**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет   
імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра інженерії програмного забезпечення**

**ЗВІТ**

з лабораторної роботи № 8

**«Enum. Узагальнення»**

**Варіант 5**

Дисципліна «Об'єктно-орієнтоване програмування»

Спеціальність: **Інженерія програмного забезпечення**

121-ЛР.08-208.22120802

***Студент\_\_Д. М.  Дзундза***

*(підпис)*

*\_\_10.26.2021* \_\_

*(дата)*

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

*(підпис)*

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

*(дата)*

Миколаїв – 2021

**Лабораторна робота №8**

**Завдання:**

1. Використовуючи класи із свого варіанту, створіть додаток для демонстрації в коді використання інтерфейсів та поліморфізму з використанням інтерфейсів:

a.    Один із варіантів класу використайте в якості інтерфейсу.

b.    Два інших класи повинні реалізовувати оголошений інтерфейс.

c.    Кпас Program повинен містити метод, який в якості параметра приймає інтерфейс. При виклику методу передавати в якості параметра об‘єкти своїх класів.

d.    Створіть масив об‘єктів одного із класів свого варіанту.  Забезпечте можливість сортування масиву по кожному із трьох полів класу. Необхідно використати хоча б один раз узагальнену та неузагальнену форму інтерфейсів.

2.     Використовуючи класи свого варіанту(перелік 2),  виконайте наступне завдання:

Оголосіть класи таким чином, щоб один із класів, підходящий за змістом, містив масив об‘єктів іншого класу.

3.   Організуйте використання циклу foreach для перегляду об‘єктів масиву (використання інтерфейсів  IEnumarable та IEnumerator)

**Лістинг коду:**

using System;

using System.Collections;

using System.Text;

namespace ConsoleApp1

{

interface IJournal

{

int NumOfPages { get; }

string Text { get; set; }

void Show();

void PrintNumOfPages(IJournal ij);

}

class Book : IJournal, IComparable

{

string text, author;

public string Author { get => author; set => author = value; }

public int NumOfPages => (int)Math.Ceiling((double)(text.Length / 1000 + 1));

public string Text { get => text; set => text = value; }

public Book() { }

public Book(string text) { this.text = text; }

public Book(string text, string author) { this.text = text; this.author = author; }

public void Show()

{

Console.WriteLine(text);

}

public void PrintNumOfPages(IJournal ij)

{

Console.WriteLine($"PrintNumOfPages {ij.NumOfPages}");

}

public int CompareTo(object obj)

{

Book book = (Book)obj;

return text.CompareTo(book.text);

}

}

class PrintedEdition : IJournal

{

int height = 40, width = 1;

double sizeOfText = 1;

string text;

public int NumOfPages => (int)Math.Ceiling(sizeOfText \* Text.Length / height \* width);

public string Text { get => text; set => text = value; }

public PrintedEdition() { }

public PrintedEdition(string text) { this.text = text; }

public PrintedEdition(string text, int height) { this.text = text; this.height = height; this.width = height; }

public PrintedEdition(string text, int height, int width) { this.text = text; this.height = height; this.width = width; }

public void Show()

{

string[] separator = { "-\*-", "\n\n\n" };

int page = 0;

var texts = text.Split(separator, StringSplitOptions.RemoveEmptyEntries);

foreach (var txt in texts)

{

int lastHeight = 0;

StringBuilder new\_line = new StringBuilder("");

for (int iteration = 0; iteration < txt.Length / width / height + 1; iteration++)

{

lastHeight = (lastHeight == 0) ? 0 : lastHeight + 1;

int maxHeight = (txt.Length / width - lastHeight < height) ? txt.Length / width : lastHeight + height;

for (int h = lastHeight; h < maxHeight; h++)

{

int maxWidth = (txt.Length - width \* h > width) ? width \* h + width : txt.Length;

new\_line.Clear();

for (int w = width \* h; w < maxWidth; w++)

{

new\_line.Append(txt[w]);

}

Console.WriteLine(new\_line);

lastHeight = h;

}

page++;

Console.WriteLine($"Page {page}");

}

}

}

public void PrintNumOfPages(IJournal ij)

{

Console.WriteLine($"PrintNumOfPages {ij.NumOfPages}");

}

}

class SortNumOfPages : IComparer

{

public int Compare(object x, object y)

{

PrintedEdition first = (PrintedEdition)x;

PrintedEdition second = (PrintedEdition)y;

return first.NumOfPages.CompareTo(second.NumOfPages);

}

}

class SortLenOfText : IComparer

{

public int Compare(object x, object y)

{

Book first = (Book)x;

Book second = (Book)y;

return first.Text.Length.CompareTo(second.Text.Length);

}

}

class Animal

{

double price;

string name;

int age;

public double Price { get => price; set => price = value; }

public string Name { get => name; set => name = value; }

public int Age { get => age; set => age = value; }

public Animal(string name, int age, double price) { this.price = price; this.name = name;this.age = age; }

}

class Zoo : IEnumerable, IEnumerator

{

int pos = -1;

Animal[] animals;

public Zoo(Animal[] arrOfAnimals)

{

animals = new Animal[arrOfAnimals.Length];

for (int i = 0; i < arrOfAnimals.Length; i++)

{

animals[i] = arrOfAnimals[i];

}

}

public object Current => animals[pos];

public IEnumerator GetEnumerator()

{

return (IEnumerator)this;

}

public bool MoveNext()

{

pos++;

if (pos < animals.Length)

return true;

return false;

}

public void Reset()

{

pos = -1;

}

}

class Program

{

static void Main(string[] args)

{

Book book = new Book("Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.-\*-Sed ut perspiciatis, unde omnis iste natus error sit voluptatem", "Dima Dzundza");

Book book2 = new Book("quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio, cumque nihil impedit, quo minus id, quod maxime placeat, facere possimus, omnis voluptas assumenda ", "Dima Dzundza2");

Book book3 = new Book("khisguhsjkdbfdjshf", "Kul Ano");

PrintedEdition printedEdition1 = new PrintedEdition("Nemo enim ipsam voluptatem, quia voluptas sit, aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos, qui ratione voluptatem sequi nesciunt, neque porro quisquam est, qui dolorem ipsum, quia dolor sit, amet, consectetur.-\*-adipisci velit, sed quia non numquam eius modi tempora incidunt, ut labore et dolore magnam aliquam quaerat voluptatem", 10);

PrintedEdition printedEdition2 = new PrintedEdition("Second Text we need to sort", 10);

PrintedEdition printedEdition3 = new PrintedEdition("Third text we need to sort dolore magnam aliquam quaerat voluptatem", 10);

book.Show();

Console.WriteLine("PRINTED EDITION:::");

printedEdition1.Show();

printedEdition1.PrintNumOfPages(book);

Book[] books = { book, book2, book3 };

Array.Sort(books);

foreach (var item in books)

{

Console.WriteLine(item.Author);

}

Array.Sort(books, new SortLenOfText());

foreach (var item in books)

{

Console.WriteLine(item.Author);

}

PrintedEdition[] printedEditions = { printedEdition1, printedEdition2, printedEdition3 };

Array.Sort(printedEditions, new SortNumOfPages());

foreach (var item in printedEditions)

{

item.PrintNumOfPages(item);

}

Animal[] animals = {

new Animal("Wolf", 15, 9.5),

new Animal("Fox", 4, 2),

new Animal("Rat", 6, 5),

new Animal("Slime", 123, 3),

new Animal("Nox", 4, 12),

new Animal("Lizard", 7, 15),

};

Zoo zoo = new Zoo(animals);

foreach (Animal animal in zoo)

{

Console.WriteLine($"Name {animal.Name} Price {animal.Price} Age {animal.Age}");

}

Console.ReadKey();

}

}

}

**Питання до захисту:**

1. Перерахуйте модифікатори доступу, які може мати інтерфейс.?

Internal public

1. Перерахуйте властивості інтерфейсів.

* Інтерфейси можуть містити лише оголошення методів, властивостей, індексаторів та подій.
* Інтерфейси не можуть містити змінних.
* При оголошенні членів інтерфейсів не дозволяються модифікатори. Члени інтерфейсів завжди неявно є public і не можуть бути static.
* Інтерфейс верхнього рівня може бути оголошений з модифікатором public або internal, а внутрішній – з тими ж модифікаторами, що і інші члени класу.
* Не можна напряму створити об'єкт інтерфейсу. Ім'я інтерфейсу може бути типом змінної, але містити змінна може тільки посилання на об'єкт класу, який реалізує цей інтерфейс.

1. IComparable (дві форми)

class Class1: IComparable

public int CompareTo(object obj){

return …

}

class Class2 : IComparable<ТИП>{

public int CompareTo(<Тип параметру>){

return …

}

}

1. IComparer (дві форми)

class Class1 : IComparer{

public int Compare(object x, object y){

return …

}

}

class Class2: IComparer<ТИП>{

public int Compare(<Тип> x, <Тип> y){

return …

}}