### Кам’янець-Подільський національний університет імені Івана Огієнка

### КАФЕДРА КОМП’ЮТЕРНИХ НАУК

### Навчальна дисципліна «Об’єктно-орієнтоване програмування»

### ЛАБОРАТОРНА РОБОТА #0105

## Тема:

Інтерфейси

Варіант №2

**Виконав**:  
студент 1-го курсу  
групи KNms1-B23  
Веретко С. І.

**Прийняла**:  
старший викладач,  
Мястковська М.О.

Кам’янець-Подільський – 2023

1. Теоретичні відомості

*Інтерфейс* – іменований набір абстрактних членів. Інтерфейс схожий на клас, але надає лише специфікацію, а не реалізацію для своїх членів. Специфічні члени, що визначаються інтерфейсом, залежать від того, яку точно поведінку він моделює. Іншими словами, інтерфейс виражає поведінку, яку заданий клас або структура може обрати для підтримки. Інтерфейс (інтерфейсний тип) володіє наступними особливостями:

* всі члени інтерфейсу є неявно абстрактними; на противагу цьому, клас може мати як абстрактні члени, так і конкретні члени з реалізацією;
* клас (або структура) може реалізовувати декілька інтерфейсів; на противагу цьому клас може бути успадкований лише від одного класу, а структура взагалі не підтримує успадкування.

Опис інтерфейсу схожий на опис класу, але при цьому ніякої реалізації не надається, оскільки всі члени інтерфейсу неявно абстрактні. Всі ці члени обов’язково мають бути реалізовані класами і структурами, які включають даний інтерфейс.

Інтерфейс може містити лише методи, властивості, події та індексатори, що відповідає членам класу, які можуть бути абстрактними. Проте інтерфейс, що встановлюється абстрактним класом має суттєве обмеження: члени, що визначені абстрактним батьківським класом, підтримуються лише дочірніми класами. Тому класи, що не мають спільного батьківського класу, окрім System.Object, не можуть реалізовувати спільні інтерфейси. Крім того, класи можуть мати лише один батьківський клас, а реалізовувати можуть декілька інтерфейсів.

Члени інтерфейсу завжди неявно public, і для них не можна оголошувати модифікатори доступу. Реалізація інтерфейсу означає надання реалізації public для всіх його членів. Жоден з членів інтерфейсу не може бути оголошений як static

# 2. Завдання для виконання

Завдання 1

**Варіант 2.**

Описати інтерфейс **IFigure** з членами:

1) метод, що виводить на екран тип фігури;

2) метод, що виводить на екран площу фігури;

3) властивість, що відповідає за перший лінійний розмір фігури;

4) властивість, що відповідає за другий лінійний розмір фігури;

5) метод, що виводить на екран довжину діагоналі фігури.

Описати інтерфейс IColoredFigure, що успадковується від інтерфейсу IFigure та містить властивість, що відповідає за колір фігури, та метод, що виводить на екран колір фігури.

Описати клас Rectangle, та ColoredRectangle що реалізовують інтерфейси IFigure та IColoredFigure відповідно.

Створити не менше трьох екземплярів кожного класу, сформувати з них один масив.

Створити метод, який з масиву вибирає лише ті елементи, що підтримують колір.

Завдання 2

### Варіант 2.

Створити інтерфейс **Інструмент** та реалізувати його класи Гітара, Барабан та Труба.

Інтерфейс Інструмент повинен містити метод play() та змінну String KEY = "До мажор".

Гітара містить змінні класу Гітара - *кількість* *струн*, Барабан - *розмір*, Труба - *діаметр*.

Створити масив типу Інструмент, що містить інструменти різного типу.

У циклі викликати метод play() для кожного інструменту, який повинен виводити рядок "Грає такий-то інструмент з такими-то характеристиками".

Завдання 3

Створити клас **Shop**, що містить масив із розмірами одягу (XXS, XS, S, M, L). Клас містить метод getDescription, який повертає рядок "Дорослий розмір".

Перевизначити метод getDescription - для константи XXS метод має повертати рядок "Дитячий розмір".

Також клас має містити числове значення euroSize (32, 34, 36, 38, 40), що відповідає кожному розміру. Створити конструктор, який приймає на вхід euroSize.

Створити інтерфейси "Чоловічий Одяг" з методом "одягти Чоловіка" та "Жіночий Одяг" з методом "одягти Жінку".

Створити абстрактний клас Одяг, що містить змінні – розмір одягу, вартість, колір.

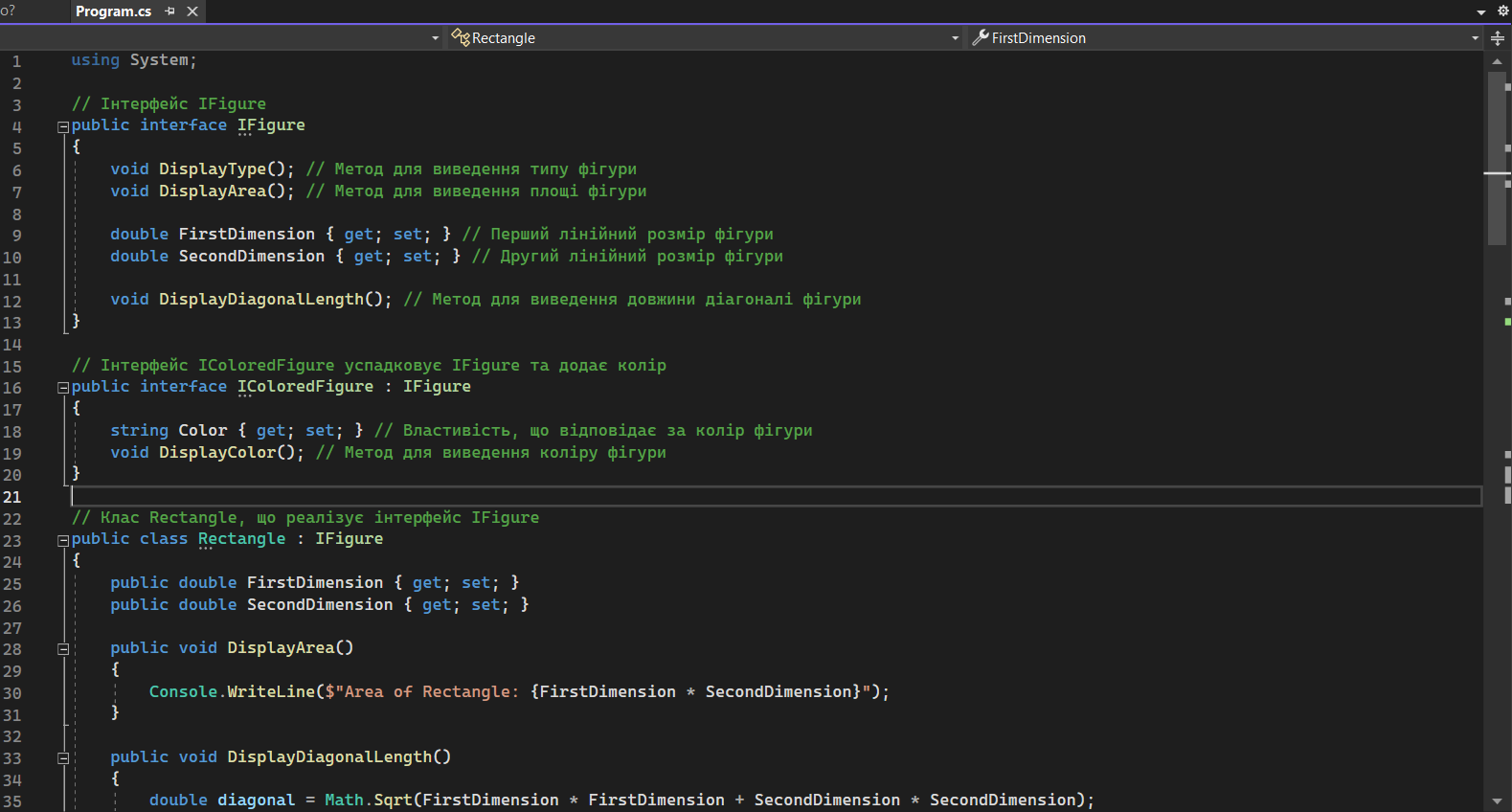
Створити класи спадкоємці Одягу - Футболка (реалізує інтерфейси "Чоловічий Одяг" та "Жіночий Одяг"), Штани (реалізує інтерфейси "Чоловічий Одяг" та "Жіночий Одяг"), Спідниця (реалізує інтерфейси "Жіночий Одяг"), Краватка (реалізує інтерфейси " Чоловічий одяг").

Створити масив, що містить усі типи одягу.

Створити клас Ательє, що містить методи одягнути Жінку, одягнути Чоловіка, на вхід яких надходитиме масив, що містить всі типи одягу.

Метод одягу Жінку виводить на консоль всю інформацію про жіночий одяг. Те ж саме для методу одягнути Чоловіка.

3. Послідовність виконання завданнь



4. Лістинг програми

Завдання 5.1  
using System;

// Інтерфейс IFigure

public interface IFigure

{

void DisplayType(); // Метод для виведення типу фігури

void DisplayArea(); // Метод для виведення площі фігури

double FirstDimension { get; set; } // Перший лінійний розмір фігури

double SecondDimension { get; set; } // Другий лінійний розмір фігури

void DisplayDiagonalLength(); // Метод для виведення довжини діагоналі фігури

}

// Інтерфейс IColoredFigure успадковує IFigure та додає колір

public interface IColoredFigure : IFigure

{

string Color { get; set; } // Властивість, що відповідає за колір фігури

void DisplayColor(); // Метод для виведення коліру фігури

}

// Клас Rectangle, що реалізує інтерфейс IFigure

public class Rectangle : IFigure

{

public double FirstDimension { get; set; }

public double SecondDimension { get; set; }

public void DisplayArea()

{

Console.WriteLine($"Area of Rectangle: {FirstDimension \* SecondDimension}");

}

public void DisplayDiagonalLength()

{

double diagonal = Math.Sqrt(FirstDimension \* FirstDimension + SecondDimension \* SecondDimension);

Console.WriteLine($"Diagonal length of Rectangle: {diagonal}");

}

public void DisplayType()

{

Console.WriteLine("Type: Rectangle");

}

}

// Клас ColoredRectangle, що реалізує інтерфейс IColoredFigure

public class ColoredRectangle : IColoredFigure

{

public double FirstDimension { get; set; }

public double SecondDimension { get; set; }

public string Color { get; set; }

public void DisplayArea()

{

Console.WriteLine($"Area of ColoredRectangle: {FirstDimension \* SecondDimension}");

}

public void DisplayDiagonalLength()

{

double diagonal = Math.Sqrt(FirstDimension \* FirstDimension + SecondDimension \* SecondDimension);

Console.WriteLine($"Diagonal length of ColoredRectangle: {diagonal}");

}

public void DisplayType()

{

Console.WriteLine("Type: ColoredRectangle");

}

public void DisplayColor()

{

Console.WriteLine($"Color of ColoredRectangle: {Color}");

}

}

class Program

{

static void Main(string[] args)

{

// Створення екземплярів класів Rectangle та ColoredRectangle

Rectangle rect1 = new Rectangle { FirstDimension = 5, SecondDimension = 10 };

Rectangle rect2 = new Rectangle { FirstDimension = 3, SecondDimension = 7 };

Rectangle rect3 = new Rectangle { FirstDimension = 4, SecondDimension = 8 };

ColoredRectangle coloredRect1 = new ColoredRectangle { FirstDimension = 6, SecondDimension = 12, Color = "Blue" };

ColoredRectangle coloredRect2 = new ColoredRectangle { FirstDimension = 2, SecondDimension = 4, Color = "Red" };

ColoredRectangle coloredRect3 = new ColoredRectangle { FirstDimension = 7, SecondDimension = 14, Color = "Green" };

// Створення масиву з екземплярів

IFigure[] figures = { rect1, rect2, rect3, coloredRect1, coloredRect2, coloredRect3 };

// Метод для вибору лише елементів, що підтримують колір

DisplayColoredFigures(figures);

}

static void DisplayColoredFigures(IFigure[] figures)

{

Console.WriteLine("Colored Figures:");

foreach (var figure in figures)

{

if (figure is IColoredFigure coloredFigure)

{

coloredFigure.DisplayType();

coloredFigure.DisplayColor();

}

}

}

}

Завдання 5.2

using System;

// Інтерфейс Інструмент

public interface IInstrument

{

string KEY { get; } // Змінна інтерфейсу

void Play(); // Метод Play()

}

// Клас Гітара, що реалізує інтерфейс Інструмент

public class Guitar : IInstrument

{

public string KEY { get; } = "До мажор"; // Реалізація змінної інтерфейсу

public int NumberOfStrings { get; private set; } // Змінна класу Гітара - кількість струн

public Guitar(int numberOfStrings)

{

NumberOfStrings = numberOfStrings;

}

public void Play()

{

Console.WriteLine($"Playing guitar with {NumberOfStrings} strings");

}

}

// Клас Барабан, що реалізує інтерфейс Інструмент

public class Drum : IInstrument

{

public string KEY { get; } = "До мажор"; // Реалізація змінної інтерфейсу

public int Size { get; private set; } // Змінна класу Барабан - розмір

public Drum(int size)

{

Size = size;

}

public void Play()

{

Console.WriteLine($"Playing drum with size {Size}");

}

}

// Клас Труба, що реалізує інтерфейс Інструмент

public class Trumpet : IInstrument

{

public string KEY { get; } = "До мажор"; // Реалізація змінної інтерфейсу

public int Diameter { get; private set; } // Змінна класу Труба - діаметр

public Trumpet(int diameter)

{

Diameter = diameter;

}

public void Play()

{

Console.WriteLine($"Playing trumpet with diameter {Diameter}");

}

}

class Program

{

static void Main(string[] args)

{

// Створення масиву типу Інструмент

IInstrument[] instruments = {

new Guitar(6),

new Drum(20),

new Trumpet(10)

};

// Виклик методу Play() для кожного інструменту у циклі

foreach (var instrument in instruments)

{

instrument.Play();

}

}

}

Завдання 5.3

using System.IO;

// Оголошуємо інтерфейс "Чоловічий Одяг" з методом "одягти Чоловіка".

interface IMensClothing

{

void DressMan();

}

// Оголошуємо інтерфейс "Жіночий Одяг" з методом "одягти Жінку".

interface IWomensClothing

{

void DressWoman();

}

// Абстрактний клас Одяг, що містить загальні характеристики одягу.

abstract class Clothing

{

public string Size { get; set; }

public decimal Price { get; set; }

public string Color { get; set; }

public Clothing(string size, decimal price, string color)

{

Size = size;

Price = price;

Color = color;

}

}

// Клас Футболка, який реалізує інтерфейси "Чоловічий Одяг" та "Жіночий Одяг".

class TShirt : Clothing, IMensClothing, IWomensClothing

{

public TShirt(string size, decimal price, string color) : base(size, price, color) { }

public void DressMan()

{

Console.WriteLine($"Men's T-Shirt: Size {Size}, Color {Color}, Price ${Price}");

}

public void DressWoman()

{

Console.WriteLine($"Women's T-Shirt: Size {Size}, Color {Color}, Price ${Price}");

}

}

// Клас Штани, який реалізує інтерфейси "Чоловічий Одяг" та "Жіночий Одяг".

class Pants : Clothing, IMensClothing, IWomensClothing

{

public Pants(string size, decimal price, string color) : base(size, price, color) { }

public void DressMan()

{

Console.WriteLine($"Men's Pants: Size {Size}, Color {Color}, Price ${Price}");

}

public void DressWoman()

{

Console.WriteLine($"Women's Pants: Size {Size}, Color {Color}, Price ${Price}");

}

}

// Клас Спідниця, який реалізує інтерфейс "Жіночий Одяг".

class Skirt : Clothing, IWomensClothing

{

public Skirt(string size, decimal price, string color) : base(size, price, color) { }

public void DressWoman()

{

Console.WriteLine($"Women's Skirt: Size {Size}, Color {Color}, Price ${Price}");

}

}

// Клас Краватка, який реалізує інтерфейс "Чоловічий Одяг".

class Tie : Clothing, IMensClothing

{

public Tie(string size, decimal price, string color) : base(size, price, color) { }

public void DressMan()

{

Console.WriteLine($"Men's Tie: Size {Size}, Color {Color}, Price ${Price}");

}

}

// Клас Shop, який містить масив із розмірами одягу та має метод getDescription.

class Shop

{

public List<string> ClothingSizes { get; } = new List<string> { "XXS", "XS", "S", "M", "L" };

public int EuroSize { get; set; }

public Shop(int euroSize)

{

EuroSize = euroSize;

}

public virtual string GetDescription()

{

return "Дорослий розмір";

}

public string GetSizeByEuroSize()

{

if (EuroSize >= 32 && EuroSize <= 40)

{

return ClothingSizes[EuroSize - 32];

}

return "Невідомий розмір";

}

}

// Клас-спадкоємець від Shop, що перевизначає метод getDescription для XXS.

class ShopWithChildSize : Shop

{

public ShopWithChildSize(int euroSize) : base(euroSize) { }

public override string GetDescription()

{

if (GetSizeByEuroSize() == "XXS")

{

return "Дитячий розмір";

}

return "Дорослий розмір";

}

}

// Клас Ательє, що містить методи одягнути Жінку та одягнути Чоловіка.

class Atelier

{

public static void DressWoman(List<IWomensClothing> clothes)

{

Console.WriteLine("Одягаємо Жінку:");

foreach (IWomensClothing item in clothes)

{

item.DressWoman();

}

}

public static void DressMan(List<IMensClothing> clothes)

{

Console.WriteLine("Одягаємо Чоловіка:");

foreach (IMensClothing item in clothes)

{

item.DressMan();

}

}

}

class Program

{

public static void Main()

{

// Створюємо список об'єктів одягу різних типів.

List<Clothing> clothes = new List<Clothing>

{

new TShirt("M", 25.99M, "Red"),

new TShirt("S", 19.99M, "Blue"),

new Pants("L", 49.99M, "Black"),

new Skirt("XS", 39.99M, "Green"),

new Tie("L", 29.99M, "Navy")

};

// Виводимо інформацію про всі види одягу.

Console.WriteLine("Інформація про одяг:");

foreach (Clothing item in clothes)

{

Console.WriteLine($"Одяг: {item.GetType().Name}, Розмір: {item.Size}, Ціна: ${item.Price}, Колір: {item.Color}");

}

// Створюємо магазин з розміром XXS.

ShopWithChildSize shop = new ShopWithChildSize(32);

// Виводимо опис розміру з магазину.

Console.WriteLine($"Опис розміру з магазину: {shop.GetDescription()}");

// Використовуємо ательє для одягу Жінку та Чоловіка.

Console.WriteLine("\nВиведення одягу Жінки:");

Atelier.DressWoman(clothes.OfType<IWomensClothing>().ToList());

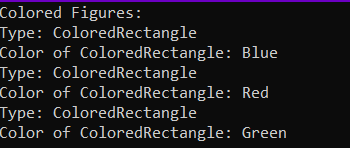
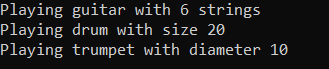
Console.WriteLine("\nВиведення одягу Чоловіка:");

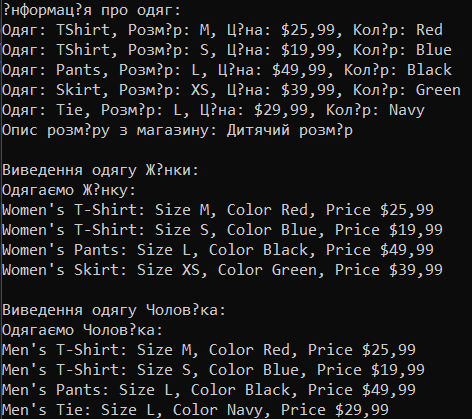
Atelier.DressMan(clothes.OfType<IMensClothing>().ToList());

}

}

5. Результат роботи програми



6. [xanax4rl/lab (github.com)](https://github.com/xanax4rl/lab)