Лабораторна робота №5

Тема: Спадкування класів в C#. Ієрархія класів.

Мета: Вивчити особливості роботи з класами та їх спадкуванням у C#.

Завдання: Здобути навички створення, відлагодження та тестування проектів обробки даних з використанням спадкування класів мовою ***C#***.

(max: 5 балів)

Теоретичні відомості:

1. Організація ієрархії класів

Спадкоємство (наслідування)є однією з основних концепцій об'єктно-орієнтовного програмування (ООП). З його допомогою створюється ієрархія класів. Спадкоємство використовується для розширення функціональних можливостей класу. При цьому 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",
"КСІТ", "К-401", 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.**

2. Використання в похідному класі конструктора базового класу з параметрами

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",
"КСІТ", "К-401", 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`.

3. 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`.

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

Приклад 3.

```
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 Course) { 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",
"КСІТ", "К-401", 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.

4. 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 4.

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

namespace Person
{
    abstract class Person
    {
        public string Name; //ім'я
        public int Age;  // вік
        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",
"КСІТ", "К-401", 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();
        }
    }
}
```

5. 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.

6. Види відношень між класами

Механізм спадкоємства класів надає великі можливості організації коду і його багатократного використання. Вибір найбільш відповідних засобів для цілей конкретного проекту ґрунтується на знанні механізму їх роботи і взаємодії.

Коли потрібно використовувати спадкоємство? Це залежить від різних причин, обумовлених вирішуваною задачею, наприклад:

- Спеціалізація. Клас-нащадок є спеціалізованою формою базового класу — в нащадку просто перевизначаються методи.
- Специфікація. Клас-нащадок реалізує поведінку, описану в батьківському класі. У C# ця форма реалізується спадкоємством від абстрактного класу.
- Конструювання. Клас-нащадок використовує методи базового класу, але не є його підтипом.
- Розширення. У клас-нащадок додають нові методи, розширюючи поведінку батьківського класу.
- Узагальнення. Клас-нащадок узагальнює поведінку базового класу. Звичайно таке спадкоємство використовується в тих випадках, коли змінити поведінку базового класу неможливо (наприклад, базовий клас є бібліотечним класом).
- Обмеження. Клас-нащадок обмежує поведінку батьківського класу.
- Комбінування. Клас-нащадок успадковує риси декількох класів — це множинне спадкоємство (у C# не використовується, оскільки множинне спадкоємство заборонене, а спадкоємство від декількох інтерфейсів має інший сенс).

Альтернативою спадкоємству при проектуванні ієрархії класів є **вкладення**, коли один клас включає поля, які самі є класами. Наприклад, якщо є клас "**двигун**", а потрібно описати клас "літак", логічно зробити двигун полем цього класу, а не його предком. Вкладення представляє відношення класів "Y містить X" або "Y реалізується за допомогою X" і зазвичай реалізується за допомогою моделі "включення-делегування". Приклад 5. ілюструє цю можливість.

Приклад 5

```
using System;
namespace ConsoleApplication1
{
    class Двигун
    {   public void Запуск()
        {
            Console.WriteLine( "вжжжж!!" );
        }
    }
    class Літак
    {  public Літак ()
       {
            лівий  = new Двигун();
            правий = new Двигун();
        }
        public void Запустити_двигуни()
        {
            лівий.Запуск();
            правий.Запуск();
        }
        Двигун лівий, правий;
```

```
        }

        class Class1
        {   static void Main()
            {
            Літак АН24_1 = new Літак();
            АН24_1. Запустити_двигуни ();
            }
        }
    }
```

7. Клас object – базовий клас ієрархії класів C#

Кореневий клас `System.Object` всієї ієрархії об'єктів .NET, який має в C# назву **`object`**, забезпечує всіх нащадків кількома важливими методами. Похідні класи можуть використовувати ці методи безпосередньо або перевизначати їх.

Клас `object` часто використовується і безпосередньо при описі типу параметрів методів для додавання їм універсальності, а також для зберігання посилань на об'єкти різних типів — таким чином реалізується **поліморфізм**.

Відкриті методи класу `System.Object` перелічені нижче.

1. Метод **`Equals`** з одним параметром повертає значення true, якщо параметр і об'єкт, який його викликає посилаються на одну і ту саму ділянку пам'яті. Синтаксис:

```
public virtual bool Equals(object obj);
```

2. Метод **`Equals`** з двома параметрами повертає значення true, якщо обидва параметри посилаються на одну і ту саму ділянку пам'яті. Синтаксис:

```
public static bool Equals(object ob1, object ob2);
```

3. Метод **`GetHashCod`**e формує хеш-код об'єкту і повертає число, що однозначно ідентифікує об'єкт. Це число використовується в різних структурах і алгоритмах бібліотеки. Якщо перевизначається метод `Equals`, необхідно перенавантажувати і метод `GetHashCode`. Синтаксис:

```
public virtual int GetHashCode();
```

4. Метод **`GetType`** повертає поточний поліморфний тип об'єкту, тобто не тип посилання, а тип об'єкту, на який посилання вказує в даний момент. Значення, що повертається має тип `Type`. Це абстрактний базовий клас ієрархії, що використовується для отримання інформації про типи під час виконання. Синтаксис:

```
public Type GetType();
```

5. Метод **`ReferenceEquals`** повертає значення true, якщо обидва параметри посилаються на одну і ту саму ділянку пам'яті. Синтаксис:

```
public static bool ReferenceEquals(object ob1, object ob2);
```

6. Метод **`ToString`** за замовчанням повертає для посилкових типів повне ім'я класу у вигляді рядка, а для значимих — значення величини, перетворене в рядок. Цей метод перевизначають для того, щоб можна було виводити інформацію про стан об'єкту. Синтаксис:

```
public virtual string ToString()
```

У похідних об'єктах ці методи часто перевизначають. Наприклад, можна перевизначити метод `Equals` для того, щоб задати власні критерії порівняння об'єктів.

Індивідуальні завдання:

(Для виконання індивідуальних завдань № варінта є порядковим номером прізвища студента в списку групи. Усі проекти завантажити у власні репозиторії на Git Hub та виконати їх збірку/build у середовищі Travis CI. Посилання на репозиторії проектів вказати у звіті та обов'язково долучати скріни успішної збірки в Travis CI.)

1. Розробити консольний застосунок для роботи з базою даних, що зберігається у текстовому файлі (початковий масив не менше 5 записів). Структура бази даних описується ієрархією класів згідно вашого варіанта. Для ідентифікації спроби введення з клавіатури некоректних даних описати виключення. Реалізувати методи <u>у базовому класі</u> для:

– додавання записів;

– редагування записів;

– знищення записів;

– виведення інформації з файла на екран;

– обчислення та виведення на екран результатів згідно свого варіанта індивідуального завдання.

Меню програми реалізувати по натисненню на певні клавіші: наприклад, Enter – вихід, п - пошук, р – редагування тощо.

Один з методів індивідуального завдання зробити віртуальним.

(3 бала)

Варіант №	Батьківський (базовий) клас		Похідний клас		Реалізувати з допомогою окремих методів обчислення та виведення на екран таких даних:
	Сутність	Обов'язкові поля	Сутність	Обов'язкові поля	
1.	Навчальний курс	Назва, наявність іспиту	Практичне заняття	Дата, тема, кількість студентів	Середня кількість студентів, заняття з максимальною кількістю студентів, список тем з певним словом у назві
2.	Трамвайна зупинка	Назва, список номерів маршрутів	Година	Кількість пасажирів, коментар	Загальна кількість пасажирів, година з найменшою кількістю пасажирів, найдовший коментар
3.	Навчальний курс	Назва, прізвище викладача	Лекція	Дата, тема, кількість студентів	Лекція з мінімальною кількістю студентів, список тем з певним словом у назві, остання літера у прізвищі викладача
4.	Конференція	Назва, місце проведення	Засідання	Дата, тема, кількість учасників	Середня кількість учасників на засіданні, засідання з найбільшою кількістю учасників, довжина назви
5.	Виставка	Назва, прізвище художника	День	Кількість відвідувачів, коментар	Сумарна кількість відвідувачів, день з найменшою кількістю відвідувачів, список коментарів з певним словом
6.	Станція метрополітену	Назва, рік відкриття	Година	Кількість пасажирів, коментар	Сумарна кількість пасажирів, години з найменшою кількістю пасажирів та найбільшою кількістю слів у коментарі
7.	Лікар	Прізвище, фах	Прийом	День, зміна, кількість відвідувачів	Загальна кількість відвідувачів, прийом з мінімальною кількістю відвідувачів, довжина прізвища
8.	Поет	Прізвище, мова, кількість збірок	Виступ	Дата, місце, кількість слухачів	Сумарна кількість слухачів, день з найбільшою кількістю слухачів, довжина прізвища
9.	Лікар	Прізвище, стаж	Прийом	День, кількість відвідувачів, коментар	Середня кількість відвідувачів, прийом з мінімальною кількістю відвідувачів, найдовшим коментарем
10.	Трамвайний маршрут	Номер, середній інтервал руху	Зупинка	Назва, кількість пасажирів	Загальна кількість пасажирів, зупинки з найменшою кількістю пасажирів, найдовшою назвою
11.	Цілодобовий кіоск	Назва, адреса	Година	Кількість покупців, коментар	Загальна кількість покупців, година з найменшою кількістю покупців, коментарями з певними словами
12.	Виконавець	Прізвище, жанр	Концерт	Дата, кількість глядачів	Загальна кількість глядачів, концерт з максимальною кількістю глядачів, кількість слів у назві жанру

13.	Музичний гурт	Назва, прізвище керівника	Гастрольна поїздка	Місто, рік, кількість концертів	Гастрольна поїздка з максимальною кількістю концертів, список гастрольних поїздок у певне місто, остання літера в прізвищі керівника
14.	Навчальний курс	Назва курсу, назва кафедри	Лекція	Дата, група, кількість студентів	Середня кількість студентів, лекції з найбільшою кількістю студентів, кількість слів у назві кафедри
15.	Виставка	Назва, прізвище скульптора	День	Кількість відвідувачів, коментар	Сумарна кількість відвідувачів, день з найбільшою кількістю відвідувачів, день з найбільшою кількістю слів у коментарі
16.	Письменник	Прізвище, мова, кількість книжок	Виступ	Дата, місце, кількість слухачів	Сумарна кількість слухачів, день з найменшою кількістю слухачів, довжина прізвища
17.	Співробітник	ПІБ, посада	Робочій день	Дата, кількість годин, назва проекту	Середня кількість робочих годин за період; Кількість годин на проекті; Дні з максимальним навантаженням
18.	Лікар	ПІБ, спеціальність	Робочій день	Дата, кількість пацієнтів, час початку роботи	Середня кількість пацієнтів в день за період; Кількість днів з максимальним навантаженням; Дні, коли починав приймати після зазначеного часу
19.	Піцерія	Назва, адреса	Робочій день	Дата, кількість замовлень, піца дня	Середня кількість замовлень в день за період; Дні з максимальним відвідуванням; Сумарна кількість замовлень для днів з визначеною піцою дня
20.	Басейн	Назва, адреса	Робочій день	Дата, кількість відвідувачів, кількість доступних доріжок	Середня кількість відвідувань в день за період; Дні з мінімальною кількістю доступних доріжок; Кількість днів, коли було доступно не менше зазначеної кількості доріжок
21.	Бібліотека	Назва, адреса	Робочій день	Дата, кількість книг, що видано, кількість книг, що повернуто	Середній рух книжок в день за період; Кількість днів, коли було видано книг більше, ніж повернуто; Дні, коли видана парна кількість книг, а повернута – непарна
22.	Сайт	Назва, URL	Відвідування	Дата, кількість унікальних хостів, кількість завантажених сторінок	Середня кількість хостів в день за період; Дні з максимальною кількістю завантажених сторінок; Кількість днів, коли співвідношення хостів до сторінок перевищує задане значення
23.	Акаунт електронної пошти	E-mail, ПІБ володаря	Спам	Дата, кількість спам повідомлень, загальна кількість повідомлень	Середня кількість спаму в день за період; Кількість днів, коли відсоток спам повідомлень був менший за задане значення; Дні, коли кількість спаму збільшувалась
24.	Акція компанії на біржі	Назва компанії, код на біржі	Курс	Дата, курс відкриття, курс закриття	Середня вартість акцій по закриттю за період; Кількість днів, коли курс зростав протягом дня; Дні, коли зміна курсу за день перевищувала задане значення
25.	Телефонний номер	Номер, оператор	Дзвінки	Дата, кількість хвилин розмов, кошти, що використано на розмови	Середня платня в день за період; Кількість днів, коли вартість хвилини розмови перевищувала задане значення; Дні, коли кількість хвилин розмов була парна

2. Модифікувати консольний застосунок з попереднього завдання таким чином, щоб:
А) Базовий клас був абстрактним.
Б) Реалізація методів індивідуального завдання була перенесена у похідний клас.
(1 бал)

3. Створити модульний тест до консольного застосунку з першого завдання для перевірки коректності роботи <u>одного</u> (на ваш вибір) з методів, створених до індивідуального завдання.
(1 бал)