Лабораторна робота №6

Тема: Інтерфейси в C#.

Мета: Вивчити особливості роботи з інтерфейсами у C#.

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

(max: 5 балів)

Теоретичні відомості:

1. Інтерфейс як окремий випадок абстрактного класу

Інтерфейс це повністю абстрактний клас, всі методи якого абстрактні (не містять реалізації). Від абстрактного класу інтерфейс відрізняється деякими деталями в синтаксисі і поведінці.

Синтаксична відмінність полягає в тому, що методи інтерфейсу оголошуються без вказівки модифікатора доступу.

Відмінність в поведінці полягає в жорсткіших вимогах до похідних класів - в похідних класах повинні бути реалізовані всі методи інтерфейсного класу.

Це означає, що клас, який наслідує інтерфейс, зобов'язаний повністю реалізувати всі методи інтерфейсу. У цьому відмінність від класу, що наслідує абстрактний клас, де похідний клас може реалізувати лише деякі методи базового абстрактного класу.

Головне призначення інтерфейсів – забезпечення множинного наслідування класів.

Є ще одне важливе призначення інтерфейсів, що відрізняє їх від абстрактних класів. Абстрактний клас є початковим етапом проектування класу, який в майбутньому отримає конкретну реалізацію. Інтерфейси задають додаткові властивості класу. Один і той самий інтерфейс дозволяє описувати властивості, які можуть мати різні класи.

Кожен клас може визначати елементи інтерфейсу по-своєму. Так досягається поліморфізм: об'єкти різних класів по-різному реагують на виклики одного і того ж методу.

Синтаксис інтерфейсу аналогічний синтаксису класу:

[атрибути] [специфікатори] interface ім'я_інтерфейсу [: предки]
тіло_інтерфейсу [;]

Для інтерфейсу можуть бути вказані специфікатори `new`, `public`, `protected`, `internal` і `private`. Специфікатор `new` застосовується для вкладених інтерфейсів і має такий самий сенс, як і відповідний модифікатор методу класу. Інші специфікатори управляють видимістю інтерфейсу. За замовчанням інтерфейс доступний лише із збірки, в якій він описаний (internal).

Інтерфейс може успадковувати методи декількох інтерфейсів, в цьому випадку предки перераховуються через кому.

Тіло інтерфейсу складають абстрактні методи, шаблони властивостей і індексаторів, а також події.

2. Стратегії реалізації інтерфейсу

Опишемо деякий інтерфейс, реалізація методів якого дозволить класу робити певні перетворення над об'єктами класу і повертати рядки, що представляють результат цих перетворень.

```
interface IStrings
 {
   /// <summary>
   /// Перетворення
   /// </summary>
   /// <returns> результат перетворення </returns>
   string Convert();

   /// <summary>
   /// Шифрування
   /// </summary>
   /// <param name="code"> код шифру </param>
   /// <returns> результат шифрування</returns>
   string Cipher(string[] code);
 }
```

В цьому інтерфейсі є два методи, які будуть реалізувати всі класи, які будуть наслідувати цей інтерфейс. Метод `Convert` повинен, слідуючи алгоритму, вибраному похідним класом, перетворити об'єкт, повертаючи рядок, а метод `Chiper`, що повертає рядок, розглядається як шифрування, в алгоритмі якого використовується масив рядків code, переданий методу.

Загалом існує дві стратегії реалізації методів інтерфейсу: як відкритих методів і як закритих. Ми розглянемо тільки першу стратегію, яка використовується найчастіше.

Клас, що успадковує інтерфейс і реалізує його методи, може оголосити відповідні методи класу **відкритими (public)**. У методів інтерфейсу не задані модифікатори доступу. Побудуємо приклад класу, що успадковує інтерфейс `IStrings`:

Приклад 1.

```
    /// <summary>
    /// Успадковує інтерфейс IStrings
    /// реалізуючи його методи як відкриті (public)
    /// </summary>
    class SimpleText : IStrings
    {
   //поля класу
   string text;
   static string[] codeTable =
   {
     "абвгдеёжзийклмнопрстуфхцчшщыьъэюя ,.!?;:",
     "ъьыщшчцхфуэюя ,.!?;:тсрпонмлкйабвгдеёжзи"
   };
   //Конструктори
   public SimpleText()
   {
       text = "Простий текст!";
   }
   public SimpleText(string txt)
   {
       text = txt;
   }
   public string Text
   {
       get { return text; }
   }
}
```

Побудований клас є звичайним класом, що містить текстове поле `text`, 2 конструктори, метод - властивість, що забезпечує доступ до закритого поля. Але оскільки клас оголосив себе спадкоємцем інтерфейсу `IStrings`, він зобов'язаний реалізувати методи інтерфейсу, запропонувавши деяку їх реалізацію. Ось приклад подібної реалізації:

```
   /// Реалізація інтерфейсів
   /// <summary>
   /// Видалення пробілів в полі text
   /// перетворення до нижнього регістра
   /// </summary>
   /// <returns>  перетворений рядок </returns>
   public string Convert()
   {
      string res = "";
       foreach (char sym in text)
      if (sym != ' ') res += sym.ToString();
       res = res.ToLower();
       return res;
   }
   /// <summary>
   /// шифрування поля text
   /// з використанням таблиці кодування символів
   /// </summary>
```

```
    /// <param name="code"> таблиця кодування </param>
    /// <returns> зашифрований текст </returns>
    public string Cipher(string[] code)
    {
        string res = "";
        foreach (char sym in text)
        {
       int k = code[0].IndexOf(sym);
       if (k >= 0) res += code[1][k];
       else res += sym.ToString();
        }
        return res;
    }
```

Клас реалізує методи інтерфейсу, роблячи їх відкритими для клієнтів класу і спадкоємців. Клас може побудувати власні методи, використовуючи реалізацію методів інтерфейсу.

```
/// <summary>
    /// Перевірка поля text,  чи є він паліндромом
    ///  після перетворення Convert
    /// </summary>
    /// <returns> true, якщо паліндром  </returns>
    public bool IsPalindrom()
    {
        string txt = Convert();
        for(int i=0, j = txt.Length-1; i<j; i++, j--)
       if(txt[i] != txt[j])  return false;
        return true;
    }
    /// <summary>
    /// Шифрування, задане власною таблицею кодування
    /// </summary>
    /// <returns> зашифрований текст</returns>
    public string Coding()
    {
        return Cipher(codeTable);
    }
```

Розглянемо клас `Testing`, методи якого дозволять виконати тестування об'єктів створюваних нами класів:

```
/// <summary>
     /// тестовий клас
     /// </summary>
     class Testing
     {
    //поля класу
    const string PAL =
        "А роза упала на лапу Азора";
    static string[] CODE =
    {
        "абвгдежзиклмнопрстуфхцчшыъьэюя",
        "abvgdejziklmnoprstyfhc461w'qux"
    };
    /// <summary>
    /// Тестування класу SimpleText
    /// </summary>
    public void TestText()
    {
        Console.WriteLine("Работа с объектом класу SimpleText! ");
        SimpleText simpleText = new SimpleText(PAL);
        Console.WriteLine("Початковий текст : " + PAL);
        string text;
        text = simpleText.Convert();
```

```
            Console.WriteLine("Перетворений текст : " + text);
            if(simpleText.IsPalindrom())
           Console.WriteLine("Це паліндром!");
            text = simpleText.Coding();
            Console.WriteLine("Шифрований текст : " + text);
            Console.WriteLine("Робота з об'єктом інтерфейсу IStrings! ");
            IStrings istrings;
            text = "Це простий текст!";
            Console.WriteLine("Початковий  текст : " + text);
            simpleText = new SimpleText(text);
            istrings = (IStrings)simpleText;
            text = istrings.Convert();
            Console.WriteLine("Перетворений текст : " + text);
            text = istrings.Cipher(CODE);
            Console.WriteLine("Шифрованний текст : " + text);
        }
         }
```

Зверніть увагу, що у класі `Testing` оголошений як об'єкт класу `SimpleText`, так і об'єкт `istrings` інтерфейсу `IStrings`. У методі `TestText` об'єкт `simpleText` створюється звичайним способом при виклику конструктора класу. Потім цей об'єкт викликає як відкритий метод `Convert`, успадкований від інтерфейсу, так і відкриті методи класу `IsPalindrom`, `Coding`, що використовують методи інтерфейсу.

Повний код програми:

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

namespace ConsoleInterface1
{
    interface IStrings
    {
        string Convert();
        string Cipher(string[] code);
    }
    /// <summary>
    /// Успадковує інтерфейс IStrings
    /// реалізуючи його методи як відкриті (public)
    /// </summary>
    class SimpleText : IStrings
    {
        //поля класу
        string text;
        static string[] codeTable =
   {
     "абвгдеёжзийклмнопрстуфхцчшщыьъэюя ,.!?;:",
     "ъьыщшчцхфуэюя ,.!?;:тсрпонмлкйабвгдеёжзи"
   };
        //Конструктори
        public SimpleText()
        {
            text = "Простий текст!";
        }
        public SimpleText(string txt)
        {
            text = txt;
        }
        public string Text
```

```
        {
            get { return text; }
        }
        /// Реалізація інтерфейсів
        /// <summary>
        /// Видалення пробілів в полі text
        /// перетворення до нижнього регістра
        /// </summary>
        /// <returns>  перетворений рядок </returns>
        public string Convert()
        {
            string res = "";
            foreach (char sym in text)
                if (sym != ' ') res += sym.ToString();
            res = res.ToLower();
            return res;
        }
        /// <summary>
        /// шифрування поля text
        /// з використанням таблиці кодування символів
        /// </summary>
        /// <param name="code"> таблиця кодування </param>
        /// <returns> зашифрований текст </returns>
        public string Cipher(string[] code)
        {
            string res = "";
            foreach (char sym in text)
            {
                int k = code[0].IndexOf(sym);
                if (k >= 0) res += code[1][k];
                else res += sym.ToString();
            }
            return res;
        }
/// <summary>
        /// Перевірка поля text,  чи є він паліндромом
        ///  після перетворення Convert
        /// </summary>
        /// <returns> true, якщо паліндром  </returns>
        public bool IsPalindrom()
        {
            string txt = Convert();
            for (int i = 0, j = txt.Length - 1; i < j; i++, j--)
                if (txt[i] != txt[j]) return false;
            return true;
        }
        /// <summary>
        /// Шифрування, задане власною таблицею кодування
        /// </summary>
        /// <returns> зашифрований текст</returns>
        public string Coding()
        {
            return Cipher(codeTable);
        }
    }
    class Testing
    {
        //поля класу
        const string PAL =
            "А роза упала на лапу Азора";
```

```
        static string[] CODE =
    {
        "абвгдежзиклмнопрстуфхцчшыъьэюя",
        "abvgdejziklmnoprstyfhc461w'qux"
    };
        /// <summary>
        /// Тестування класу SimpleText
        /// </summary>
        public void TestText()
        {
            Console.WriteLine("Работа с объектом класса SimpleText! ");
            SimpleText simpleText = new SimpleText(PAL);
            Console.WriteLine("Початковий текст : " + PAL);
            string text;
            text = simpleText.Convert();
            Console.WriteLine("Перетворений текст : " + text);
            if (simpleText.IsPalindrom())
                Console.WriteLine("Це паліндром!");
            text = simpleText.Coding();
            Console.WriteLine("Шифрований текст : " + text);
         Console.WriteLine("Робота с об'єктом інтерфейсуа IStrings! ");
            IStrings istrings;
            text = "Це простий текст!";
            Console.WriteLine("Початковий текст : " + text);
            simpleText = new SimpleText(text);
            istrings = (IStrings)simpleText;
            text = istrings.Convert();
            Console.WriteLine("Перетворений текст : " + text);
            text = istrings.Cipher(CODE);
            Console.WriteLine("Шифрований текст : " + text);
        }
    }
    class Program
    {

        static void Main(string[] args)
        {
            Testing tint = new Testing();
            tint.TestText();
            Console.ReadKey();
        }
    }
}
```

Розглянемо ще один приклад. Класи `Person` і `Student`.

Приклад 2.
Створимо інтерфейсний клас з двома абстрактними методами:

```
interface IRating
    {
        void Rating(double R);
        string GetRole(int Course);
    }
// В класі Person створимо метод  GetAge()
class Person
    {
        public string Name;       //ім'я
        public int Age;           // вік
```

```
        public string Role;        // роль
        public string GetName() { return Name; }
        public int GetAge() { return Age; }
    }
```

Модифікуємо клас Student. Додамо наслідування інтерфейсу IRating. В цьому класі є реалізація методів інтерфейсу IRating.

```
class Student : Person,IRating
    {
        public string Facultet;
        public string Group;
        public int Course;
   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 Rating(double Student_Rating)
        {
            if (Student_Rating >= 82)
                Console.WriteLine("Привіт відмінникам");
            else
                if (Student_Rating <= 45)
                 Console.WriteLine("Перездача! Треба краще вчитися!");
                else
                    Console.WriteLine("Можна  вчитися ще краще!");
        }
    }
```

Створимо новий клас `Prepod`, похідний від класу `Person` з інтерфейсом `IRating`. В класі `Prepod` створимо власний конструктор і реалізацію методів інтерфейсу `IRating`.

```
class Prepod : Person
    {
        string kafedra;

        public Prepod(string N, string R)
        {
            Name = N;
            Role = R;
        }

        public string GetRole(int Course)
        {
            switch (Course)
```

```
                {
                    case 1:
                    return ("Не читаю");
                    case 2:
                     return ("Matlab");
                    case 3:
                     return ("C#");
                    case 4:
                     return ("Технологія програмування");
                    case 5:
                     return ("Semantic web and XML");
                    case 6:
                     return ("Парадигми програмування");
                    default:
                     return ("Неправильно заданий курс");
                }
            }

            public void Rating(double Prepod_Rating)
            {
                Console.WriteLine("Рейтинг викладача" + Prepod_Rating);
            }
            public  string Kafedra
            {
                get { return kafedra; }
                set
                {
                    kafedra = value;
                }
            }
        }
```

В класі `Program` напишемо код для перевірки роботи класів.

```
    class Program
        {
            static void Main(string[] args)
            {
                string r;
                string newRole;
                //дані про студента
                Student newSt = new Student("Іванов", 20, "студент",
"КННІ", "К-81", 4);
                Console.WriteLine("Дані про студента");
                string Name = newSt.GetName();
                int Age = newSt.GetAge();
                Console.WriteLine("Прізвище = " + Name);
                Console.WriteLine("Вік= " + Age);
                newRole = newSt.GetRole(newSt.Course);
                Console.WriteLine("Ви ще = " + newSt.Role);
                Console.WriteLine("Факультет = " + newSt.Facultet);
                Console.WriteLine("група= " + newSt.Group);
                Console.WriteLine("курс= " + newSt.Course);
                Console.WriteLine("Ваш рейтинг?");
                r = Console.ReadLine();
                double rating = Convert.ToDouble(r);
                newSt.Rating(rating);
                //дані про викладача
                Prepod pr = new Prepod("Коротун Т.М.", "зав.кафедрою");
                newRole = pr.Role;
```

```
            pr.Kafedra = "КНІС"; // ініціалізація через властивість
Kafedra
            Console.WriteLine("Дані про викладача:  " + pr.GetName());
            Console.WriteLine("Посада:  " + newRole);
            Console.WriteLine("Кафедра: " + pr.Kafedra);
            pr.Rating(100);      // інтерфейсний метод
            Console.WriteLine("Які дисципліни ви викладаєте?");
            Console.WriteLine("На якому курсі?");
            int Cource = int.Parse(Console.ReadLine());
            string disc = pr.GetRole(Cource);
            Console.WriteLine("на " + Cource + "курсі дисципліну  " +
disc);
            Console.ReadLine();
        }
    }
```

3. Інтерфейси і поля

Ми говорили, що в інтерфейсі можна оголошувати тільки методи, а поля не можна (в явному вигляді).

Але в інтерфейсі можна оголосити властивість (шаблон властивості) з методами `get` і `set`, що забезпечують доступ до поля.

Ось приклад такого інтерфейсу:

```
/// <summary>
/// Доступ до полів Name і Age
/// </summary>
interface IFields
{
    string Name { get; set; }
    int Age {get;}
}
```

Створимо клас, що успадковує цей інтерфейс:

```
/// <summary>
/// Клас, що успадковує інтерфейс IFields
/// Має поля Name і Age
/// доступ до поля Name відкритий клієнтам класу
/// доступ до поля Age закритий і відкритий з перейменуванням!
/// </summary>
class TwoFields:IFields
{
    string name;
    int age;
    public TwoFields()
    {
        name = "Nemo"; age = 37;
    }
    public TwoFields(string name, int age)
    {
        this.name = name; this.age = age;
    }
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
    int IFields.Age
    {
        get { return age; }
```

```
        }
        public int WhatAge()
        {
            return age;
        }
    }
```

Зверніть увагу: методи доступу до полів, успадковані від інтерфейсу, можна реалізувати як відкриті так і як закриті, відкриваючи їх потім під іншим іменем. У класу завжди має бути можливість перейменовувати імена методів і полів, успадкованих від інтерфейсу.

Додамо в клас `Program` в метод `main` код, що дозволяє протестувати роботу з успадкованими полями.

```
public void TestFields()
  {
      Console.WriteLine("Робота з об'єктом класу TwoFields!");
      TwoFields twofields = new TwoFields();
      Console.WriteLine("Ім'я: {0}, Вік: {1}",
          twofields.Name, twofields.WhatAge());
      twofields.Name = "Captain Nemo";
      Console.WriteLine("Робота з інтерфейсним об'єктом  IFields!");
      IFields ifields = (IFields)twofields;
      Console.WriteLine("Ім'я: {0}, Вік: {1}",
          ifields.Name,ifields.Age);
  }
```

Повний код програми.

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

namespace Console3_Lab10
{
    interface IFields
    {
        string Name { get; set; }
        int Age { get; }
    }

    /// <summary>
    /// Клас, що успадковує інтерфейс IFields
    /// Має поля Name і Age
    /// доступ до поля Name відкритий клієнтам класу
    /// доступ до поля Age закритий і відкритий з перейменуванням!
    /// </summary>
    class TwoFields : IFields
    {
        string name;
        int age;
        public TwoFields()
        {
            name = "Nemo"; age = 37;
        }
        public TwoFields(string name, int age)
        {
            this.name = name; this.age = age;
        }
```

```
            public string Name
            {
                get { return name; }
                set { name = value; }
            }
            int IFields.Age
            {
                get { return age; }
            }
            public int WhatAge()
            {
                return age;
            }
        }

        class Program
        {
            static void Main(string[] args)
            {
                Console.WriteLine("Робота з об'єктом класу TwoFields!");
                TwoFields twofields = new TwoFields();
                Console.WriteLine("Ім'я: {0}, Вік: {1}",twofields.Name,
twofields.WhatAge());
                twofields.Name = "Captain Nemo";
                Console.WriteLine("Робота з інтерфейсним об'єктом
IFields!");
                IFields ifields = (IFields)twofields;
                Console.WriteLine("Ім'я: {0}, Вік: {1}",ifields.Name,
ifields.Age);
                Console.ReadLine();
            }
        }
    }
```

4. Інтерфейси і спадкоємство

Інтерфейс може не мати або мати скільки завгодно інтерфейсів-предків, в останньому випадку він успадковує всі елементи всіх своїх базових інтерфейсів, починаючи з самого верхнього рівня.

Як і в звичайній ієрархії класів, базові інтерфейси визначають загальну поведінку, а їх нащадки конкретизують і доповнюють його. У інтерфейсі-нащадку можна також вказати елементи, які перевизначають успадковані елементи з такою ж сигнатурою. В цьому випадку перед елементом вказується ключове слово **new**, як і в аналогічній ситуації в класах. За допомогою цього слова відповідний елемент базового інтерфейсу приховується. Клас, що реалізує інтерфейс, повинен визначати всі його елементи, у тому числі успадковані.

Інтерфейс, на власні або успадковані елементи якого є явне посилання, має бути вказаний в списку предків класу.

Клас успадковує всі методи свого предка, у тому числі ті, які реалізували інтерфейси. Він може перевизначити ці методи за допомогою специфікатора **new**, але звертатися до них можна буде лише через об'єкт класу. Якщо використовувати для звернення посилання на інтерфейс, викликається не перевизначена версія:

```
    interface IBase
    {
       void A();
    }

    class Base : IBase
    {
       public void A() { ... }
    }
    class Derived: Base
```

```
{
   new public void A() { ... }
}
...
Derived d = new Derived ();
d.A();                          // викликається Derived.A();
IBase id = d;
id.A();                         // викликається Base.A();
```

Проте якщо інтерфейс реалізується за допомогою віртуального методу класу, після його перевизначення в нащадку будь-який варіант звернення (через клас або через інтерфейс) призведе до одного і того самого результату.

Метод інтерфейсу, реалізований явною вказівкою імені, оголошувати віртуальним забороняється.

Існує можливість повторно реалізувати інтерфейс, вказавши його ім'я в списку предків класу разом з класом-предком, що вже реалізував цей інтерфейс. При цьому реалізація перевизначених методів базового класу в увагу не береться:

```
interface IBase
{
   void A();
}

class Base : IBase
{
   void IBase.A() { ... }        // не використовується в Derived
}

class Derived : Base, IBase
{
   public void A() { ... }
}
```

Якщо клас успадковує від класу і інтерфейсу, які містять методи з однаковими сигнатурами, успадкований метод класу сприймається як реалізація інтерфейсу. Взагалі при реалізації інтерфейсу враховується наявність "відповідних" методів в класі незалежно від їх походження. Це можуть бути методи, описані в поточному або базовому класі, які реалізують інтерфейс явним або неявним чином.

Головне призначення інтерфейсів – забезпечення множинного наслідування класів. Клас може успадковувати (реалізовувати) методи декількох інтерфейсів. Через те, що інтерфейсний клас не містить реалізації методів, то кожний клас, який успадковує інтерфейс, повинен реалізувати всі його методи. Кожен клас може визначати методи інтерфейсу по-своєму. Так досягається поліморфізм: об'єкти різних класів по-різному реагують на виклики одного і того ж методу.

Клас, що успадковує інтерфейс і реалізує його методи, може оголосити відповідні методи класу відкритими (public) чи закритими (цю стратегію реалізації ми не розглядали).

В інтерфейсі не можна оголошувати в явному вигляді поля, але можна оголосити властивість (шаблон властивості) з методами get і set, що забезпечують доступ до поля.

Інтерфейс може не мати або мати скільки завгодно інтерфейсів-предків, в останньому випадку він успадковує всі елементи всіх своїх базових інтерфейсів, починаючи з самого верхнього рівня. Як і в звичайній ієрархії класів, базові інтерфейси визначають загальну поведінку, а їх нащадки конкретизують і доповнюють його.

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. Перетворити консольний за стосунок, створений вами у другому завданні лабораторної роботи №4 таким чином, щоб в ньому з'явився інтерфейсний клас, у який перенести усі методи (реалізація методів повинна залишитись у класі, що успадковує інтерфейс). Варіанти індивідуальних завдань лабораторної роботи №4 для зручності продубльовано в таблиці.

(2 бала)

1. Описати клас для бази зданих з інформацією про книги у формі: автор, назва книги, видавництво, рік видання, кількість штук. Вивести масив на екран у формі таблиці, згрупувавши книги з однаковим роком видання. Порахувати загальну кількість книг для кожного року видання та вивести цю інформацію у вигляді таблиці на екран.
2. Описати клас для бази зданих з інформацією про книги у формі: автор, назва книги, видавництво, рік видання. Вивести дані про книги з програмування (перевіряти, чи є частиною назви книги слово «програмування» з малої або великої літери) у порядку спадання років видань.
3. Описати клас для бази зданих з інформацією про конфігурацію комп'ютера з полями: тип процесора, тактова частота, обсяг оперативної пам'яті, обсяг дискової пам'яті, характеристика монітора та ін. Відсортувати записи по типу процесора та вивести на екран у формі таблиці. Визначити комп'ютери з найбільшим обсягом оперативної і дискової пам'яті.
4. Описати клас для бази зданих з інформацією про результати роботи підприємства протягом року у форматі: місяць, план випуску продукції, фактичний випуск продукції. изначити назви місяців з недовиконанням плану випуску продукції та вивести цю інформацію на екран у вигляді таблиці.
5. Описати клас для бази зданих з інформацією про результати роботи підприємства впродовж року з полями: місяць, план випуску продукції, фактичний випуск продукції. Відсортувати масив записів у порядку зростання відсотку виконання плану та вивести його на екран у формі таблиці. Визначити місяці з найбільшим та найменшим відсотком виконання плану.
6. Описати клас для бази зданих з інформацією про рейтинг студентів однієї групи: прізвище студента, № залікової книжки, рейтинг у 100 бальній шкалі. Впорядкувати записи по спаданню рейтингу та вивести його на екран у формі таблиці. Обчислити середній рейтинг групи та кількість студентів з рейтингом нижче середнього.
7. Описати клас для бази зданих з інформацією про студентів з полями: прізвище, дата народження, місце народження. Дата народження задається у вигляді ДД:ММ:РР. Відсортувати записи за зростанням дат народження та вивести його на екран у формі таблиці. Якщо є студенти з однаковою датою народження (співпадають день та місяць), то вивести записи про них окремо в таблиці «двійнята».
8. Описати клас для бази зданих з інформацією про успішність групи студентів з полями: прізвище та ім'я, № залікової книжки, оцінки за 100 бальною шкалою з п'яти предметів. Впорядкувати записи у порядку зростання зростання середнього балу і вивести їх на екран у формі таблиці. Визначити відсоток студентів, що мають незадовільні оцінки.
9. Описати клас для бази зданих з інформацією про успішність групи студентів з полями: прізвище та ім'я, № залікової книжки, оцінки з п'яти предметів за 100 бальною шкалою. Впорядкувати записи у порядку спадання середньої оцінки та вивести їх на екран у формі таблиці. Визначити відсоток студентів, середній бал яких відповідає оцінкам "добре" та "відмінно".
10. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом місяця у форматі: дата, температура повітря, атмосферний тиск. Впорядкувати записи у порядку спадання температури повітря та вивести його на екран у формі таблиці. Визначити два дні з найбільшим перепадом температури повітря.
11. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом місяця у форматі: дата, температура повітря, атмосферний тиск. Визначити дні з атмосферним тиском, більшим від середнього начення цього показника за весь період. Результат вивести на екран у формі таблиці.
12. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом року у форматі: дата (ДД:ММ), температура повітря. Знайти середню температуру повітря кожного місяця та вивести на екран таблицю: місяць, середня температура. Визначити назву місяця з найбільшою середньою температурою повітря.
13. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом місяця у форматі: дата, температура, атмосферний тиск. Впорядкувати дані у порядку зростання тиску та вивести результати на екран у формі таблиці. Визначити два дні з найбільшим перепадом тиску.
14. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом місяця у форматі: дата, температура, тиск. Визначити дні, температура яких є більшою від середнього значення температури. Результат вивести на екран у формі таблиці.
15. Описати клас для бази зданих з інформацією про власників авто з полями: марка автомобіля, номер автомобіля, колір, дані про власника. Вивести на екран у формі таблиці дані про власників автомобілів заданного кольору та заданої марки.
16. Описати клас для бази зданих з інформацією про успішність з полями: прізвище студента, оцінки з п'яти предметів. Знайти середній бал по кожному предмету. Вивести результати на екран у формі таблиці.

17. Описати два класи для бази зданих з інформацією про виробництво. Перший містить поля: назва виробу, вартість одиниці виробу. Другий про результати виробництва за звітний період: дата, назва виробу, кількість. Дані про один виріб можуть повторюватися в різні дати. Обчислити загальну вартість виготовленої за звітний період продукції за кожним видом виробів. Відсортувати утворений масив по зростанню вартості виробів та вивести його на екран у формі таблиці.

18. Описати клас для бази зданих з інформацією про автомобілі з полями: марка, колір, номер, рік випуску, дані про власника. Відсортувати масив даних по марках автомобілів та вивести його на екран у формі таблиці. Визначити кількість різних кольорів кожної марки.

19. Описати клас для бази зданих з інформацією про автомобілі з полями: марка, колір, номер, рік випуску, дані про власника. Визначити кількість автомобілів кожної марки та вивести на екран у формі таблиці впорядкувавши по спаданню кількості автомобілів.

20. Описати клас для бази зданих з інформацією про дні народження декількох студентів з полями: дата, прізвище та ініціали. Дата задається у форматі ДД:ММ:РР. З клавіатури ввести поточну дату. Визначити студента з найближчим днем народження. Вивести на екран дані про студентів, дні народження яких ще будуть у поточному році (відлік вести від поточної дати).

21. Описати клас для бази зданих з інформацією про про клієнтів банку з полями: № рахунку, прізвище та ім'я, сума вкладу, дата проведення операції. Визначити клієнтів банку з найбільшою кількістю банківських операцій та вивести дані про них на екран у формі таблиці.

22. Описати клас для бази зданих з інформацією про клієнтів банку з полями: дата проведення операції прізвище та ім'я, № рахунку, сума безготівкового отримання/переведення, отримано/видано готівкою, залишок вкладу. Вивести на екран у формі таблиці дані про клієнтів банку, які на протязі заданого періоду часу мають найбільшу суму безготівкового отримання коштів на рахунок.

23. Описати клас для бази зданих з інформацією про клієнтів банку з полями: дата проведення операції прізвище та ім'я, № рахунку, сума безготівкового отримання/переведення, отримано/видано готівкою, залишок вкладу. Вивести на екран у формі таблиці дані про клієнтів банку, які на протязі заданого періоду часу виконали безготівкове переведення на загальну суму, яка перевищує задане користувачем граничне значення.

24. Описати клас для бази зданих з інформацією про книги бібліотеки з полями: автор, назва книги, видавництво, рік видання. Відсортувати масив за прізвищами авторів та вивести на екран у формі таблиці. Підрахувати кількість книг від кожного видавництва.

25. Описати клас для бази зданих з інформацією про книги бібліотеки з полями: автор, назва книги, видавництво, рік видання, кількість штук. Дані про книги, видані після 2000 року вивести на екран у формі таблиці, порахувати загальну кількість таких книг.

2. Для консольного застосунку з ієрархією класів з попередньої лабораторної роботи (№5) організувати інтерфейсний клас з двома абстрактними методами (які з існуючих у проекті методів робити абстрактними – обирайте самі).

Варіанти індивідуальних завдань попередньої лабораторної роботи для зручності продубльовано в таблиці.

(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