# Лабораторная работа № 16-17

**Тема:** Разработка и тестирование программ на языке С# с использованием принципов наследования и полиморфизма. Иерархия классов.

**Цель работы:**

1. Формирование навыков разработки программ на языке С# с использованием принципов наследования и полиморфизма.
2. Изучение принципа наследования при использовании классов на языке С#.

. **Оборудование:**

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

**Контрольные вопросы:**

1. Что такое наследование?

Наследование — это механизм в объектно-ориентированном программировании, который позволяет создавать новый класс на основе уже существующего класса, называемого базовым или родительским классом. Наследующий класс, называемый производным или дочерним классом, наследует свойства и поведение базового класса и имеет возможность расширять или изменять его функциональность.

1. Что такое родительский (базовый) класс?

Родительский (базовый) класс - это класс, от которого происходит наследование. Он содержит общие свойства и методы, которые могут быть использованы другими классами.

1. Как называется класс, который создается на основе базового класса?

Класс, который создается на основе базового класса, называется производным классом или классом-наследником.

1. Как объявить производный класс?

Производный класс объявляется с использованием ключевого слова extends после имени класса, за которым следует имя родительского класса. Например: public class DerivedClass extends BaseClass { ... }.

1. Для чего используют ключевое слово ***base***?

Ключевое слово base используется в производном классе для доступа к членам базового класса. Оно позволяет вызвать конструктор базового класса или обратиться к его методам и полям.

1. Как получить доступ к членам базового класса из класса - наследника и какие спецификаторы доступа они должны иметь?

Для доступа к членам базового класса из класса-наследника используются модификаторы доступа. Если член базового класса объявлен с модификаторами public или protected, то он доступен из класса-наследника. Если модификатор доступа отсутствует, то член базового класса доступен только внутри пакета.

1. Что означает спецификатор доступа **sealed** для базового класса?

Спецификатор доступа sealed для базового класса означает, что он не может быть дальше унаследован другими классами. То есть класс-наследник не может быть создан от класса, помеченного как sealed.

1. Возможно ли наследование от статического класса?

Нет, наследование от статического класса не возможно, так как статический класс не предназначен для создания экземпляров или наследования.

1. Передаются ли конструкторы производному классу при наследовании?

Да, конструкторы также передаются производному классу при наследовании. Производный класс может вызывать конструктор базового класса с помощью ключевого слова super или base в своем конструкторе.

1. Что такое полиморфизм?

Полиморфизм - это возможность объекта быть использованным в виде одного из нескольких типов. Он позволяет вызывать один и тот же метод у разных объектов и получать разные результаты в зависимости от типа объекта.

1. Дать объяснение для статического и динамического полиморфизма.
2. Как реализуется статический полиморфизм?

Статический полиморфизм реализуется через перегрузку методов и операторов. Перегрузка методов позволяет определить несколько методов с одним и тем же именем, но с различными параметрами. Компилятор выбирает подходящий метод на основе переданных аргументов. Перегрузка операторов позволяет определить различное поведение операторов в зависимости от типов операндов.

1. В чем заключается перегрузка методов?

Перегрузка методов - это возможность определить несколько методов с одним и тем же именем, но с различными параметрами. При вызове метода происходит выбор подходящей версии метода на основе типов аргументов, переданных во время вызова.

1. В чем заключается перегрузка операторов?

Перегрузка операторов - это возможность определить различное поведение операторов для различных типов операндов. Например, можно определить оператор "+" для опередления сложения двух чисел и для присоединения двух строк.

1. Что такое абстрактный класс?

Абстрактный класс - это класс, который не может быть использован для создания объектов, но может содержать абстрактные методы, которые должны быть реализованы в классах-наследниках. Абстрактные классы могут содержать и обычные методы и поля данных. Они предоставляют базовую реализацию для классов-наследников, но самостоятельно не могут быть инстанцированы.

1. Что такое виртуальный метод?

Виртуальный метод - это метод, которые может быть переопределен в классах-наследниках. Ключевое слово "virtual" используется при объявлении метода в базовом классе, а ключевое слово "override" используется при переопределении метода в производных классах. Виртуальные методы позволяют вызывать специфичную реализацию метода, связанную с типом объекта, который вызывает метод, во время выполнения программы (динамическое связывание).

**Ход работы:**

1. **В соответствии с вариантом задания разработать класс и программу, иллюстрирующую его работу. Требования к классу:**

* **обязательно наличие открытой (public) и закрытой (private) частей;**
* **класс должен иметь по крайней мере два конструктора, определенных программистом (один конструктор с параметрами, второй по – умолчанию;**
* **необходимо задать набор свойств для получения значений и модификации полей данных, находящихся в закрытой части класса.**

**Пример. Вариант задания:**

Персона

Родительский класс Персона. ФИО, Год рождения, Год поступления на работу или учебу, Базовый размер заработной платы или стипендии, Размер надбавки (за совмещение должностей или за отличную учебу).

Действия над объектами класса: начисление заработной платы или стипендии, вывод личных данных, операции сравнения объектов, присваивание.

1. **Для всех вариантов сделать по аналогии**

**Построить иерархию классов в соответствии с вариантом задания (базовый класс и два или 3 производных класса). Базовый класс сделать абстрактным. В базовый класс добавить абстрактный метод и реализовать его в производных классах. Другие методы базового класса сделать виртуальными и переопределить их в классах – наследниках. Производные классы должны иметь собственные поля данных, отличные от полей базового класса.**

1. **Для разработанной Вами иерархии классов в методе Main:**

* **Описать массив объектов базового класса;**
* **Занести в этот массив из файла объекты дочерних классов;**
* **Продемонстрировать работу методов класса у всех элементов этого массива.**

1. Разработать алгоритм задачи и представить его в виде  программы на С#
2. Используя средства Visual Studio.Net создать файл с программой  и выполнить тестирование и отладку программы
3. Результаты представить в виде отчета
4. Сделать вывод о проделанной работе

**Пример. Вариант задания:**

Персона

Родительский класс Персона. ФИО, Год рождения, Год поступления на работу или учебу, Базовый размер заработной платы или стипендии, Размер надбавки (за совмещение должностей или за отличную учебу).

Действия над объектами класса: начисление заработной платы или стипендии, вывод личных данных, операции сравнения объектов, присваивание.

**Варианты заданий для лабораторной работы №16**

**Вариант 12**

Двигатель, дизель, двигатель внутреннего сгорания, реактивный двигатель.

Листинг с исходным кодом

using System;

abstract class Двигатель

{

public abstract void запуск();

public virtual void остановка()

{

Console.WriteLine("Двигатель остановлен");

}

public virtual void информация()

{

Console.WriteLine("Информация о двигателе");

}

}

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);

}

}

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);

foreach (Двигатель двигатель in двигатели)

{

двигатель.запуск();

двигатель.информация();

двигатель.остановка();

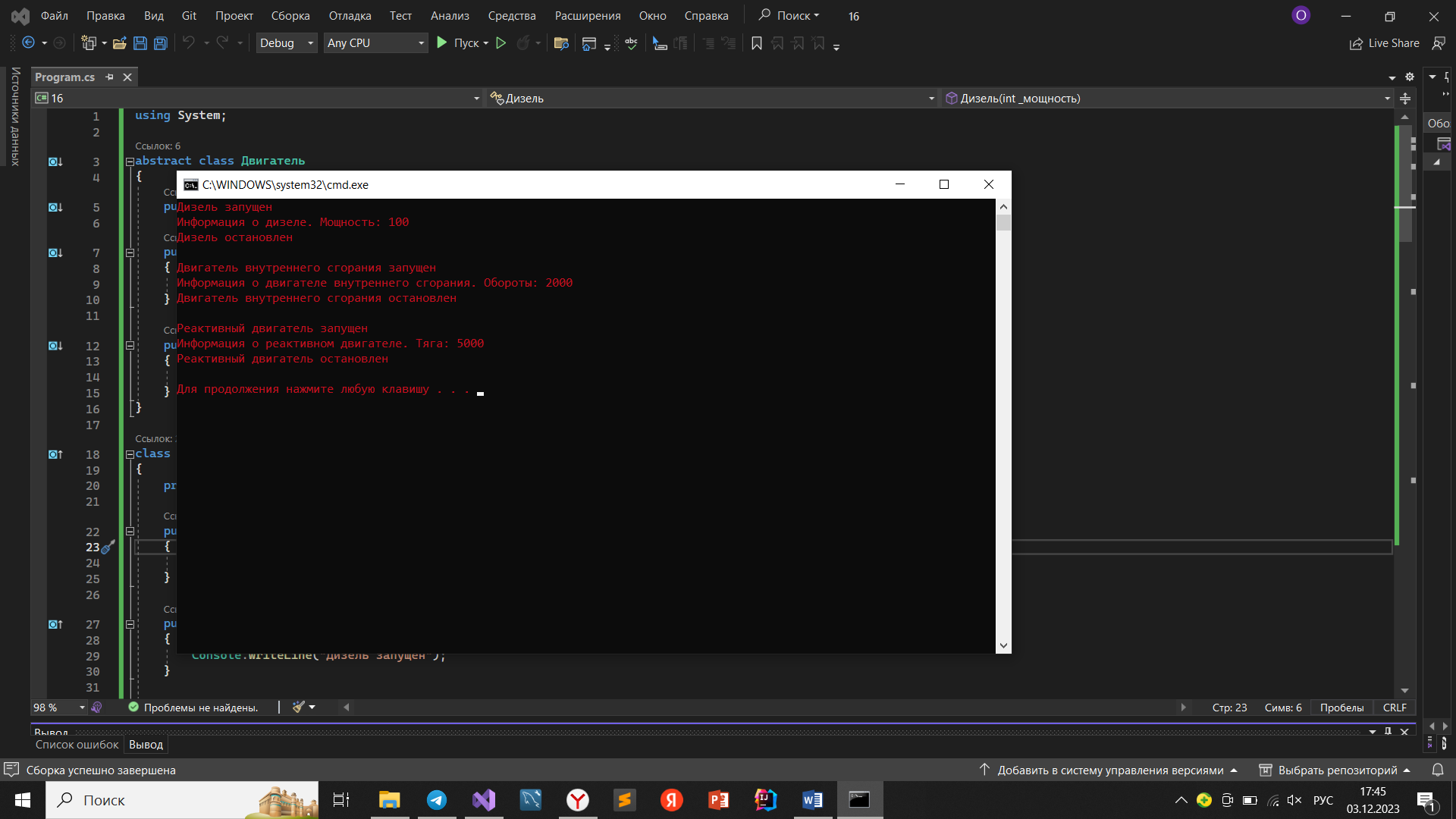
Console.WriteLine();

}

}

}

Результаты тестирования



Вывод по работе: я сформировала навыки разработки программ на языке С# с использованием принципов наследования и полиморфизма, изучила принцип наследования при использовании классов на языке С#.