Кам’янець-Подільський національний університет імені Івана Огієнка

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

Навчальна дисципліна «Об’єктно-орієнтоване програмування»

ЛАБОРАТОРНА РОБОТА #0101

Тема:

Основи програмування мовою C#

Варіант №3

**Виконала**:  
студентка 1-го курсу  
групи KNms1-B24  
Некрасова Ю.Д.

**Прийняв**:  
Слободянюк О.В.

Кам’янець-Подільський – 2025

1. **Короткі теоретичні відомості**

Поліморфізм — це принцип об'єктно-орієнтованого програмування, який дозволяє об'єктам різних типів бути обробленими через спільний інтерфейс. Це означає, що одна і та ж операція може бути виконана по-різному в залежності від типу об'єкта, на якому вона виконується.

У C# поліморфізм реалізується через два основні механізми:

1. Поліморфізм під час виконання (динамічний поліморфізм) — використовує механізм перевизначення методів (override).
2. Поліморфізм під час компіляції (статичний поліморфізм) — використовує механізм перевантаження методів (overload).

Поліморфізм під час виконання (динамічний поліморфізм) досягається через наслідування та перевизначення методів. Клас-потік може перевизначити методи свого базового класу. Коли викликається метод, виконується версія методу, що відповідає типу об'єкта, на якому він викликається, а не типу змінної, через яку цей метод викликається.

Поліморфізм під час компіляції (статичний поліморфізм) досягається через перевантаження методів (overloading), коли метод має однакове ім'я, але різні параметри (типи або кількість параметрів).

1. **Умови завдання**

Створити додаток, який задовольняє вимогам, наведеним в завданні. Успадкування застосовувати тільки в тих завданнях, в яких це логічно обґрунтоване. Аргументувати приналежність класу кожного створюваного методу і коректно перевизначити для кожного класу методи equals(), hashCode(), toString().

Створити об'єкт класу Літак, використовуючи класи Крило, Шасі, Двигун. Методи: літати, задавати маршрут, вивести на консоль маршрут.

1. **Послідовність виконання роботи**

Клас Engine (Двигун):

* Має властивість Power, що описує потужність двигуна.Метод ToString() для виведення інформації про двигун.

Клас Wing (Крило):

* Має властивість Span, що описує розмах крила.
* Метод ToString() для виведення інформації про крило.

Клас Chassis (Шасі):

* Має властивість Material, що описує матеріал шасі.
* Метод ToString() для виведення інформації про шасі.

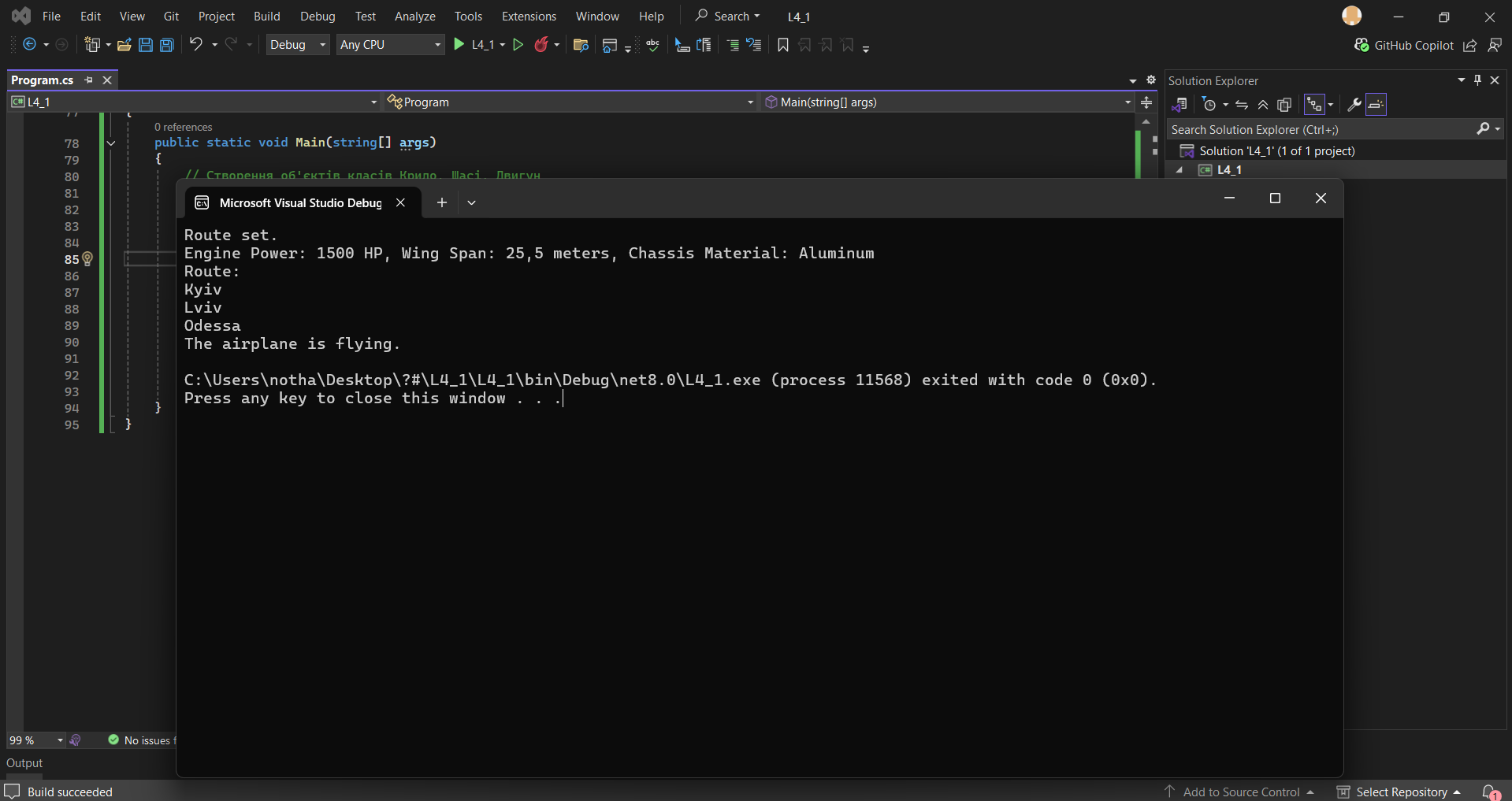
Клас Airplane (Літак):

* Має властивості для збереження об'єктів типу Engine, Wing і Chassis.
* Має методи:
  + Fly(), який виводить повідомлення про політ літака.
  + SetRoute(), який задає маршрут.
  + DisplayRoute(), який виводить на консоль маршрут.
  + Метод ToString() для виведення інформації про літак.

Клас Program:

* Створюється об'єкт Airplane, який ініціалізується через об'єкти класів Engine, Wing і Chassis.
* Встановлюється маршрут, і після цього виводиться інформація про літак і маршрут.
* Викликається метод Fly(), що виводить повідомлення про політ літака.

1. **Код програми**
2. using System;
3. using System.Collections.Generic;
4. public class Engine
5. {
6. public int Power { get; set; }
7. public Engine(int power)
8. {
9. Power = power;
10. }
11. public override string ToString()
12. {
13. return $"Engine Power: {Power} HP";
14. }
15. }
16. public class Wing
17. {
18. public double Span { get; set; }
19. public Wing(double span)
20. {
21. Span = span;
22. }
23. public override string ToString()
24. {
25. return $"Wing Span: {Span} meters";
26. }
27. }
28. public class Chassis
29. {
30. public string Material { get; set; }
31. public Chassis(string material)
32. {
33. Material = material;
34. }
35. public override string ToString()
36. {
37. return $"Chassis Material: {Material}";
38. }
39. }
40. public class Airplane
41. {
42. public Engine Engine { get; set; }
43. public Wing Wing { get; set; }
44. public Chassis Chassis { get; set; }
45. public List<string> Route { get; set; } = new List<string>();
46. public Airplane(Engine engine, Wing wing, Chassis chassis)
47. {
48. Engine = engine;
49. Wing = wing;
50. Chassis = chassis;
51. }
52. // Метод для польоту
53. public void Fly()
54. {
55. Console.WriteLine("The airplane is flying.");
56. }
57. // Метод для встановлення маршруту
58. public void SetRoute(List<string> route)
59. {
60. Route = route;
61. Console.WriteLine("Route set.");
62. }
63. // Метод для виведення маршруту
64. public void DisplayRoute()
65. {
66. Console.WriteLine("Route:");
67. foreach (var point in Route)
68. {
69. Console.WriteLine(point);
70. }
71. }
72. public override string ToString()
73. {
74. return $"{Engine}, {Wing}, {Chassis}";
75. }
76. }
77. public class Program
78. {
79. public static void Main(string[] args)
80. {
81. // Створення об'єктів класів Крило, Шасі, Двигун
82. Engine engine = new Engine(1500);
83. Wing wing = new Wing(25.5);
84. Chassis chassis = new Chassis("Aluminum");
85. // Створення об'єкта Літак
86. Airplane airplane = new Airplane(engine, wing, chassis);
87. // Встановлення маршруту
88. airplane.SetRoute(new List<string> { "Kyiv", "Lviv", "Odessa" });
89. // Виведення інформації про літак
90. Console.WriteLine(airplane);
91. // Виведення маршруту
92. airplane.DisplayRoute();
93. // Політ
94. airplane.Fly();
95. }
96. }
97. **Приклад виконання програми**



1. **Умови завдання**

У кожному варіанті завдання при описі класів самостійно визначити необхідні поля, властивості та методи вводу/виводу. Деякі методи класу-предка повинні бути віртуальними і абстрактними. У програмі-клієнті для збереження сукупності об’єктів використати масив.

Створити клас TBody, який представляє просторову геометричну фігуру з методами обчислення площі поверхні та об’єму. На основі цього класу створити класи нащадки TParallelepiped та TBall. Випадковим чином створити певну кількість паралелепіпедів та куль, щоб їх сумарна кількість дорівнювала *n* . Знайти сумарну площу поверхонь усіх геометричних тіл.

1. **Послідовність виконання роботи**

Класи:

Клас TBody — це базовий клас для просторових геометричних фігур, в якому будуть абстрактні методи для обчислення площі поверхні та об'єму.

Клас TParallelepiped — нащадок класу TBody, який реалізує методи для обчислення площі поверхні та об'єму прямокутного паралелепіпеда.

Клас TBall — нащадок класу TBody, який реалізує методи для обчислення площі поверхні та об'єму кулі.

Програма-клієнт — створення випадкової кількості паралелепіпедів і куль, підсумування площ поверхонь та виведення результату.

Пояснення:

Клас TBody:

* Це абстрактний клас для геометричних тіл, який має два абстрактні методи:
  + SurfaceArea() — для обчислення площі поверхні.
  + Volume() — для обчислення об'єму.
* Також є метод DisplayInfo(), який виводить площу поверхні та об'єм для кожного тіла.

Клас TParallelepiped:

* Це нащадок класу TBody, що представляє прямокутний паралелепіпед.
* Має три властивості: довжину, ширину і висоту.
* Методи SurfaceArea() і Volume() реалізують відповідні формули для площі поверхні і об'єму паралелепіпеда:
  + Площа поверхні: 2×(L×W+W×H+H×L)
  + Об'єм: L×W×H

Клас TBall:

* Це нащадок класу TBody, що представляє кулю.
* Має одну властивість: радіус.
* Методи SurfaceArea() і Volume() реалізують відповідні формули для площі поверхні і об'єму кулі:
  + Площа поверхні: 4×π×r2
  + Об'єм: ×π×r3

Програма-клієнт:

* У програмі створюється масив об'єктів TBody, який містить випадково створені об'єкти класів TParallelepiped і TBall.
* Кількість тіл задається змінною n (в даному випадку 10).
* Випадковим чином створюються або паралелепіпеди, або кулі з випадковими розмірами (довжина, ширина, висота для паралелепіпедів і радіус для куль).
* Потім обчислюється і виводиться сумарна площа поверхні всіх тіл, а також виводяться деталі кожного тіла.

1. **Код програми**
2. using System;
3. public abstract class TBody
4. {
5. // Абстрактні методи для обчислення площі поверхні та об'єму
6. public abstract double SurfaceArea();
7. public abstract double Volume();
8. // Метод для виведення даних про геометричне тіло
9. public void DisplayInfo()
10. {
11. Console.WriteLine($"Surface Area: {SurfaceArea():F2}, Volume: {Volume():F2}");
12. }
13. }
14. public class TParallelepiped : TBody
15. {
16. public double Length { get; set; }
17. public double Width { get; set; }
18. public double Height { get; set; }
19. public TParallelepiped(double length, double width, double height)
20. {
21. Length = length;
22. Width = width;
23. Height = height;
24. }
25. // Перевизначення методу для обчислення площі поверхні прямокутного паралелепіпеда
26. public override double SurfaceArea()
27. {
28. return 2 \* (Length \* Width + Width \* Height + Height \* Length);
29. }
30. // Перевизначення методу для обчислення об'єму прямокутного паралелепіпеда
31. public override double Volume()
32. {
33. return Length \* Width \* Height;
34. }
35. }
36. public class TBall : TBody
37. {
38. public double Radius { get; set; }
39. public TBall(double radius)
40. {
41. Radius = radius;
42. }
43. // Перевизначення методу для обчислення площі поверхні кулі
44. public override double SurfaceArea()
45. {
46. return 4 \* Math.PI \* Radius \* Radius;
47. }
48. // Перевизначення методу для обчислення об'єму кулі
49. public override double Volume()
50. {
51. return (4 / 3.0) \* Math.PI \* Math.Pow(Radius, 3);
52. }
53. }
54. public class Program
55. {
56. public static void Main(string[] args)
57. {
58. Random random = new Random();
59. int n = 10; // Загальна кількість тіл
60. TBody[] bodies = new TBody[n]; // Масив для збереження геометричних тіл
61. // Створення випадкових паралелепіпедів та куль
62. for (int i = 0; i < n; i++)
63. {
64. if (random.Next(2) == 0)
65. {
66. // Створення випадкового паралелепіпеда
67. double length = random.Next(1, 10);
68. double width = random.Next(1, 10);
69. double height = random.Next(1, 10);
70. bodies[i] = new TParallelepiped(length, width, height);
71. }
72. else
73. {
74. // Створення випадкової кулі
75. double radius = random.Next(1, 10);
76. bodies[i] = new TBall(radius);
77. }
78. }
79. // Обчислення сумарної площі поверхні всіх геометричних тіл
80. double totalSurfaceArea = 0;
81. foreach (var body in bodies)
82. {
83. totalSurfaceArea += body.SurfaceArea();
84. }
85. // Виведення результату
86. Console.WriteLine($"Total Surface Area of all bodies: {totalSurfaceArea:F2}");
87. // Виведення даних для кожного тіла
88. for (int i = 0; i < n; i++)
89. {
90. Console.WriteLine($"Body {i + 1}:");
91. bodies[i].DisplayInfo();
92. }
93. }
94. }
95. **Приклад виконання програми**

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Вміст, створений ШІ, може бути неправильним.