**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет   
імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інженерії програмного забезпечення**

**ЗВІТ**

з лабораторної роботи № 5

**«Клас String. Наслідування»**

**Варіант 5**

Дисципліна «Об'єктно-орієнтоване програмування»

Спеціальність: **Інженерія програмного забезпечення**

121-ЛР.05-208.22120802

***Студент\_\_Д. М.  Дзундза***

*(підпис)*

*\_\_10.04.2021* \_\_

*(дата)*

***Викладач \_\_\_\_\_\_\_\_С. Ю. Боровльова***

*(підпис)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

Миколаїв – 2021

**Лабораторна робота №5**

|  |  |
| --- | --- |
| Тема роботи: | Клас String. Наслідування |

|  |
| --- |
| Мета роботи: |

Ознайомитись з використанням абстрактних та запечатаних класів, використання ключового слова this. Навчитись використовувати часткові класи. Ознайомитись з класом String. **Завдання:**

1.На базі класів із попередньої лабораторної роботи створіть додаток для демонстрації в коді наступних можливостей:

* Використання абстрактного та запечатаного класів
* Використання ключового слова this
* Використання часткових класів
* Не менше 3 конструкторів класу String
* Не менше 5 методів класу String
* Навести в класі приклад інтернування рядків та доказати в коді, що вони дійсно інтерновані.
* Порівняння методів Equals класу Object та класу String.
* Показати в коді MSIL пул рядків та показати в коді особливості збереження рядків констант.
* Додатково\*: Використання ініціалізаторів (0.1 бал при умові своєчасного захисту лабораторної роботи)

2.Використовуючи класи із свого варіанту, створіть додаток для демонстрації в коді наслідування та поліморфізму:

* Створити метод, який в якості параметра приймає тип Object При виклику методу передавати в якості параметра об‘єкти свого класу(в методі використати as).
* Створити метод, який в якості параметра приймає тип Вашого базового класу При виклику методу передавати в якості параметра об‘єкти своїх класів. Розібратись в різниці виконання цього методу і методу із пункту 1.
* Створити масив об‘єктів базового та похідних класів та продемонструвати поліморфізм з використанням об‘єктів масиву.
* Додатково\*:Продемонструвати роботу new. Для зарахування обов‘язково розібрати теоретично необхідність використання new, використавши приклад із Ріхтера, посилання є в перезентації (0.4 бали при умові своєчасного захисту лабораторної роботи і розуміння процесу роботи)

**Лістинг коду:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

sealed class Utils

{

public const string OSNAME = "WINDOWS";

public void Print(string[] args)

{

foreach (var arg in args)

{

Console.WriteLine(arg);

}

}

}

abstract class Person

{

protected int age;

protected string name;

abstract public string Name { get; set; }

abstract public int Age { get; set; }

}

partial class Worker : Person

{

int paidDays;

override public int Age

{

get => this.age;

set => this.age = value;

}

override public string Name

{

get => this.name;

set => this.name = value;

}

public Worker() : this("", 0)

{

}

public Worker(string name, int age)

{

this.name = name;

this.age = age;

}

}

class Template

{

protected int numOfPages;

protected string[] text;

protected int maxNumberOfCharactersOnPage;

public Template()

{

}

public void Out()

{

Console.WriteLine("Template Out");

}

}

class Book : Template

{

public Book()

{

}

public void Convert(Object obj)

{

Book book = obj as Book;

if (book != null)

{

Console.WriteLine("Can convert");

}

else

{

Console.WriteLine("Cannot convert");

}

}

public void Convert2(Template obj)

{

Book book = obj as Book;

if (book != null)

{

Console.WriteLine("Can convert");

}

else

{

Console.WriteLine("Cannot convert");

}

}

public new virtual void Out()

{

Console.WriteLine("Book Out");

}

public virtual void Out(double i)

{

Console.WriteLine("Book Out double");

}

public virtual void Out(int i)

{

Console.WriteLine("Book Out int");

}

}

class Magazine : Book

{

public new void Out()

{

Console.WriteLine("Magazine new Out");

}

public new void Out(int i)

{

Console.WriteLine("Magazine new Out int");

}

public new void Out(double i)

{

Console.WriteLine("Magazine new Out double");

}

}

class PrintedEdition : Book

{

public override void Out()

{

Console.WriteLine("PrintedEdition override Out");

}

public override void Out(int i)

{

Console.WriteLine("PrintedEdition override Out int");

}

}

class Program

{

static void Main(string[] args)

{

String str1 = "Dont like Funkner";

char[] sim = { 'I', 'o' };

String str2 = new String(sim);

String str3 = str1.Insert(str1.Length, " Unless u like Shon.");

String str4 = "Row";

Console.WriteLine(str1);

Console.WriteLine(str2);

Console.WriteLine(str3);

Console.WriteLine(str1[0]);

Console.WriteLine(str1.StartsWith("Dont"));

Console.WriteLine(str1.Substring(4, 5));

Console.WriteLine(str1.ToLower());

Console.WriteLine(str1.ToUpper());

String intern1 = "Hello", intern2 = "Hello", intern3 = "people", intern4 = "people";

Console.WriteLine(Object.ReferenceEquals(intern1, intern2));

Console.WriteLine(Object.ReferenceEquals(intern1, intern4));

Console.WriteLine("reference = " + String.IsInterned(intern1));

str4 = Console.ReadLine();

Console.WriteLine("reference = " + String.IsInterned(str4));

Console.WriteLine("intern1 == intern2: " + intern1.Equals(intern2));

Console.WriteLine("intern2 == intern4: " + intern2.Equals(intern4));

Worker worker = new Worker();

Worker worker2 = new Worker();

Console.WriteLine("worker1 == worker2: " + worker.Equals(worker2));

worker.Print();

worker.Age = 55;

Console.WriteLine("Age: " + worker.Age);

Console.WriteLine(Utils.OSNAME);

Book book1 = new Book();

Book book2 = new Book();

Book book3 = new Book();

book1.Convert(book1);

book2.Convert2(book3);

Console.WriteLine();

Console.WriteLine("Polimorfism");

Magazine magazine = new Magazine();

PrintedEdition printedEdition = new PrintedEdition();

Book bookPrintedEdition = new PrintedEdition();

Book bookMagazine = new Magazine();

Book book = new Book();

Book[] books = { magazine, bookPrintedEdition, printedEdition, bookMagazine, book };

foreach (var b in books)

{

b.Out();

}

magazine.Out();

printedEdition.Out();

Console.WriteLine();

Console.WriteLine("Out with arg");

foreach (var b in books)

{

b.Out(5);

}

magazine.Out(5);

printedEdition.Out(5);

Console.WriteLine();

Console.WriteLine("Out with arg double");

foreach (var b in books)

{

b.Out(5.7);

}

magazine.Out(5.5);

printedEdition.Out(5.5);

Console.ReadKey();

Console.ReadKey();

}

}

}

**Результат роботи програми:**

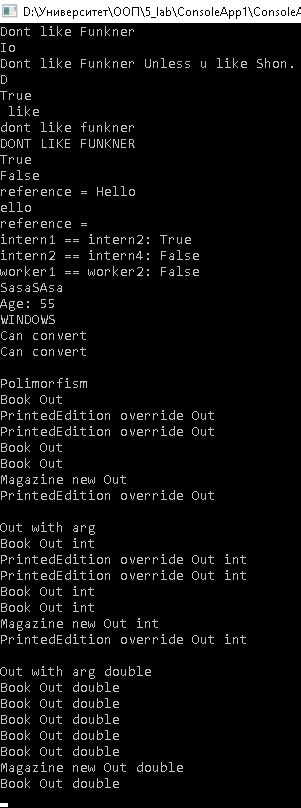


Рисунок 1 – Результат роботи програми

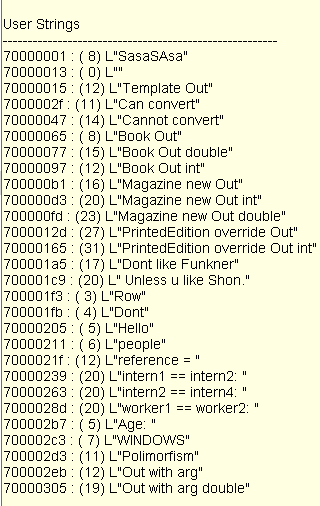
****

Рисунок 2 – Зберігання рядків у ildasm

**Питання до захисту:**

1. Незмінність рядків

Рядок є незмінним, при присвоєнні нового значення старий об‘єкт звільняється, а новий створюється.

2.Алгоритм роботи з рядками при великій кількості операцій зміни рядків

Можна використати наступний прийом:

1. Створити об’єкт класу String, використати його,
2. При небхідності декількох змін цього об’єкту, перетворити його в об’єкт класу StringBuilder, виконати зміни
3. Перетворити об’єкт в об’єкт класу String

3.Методи класу клас String

Clone, Compare, CompareTo, Concat, Contains, EndsWith, Equals, Format, IndexOf, IndexOfAny, Insert, Join, Normalize, PadLeft…

4.Інтернування рядків.

Інтернування - це спосіб зберігання однієї копії будь якого - рядка.

При необхідності, можна перевірити, чи зберігається рядок в пулі рядків. Метод IsInterned повертає посилання на рядок, якщо такий рядок міститься в пулі рядків і null, якщо ні.

Під час виконання програми можна додати рядок в пул рядків. Для цього існує метод Intern, який перевіряє, чи міститься рядок в пулі рядків, і якщо ні, то додає його в пул.

5.Переваги та недоліки використання пула рядків

Недоліки:   
Коли постійно змінюємо рядки, то використовується багато ресурсів.

Переваги:  
Використовується менше пам’яті

6.Перерахуйте принципи ООП

Спадкування

Iнкапсуляцiя

Полiморфiзм

7.Типи наслідування в С#

Використовується тільки просте наслідування.

8.Поліморфізм

Поліморфізм реалізується за допомогою віртуальних методів. Віртуальні методи дозволяють працювати з групами зв’язаних об’єктів універсальним способом.

Наприклад:

1. Є масив, який містить об’єкти базового та похідних класів. Щоб це було можливо, в якості типу масиву необхідно вказати базовий клас.
2. Методу, в якості аргумента, будуть передаватись об’єкти базового або похідних класів. В цьому випадку в якості типу параметру методу необхідно вказати базовий клас

9.Виклик конструктора базового класу.

Виклик конструктора базового класу

При створенні об’єкту похідного класу, як завжди, спочатку виконується конструктор базового класу, а потім – похідного.

Який саме конструктор базового класу буде викликатись?

Це вказується після списку параметрів відповідного конструктора похідного класу за допомогою ключового слова base з аргументами. Якщо в списку параметрів не вказано base з аргументами, то викликається конструктор за замовчуванням.

10.Перевизначені методи

Приклад використання

class Base {

public virtual void Display()

{

}

}

class Classics : Base {

public override void Display()

{

Console.WriteLine("Pages in Exem "+ Pagenumber);

}

11.Приховання методів

Приховування представляє визначення в класі-спадкоємця методу або властивості, які відповідає на ім'я і набору параметрів методу або властивості базового класу. Для приховування членів класу застосовується ключове слово new.

12.Абстрактні класи та методи

Ключове слово abstract дозволяє створювати класи і члени класів, які є неповними і повинні бути реалізовані в похідному класі.

Створювати екземпляри абстрактного класу не можна. Призначення абстрактного класу полягає в наданні загального визначення для базового класу, яке можуть спільно використовувати кілька похідних класів.

Абстрактні методи не мають реалізації, тому визначення такого методу закінчується крапкою з комою замість звичайного блоку методу. Класи, похідні від абстрактного класу, повинні реалізовувати всі абстрактні методи. Якщо абстрактний клас успадковує віртуальний метод з базового класу, абстрактний клас може перевизначити віртуальний метод за допомогою абстрактного методу.

13.Перевірка можливості приведення об‘єкта до відповідного типу та приведення об‘єктів до відповідного типу.

Оператор аs використовує явне перетворення. Оператор as не можна перевизначати, але можна перевизначити неявне та явне перетворення. Також при перетворенні об’єкту до іншого з використанням as, повертається NULL якщо перетворення не може відбутися.

14.Спеціфікатори доступу для типів верхнього рівня.

Abstract, virtual, override, new

15.Спеціфікатори доступу для членів типів.

Abstract, virtual, override, new

**Висновок:**

Протягом процесу виконання лабораторної роботи було створено 1 консольний застосунок що використовує передачу параметрів за посиланням та значенням. Було показано різницю в їх використанні. Було використано статичні методи та статичні змінні.