, і як з ним працювати.

Однак, може виникнути питання, як нам бути, якщо, припустимо, ми хочемо ввести вік у змінну типу int або іншу інформацію у змінні типу double чи decimal? За замовчуванням платформа .NET надає ряд методів, які дозволяють конвертувати різні значення до типів int, double і т.д. Деякі з цих методів:

* **Convert.ToInt32()** (перетворює до типу int)
* **Convert.ToDouble()** (перетворює до типу double)
* **Convert.ToDecimal()** (перетворює до типу decimal)

Приклад введення значень:

Console.Write("Введите имя: ");

string? name = Console.ReadLine();

Console.Write("Введите возраст: ");

int age = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите рост: ");

double height = Convert.ToDouble(Console.ReadLine());

Console.Write("Введите размер зарплаты: ");

decimal salary = Convert.ToDecimal(Console.ReadLine());

Console.WriteLine($"Имя: {name}; Возраст: {age}: Рост: {height}м:  Зарплата: {salary}$");

При введенні важливо враховувати поточну операційну систему. В одних культурах роздільником між цілою та дробовою частиною є точка (США, Великобританія...), в інших – кома (Росія, Німеччина...). Наприклад, якщо поточна ОС - російськомовна, то треба вводити дробові числа з роздільником коми. Якщо локалізація англомовна, значить, роздільником цілої та дробової частини під час введення буде точка.

Приклад роботи програми:

Введите имя: Том

Введите возраст: 25

Введите рост: 1,75

Введите размер зарплаты: 300,67

Имя: Том Возраст: 25 Рост: 1,75м Зарплата: 300,67$

**Індивідуальні завдання**

1. Предметна область: **ATC**. У класі зберігати інформацію про адресу АТС, кількість абонентів, абонентську плату (для всіх абонентів одна). Реалізувати метод підрахунку абонентської плати всіх клієнтів.

2. Предметна область: **Вокзал**. У класі зберігати інформацію про найменування станції, вартість квитка (вартість однакова для всіх напрямків), кількість місць, кількість проданих квитків. Реалізувати метод для розрахунку загальної вартості всіх непроданих квитків.

3. Предметна область: **ОСББ**. У класі зберігати інформацію про район, до якого належить **ОСББ**, адресу будинку, кількість мешканців, оплату за місяць (для всіх мешканців однакова), кількість тих, хто оплатив. Реалізувати метод для підрахунку загальної заборгованості мешканців.

4. Предметна область: **Аеропорт**. У класі зберігати інформацію про назву аеропорту, вартість квитка (вартість однакова), загальну кількість місць у всіх літаках, кількість проданих квитків. Реалізувати метод для розрахунку загальної вартості всіх проданих квитків.

5. Предметна область: **Банк**. У класі зберігати інформацію про найменування банку, кількість вкладів, розмір вкладу (всі вклади однакові), розмір процентної ставки. Реалізувати метод для підрахунку загальної виплати за відсотками.

6. Предметна область: **Відділ кадрів**. У класі зберігати інформацію про найменування підприємства, кількість працівників, норму виробітку годин на місяць (одна для всіх працівників), оплату за годину, прибутковий податок. Реалізувати метод для підрахунку загальної виплати з прибуткового податку.

7. Предметна область: **Фірма вантажоперевезень**. У класі зберігати інформацію про оплату за перевезення однієї тони вантажів (не залежить від напрямку), про масу перевезених вантажів, найменування фірми. Реалізувати спосіб підрахунку загальної виручки фірми.

8. Предметна область: **Інтернет-оператор**. У класі зберігати інформацію про вартість тарифу (одна для всіх користувачів), найменування оператора, кількість абонентів. Реалізувати метод підрахунку загальної виручки.

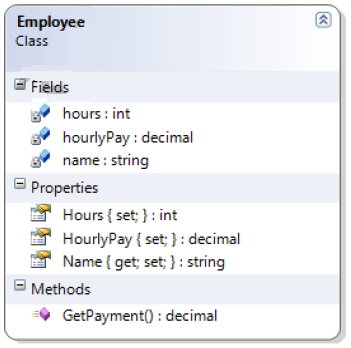
9. Предметна область: **Інтернет-магазин** із продажу телевізорів. У класі зберігати інформацію про вартість телевізора (одна для всіх моделей), найменування магазину, кількість покупок. Реалізувати метод підрахунку загальної виручки.

10. Предметна область: **Готель**. У класі зберігати інформацію про назву готелю, кількість заселених місць, загальну кількість місць, оплату за день проживання (для всіх мешканців однакова вартість). Реалізувати спосіб підрахунку загальної виручки готелю.

**Приклад виконання роботи**

Нехай задана предметна область: **Завод**. У працівника заводу зберігати прізвище, розмір погодинної оплати та кількість відпрацьованих годин. У класі реалізувати метод для підрахунку заробітної плати працівника, виходячи з величини погодинної оплати та відпрацьованих годин.

Діаграма спроектованого класу:



Текст программы:

using System;

namespace ConsoleAppE\_ampleLab\_cs1

{

class Employee

{

string name;

public string Name

{

get {return name;}

set {name = value;}

}

decimal hourlyPay;

public decimal HourlyPay

{

get {return hourlyPay;}

set {hourlyPay = value;}

}

private int hours;

public int Hours

{

set {hours = value;}

}

public decimal GetPayment ()

{

return hours \* hourlyPay;

}

public void GetInfo ()

{

Console.WriteLine($"Прiзвище: {name}; погодинна платня: {hourlyPay}; працював {hours} годин; Заробiток = {GetPayment ()}");

}

}

class Program

{

static void Main(string[] args)

{

Employee worker1 = new Employee();

Console.WriteLine("Введiть прiзвище 1-го працiвника:");

worker1.Name = Console.ReadLine();

Console.WriteLine("Введiть для нього платню за годину роботи:");

worker1.HourlyPay = Convert.ToDecimal(Console.ReadLine());

Console.WriteLine("Скiлькi годин вiн працював:");

worker1.Hours = Convert.ToUInt32(Console.ReadLine());

worker1.GetInfo();

Console.WriteLine();

Console.WriteLine("Введiть прiзвище 1-го працiвника:");

string name2 = worker1.Name = Console.ReadLine();

Console.WriteLine("Введiть для нього платню за годину роботи:");

decimal hourly\_Pay = Convert.ToDecimal(Console.ReadLine());

Console.WriteLine("Скiлькi годин вiн працював:");

int hours2 = Convert.ToUInt32(Console.ReadLine());

Employee worker2 = new Employee () {Name = name2, HourlyPay = hourly\_Pay, Hours = hours2};

worker2.GetInfo();

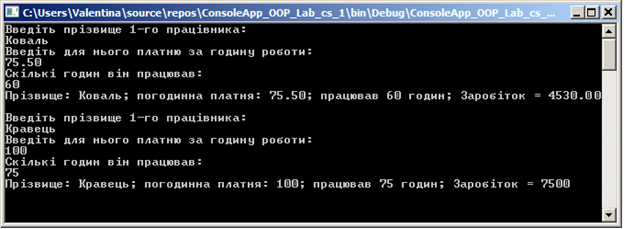
Console.ReadKey();

}

}

}

Консольне виведення програми:



# Лабораторна робота № 2

# «КОНСТРУКТОРИ. СТАТИЧЕСКИЕ ЧЛЕНЫ КЛАССА. ШАБЛОН ПРОЕКТИРОВАНИЯ SINGLETON»

**Мета роботи:** вивчити роботу та призначення конструкторів. Засвоїти можливості членів класу з модифікатором static. Ознайомитись з шаблоном проектування Singleton.

**Завдання до роботи**

1. Спроектувати класи для обраної предметної області.

2. Намалювати діаграму класів.

3. Застосувати до одного з класів шаблон проектування Singleton.

**Контрольні питання**

1. Чим конструктор відрізняється від звичайного методу?

2. Які переваги дає перевантаження конструкторів?

3. Чи можливий доступ до статичного методу з екземплярного методу та навпаки?

4. Яке призначення шаблону проектування Singleton?

**Література:[2], [4], [5]**

**Короткі теоретичні дані**

Конструктор – це метод, що має ім'я таке саме, як і ім'я класу і не повертає значення. Конструктор викликається при створенні об'єкта класу та служить для початкової ініціалізації полів та властивостей. Як і будь-який метод класу, можна перевантажити конструктор, виклик відповідного конструктора буде визначатися за списком параметрів, який вказується при створенні об'єкта класу.

Приклад перевантаження конструктора:

class Item

{

decimal price;

public decimal Price

{

get { return price; }

set { price = value; }

}

public Item()

{

price = 10;

}

public Item(decimal p)

{

price = p;

}

}

Для виклику конструктора без параметрів, який зветься конструктором по замовчуванню, слід написати:

Item i1=new Item();

Коли потрібно викликати конструктор з параметрами, фактичні значення, що передаються, вказуються в круглих дужках, як показано нижче:

Item i1=new Item(50);

Якщо одному конструкторі слід викликати інший перевантажений конструктор, то після оголошення першого слід поставити двокрапку і вказати ключове слово this. У круглих дужках після цього через кому вказуються фактичні значення, що передаються конструктору.

Статичний член класу оголошується із ключовим словом static і є незалежним від усіх об'єктів класу. Статичне поле або метод стає доступним до створення об'єкта класу. Для доступу до статичного члена за межами класу достатньо вказати ім'я цього класу та через оператор «крапка» ім'я статичного члена.

Шаблон проектування Singleton – це шаблон, що породжує, завданням якого є гарантія можливості створення тільки одного об'єкта класу та надання до цього об'єкта глобальної точки доступу.

Для застосування шаблону проектування Singleton до конкретного класу потрібно: оголосити закритий конструктор в цьому класі; оголосити закрите статичне посилання цей клас; додати відкритий статичний метод, який повертає посилання єдиний створений об'єкт класу. Статичний метод повертає посилання на єдиний створений об'єкт класу і є глобальної точкою доступу до цього об'єкту.

Приклад класу, написаного відповідно до шаблону Singleton:

class World

{

private static World world;

private World()

{ }

public static World GetWorld()

{

if (world==null) world = new World(); return world;

}

}

**Індивідуальні завдання**

Розробити два класи: клас-контейнер, керуючий контейнеризованим класом, і контейнеризований клас. Для класу-контейнера застосувати шаблон проектування Singleton.

Об’єкт контейнеризованого класу може вмикатись/вимикатись (або відкриватись/закриватись). Тому клас повинен мати поле логічного типу, яке показує, чи є об’єкт включеним (відкритим). Також потрібно у контейнеризованому класу виявити атрибут кількісного типу, який може зменшувати та збільшувати своє значення: кількість людей (пацієнтів, працівників); одиниць зберігання (деталь, багаж, книга, заказ); потужність освітлення.

**Варіанти індивідуальних завдань**

Опис предметної області:

1. Будівля – Опалювальна система.

2. Лікарня – Приймальне відділення.

3. Завод – Склад деталей.

4. Аеропорт – Злітна смуга.

5. Вокзал – Багажне відділення.

6. Фірма – Відділ кадрів.

7. Ресторан – Кухня.

8. Комп'ютер – Монітор.

9. Бібліотека – Книгосховище.

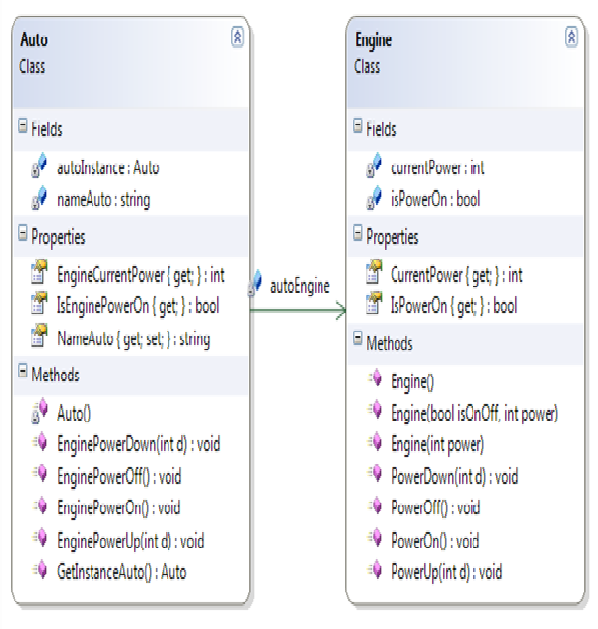
**Приклад виконання роботи**

Нехай задано предметну область: **Автомобіль – Двигун автомобіля**. Клас «автомобіль» є контейнерним класом, а клас «двигун» – контейнеризованим.

У класі «двигун» зберігається інформація про його стан (увімкнений/вимкнений) та про поточну потужність.

У програмі доступ до полів, властивостей та методів об'єкта класу «двигун» здійснюється через клас-контейнер «автомобіль».

Діаграма класів:



**Текст програми.**

using System;

namespace ConsoleAppLab4\_OOP\_2cs

{

//клас Двигун

class Engine

{

//стан двигуна

private bool isPowerOn;

public bool IsPowerOn

{

get { return isPowerOn; }

}

//поточна потужність

private uint currentPower;

public uint CurrentPower

{

get { return currentPower; }

}

//конструктор

public Engine()

{

isPowerOn = true;

currentPower = 10;

}

//конструктор

public Engine(uint power)

{

isPowerOn = true;

currentPower = power;

}

//конструктор

public Engine(bool isOnOff, uint power)

{

isPowerOn = isOnOff;

if (isPowerOn)

currentPower = power;

else currentPower = 0;

}

//увімкнути двигун

public void PowerOn()

{

isPowerOn = true;

}

//вимкнути двигун

public void PowerOff()

{

isPowerOn = false;

currentPower = 0;//эта строка для выключения (мощности), но не для закрытия (приемное отделение, отдел кадров, склад)

}

//збільшити потужність

public void PowerUp(uint d)

{

if (isPowerOn)

currentPower += d;

}

//зменшити потужність

public void PowerDown(uint d)

{

if (currentPower > d)

currentPower -= d;

else currentPower = 0;

}

}

//

class Auto

{

//закрите посилання на об'єкт класу Автомобіль

private static Auto autoInstance;

//закрите посилання на об'єкт класу Двигун

private Engine autoEngine;

//закритий конструктор

private Auto()

{

autoEngine = new Engine(false, 0);

}

public static Auto GetInstanceAuto()

{

if(autoInstance==null)

{

autoInstance = new Auto();

}

return autoInstance;

}

//назва марки автомобіля

string nameAuto;

public string NameAuto

{

get { return nameAuto; }

set { nameAuto = value; }

}

//увімкнути двигун автомабіля

public bool IsEnginePowerOn

{

get { return autoEngine.IsPowerOn; }

}

//

public void EnginePowerOn()

{

autoEngine.PowerOn();

}

//вимкнути двигун автомабіля

public void EnginePowerOff()

{

autoEngine.PowerOff();

}

//поточна потужність двигуна автомабіля

public uint EngineCurrentPower

{

get { return autoEngine.CurrentPower; }

}

public void EnginePowerUp(uint d)

{

autoEngine.PowerUp(d);

}

public void EnginePowerDown(uint d)

{

autoEngine.PowerDown(d);

}

}

class Program

{

static void Main(string[] args)

{

//Створення об'єкту класу автомобіля та разом з цим об'єкту класу двигуна

Auto auto1 = Auto.GetInstanceAuto();

uint d;

string Yes;

Console.WriteLine("Введіть назву марки автомобіля");

auto1.NameAuto = Console.ReadLine();

PrintInfoOfAuto(auto1);

Console.WriteLine("Увімкнути двигун? (Y/N)");

Yes = (Console.ReadLine());

char yes=Yes[0];

if (yes == 'y' || yes=='Y')

{

auto1.EnginePowerOn();

//

PrintInfoOfAuto(auto1);

do

{

Console.WriteLine("Введіть, на скільки збільшити потужність");

d = Convert.ToUInt32(Console.ReadLine());

auto1.EnginePowerUp(d);

PrintInfoOfAuto(auto1);

Console.WriteLine("Введіть, на скільки зменшити потужність");

d = Convert.ToUInt32(Console.ReadLine());

auto1.EnginePowerDown(d);

PrintInfoOfAuto(auto1);

Console.WriteLine("Вимкнути двигун? (Y/N)");

yes = (Console.ReadLine())[0];

} while (yes != 'y' && yes != 'Y');

auto1.EnginePowerOff();

PrintInfoOfAuto(auto1);

}

}

//метод для отримання інформації про автомобіль та його двигун

static void PrintInfoOfAuto(Auto auto1)

{

Console.Write($"Автомобiль {auto1.NameAuto} двигун увiмкнутий {auto1.IsEnginePowerOn}. ");

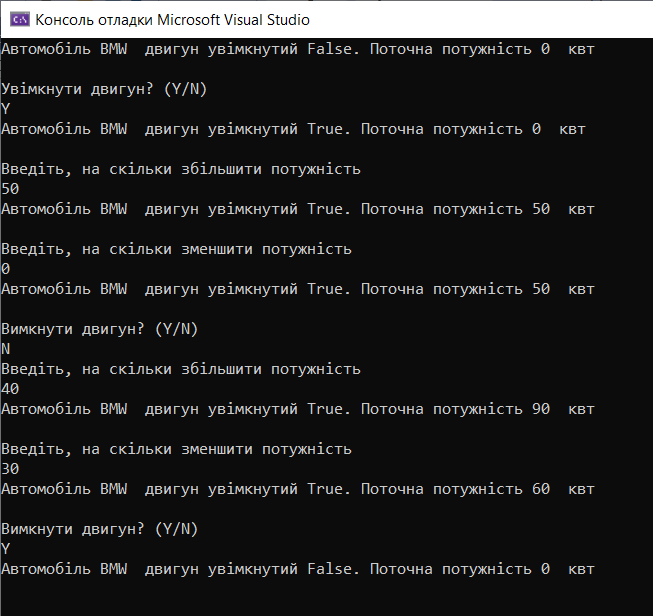
Console.WriteLine($"Поточна потужнiсть {auto1.EngineCurrentPower} квт"); Console.WriteLine();

}

}

}

**Результати роботи програми:**



# ОСНОВНА ЛІТЕРАТУРА

1. Руководства по Visual Studio | C# [Електронний ресурс]. – Режим доступу: URL: https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/?view=vs-2022 – Назва з екрана.
2. .NET Framework [Електронний ресурс]. – Режим доступу : URL https://msdn.microsoft.com/ru-ru/library/w0x726c2(v=vs.110).aspx – Назва екрана.
3. Справочник по C# [Електронний ресурс]. – Режим доступу : URL https://msdn.microsoft.com/ru-ru/library/618ayhy6.aspx – Назва з екрана.
4. Об’єктно орієнтоване програмування на Java. Конспект лекцій з дисципліни «Об’єктно орієнтоване програмування» для студентів спеціальностей 121 – «Програмна інженерія», 123 – «Комп’ютерна інженерія». /Укл.: Бивойно П.Г. – Чернігів: ЧНТУ, 2019. – 136 с. URL: <http://www.dgma.donetsk.ua/docs/kafedry/avp/metod/%D0%9E%D0%9E%D0%9F%20%D0%9A%D0%BE%D0%BD%D1%81%D0%BF%D0%B5%D0%BA%D1%82%20%D0%BB%D0%B5%D0%BA%D1%86_%D0%B9%20%D0%91%D0%B8%D0%B2%D0%BE%D0%B9%D0%BD%D0%BE.pdf>

**ДОДАТКОВА ЛІТЕРАТУРА**

5. Кравець П.О. Об'єктно-орієнтоване програмування: навч. посібник / П.О. Кравець. - Львів: Видавництво Львівської політехніки, 2012. – 624 с.