**Занятие № 8**

**Дата выполнения работы:** 28.04.2023

# **Тема работы: «Перечисления. Структуры. Интерфейсы»**

**Ход работы**

**Задание 1  
interface Корабль←abstract class Военный Корабль←class Авианосец.**

**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 \_1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Корабль корабль1 = new Авианосец();

textBox1.AppendText(корабль1.Передвижение() + Environment.NewLine);

Военный\_Корабль корабль2 = new Авианосец();

textBox1.AppendText(корабль2.Передвижение() + Environment.NewLine);

textBox1.AppendText(корабль2.Огонь() + Environment.NewLine);

}

interface Корабль

{

string Передвижение();

}

abstract class Военный\_Корабль : Корабль

{

public string Передвижение()

{

return "Корабль плывет";

}

public abstract string Огонь();

}

class Авианосец : Военный\_Корабль

{

public override string Огонь()

{

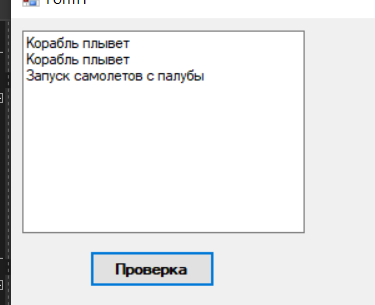
return "Запуск самолетов с палубы";

}

}

}

}

**Результат:**

**Задание 2  
Создать абстрактный класс Работник фирмы и подклассы Менеджер, Аналитик, Программист, Тестировщик, Дизайнер.**

**Листинг программы:**using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.Design;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace \_2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

// Создание массива объектов суперкласса

Employee[] employees = new Employee[5];

// Заполнение массива объектами подклассов

employees[0] = new Manager { Name = "John", Age = 35, Position = "Manager", Salary = 5000, Department = "Sales", Bonus = 1000 };

employees[1] = new Analyst { Name = "Mary", Age = 28, Position = "Analyst", Salary = 4000, Project = "Marketing Research" };

employees[2] = new Programmer { Name = "Tom", Age = 40, Position = "Programmer", Salary = 6000, Language = "C#" };

employees[3] = new Tester { Name = "Kate", Age = 25, Position = "Tester", Salary = 3500, Tool = "Selenium" };

employees[4] = new Designer { Name = "Alex", Age = 30, Position = "Designer", Salary = 4500, Type = "UI/UX" };

// Вывод информации о каждом работнике в TextBox

foreach (Employee employee in employees)

{

string info = "Name: " + employee.Name + "\r\n";

info += "Age: " + employee.Age + "\r\n";

info += "Position: " + employee.Position + "\r\n";

info += "Salary: " + employee.Salary + "\r\n";

info += "Calculated Salary: " + employee.CalculateSalary() + "\r\n\r\n";

textBox1.Text += info;

}

}

}

// Абстрактный класс Работник фирмы

public abstract class Employee

{

public string Name { get; set; }

public int Age { get; set; }

public string Position { get; set; }

public double Salary { get; set; }

// Абстрактный метод для расчета зарплаты

public abstract double CalculateSalary();

}

// Подкласс Менеджер

public class Manager : Employee

{

public string Department { get; set; }

public double Bonus { get; set; }

// Реализация метода для расчета зарплаты

public override double CalculateSalary()

{

return Salary + Bonus;

}

}

// Подкласс Аналитик

public class Analyst : Employee

{

public string Project { get; set; }

// Реализация метода для расчета зарплаты

public override double CalculateSalary()

{

return Salary \* 1.2;

}

}

// Подкласс Программист

public class Programmer : Employee

{

public string Language { get; set; }

// Реализация метода для расчета зарплаты

public override double CalculateSalary()

{

return Salary \* 1.5;

}

}

// Подкласс Тестировщик

public class Tester : Employee

{

public string Tool { get; set; }

// Реализация метода для расчета зарплаты

public override double CalculateSalary()

{

return Salary \* 1.1;

}

}

// Подкласс Дизайнер

public class Designer : Employee

{

public string Type { get; set; }

// Реализация метода для расчета зарплаты

public override double CalculateSalary()

{

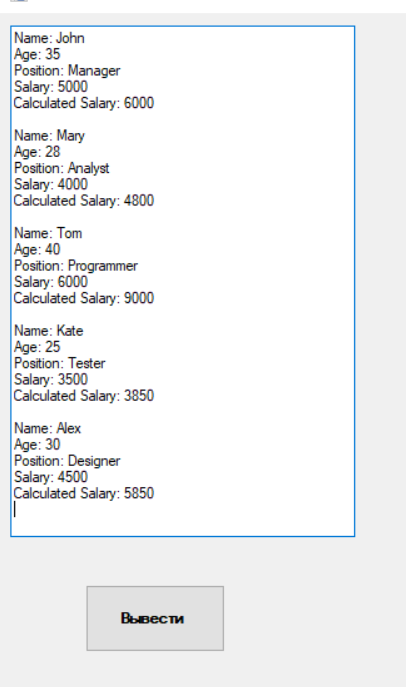
return Salary \* 1.3;

}

}

}

**Результат:**



**Вывод:**Интерфейсы в C# - это синтаксический элемент языка, который позволяет определить контракт, который должны реализовать классы или структуры. Интерфейсы определяют методы, свойства, индексаторы и события, которые должны быть реализованы в классах, которые реализуют данный интерфейс.

Интерфейсы предоставляют механизм полиморфизма, который позволяет использовать один и тот же код с различными типами объектов, которые реализуют интерфейс. Это делает код более гибким и позволяет упростить его поддержку и расширение.