Кам’янець-Подільський національний університет імені Івана Огієнка

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

Навчальна дисципліна «Об’єктно-орієнтоване програмування»

ЛАБОРАТОРНА РОБОТА #0103

Тема:

Основи об’єктно-орієнтованого програмування

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**

Наслідування в C# дозволяє одному класу успадковувати поведінку та властивості іншого класу. Це дає змогу створювати ієрархії класів, де базовий клас містить загальну функціональність, а похідні класи можуть розширювати або змінювати її. При наслідуванні можливе перетворення типів: екземпляр похідного класу може зберігатися в змінній базового класу, що корисно для поліморфізму. Однак зворотне перетворення (casting) потребує перевірки, яку можна виконати за допомогою оператора as або is, а також Convert.ChangeType(), якщо це можливо.

Віртуальні методи та властивості дозволяють похідним класам змінювати поведінку базових класів. Метод або властивість позначаються ключовим словом virtual, а в похідному класі перевизначаються за допомогою override. Це дає змогу реалізувати поліморфізм, коли виклик методу здійснюється не на основі типу змінної, а на основі реального типу об'єкта. Віртуальні властивості працюють аналогічно методам, дозволяючи змінювати логіку доступу до даних у похідних класах.

Приховування методів і властивостей реалізується за допомогою ключового слова new, яке вказує, що нова версія методу або властивості приховує базову. На відміну від перевизначення, прихований метод не виконується, якщо об'єкт викликається через змінну базового типу. Це означає, що приховування працює на рівні компіляції, тоді як перевизначення забезпечує динамічне зв’язування (run-time binding).

Абстрактні класи містять хоча б один абстрактний метод або властивість, які не мають реалізації та повинні бути перевизначені в похідних класах. Вони не можуть створювати екземпляри, а лише служать основою для конкретніших класів. System.Object є базовим класом для всіх типів у C# і містить методи, такі як ToString(), Equals(), GetHashCode() та GetType(), які можна перевизначати для зміни поведінки.

Узагальнені (generic) типи дозволяють створювати класи та методи, які працюють із будь-якими типами, зберігаючи безпеку типів і уникаючи зайвого перетворення. Вони підтримують обмеження (where), які дозволяють вказувати, що узагальнений тип повинен реалізовувати певний інтерфейс або успадковувати конкретний клас. Узагальнені класи можуть успадковувати як звичайні, так і інші узагальнені класи, що дає гнучкість у проєктуванні коду.

1. **Задача №1**

Ієрархія працівників. Створіть ієрархію працівників із перевантаженням методів.

1. **Послідовність виконання роботи**
   * Ознайомлення із теоретичними відомостями.
   * Створення нового проекту у Visual Studio Code.
   * Відкриття файлу.
   * Написання коду. Тестування програми та редагування коду.
     + Підключення бібліотеки.
     + Оголошення класу Workers, його полів Name i Position.
     + Ініціалізація цих властивостей при створенні об'єкта.
     + Опис методу Show(), який виводить ім'я та посаду працівника.

Зображення, що містить текст, програмне забезпечення, Мультимедійне програмне забезпечення, Графічний редактор

Вміст, створений ШІ, може бути неправильним.

* + - Клас Director успадковує клас Workers.
    - Конструктор класу Director приймає лише ім'я і передає його в базовий клас Workers разом з посадою "Director".
    - Перевизначення методу Show() у класі Director.
    - Повторення цих дій для класів Manager i Programmer.

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

Вміст, створений ШІ, може бути неправильним.

* + - Опис основного класу Program, методу Main().
    - Створення нових об’єктів, виведення їх на екран за допомогою методу Show().

**Зображення, що містить текст, Мультимедійне програмне забезпечення, програмне забезпечення, Графічний редактор

Вміст, створений ШІ, може бути неправильним.**

1. **Код програми**
2. using System;
3. public class Workers
4. {
5. public string Name { get; set; }
6. public string Position { get; set; }
7. public Workers(string name, string position)
8. {
9. Name = name;
10. Position = position;
11. }
12. public virtual void Show()
13. {
14. Console.WriteLine($"{Name} - {Position}");
15. }
16. }
17. class Director : Workers
18. {
19. public Director(string name) : base(name, "Director") { }
20. public override void Show()
21. {
22. Console.WriteLine($"{Name} - {Position}");
23. }
24. }
25. class Manager : Workers
26. {
27. public Manager(string name) : base(name, "Manager") { }
28. public override void Show()
29. {
30. Console.WriteLine($"{Name} - {Position}");
31. }
32. }
33. class Programmer : Workers
34. {
35. public Programmer(string name) : base(name, "Prorammer") { }
36. public override void Show()
37. {
38. Console.WriteLine($"{Name} - {Position}");
39. }
40. }
41. class Program
42. {
43. static void Main(string[] args)
44. {
45. Workers[] workers = new Workers[4];
46. workers[0] = new Workers("Sofiia", "Accountant");
47. workers[1] = new Director("Andrew");
48. workers[2] = new Manager("Oleksandr");
49. workers[3] = new Programmer("Maria");
50. foreach (Workers w in workers)
51. {
52. w.Show();
53. }
54. Console.ReadKey();
55. }
56. }
57. **Приклад виконання програми**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

1. **Задача №2**

Приховування проти перевизначення. Порівняйте поведінку прихованих і перевизначених методів.

1. **Послідовність виконання роботи**
   * Ознайомлення із теоретичними відомостями.
   * Створення нового проекту у Visual Studio Code.
   * Відкриття файлу.
   * Написання коду. Тестування програми та редагування коду.
     + Підключення бібліотеки.
     + Оголошення класу BankAccount, методу GetAccountType(), який виводить тип рахунку, методу CalculateInterest(), що виводить стандартну процентну ставку для банківських рахунків.

Зображення, що містить текст, програмне забезпечення, Мультимедійне програмне забезпечення, Графічний редактор

Вміст, створений ШІ, може бути неправильним.

* + - Оголошення класу BankAccount.
    - Метод GetAccountType() приховано з базового класу.
    - Метод CalculateInterest() перевизначено з базового класу.

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

Вміст, створений ШІ, може бути неправильним.**

* + - Опис основного класу Program, методу Main().
    - Створення нових об’єктів, виведення інформації про них на екран за допомогою методів GetAccountType() та CalculateInterest().

**Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.**

1. **Код програми**
2. using System;
3. class BankAccount
4. {
5. public void GetAccountType()
6. {
7. Console.WriteLine("Account: General bank account");
8. }
9. public virtual void CalculateInterest()
10. {
11. Console.WriteLine("Interest rate: Standard rate for bank accounts");
12. }
13. }
14. class SavingsAccount : BankAccount
15. {
16. public new void GetAccountType()
17. {
18. Console.WriteLine("Account: Savings account");
19. }
20. public override void CalculateInterest()
21. {
22. Console.WriteLine("Interest rate: Higher rate for savings accounts");
23. }
24. }
25. class Program
26. {
27. static void Main()
28. {
29. BankAccount generalAccount = new BankAccount();
30. generalAccount.GetAccountType();
31. generalAccount.CalculateInterest();
32. SavingsAccount savingsAccount = new SavingsAccount();
33. savingsAccount.GetAccountType();
34. savingsAccount.CalculateInterest();
35. BankAccount referenceAccount = new SavingsAccount();
36. referenceAccount.GetAccountType();
37. referenceAccount.CalculateInterest();
38. }
39. }
40. **Приклад виконання програми**

**Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.**

1. **Задача №3**

Узагальнені обмеження. Реалізуйте узагальнений клас із кількома обмеженнями.

1. **Послідовність виконання роботи**
   * Ознайомлення із теоретичними відомостями.
   * Створення нового проекту у Visual Studio Code.
   * Відкриття файлу.
   * Написання коду. Тестування програми та редагування коду.
     + Підключення бібліотеки.
     + Оголошення Container<T>, що працює з узагальненим типом T і зберігає один елемент цього типу (поле \_item).
     + Обмеження where T : IComparable<T>, що означає, що тип T має підтримувати порівняння між його об'єктами (тобто реалізовувати інтерфейс IComparable<T>).
     + Конструктор приймає об'єкт типу T і зберігає його в полі \_item, метод Compare приймає ще один об'єкт типу T і порівнює його з елементом, який зберігається в контейнері (тобто в полі \_item).

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

Вміст, створений ШІ, може бути неправильним.

* + - Опис класу Person, який представляє людину з властивістю Age.
    - Реалізація інтерфейсу IComparable<Person>, який має перевизначити метод CompareTo, який порівнює два об'єкти типу Person.
    - Метод CompareTo порівнює двох осіб за віком. Якщо вік першої особи більший за іншу, повертається значення більше за 0. Якщо менший − значення менше за 0. Якщо рівні − значення 0.

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

Вміст, створений ШІ, може бути неправильним.

* + - Опис основного класу Program, методу Main().
    - Створення нових об’єктів класу Person, заповнення їх полів.
    - Створення об'єкту класу Container<Person>.
    - Виклик методу Compare, що порівнює person1 з person2.

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

1. **Код програми**
2. using System;
3. public class Container<T> where T : IComparable<T>
4. {
5. private T \_item;
6. public Container(T item) // Конструктор для встановлення елемента
7. {
8. \_item = item;
9. }
10. public void Compare(T otherItem) // Порівняння елемента з іншим
11. {
12. int result = \_item.CompareTo(otherItem);
13. if (result < 0)
14. {
15. Console.WriteLine("The first item is less than the second.");
16. }
17. else if (result > 0)
18. {
19. Console.WriteLine("The first item is greater than the second.");
20. }
21. else
22. {
23. Console.WriteLine("The items are equal.");
24. }
25. }
26. }
27. public class Person : IComparable<Person>
28. {
29. public int Age { get; set; }
30. public Person(int age)
31. {
32. Age = age;
33. }
34. public int CompareTo(Person other) // Порівняння людей за віком
35. {
36. return this.Age.CompareTo(other.Age);
37. }
38. }
39. class Program
40. {
41. static void Main()
42. {
43. Person person1 = new Person(30); // Створення двох об'єктів типу Person
44. Person person2 = new Person(25);
45. Container<Person> container = new Container<Person>(person1); // Створення контейнера для person1
46. container.Compare(person2); // Порівняння person1 з person2
47. }
48. }
49. **Приклад виконання програми**

**Зображення, що містить текст, знімок екрана, Шрифт, чорний

Вміст, створений ШІ, може бути неправильним.**