Тема 4. Пізнє і раннє зв'язування методів класів. Абстрактні класи

1. **Віртуальні методи. Пізнє і раннє зв'язування об'єктів класу**
2. **Абстрактні класи і методи**
3. **Приховані класи.**

Рекомендована література:

Базова

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд.: Пер. с англ. / Г. Буч, Р. А. Максимчук, М. У. Энг и др. – М.: Вильямс, 2008. – 720 с.
2. Брила А. Ю. Основи об’єктно-орієнтованого програмування у C#. Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності "Прикладна математика" / [А. Ю. Брила, П. П. Антосяк, М. І. Глебена та ін.]. – Ужгород, 2014. – 73 с.
3. Васильев А. C#. Объектно-ориентированное программирование. Учебный курс / Алексей Васильев. – СПб.: Питер, 2012. – 320 с.
4. Стиллмен Э. Изучаем C#. 3-е изд / Эндрю Стиллмен, Дженнифер Грин. – СПб.: Питер, 2014. – 816 с.
5. Троелсен Э. Язык программирования C# 6.0 и платформа .NET 4.6. 7-е изд. / Эндрю Троелсен, Филипп Джепикс. – СПб.: Диалектика-Вильямс, 2018. – 1440 с.
6. Албахари Д. C# 6.0. Справочник. Полное описание языка: Пер. с англ. / Джозеф Албахари, Бен Албахари. – М.: Вильямс, 2018. – 1040 с.
7. C#. Спецификация языка. Версия 5.0 / Microsoft Corporation, 2012. – 577 с.
8. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#. Мастер-класс: Пер. с англ. / Дж. Рихтер. – СПб.: Питер, 2007. – 656 с.

Допоміжна

9. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд.: Пер. с англ. / Г. Буч. – М.: "Издательство Бином" 1998. – 560 с.
10. Кнут Д. Е. Искусство программирования для ЭВМ. Т. 3: Сортировка и поиск / Д. Е. Кнут. – 2-е изд. – М.: Вильямс, 2008. – 824 с.
11. Жарков В. А. Самоучитель Жаркова по Visual Studio .Net: Visul Basic .NET, Visual C# .NET, Visual C++ .NET, Visual J# .NET / В. А. Жарков. – М.: Жарков Пресс, 2002. – 592 с.
12. Климов Л. П. C#. Советы программистам / Л. П. Климов. – СПб.: БХВ-Петербург, 2008. – 544 с.
13. Задачи по программированию / С. А. Абрамов,, Г. Г. Гнездилова, Е. Н. Капустина и др.. – М.: Наука, 2000. – 596 с.

1. Віртуальні методи. Пізнє і раннє зв'язування об'єктів класу

При **ранньому зв'язуванні** (на етапі проектування програми) програма є структурою, логіка виконання якої жорстко визначена. Якщо ж потрібно щоб рішення про те, який з однойменних методів різних об'єктів ієрархії використовувати, приймалося залежно від конкретного об'єкту, для якого виконується виклик, то заздалегідь жорстко пов'язувати ці методи з останньою частиною коду не можна.

Отже, треба якимсь чином дати знати компілятору, що ці методи оброблятимуться по-іншому. Для цього в C# існує ключове слово *virtual.* Воно записується в заголовку методу *базового класу*, наприклад:

virtual public void Passport() ...

Оголошення методу віртуальним означає, що всі посилання на цей метод будуть визначатися у момент його виклику під час виконання програми. Цей механізм називається **пізнім зв'язуванням**.

При визначенні віртуального методу у складі базового класу перед типом значення, що повертається, указується ключове слово **virtual** а при перевизначенні віртуального методу в похідному класі використовується модифікатор **override**. Віртуальний метод не може бути визначений з модифікатором static або abstract.

Перевизначений віртуальний метод повинен мати такий самий набір параметрів, як і однойменний метод базового класу.

Приклад 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Person
{
    class Person
    {
        public string Name; //ім'я
        public int Age;     // вік
        public string Role;       // роль
        public string GetName() { return Name; }
        public virtual string GetRole(int Age) { return Role; }
    }
    class Student : Person
    {
        public string Facultet;
        public string Group;
        public int Course;
        public double Rating;
        public Student(string N, int A, string R, string F, string G, int C)
        {
            //конструктор з параметрами
            Name = N;
            Age = A;
            Role = R;
            Facultet = F;
            Group = G;
            Course = C;
```

```
        }
        public override string GetRole(int Course)
        {
            if (Course <= 4)
                Role = "бакалавр";
            else
                Role = "магістр";
            return Role;
        }
        public void Student_Rating(double Rating)
        {
            if (Rating >= 82)
                Console.WriteLine("Привіт відмінникам");
            else
                if (Rating <= 45)
                    Console.WriteLine("Перездача! Треба краще вчитися!");
                else
                    Console.WriteLine("Можна  вчитися ще краще!");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            string r;
            string newRole;
            //дані рейтингу
            Student newSt = new Student("Іванов", 20, "студент", "КННІ", "К-
61", 4);
            Console.WriteLine("Ваш рейтинг?");
            r = Console.ReadLine();
            newSt.Rating = Convert.ToDouble(r);
            newSt.Student_Rating(newSt.Rating);
            newRole = newSt.GetRole(newSt.Course);
            Console.WriteLine("Прізвище = " + newSt.Name);
            Console.WriteLine("Вік= " + newSt.Age);
            Console.WriteLine("Роль= " + newSt.Role);
            Console.WriteLine("Факультет = " + newSt.Facultet);
            Console.WriteLine("група= " + newSt.Group);
            Console.WriteLine("курс= " + newSt.Course);
            Console.ReadLine();
        }
    }
}
```

Примітка

Використання віртуальних методів ускладнює тестування класу.

Віртуальні методи базового класу визначають інтерфейс всієї ієрархії. Цей інтерфейс може розширюватися в нащадках за рахунок додавання нових віртуальних методів.

Перевизначати віртуальний метод в кожному з нащадків не обов'язково: якщо він виконує дії, що влаштовують нащадка, метод успадковується.

За допомогою віртуальних методів реалізується один з основних принципів об'єктно-орієнтованого програмування — поліморфізм. Це слово в перекладі з грецького означає "багато форм", що в даному випадку означає "один виклик — багато методів".

Віртуальні методи незамінні і при передачі об'єктів в методи як параметри. У параметрах методу описується об'єкт базового типа, а при

виклику в нього передається об'єкт похідного класу. Віртуальні методи, що викликаються для об'єкту з методу, відповідатимуть типу аргументу, а не параметра.

2. Абстрактні класи і методи

При створенні ієрархії об'єктів для виключення коду, що повторюється, часто буває логічно виділити їх загальні властивості в один базовий клас. При цьому може виявитися, що створювати екземпляри такого класу не має сенсу, тому що жодні реальні об'єкти їм не відповідають.

Такі класи називають абстрактними.

В абстрактному класі визначаються лише загальні призначення методів, які повинні бути реалізовані в похідних класах, але сам по собі цей клас не містить реалізації методів, а тільки їх сигнатуру (тип значення, що повертається, ім'я методу і список параметрів).

При оголошенні абстрактного методу використовується модифікатор *abstract*. Абстрактний метод автоматично стає віртуальним, так що модифікатор virtual при оголошенні методу не використовується.

Абстрактний клас призначений тільки для створення ієрархії класів, не можна створити *об'єкт абстрактного класу.*

Якщо в класі є хоч би один абстрактний метод, весь клас також має бути описаний як абстрактний, наприклад:

Приклад 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Person
{
    abstract class Person
    {
        public string Name; //имя
        public int Age;  // возраст
        public string Role;       // роль
        public string GetName() { return Name; }
        abstract public string GetRole(int Course);
    }
    class Student : Person
    {
        public string Facultet;
        public string Group;
        public int Course;
        public double Rating;
        public Student(string N, int A, string R, string F, string G, int C)
        {
            //конструктор з параметрами
            Name = N;
            Age = A;
            Role = R;
            Facultet = F;
            Group = G;
            Course = C;
```

```
        }
        public override string GetRole(int Course)
        {
            if (Course <= 4)
                Role = "бакалавр";
            else
                Role = "магістр";
            return Role;
        }
        public void Student_Rating(double Rating)
        {
            if (Rating >= 82)
                Console.WriteLine("Привіт відмінникам");
            else
                if (Rating <= 45)
                    Console.WriteLine("Перездача! Треба краще вчитися!");
                else
                    Console.WriteLine("Можна  вчитися ще краще!");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            string r;
            string newRole;
            //дані рейтингу
            Student newSt = new Student("Іванов", 20, "студент", "КННІ", "К-
81", 4);
            Console.WriteLine("Ваш рейтинг?");
            r = Console.ReadLine();
            newSt.Rating = Convert.ToDouble(r);
            newSt.Student_Rating(newSt.Rating);
            newRole = newSt.GetRole(newSt.Course);
            Console.WriteLine("Прізвище = " + newSt.Name);
            Console.WriteLine("Вік= " + newSt.Age);
            Console.WriteLine("Роль= " + newSt.Role);
            Console.WriteLine("Факультет = " + newSt.Facultet);
            Console.WriteLine("група= " + newSt.Group);
            Console.WriteLine("курс= " + newSt.Course);
            Console.ReadLine();
        }
    }
}
```

3. Приховані (закриті) класи

В C# можна визначати класи і методи як sealed (закриті). У випадку класу це означає, що ви не можете успадковувати від нього. У випадку методу це означає, що його не можна перевизначити.

```
sealed class FinalClass
{
  //тіло класу
}
class DerivedClass : FinalClass  //неправильно. Помилка компіляції
{
  //тіло класу
}
```

Більшість вбудованих типів даних описана як sealed. Якщо необхідно використовувати функціональність такого класу, застосовується не спадкоємство, а включення: у класі описується поле відповідного типу.

Включення класів, коли один клас включає поля, що є класами, є альтернативою спадкоємству при проектуванні. Наприклад, якщо є об'єкт "двигун", а потрібно описати об'єкт "літак", логічно зробити двигун полем цього об'єкту, а не його предком.

Висновки

Спадкоємство – важлива риса об'єктно-орієнтованого програмування. Воно призначене для відображення такої риси програмних систем як ієрархічність. Створення ієрархії класів дозволяє розширювати функціональні можливості класів в похідних класах, а також повторно використовувати код методів базових класів.

На відміну від C++, у C# заборонено множинне спадкоємство, тобто похідний клас (нащадок) може мати тільки один базовий клас (предок).

При спадкоємстві конструктори класів не успадковуються, тому похідний клас повинен мати власні конструктори.

Якщо конструктор базового класу має параметри, він має бути явним чином викликаний в конструкторі похідного класу за допомогою ключового слова **base.**

Віртуальні методи є ще однією рисою поліморфізму, реалізованому в C#. На відміну від перегружених методів, віртуальні методи повинні мати однакові списки параметрів. Віртуальні методи надають можливості пізнього зв'язування об'єктів (під час виконання).

Питання для самоконтролю

1. Які ключові слова використовуються при перевизначенні методів базового класу в похідному?
2. Опишіть механізми раннього і пізнього зв'язування.
3. Опишіть процес виклику віртуального методу. Для чого застосовуються віртуальні методи?
4. Чи всі методи слід описувати як віртуальні?
5. Для чого використовуються абстрактні класи?