ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Освітньо-кваліфікаційний рівень_____бакалавр_____

Напрям підготовки ___Компютерні_науки_____

Спеціальність_____ІТІНФ_____ Семестр_____3_____
(назва)

Навчальна дисципліна _____Програмне забезпечення обчислювальних систем_____

ЕКЗАМЕНАЦІЙНИЙ БІЛЕТ № __21___

1.Теоретична частина.

1.1. Масиви. Масиви та типи. Ініціалізація масивів. Багатовимірні масиви

1.2. Дати визначення та описати призначення інтерфейсу. Навести способи реалізації інтерфейсів. Дати визначення та привести пример використання поняття “кастінг”..

2.Практична частина.

2.1. Створити (абстрактний) клас Телефонний_довідник із методами, що дозволяють вивести на екран інформацію про записи в телефонному довіднику, а також визначити відповідність запису критерію пошуку.

1. Створити похідні класи: Персона (прізвище, адреса, номер телефону), Організація (назва, адреса, телефон, факс, контактна особа), Друг (прізвище, адреса, номер телефону, дата народження) зі своїми методами виведення інформації на екран, та визначення відповідності шуканому типу.

2. Створити базу (масив) з n товарів, вивести повну інформацію з бази на екран, а також організувати пошук у базі на прізвище.

Затверджено на засіданні кафедри _____інформатики_____
Протокол №_6___ від „_01_” ___грудня_____ 20_21_року

Лектор: Сінельникова Т.Ф.

Зав. кафедри Кобилін О.А.

Комбінований іспит

З дисципліни ПЗоС

Студента групи ІТІНФ-20-1

Самченка Станіслава

Білет №21

Теоритична частина

Питання:

1. Масиви. Масиви та типи. Ініціалізація масивів. Багатовимірні масиви

2. Дати визначення та описати призначення інтерфейсу. Навести способи реалізації інтерфейсів. Дати визначення та привести пример використання поняття “кастінг”

1) **Масив** є сукупність змінних одного типу із загальним для звернення до них ім'ям. C# масиви можуть бути як одномірними, так і багатовимірними. Масиви служать різним цілям, оскільки вони надають зручні засоби для об'єднання пов'язаних разом змінних.

Масивами C# можна користуватися практично так само, як і в інших мовах програмування. Проте вони мають одна особливість: вони реалізовані як об'єктів.

Для того щоб скористатися масивом у програмі, потрібна двоетапна процедура, оскільки в C# масиви реалізовані у вигляді об'єктів. По-перше, необхідно оголосити змінну, яка може звертатися до масиву. І, по-друге, потрібно створити екземпляр масиву, використовуючи оператор new.

Приклад: `int[] myArr = new int[5];`

Ініціалізація масиву

Крім заповнення масиву елемент за елементом (як показано в попередньому прикладі), можна заповнювати його з використанням спеціального синтаксису ініціалізації масивів. Для цього необхідно перерахувати елементи, що включаються в масив, у фігурних дужках { }. Такий синтаксис зручний при створенні масиву відомого розміру, коли потрібно швидко задати початкові значення:

```
int[] myArr = new int[] {10,20,30,40,50};
string[] info = { "Фамилия", "Имя", "Отчество" };
char[] symbol = new char[4] { 'X','Y','Z','M' };
```

Багатовимірним називається такий **масив**, який відрізняється двома або більше вимірами, причому доступ до кожного елемента такого масиву здійснюється за допомогою певної комбінації двох або більше індексів. Багатовимірний масив індексується двома та більш цілими числами.

Найпростішою формою багатовимірного масиву є двовимірний масив. Розташування будь-якого елемента у двовимірному масиві позначається двома індексами. Такий масив можна у вигляді таблиці, на рядки якої вказує один індекс, але в стовпці — інший. Приклад оголошення та ініціалізації двовимірного масиву:

```
// Объявляем двумерный массив
int[,] myArr = new int[4, 5];

Random ran = new Random();

// Инициализируем данный массив
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 5; j++) {
       myArr[i, j] = ran.Next(1, 15);
       Console.Write("{0}\t", myArr[i, j]);
    }
    Console.WriteLine();
}
```

C# допускаються масиви трьох і більше вимірювань. Нижче наведено загальну форму оголошення багатовимірного масиву:

```
тип[,...,] имя_массива = new тип[размер1, размер2, ... размеры];
```

Використовує тривимірний масив:

```
int[,,] myArr = new int[5,5,5];

for (int i = 0; i < 5; i++)
   for (int j = 0; j < 5; j++)
      for (int k = 0; k < 5; k++)
         myArr[i, j, k] = i + j + k;
```

2) **Інтерфейс** представляє тип посилання, який може визначати деякий функц іонал - набір методів і властивостей без реалізації. Потім цей функціонал реа лізують класи та структури, які застосовують дані інтерфейси. Для визначенн я інтерфейсу використається ключове слово interface.

Інтерфейс є повністю абстрактним класом, всі методи якого абстрактні. Від абстрактного класу інтерфейс відрізняється деякими деталями в синтаксисі та поведінці.

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

Відмінність у поведінці полягає у жорсткіших вимогах до нащадків. Клас, наслідуючий інтерфейс, повинен повністю продати всі способи інтерфейсу. В цьому - на відміну від класу, що успадковує абстрактний клас, де нащадок мо же реалізувати лише деякі методи батьківського абстрактного класу, залиша ючись абстрактним класом.

Призначення інтерфейсів

Інтерфейси дозволяють частково впоратися з таким суттєвим недоліком мови , як відсутність множинного успадкування класів. У мові C# повного

множинного успадкування класів немає. Щоб частково згладити цю прогалин у, допускається множинне наслідування інтерфейсів.

Забезпечити можливість класу мати кілька батьків – один повноцінний клас, і нші як інтерфейсів, – у цьому полягає основне призначення інтерфейсів.

Також інтерфейс дозволяє описувати деякі бажані властивості, які можуть ма ти об'єкти різних класів. У бібліотеці FCL є велика кількість подібних інтерф ейсів, з деякими їх ми познайомимося у цій лекції.

Для **реалізації** інтерфейсу є ключове слово interface.

Приклад реалізації:

```
public interface IProps {
  void Prop1 (string s);
  void Prop2 (string name, int val);

}
```

Цей інтерфейс має два методи, які і повинні будуть реалізувати всі класи – сп адкоємці інтерфейсу. Зауважте, методи не мають модифікаторів доступу.

Клас, успадковуючи інтерфейс і його методи, може реалізувати їх явно, оголошуючи відповідні методи класу відкритими.

Приклад:

```
public class Clain : IProps {
  …
  public void Prop1(string s)
  { Console.WriteLine(s);}
  public void Prop2(string name, int val)
  {Console.WriteLine("name = {0}, val ={1}", name, val);}
…
}
```

Клас реалізує методи інтерфейсу, роблячи їх відкритими для клієнтів класу та спадкоємців

Інша стратегія реалізації полягає в тому, щоб усі або деякі методи інтерфейсу зробити закритими. Для реалізації цієї стратегії клас, успадковуючий інтерфейс, оголошує методи без модифікатора доступу, що відповідає модифікатору private, і уточнює ім'я методу ім'ям інтерфейсу.

Приклад:

```
public class ClainP:IProps {
…
 void IProps.Prop1(string s)
 {Console.WriteLine(s); }
 void IProps.Prop2(string name, int val)
 {Console.WriteLine("name={0}, val={1}", name, val); }

}
```

Клас ClainP реалізував методи інтерфейсу IProps, але зробив їх закритими та недоступними для виклику клієнтів та спадкоємців класу.

Створити об'єкт класу інтерфейсу звичайним шляхом з використанням конструктора та операції new не можна. Проте, можна оголосити об'єкт інтерфейсного класу і пов'язати його з цим об'єктом шляхом приведення (**кастингу**) об'єкта спадкоємця до класу інтерфейсу. Це перетворення задається очевидно. Маючи об'єкт, можна викликати методи інтерфейсу – навіть якщо вони закриті у класі, для інтерфейсних об'єктів є відкритими

Кастинг приклад:

```
public void TestClainIProps() {
  Console.WriteLine("Об'єкт класу Clain викликає відкриті методи!");
  Clain Clain = New Clain ();
  clain.Prop1(" властивість 1 об'єкта");
  clain.Prop2("Володимир", 44);
  Console.WriteLine("Об'єкт класу IProps викликає відкриті методи!");
  IProps ip = (IProps)clain;
  ip.Prop1("інтерфейс: властивість");
  ip.Prop2 ("інтерфейс: властивість", 77);
  Console.WriteLine("Об'єкт класу ClainP викликає відкриті методи!");
  ClainP clainp = New ClainP();
  clainp.MyProp1(" властивість 1 об'єкта");
  clainp.MyProp2("Володимир", 44);
  Console.WriteLine("Об'єкт класу IProps викликає закриті методи!");
  IProps ipp=(IProps)clainp;
  ipp.Prop1("інтерфейс: властивість");
  ipp.Prop2 ("інтерфейс: властивість",77);
}
```

Комбінований іспит

З дисципліни ПЗоС

Студента групи ІТІНФ-20-1

Самченка Станіслава

Білет №21

Практична частина

Питання:

1. Створити (абстрактний) клас Телефонний_довідник із методами, що дозволяють вивести на екран інформацію про записи у телефонному справочнику, а також визначити відповідність запису критерію пошуку.

2. Створити похідні класи: Персона (прізвище, адреса, номер телефону), Організація (назва, адреса, телефон, факс, контактна особа), Друг (прізвище, адреса, номер телефону, дата народження) зі своїми методами виведення інформації на екран, та визначення відповідності шуканому типу.

3. Створити базу (масив) з n товарів, вивести повну інформацію з бази на екран, а також організувати пошук у базі на прізвище.

Program.cs

```
using System;

namespace Ticket21
{
    class Program
    {
        static void Main(string[] args)
        {
            int n;
            Console.WriteLine("Введите количество элементов");
            n = Convert.ToInt32(Console.ReadLine());
            int choice;
            TelephoneJournal[] list = new TelephoneJournal[5];
            for (int i = 0; i < n; i++)
            {
                Console.WriteLine("О ком ввести информацию:\n1-О персоне\n2-Об организации\n3-О друге");
                choice = Convert.ToInt32(Console.ReadLine());
                if (choice == 1)
                {

                    String s, a, p;

                    Console.WriteLine("Введите фамилию");
                    s = Console.ReadLine();
                    Console.WriteLine("Введите адрес");
                    a = Console.ReadLine();
                    Console.WriteLine("Введите телефон");
                    p = Console.ReadLine();
                    list[i] = new Persona(s, a, p);
                }
                else if (choice == 2)
                {

                    String s, a, p, f, cp;

                    Console.WriteLine("Введите название");
                    s = Console.ReadLine();
                    Console.WriteLine("Введите адрес");
                    a = Console.ReadLine();
                    Console.WriteLine("Введите телефон");
                    p = Console.ReadLine();
                    Console.WriteLine("Введите факс");
```

```
                f = Console.ReadLine();
                Console.WriteLine("Введите контактное лицо");
                cp = Console.ReadLine();
                list[i] = new Organization(s, a, p, f, cp);

            }

            else if (choice == 3)
            {

                String s, a, p, d;

                Console.WriteLine("Введите фамилию");
                s = Console.ReadLine();
                Console.WriteLine("Введите адрес");
                a = Console.ReadLine();
                Console.WriteLine("Введите телефон");
                p = Console.ReadLine();
                Console.WriteLine("Введите дату рождения");
                d = Console.ReadLine();
                list[i] = new Friend(s, a, p, d);

            }


        }
            for (int i = 0; i < n; i++)
            {
                Console.WriteLine(list[i].ToString());
            }

            while (true)
            {
                String search;
                Console.WriteLine("Введите фамилию для поиска");
                search = Console.ReadLine();

                for (int i  = 0; i < n; i++)
                {
                    if (list[i].findSurname(search))
                        Console.WriteLine(list[i].ToString());
                }
            }
        }
    }
}
```

TelephoneJournal.cs

```
using System;

namespace Ticket21
{
    abstract class TelephoneJournal
    {
        public abstract String ToString();
        public abstract Boolean findSurname(String surname2);
    }
}
```

Personal.cs

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

namespace Ticket21
{
    class Persona:TelephoneJournal
    {
        String surname;
        String address;
        String phone;

        public Persona(String s,String a,String p)
        {
            surname = s;
            address = a;
            phone = p;
        }
```

```
        public override string ToString()
        {
            return "\nФамилия:" + surname + "\nАдрес:" + address + "\nТелефон:" + phone;
        }

        public override Boolean findSurname(String surname2)
        {
            if (surname.Equals(surname2))
                return true;
            return false;
        }
    }
}
```

Organization.cs

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

namespace Ticket21
{
    class Organization : TelephoneJournal
    {
        String surname;
        String address;
        String phone;
        String fax;
        String contactPerson;

        public Organization(String s, String a, String p,String f,String cP)
        {
            surname = s;
            address = a;
            phone = p;
            fax = f;
            contactPerson = cP;
        }

        public override string ToString()
        {
            return "\nФамилия:" + surname + "\nАдрес:" + address + "\nТелефон:" + phone + "\nФакс:" + fax + "\nКонтактная
персона:" + contactPerson;
        }

        public override Boolean findSurname(String surname2)
        {
            if (surname.Equals(surname2))
                return true;
            return false;
        }
    }
}
```

Friend.cs

```
using System;

namespace Ticket21
{
    class Friend : TelephoneJournal
    {
        String surname;
        String address;
        String phone;
        String date;

        public Friend(String s, String a, String p, String d)
        {
            surname = s;
            address = a;
            phone = p;
            date = d;
        }

        public override string ToString()
        {
            return "\nФамилия:" + surname + "\nАдрес:" + address + "\nТелефон:" + phone + "\nДата рождения:" + date;
        }
```

```
        public override Boolean findSurname(String surname2)
        {
            if (surname.Equals(surname2))
                return true;
            return false;
        }
    }
}
```