Тема 3. Спадкоємство класів як спадкоємство реалізацій

1. **Реалізація наслідування класів.**
2. **Організація ієрархії класів.**
3. **Використання в похідному класі конструктора базового класу з параметрами.**

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. На початку вивчення дисципліни нами бук розроблений клас *Паралелограм* з конструктором, доступними процедурами визначення площі, периметра та друку його даних. Створимо в цьому класі ще один **конструктор без параметрів**, який буде ініціалізувати поля об'єкта значеннями по замовчуванню:

```
class Parallelogram
 {private double a, b, alfa;

  public Parallelogram()
   {a = 5; b = 4; alfa = 60;
    Info();
   }

  public Parallelogram(double a, double b, double alfa)
   {this.a = a; this.b = b; this.alfa = alfa;
    Info();
   }

  public double area()
   {return a * b * Math.Sin(alfa / 180 * Math.PI);
   }

  public double perimeter()
   {return 2 * (a + b);
   }

  public void Info()
   {MessageBox.Show("Дані паралелограма:\nдві сторони по "+a.ToString()+
       " та дві по "+b.ToString()+" од.;\nплоща: " + area().ToString() +
       " кв. од.;\nпериметр: " + perimeter().ToString() +
       " од.;\nдва кути по " + alfa.ToString() + " і два кути по " +
       (180 - alfa).ToString() + " градусів","Інформація",
       MessageBoxButtons.OK,MessageBoxIcon.Information);
   }
 }
```

Після цього, якщо в програмі записати

```
Parallelogram p1 = new Parallelogram();    ,
```

то буде створений об'єкт-паралелограм, ініціалізований значеннями по замовчуванню.

Опишемо тепер клас квадрата, як паралелограма з однаковими сторонами і кутами:

```
class Square : Parallelogram
 {public Square(double sa) : base(sa, sa, 90)
   { }

  public void Info()
   {MessageBox.Show("Дані квадрата:\nчотири сторони по "+
       Math.Sqrt(area()).ToString()+" од.;\nплоща " + area().ToString() +
       " кв. од.;\nпериметр " + perimeter().ToString() +
       " од.;\nчотири кути по 90 градусів");
   }
 }
```

При наслідуванні породжений клас автоматично отримує всі члени батька і може ще й бути розширений власними членами.

2. Спадкоємство є основною концепцією об'єктно-орієнтовного програмування (ООП). З його допомогою створюється ієрархія класів. Спадкоємство використовується для розширення функціональних можливостей класу. При цьому, як було показано вище, похідний клас успадковує усі методи і властивості базового класу.

На відміну від C++, у C# заборонено *множинне спадкоємство*, тобто клас може успадковувати властивості і методи тільки від одного базового класу (предка).

Множинне спадкоємство можна реалізувати за допомогою *інтерфейсів*.

Таким чином, в C# є два типи спадкоємства: *спадкоємство реалізації* і *спадкоємство інтерфейсів*.

Розглянемо спадкоємство реалізації.
Синтаксис спадкоємства:
class ім'я_класу : им'я_батьківського класу
{тіло_класу}

Приклад 1

Розглянемо спадкоємство класів на прикладі. Створимо клас Person та похідний клас Student.

```
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; }
    }
    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 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();
        }
    }

      }
```

Увага!

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

В цьому прикладі в класі Student успадковуються поля базового класу і визначається конструктор з параметрами. В базовому класі конструктора не має (за замовчанням створюється конструктор без параметрів).

Якщо в базовому класі конструктор **без параметрів**, то все буде добре.

Проблеми з успадкуванням конструкторів стосуються визначення і ініціалізації конструктора в похідному класі.

Порядок виклику конструкторів визначається наведеними нижче правилами.

- Якщо в конструкторі похідного класу явний виклик конструктора базового класу відсутній, автоматично викликається конструктор базового класу без параметрів.
- Для ієрархії, що складається з декількох рівнів, конструктори базових класів викликаються, починаючи з самого верхнього рівня. Після цього виконуються конструктори тих елементів класу, які є об'єктами, в порядку їх оголошення в класі, а потім виконується конструктор класу. Таким чином, кожен конструктор ініціалізує свою частину об'єкту.
- Якщо конструктор базового класу вимагає вказівки параметрів, він має бути явним чином викликаний в конструкторі похідного класу в списку ініціалізації. Виклик виконується за допомогою ключового слова **base**. Викликається та версія конструктора, список параметрів якої відповідає списку аргументів, вказаних після слова **base.**

3. Передача управління конструктору базового класу здійснюється за допомогою конструкції

...(...):base(...){...},

яка розташовується в оголошенні конструктора похідного класу між заголовком конструктора і тілом. Після ключового слова base в дужках розташовується список значень параметрів конструктора базового класу. Очевидно, що вибір відповідного конструктора визначається типом значень в списку.

Для створення об'єктів можна застосовувати конструктори трьох ступенів захисту:

public – при створенні об'єктів в рамках даного простору імен, в методах будь-якого класу — члена даного простору імен;

protected – при створенні об'єктів в рамках похідного класу, у тому числі при побудові об'єктів похідного класу, а також для внутрішнього використовування класом — власником даного конструктора;

private – застосовується виключно для внутрішнього використовування класом-власником даного конструктора. – не успадковується.

Приклад 2. Спадкоємство конструктора з параметрами.

Спадкоємство конструктора базового класу.

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

namespace Person
{
    class Person
    {
        public string Name; //ім'я
        public int Age;  // вік
        public string Role; // роль
        public Person(string N, string R, int A)
        {
            Name = N;
            Age = A;
            Role = R;
        }
        public string GetName() { return Name; }
        public string GetRole() { 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):
base (N, R, A)

        {
            //конструктор з параметрами
            Name = N;
            Age = A;
```

```
            Role = R;
            Facultet = F;
            Group = G;
            Course = C;

        }
        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)
        {
            //дані рейтингу
            Student newSt = new Student("Іванов", 20, "студент", "КННІ", "К-
61", 4);
            Console.WriteLine("Ваш рейтинг?");
            string r = Console.ReadLine();
            newSt.Rating = Convert.ToDouble(r);
            newSt.Student_Rating(newSt.Rating);
            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();
        }
    }
}
```

Елементи базового класу, визначені як private, в похідному класі недоступні. Тому для доступу до полів класу Person вони повинні визначатися як public чи protected.

Питання для самоконтролю

1. Для чого використовується спадкоємство?
2. Який синтаксис опису похідного класу?
3. Які специфікатори доступу застосовуються в ієрархіях?
4. Як викликати метод базового класу з похідного?
5. Опишіть порядок виклику конструкторів базових класів при роботі конструктора похідного класу.