# Лабораторная работа № 18

**Тема:** Разработка программ на языке С# с использованием стандартных интерфейсов.

**Цель работы:**

1. Формирование навыков разработки программ на языке С# с использованием интерфейсов.
2. Изучение особенностей реализации стандартных интерфейсов на языке С#.

. **Оборудование:**

1. ПК
2. Программное обеспечение: ОС Windows, среда Visual Studio.Net

**Контрольные вопросы:**

1) Дайте определение интерфейса. Для каких целей используются интерфейсы?

Интерфейс представляет ссылочный тип, который может определять некоторый функционал (набор методов и свойств без реализации). Затем этот функционал реализуют классы и структуры, которые применяют данные интерфейсы.

2) Как можно получить ссылку на интерфейс? Приведите примеры.

Для получения ссылки на интерфейс нужно указать его имя. Например, чтобы обратиться к полю интерфейса IMovable нужно указать:

IMovable.maxSpeed;

3) В чем особенность иерархий интерфейсов?

Один интерфейс может наследовать другой. Синтаксис наследования интерфейсов такой же, как и у классов. Когда в классе реализуется один интерфейс, наследующий другой, в нем должны быть реализованы все члены, определенные в цепочке наследования интерфейсов.

Таким образом, интерфейсы могут быть организованы в иерархии. Как и в иерархии интерфейсов, когда какой-то интерфейс расширяет существующий, он наследует все абстрактные члены своего родителя (или родителей). Конечно, в отличие от классов, производные интерфейсы никогда не наследуют саму реализацию. Вместо этого они просто расширяют собственное определение за счет добавления дополнительных абстрактных членов.

Использовать иерархию интерфейсов может быть удобно, когда нужно расширить функциональность определенного интерфейса без нарушения уже существующих кодовых баз.

Например:

public interface A

{

int Sum();

}

public interface B : A

{

int Del();

}

class MyOperation : B

{

int x = 10, y = 5;

public int Sum()

{

return x + y;

}

public int Del()

{

return x / y;

}

}

4) Какие стандартные интерфейсы Вы знаете? Приведите примеры их реализации.

В библиотеке классов .NET определено множество стандартных интерфейсов, задающих желаемое поведение объектов. Например, интерфейс IComparable задает метод сравнения объектов на «больше-меньше», что позволяет выполнять их сортировку.

Реализация интерфейсов IEnumerable и IEnumerator дает возможность просматривать содержимое объекта с помощью foreach, а реализация интерфейса ICloneable – клонировать объекты.

interface IComparable

{

int CompareTo(object obj)

}

Метод должен возвращать:

0, если текущий объект и параметры равны;

Отрицательное число, если текущий объект меньше параметра;

Положительное число, если текущий объект больше параметра

Пример реализации интерфейса

class Monster : IComparable

{

public int CompareTo(object obj)

{

monster temp = (Monster) obj;

if (this.health > temp.health) return 1;

if (this.health < temp.health) return -1;

return 0;

}

}

5) Чем отличается синтаксис интерфейса от синтаксиса абстрактного класса?

Абстрактные классы могут содержать все остальные члены, которых не может быть в интерфейсе, и не все методы/свойства в абстрактном классе должны быть абстрактными

6) Какие объекты языка С# могут быть членами интерфейсов?

Интерфейсы могут определять следующие сущности:

1. Методы

2. Свойства

3. Индексаторы

4. События

5. Статические поля и константы

7) Может ли класс реализовывать множественные интерфейсы?

Да, класс может реализовывать множественные интерфейсы. Для этого нужно указать их через запятую.

8) Какой модификатор доступа соответствует интерфейсу?

Интерфейсы по умолчанию имеют уровень доступа internal, то есть такой интерфейс доступен только в рамках текущего проекта. Но с помощью модификатора public мы можем сделать интерфейс общедоступным.

9) Возможно ли наследование интерфейсов?

Да, наследование интрефейсов возможно

10) Приведите синтаксис интерфейса в общем виде.

interface название\_интерфейса

{

тело\_интерфейса;

}

Проиллюстрируйте его фрагментом программы на языке С#.

11) Возможно ли наследование в интерфейсах?

public interface A

{

int Sum();

}

public interface B : A

{

int Del();

}

class MyOperation : B

{

int x = 10, y = 5;

public int Sum()

{

return x + y;

}

public int Del()

{

return x / y;

}

}

12) Есть ли различия в синтаксисе наследования интерфейсов и синтаксисе

наследования классов?

Различий в синтаксисе наследования интерфейсов и синтаксисе наследования классов нет

public interface A

{

int Sum();

}

public interface B : A

{

int Del();

}

**Ход работы:**

1. **Для разработанной иерархии классов в лабораторной работе №16 – 17 добавить реализацию интерфейса IComparable и отсортировать объекты базового класса.**
2. Разработать алгоритм задачи и представить его в виде  программы на С#
3. Используя средства Visual Studio.Net создать файл с программой и выполнить тестирование и отладку программы
4. Результаты представить в виде отчета
5. Сделать вывод о проделанной работе

**Содержание отчета:**

Задание на лабораторную работу

**Вариант 12**

Двигатель, дизель, двигатель внутреннего сгорания, реактивный двигатель.

Листинг с исходным кодом

using System;

abstract class Двигатель : IComparable<Двигатель> //объявление абстрактного класса "Двигатель" и указание, что он реализует интерфейс "IComparable" с типом "Двигатель".

{

public abstract void запуск(); //Объявление абстрактного метода "запуск()", который должен быть реализован в наследуемых классах.

public virtual void остановка()

{

Console.WriteLine("Двигатель остановлен");

}

public virtual void информация()

{

Console.WriteLine("Информация о двигателе");

}

public int CompareTo(Двигатель other)

// Если текущий объект меньше, чем переданный объект возвращает отрицательное число.

//Если текущий объект больше чем переданный объект возвращает положительное число.

// Если текущий объект равен переданному объекту возвращает ноль.

{

// позволяет сравнить два объекта типа "Двигатель".

Console.WriteLine(this.GetType().Name.CompareTo(other.GetType().Name));

return this.GetType().Name.CompareTo(other.GetType().Name);

}

}

class Дизель : Двигатель

{

private int мощность;

public Дизель(int \_мощность)

{

мощность = \_мощность;

}

public override void запуск()

{

Console.WriteLine("Дизель запущен");

}

public override void остановка()

{

Console.WriteLine("Дизель остановлен");

}

public override void информация()

{

Console.WriteLine("Информация о дизеле. Мощность: " + мощность);

}

}

class ДвигательВнутреннегоСгорания : Двигатель

{

private int обороты;

public ДвигательВнутреннегоСгорания(int \_обороты)

{

обороты = \_обороты;

}

public override void запуск()

{

Console.WriteLine("Двигатель внутреннего сгорания запущен");

}

public override void остановка()

{

Console.WriteLine("Двигатель внутреннего сгорания остановлен");

}

public override void информация()

{

Console.WriteLine("Информация о двигателе внутреннего сгорания. Обороты: " + обороты);

}

}

class РеактивныйДвигатель : Двигатель

{

private int тяга;

public РеактивныйДвигатель(int \_тяга)

{

тяга = \_тяга;

}

public override void запуск()

{

Console.WriteLine("Реактивный двигатель запущен");

}

public override void остановка()

{

Console.WriteLine("Реактивный двигатель остановлен");

}

public override void информация()

{

Console.WriteLine("Информация о реактивном двигателе. Тяга: " + тяга);

}

}

// Остальной код остается без изменений

class Program

{

static void Main(string[] args)

{

Двигатель[] двигатели = new Двигатель[3];

двигатели[0] = new Дизель(100);

двигатели[1] = new ДвигательВнутреннегоСгорания(2000);

двигатели[2] = new РеактивныйДвигатель(5000);

Array.Sort(двигатели);

//Сортировка массива "двигатели" в порядке возрастания с использованием метода "Sort" из класса "Array".

foreach (Двигатель двигатель in двигатели)

{

двигатель.запуск();

двигатель.информация();

двигатель.остановка();

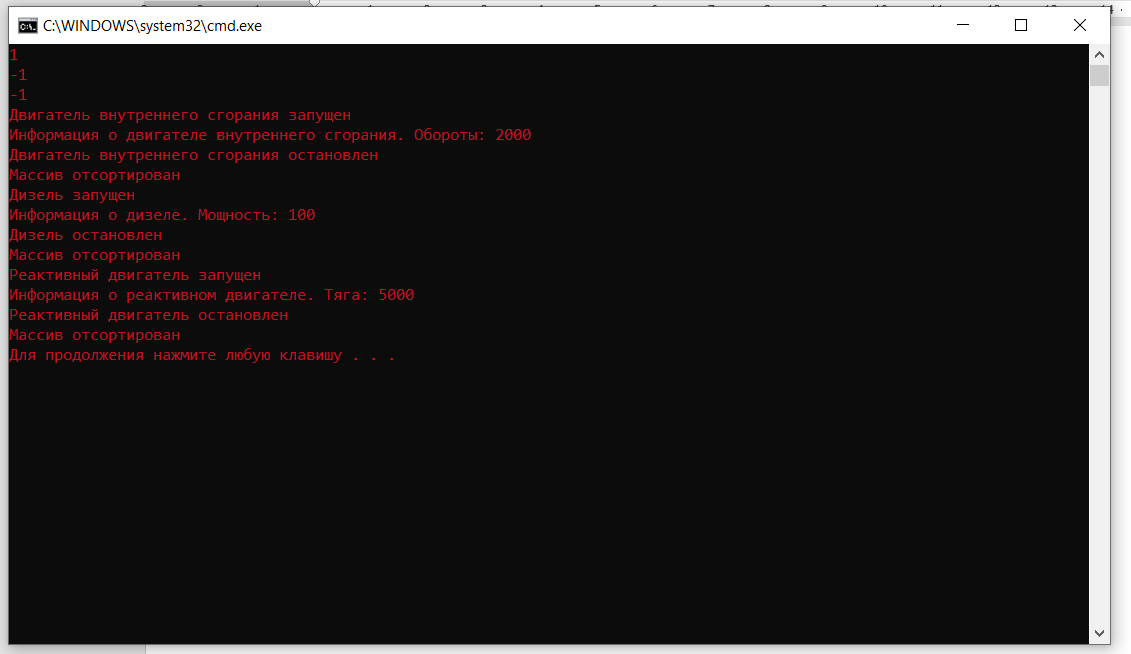
Console.WriteLine("Массив отсортирован");

}

}

}

Результаты тестирования



Вывод по работе.