**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет   
імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інженерії програмного забезпечення**

**ЗВІТ**

з лабораторної роботи № 6

**«Перевантажені операції. Перетворення типів. Методи розширення»**

**Варіант 5**

Дисципліна «Об'єктно-орієнтоване програмування»

Спеціальність: **Інженерія програмного забезпечення**

121-ЛР.06-208.22120802

***Студент\_\_Д. М.  Дзундза***

*(підпис)*

*\_\_10.10.2021* \_\_

*(дата)*

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

*(підпис)*

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

*(дата)*

Миколаїв – 2021

**Лабораторна робота №6**

**Завдання:**

1. Оголосіть відповідний клас. Вкажіть, що він є запечатаним.
2. Продемонструйте для свого класу можливість додавання нового методу, не вносячи на цьому кроці зміни в клас («методи розширення»).
3. Створіть масив із об‘єктів Вашого класу. Введіть з клавіатури значень та знайдіть, чи є в масиві об‘єкт, стрінгове поле в якому співпадає з введенем рядком. Операція порівняння повинна виконуватись за допомогою операції порівняння..
4. Перевантажте в класі свого варіанту наступні операції:
   1. Студенти з непарними номерами в журналі:
      1. операцію віднімання одного об‘єкта свого класу від іншого;
      2. операцію декримента для об‘єкта свого класу;
      3. операцію додавання числа до об‘єкту класа;
      4. визначте метод неявного перетворення типу дійсного числа в об‘єкт класу;
      5. визначте метод явного перетворення об‘єкта класа в дійсне число.

**Лістинг коду:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace L6

{

sealed class Journal

{

int numofPages;

string text;

public Journal(int numOfPages, string txt)

{

this.numofPages = numOfPages;

this.text = txt;

}

public Journal(int result)

{

this.numofPages = result;

}

public Journal()

{

}

public Journal(Journal target)

{

this.text = target.text;

this.numofPages = target.numofPages;

}

public static Journal operator - (Journal firstJournal, Journal secondJournal)

{

StringBuilder sb = new StringBuilder(firstJournal.text);

foreach (char ch in secondJournal.text)

{

sb.Replace(ch, char.MinValue);

}

string finalString = sb.ToString();

return new Journal(firstJournal.numofPages - secondJournal.numofPages, finalString);

}

public static Journal operator --(Journal target)

{

Journal ta = new Journal(target.numofPages);

ta.numofPages++;

return ta;

}

public static int operator -(Journal target, int val)

{

return target.numofPages - val;

}

public static bool operator ==(Journal target, string val)

{

return target.text == val;

}

public static bool operator !=(Journal target, string val)

{

return !(target == val);

}

public static explicit operator Journal(int value)

{

return new Journal(value);

}

public static implicit operator Int32(Journal target)

{

return target.numofPages;

}

public override string ToString()

{

return $"Num of pages: {numofPages} Text: {text}";

}

}

static class ExtendingClass

{

public static int AdditionalMethod(this Journal t)

{

int a = 40;

int b = 80;

return a + b;

}

}

class Program

{

static void Main(string[] args)

{

Journal t1 = new Journal(2, "Dima");

Journal t2 = new Journal(3, "Dzundza");

Journal inSub = new Journal(0, " ");

Console.WriteLine(t1);

Console.WriteLine(t2);

inSub = t1 - t2;

Console.WriteLine($"Minus {inSub.ToString()} {Environment.NewLine}");

Journal t3 = new Journal(40, "Alexa");

Console.WriteLine(t3);

Journal t4 = new Journal(0, " ");

t4 = t3--;

Console.WriteLine($"Decrement: {t4} {Environment.NewLine}");

Journal t5 = new Journal(100, "Mary");

Console.WriteLine(t5);

int d = t5 - 27;

Console.WriteLine($"Minus to num {d} {Environment.NewLine}");

Journal resultValue = (Journal)80;

Console.WriteLine($"Implicit: {resultValue}");

Int32 expValue = (Int32)resultValue;

Console.WriteLine($"Explicit: {expValue}");

Journal other = (Journal)44;

Console.WriteLine($"Расширение в действии {other.AdditionalMethod()}");

Journal[] myArray = { t1, t2, inSub, t3, t4, t5 };

Console.WriteLine("Please, type in a string u are looking for.");

string myLine = Console.ReadLine();

foreach (var el in myArray)

{

if (el == myLine)

{

Console.WriteLine("There is a string.");

goto ThereIs;

}

}

Console.WriteLine($"There is no string like this.");

ThereIs:

Console.ReadKey();

}

}

}

**Результат роботи програми:**

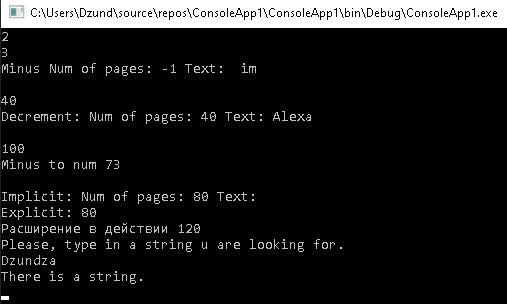


Рисунок 1 – Результат роботи програми

**Питання до захисту:**

1. Синтаксис оголошення перевантажених операцій.

public static <тип повертаємого значення> operatorXX(<список параметрів>{<тіло методу>}

static class ExtendingClass

{

public static int AdditionalMethod(this Journal t)

{

int a = 40;

int b = 80;

return a + b;

}

}

1. Вимоги до перевантаження операцій.

* не можна придумати свої знаки операцій;
* не можна змінити пріоритет операції, наприклад, якщо для чисел пріоритет операції множення (\*) вище, ніж складання (+), то і в класі користувача при перевантаженні цих операцій зберігається той самий порядок дій;
* метод, який реалізує перевантаження будь-якої операції, повинен бути статичним і відкритим;
* Параметри можна передавати в метод тільки за значенням (ref і out заборонені). Також заборонено змінювати значення параметрів всередені методів;
* не можна перевантажувати ні які форми операції присвоювання (=, + = і т.д.);
* операції порівняння необхідно реалізовувати парами (симетричними за змістом):<= и >= < и > == и !=
* якщо перевантажуються операції порівняння, то необхідно також перевантажити два методи , успадковані з класу object: Equals () і GetHashCode () (це гарний тон).
* Хоча б один з операндів повинен мати тип класу, для якого він перевантажується.

1. Як в коді MSILвідображаються перевантажені оператори?

op\_Addition…

1. Синтаксис оголошення методів перетворення типу.

public static implicit operator Point(int x){return new Point( x);}

public static explicit operator Int32(Point p) {return p.x;}

1. Що таке «методи розширення» та мета їх використання.

Це методи, що дозволяють розширювати функціонал класів без зміни класу напряму.