**Занятие № 9**

**Дата выполнения работы:** 28.04.2023

# **Тема работы:** «Наследование, полиморфизм, абстрактные классы и интерфейсы»

**Ход работы**

**Задание 1**

1) Определить иерархию и композицию классов (в соответствии с вариантом), реализовать классы. Если необходимо расширьте по своему усмотрению иерархию для выполнения всех пунктов работы.

Каждый класс должен иметь отражающее смысл название и информативный

состав. При кодировании должны быть использованы соглашения об оформлении кода code convention. В одном из классов переопределите все методы, унаследованные от Object.

2) В проекте должны быть интерфейсы и абстрактный класс(ы). Использовать виртуальные методы и переопределение.

3) Сделайте один из классов герметизированным (бесплодным).

4) Добавьте в интерфейсы (интерфейс) и абстрактный класс одноименные методы. Дайте в наследуемом классе им разную реализацию и вызовите эти методы.

5) Написать демонстрационную программу, в которой создаются объекты различных классов. Поработать с объектами через ссылки на абстрактные классы и интерфейсы. В этом случае для идентификации типов объектов использовать операторы is или as.

6) Во всех классах (иерархии) переопределить метод ToStringQ, который выводит информацию о типе объекта и его текущих значениях.

7) Создайте дополнительный класс Printer c полиморфным методом IAmPrinting(SomeAbstractClassorInterface someobj). Формальным параметром метода должна быть ссылка на абстрактный класс или наиболее общий интерфейс в вашей иерархии классов. В методе iIAmPrinting определите тип объекта и вызовите ToString(). В демонстрационной программе создайте массив, содержащий ссылки на разнотипные объекты ваших классов по иерархии, а также объект класса Printer и последовательно вызовите его метод IAmPrinting со всеми ссылками в качестве

аргументов.

**Далее приведен перечень классов:**

**Вариант 1** ПО, Набор операций, Текстовый процессор, Word, Вирус,

CConficker Игрушка, Сапер, Разработчик.

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

// Создаем новый экземпляр Word

Word word = new Word(2019);

// Создаем новый вирус и запускаем его

Virus virus = new Virus();

virus.Infect(textBox1);

// Создаем новый экземпляр CConficker и запускаем его

CConficker cconficker = new CConficker();

cconficker.Infect(textBox2);

// Создаем новый экземпляр Minesweeper и запускаем игру

Minesweeper minesweeper = new Minesweeper();

minesweeper.Play(textBox3);

// Создаем нового разработчика и пусть он разработает новое ПО

Developer developer = new Developer("Иван Иванов");

Software software = new Software("Новая программа", 1);

developer.DevelopSoftware(software, textBox4);

}

}

public abstract class OperationSet

{

public abstract void AddOperation(string operationName);

public abstract void RemoveOperation(string operationName);

}

public class TextProcessor

{

private OperationSet \_operationSet;

public TextProcessor(OperationSet operationSet)

{

\_operationSet = operationSet;

}

public void AddOperation(string operationName)

{

\_operationSet.AddOperation(operationName);

}

public void RemoveOperation(string operationName)

{

\_operationSet.RemoveOperation(operationName);

}

}

public class Software

{

private string \_name;

private int \_version;

public Software(string name, int version)

{

\_name = name;

\_version = version;

}

public string Name

{

get { return \_name; }

}

public int Version

{

get { return \_version; }

}

}

public class Word : Software

{

public Word(int version) : base("Microsoft Word", version)

{

}

}

public class Virus

{

public virtual void Infect(TextBox textBox)

{

textBox.Text = ("Компьютер заражен вирусом.");

}

}

public class CConficker : Virus

{

public override void Infect(TextBox textBox)

{

textBox.Text = ("Компьютер заражен вирусом Conficker.");

}

}

// Класс, представляющий игрушку

public class Toy

{

public virtual void Play(TextBox textbox)

{

textbox.Text = ("Игрушка играет.");

}

}

// Класс, представляющий игру Сапер

public class Minesweeper : Toy

{

public override void Play(TextBox textbox)

{

textbox.Text = ("Игра Сапер запущена.");

}

}

public class Developer

{

private string \_name;

public Developer(string name)

{

\_name = name;

}

public string Name

{

get { return \_name; }

}

public void DevelopSoftware(Software software, TextBox textbox)

{

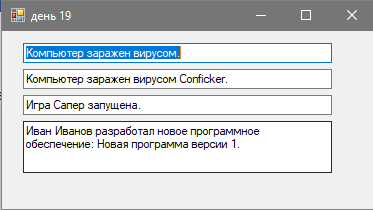
textbox.Text = ($"{\_name} разработал новое программное обеспечение: {software.Name} версии {software.Version}.");

}

}

}

**Результат:**

****

**Вывод:** ООП в C# включает в себя концепции наследования, полиморфизма, абстрактных классов и интерфейсов. Наследование позволяет создавать классы на основе уже существующих классов, наследуя их свойства и методы. Полиморфизм позволяет использовать объекты производных классов как объекты базовых классов, что упрощает программирование и улучшает читаемость кода. Абстрактные классы и интерфейсы являются абстрактными типами данных, определяющими методы и свойства, которые должны быть реализованы в производных классах. Абстрактные классы могут содержать как абстрактные, так и обычные методы и свойства, а интерфейсы - только абстрактные методы и свойства.