# Лабораторная работа № 4 – Абстрактные классы

**1 Цель работы**

Изучить основные принципы разработки абстрактных классов в C#.

1. **Порядок выполнения работы**
   * прочесть краткие теоретические сведения;
   * выполнить задания раздела;
   * составить отчет по лабораторной работе и защитить его у преподавателя.
2. **Общая часть**

**3.1 Основные понятия абстрактного класса**

Абстрактный класс в объектно-ориентированном программировании — базовый класс, который не предполагает создания экземпляров. Абстрактные классы реализуют на практике один из принципов ООП — полиморфизм. Абстрактный класс может содержать (и не содержать) абстрактные методы и свойства. Абстрактный метод не реализуется для класса, в котором объявлен, однако должен быть реализован для его неабстрактных потомков. Абстрактные классы представляют собой наиболее общие абстракции, то есть имеющие наибольший объём и наименьшее содержание.

**3.2 Общий синтаксис абстрактного класса**

Модификатор abstract указывает, что реализация изменяемого объекта является неполной или отсутствует. Модификатор abstract может использоваться с классами, методами, свойствами, индексаторами и событиями. Модификатор abstract в объявлении класса указывает, что класс предназначен только для использования в качестве базового класса для других классов. Члены, помеченные как абстрактные или включенные в абстрактный класс, должны быть реализованы с помощью классов, производных от абстрактных классов.

Общий синтаксис класса:

**[ атрибуты ] [ спецификаторы ] abstract class имя класса [ : предки ] тело класса**

## Примеры реализации абстрактных классов на C#

Зачем нужны абстрактные классы? Допустим, в нашей программе мы можем определить две основных сущности: студента и преподавателя ВУЗа. Каждая из этих сущностей будет отличаться, например, для студента надо определить его группу, а для преподавателя - должность. Соответственно студент и преподаватель будут составлять отдельные классы Student и Teacher. В то же время обе этих сущности могут иметь что-то общее, например, имя и фамилию, какую-то другую общую функциональность. И эту общую функциональность лучше вынести в какой-то отдельный класс, например, Person, который описывает человека. То есть классы Student и Teacher будут производными от класса Person. И так как все объекты в нашей системе будут представлять либо студента, либо преподавателя, то напрямую мы от класса Person создавать объекты не будем. Поэтому имеет смысл сделать его абстрактным:

public abstract class Person

{

public string Name;

public string Surname;

public Person(string name, string surname)

{

Name = name;

Surname = surname;

}

public abstract void ShowInfo();//абстрактные методы не должны иметь реализацию

}

class Student:Person

{

public string GroupName;

public Student(string name, string surname, string groupName):base(name, surname)

{

GroupName = groupName;

}

public override void ShowInfo()

{

Console.WriteLine($"Имя: {Name}\nФамилия: {Surname}\nГруппа: {GroupName}");

}

}

class Teacher : Person

{

public string Position;//Должность

public Teacher(string name, string surname, string position) : base(name, surname)

{

Position = position;

}

public override void ShowInfo()

{

Console.WriteLine($"Имя: {Name}\nФамилия: {Surname}\nДолжность: {Position}");

}

}

Затем мы сможем использовать эти классы:

Student test1 = new Student("Ольга","Васильева","19-КБ-ПИ1");

Teacher test2 = new Teacher("Иван", "Сергеев", "Старший Преподаватель");

test1.ShowInfo();

test2.ShowInfo();

Или даже так:

Person test1 = new Student("Ольга","Васильева","19-КБ-ПИ1");

Person test2 = new Teacher("Иван", "Сергеев", "Старший Преподаватель");

test1.ShowInfo();

test2.ShowInfo();

Но мы **НЕ можем** создать объект Person, используя конструктор класса Person. Однако несмотря на то, что напрямую мы не можем вызвать конструктор класса Person для создания объекта, тем не менее конструктор в абстрактных классах то же может играть важную роль, в частности, инициализировать некоторые общие для производных классов переменные и свойства. И хотя в примере выше конструктор класса Person не вызывается, тем не менее производные классы могут обращаться к нему.

Хрестоматийным примером абстрактного класса является система геометрических фигур. В реальности не существует геометрической фигуры как таковой. Есть круг, прямоугольник, квадрат, но просто фигуры нет. Однако же и круг, и прямоугольник имеют что-то общее и являются фигурами:

abstract class Figure

{

// абстрактный метод для получения периметра

public abstract float Perimeter();

// абстрактный метод для получения площади

public abstract float Area();

}

// производный класс прямоугольника

class Rectangle : Figure

{

public float Width { get; set; }

public float Height { get; set; }

public Rectangle(float width, float height)

{

this.Width = width;

this.Height = height;

}

// переопределение получения периметра

public override float Perimeter()

{

return Width \* 2 + Height \* 2;

}

// переопрелеление получения площади

public override float Area()

{

return Width \* Height;

}

}

Абстрактные классы имеют следующие особенности:

- создавать экземпляры абстрактного класса нельзя;

- абстрактные классы могут содержать абстрактные методы и свойства также, как и обычные;

- абстрактный класс с модификатором sealed (Справочник по C#) определить нельзя, поскольку эти два модификатора имеют взаимоисключающие значения. Модификатор **sealed** запрещает наследовать классу, в то время как модификатор **abstract** указывает, что класс обязан иметь производные классы.

- Неабстрактный класс, являющийся производным от абстрактного, должен содержать фактические реализации всех наследуемых абстрактных методов и методов доступа.

1. **Общее задание для выполнения на паре**

Разработать базовый абстрактный класс Товар (Product), реализующий общую сущность в интернет-магазине. Класс должен содержать конструктор, абстрактный метод вывода информации о товаре, а также общие свойства для всех товаров:

- Идентификатор (id);

- Наименование;

- Цена.

Разработать 2 класса-наследника, реализующие конкретные типы товаров (по своему усмотрению - одежда, обувь, электроника и т.д.) и содержащие дополнительные свойства (размер, цвет, диагональ экрана и пр.). Переопределить конструкторы и методы вывода информации о товаре.

1. **Индивидуальные** **задания для выполнения работы**

Сделать базовый класс из предыдущей лабораторной работы абстрактным, удалить из него реализацию метода вывода.