# ПРАКТИЧЕСКАЯ РАБОТА № 9

**Абстрактные классы**

**Цель работы:** Ознакомиться с а) разработкой абстрактных классов; б) наследованием от абстрактного класса; в) переопределения абстрактных методов классом-наследником.

|  |  |  |
| --- | --- | --- |
| №  п/п | Ф.И.О. | НОМЕР ВАРИАНТА/ЗАДАНИЯ |
|  | Андриуца Дарья Николаевна | Набор примера из лекционного материала |
|  | Варфоломеев Иван Константинович | 1 |
|  | Верховин Игорь Вадимович | 2 |
|  | Власов Тимур Михайлович | 3 |
|  | Гагауз Григорий Иванович | 4 |
|  | Голубцов Олег Русланович | Набор примера из лекционного материала |
|  | Гувирь Максим Григорьевич | Набор примера из лекционного материала |
|  | Дорожкина Алина Игоревна | 5 |
|  | Жупанов Юлиан Игоревич | 6 |
|  | Капелько Иван Юрьевич | 4 |
|  | Кирста Андрей Александрович | 7 |
|  | Ковалев Иван Алексеевич | 8 |
|  | Лютов Глеб Васильевич | 9 (дополнительные условия придумать самим) |
|  | Михалаки Василий Федорович | 10 |
|  | Москалюк Иван Сергеевич | 11 |
|  | Огурцов Алексей Михайлович | 12 |
|  | Омельченко Евгений Борисович | 13 |
|  | Полякова Анна Игоревна | 14 |
|  | Редкозубов Владислав Романович | 15 |
|  | Ротарь Федор Федорович | 5 |
|  | Рошко (был Брага) Марин Михайлович | 6 |
|  | Салкуцан Анна Александровна | 16 |
|  | Скутельник Владимир Александрович | 11 (дополнительные условия придумать самим) |
|  | Струсевич Лев Дмитриевич | 10 |
|  | Телицкий Александр | 9 |
|  | Чернюк Егор Дмитриевич | 8 |
|  | Щитченко Николь Александровна | 7 |

# Теоретические сведения

**Класс называется абстрактным**, если он имеет хотя бы один абстрактный метод.

**Метод называется абстрактным**, если при определении метода задана его сигнатура, но не задана реализация метода.

Объявление абстрактных методов и абстрактных классов должно сопровождаться модификатором*abstract*. Поскольку абстрактные классы не являются полностью определенными классами, то нельзя создавать объекты абстрактных классов. Абстрактные классы могут иметь потомков, **частично или полностью реализующих абстрактные методы** родительского класса. Поскольку абстрактный метод рассматривается как виртуальный метод, переопределяемый потомком, то к нему применяется стратегия динамического связывания.

Абстрактные классы имеют ряд свойств:

* Такие классы не могут быть *sealed*.
* Они не могут быть инициированы оператором *new*.
* При объявлении метода класса абстрактным подразумева- ется, что он виртуальный, потому метод не может быть объявлен как *virtual*.
* Все абстрактные члены должны быть определены в на- следующем классе, за исключение случая, когда наследующий класс так же является абстрактным.
* Абстрактные методы класса не могут быть объявлены как

*static*.

* При реализации абстрактным классом интерфейса, все чле-

ны интерфейса должны быть либо реализованными, либо описаны их сигнатуры и объявлены как *abstract*.

* Абстрактный класс может иметь как абстрактных так и не абстрактных членов.
* Можно объявить абстрактными методы, свойства, индекса- торы и события.
* Абстрактный класс может реализовывать несколько ин- терфейсов и может быть наследником класса, при этом он может пере- определять виртуальные методы неабстрактного класса.
* Наследующий класс при переопределении абстрактного свойства обязан реализовывать один из аксессоров (либо *set* либо *get*). Если переопределяется абстрактное свойство, имеющее только один аксессор, то переопределяться может только он. Если абстрактное свойство имеет оба аксессора, то переопределять может как один, так и оба.

Абстрактные классы являются одним из важнейших инструментов, используемых при проектировании программ, основанных на объектно- ориентированном подходе. Абстрактный класс позволяет задать ряд действий, разрешенные над объектом класса, не входя в детали реализации.

# Пример 1.

*Using System;*

*Namespace test*

*{*

*abstract class Figure*

*{ //Площадь фигуры*

*public abstract double square(); public abstract double perimeter();*

*}*

*class Triangle: Figure*

*{*

*double а, b, с; //Стороны*

*//Конструктор*

*public Triangle (double a, double b, double с) { this.a=a; this.b=b; this.c=c; } public override double square()*

*{ //Используем формулу Герона*

*double p = (a+b+c)/2; return Math.Sqrt(p\*(p-a)\*(p-b)\*(p-c));*

*}*

*public override double perimeter() { return a+b+c; }*

*}*

*class Rectangle: Figure*

*{*

*double a, b; //Стороны*

*//Конструктор*

*public Rectangle(double a, double b) { this.a=a; this.b=b; } public override double squared() { return i=a\*b; }*

*public override double perimeter() { return (a\*b)\*2; }*

*}*

*class Test*

*{*

*public static void Main()*

*{*

*Figure fl, f2;*

*f1=new Triangle(3,4,5); f2=new Rectangle(2, 6); System.Console.WriteLine(f1.perimeter()+", "+ f1.square()); System.Console.WriteLine(f2.perimeter ()+", "+ f2.square());*

*}*

*}*

В данном примере мы объявляем абстрактный класс *Figure*, от которого

производим два класса — *Rectangle* (класс прямоугольника) и *Triangle* (класс треугольника).

В классе *Figure* есть два абстрактных метода — *square* (для подсчета площади) и *perimeter* (для периметра). Так как для произвольной фигуры формул для площади и для периметра не существует, то эти методы объявле- ны в классе *Figure* и переопределены в производных классах (с ключевым словом *override*). Далее в классе *Test* проводится испытание, в котором со- здаются две переменные ссылочного типа базового класса *Figure*, затем в эти ссылки мы записываем созданные экземпляры производных классов *Triangle* и *Rectangle*. Обратите внимание, что разрешено создавать ссылку на аб- страктный класс, хотя экземпляр абстрактного класса создать нельзя. Далее мы выводим на экран результаты вычислений.

**Пример 2.** Проектирование абстрактного класса *Stack*.

Спроектируем абстрактный класс *Stack*, описывающий стек. Стек можно реализовать различным образом, например, в виде массива или в виде

списка. Можно, в дальнейшем создать классы-потомки нашего абстрактного класса - *ArrayStack* и *ListStack,* в которых реализовать различные схемы построения стека.

Ниже приведено описание абстрактного класса *Stack*:

*public abstract class Stack{ public Stack() {}*

*// втолкнуть элемент item в стек*

*public abstract void put(int item);*

*// удалить элемент в вершине стека*

*public abstract void remove();*

*// прочитать элемент в вершине стека*

*public abstract int item();*

*// определить, пуст ли стек*

*public abstract bool IsEmpty();*

*}*

Построим один из потомков этого класса, реализация которого

основана на списковом представлении. Класс *ListStack* будет потомком абстрактного класса *Stack* и клиентом класса *Linkable*, задающего элементы списка. Класс *Linkable* выглядит следующим образом:

*public class Linkable{ public Linkable() { } public int info;*

*public Linkable next;*

*}*

В нем - два поля и конструктор по умолчанию. Построим класс- наследник *ListStack*:

*public class ListStack: Stack{*

*public ListStack() { top = new Linkable(); } Linkable top;*

*// втолкнуть элемент item в стек*

*public override void put(int item) { Linkable newitem = new Linkable(); newitem.info = item;*

*newitem.next = top; top = newitem;*

*}*

*// удалить элемент в вершине стека*

*public override void remove() { top = top.next; }*

*// прочитать элемент в вершине стека*

*public override int item() { return(top.info); }*

*// определить, пуст ли стек*

*public override bool IsEmpty() { return(top.next == null); }*

*}*

Класс имеет одно поле *top* класса *Linkable* и полноценные методы,

реализующие абстрактные методы, унаследованные от абстрактного класса

*Stack*.

Пример работы со стеком:

*public void TestStack(){*

*ListStack stack = new ListStack(); stack.put(7); stack.put(9); Console.WriteLine(stack.item()); stack.remove(); Console.WriteLine(stack.item()); stack.put(11); stack.put(13); Console.WriteLine(stack.item()); stack.remove(); Console.WriteLine(stack.item()); if(!stack.IsEmpty()) stack.remove(); Console.WriteLine(stack.item());*

*}*

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

Индивидуальные задания состоят из 2-х частей, 1-я часть предполагает разработку абстрактного класса и его нескольких классов-наследников, а 2-я

– с использованием абстрактного класс, формирования класса- наследника, использующего списки для решения поставленной задачи.

# Часть 1.

Разработать программу, используя абстрактный класс и образуя его потомки:

* 1. написать программу для расчета объема и площади поверхности следующих геометрических фигур: шар, конус;
  2. написать программу для расчета объема и площади поверхности следующих геометрических фигур: куб, конус;
  3. написать программу для расчета объема и площади поверхности следующих геометрических фигур: параллелепипед, конус;
  4. написать программу для расчета объема и площади поверхности

следующих геометрических фигур: 4-х гранная пирамида, конус;

5)написать программу для расчета объема и площади поверхности

следующих геометрических фигур: конус, усеченный конус;

1. написать программу для расчета объема и площади поверхности следующих геометрических фигур: призма, конус;
2. написать программу для расчета объема и площади поверхности следующих геометрических фигур: шар, цилиндр;
3. написать программу для расчета объема и площади поверхности следующих геометрических фигур: эллипс, конус;
4. написать программу для расчета периметра и площади следующих геометрических фигур: треугольник, квадрат, круг;
5. написать программу для расчета периметра и площади следующих геометрических фигур: эллипс, ромб, прямоугольник;
6. написать программу для расчета периметра и площади следующих геометрических фигур: прямоугольник, круг, трапеция;
7. написать программу для расчета периметра и площади следующих геометрических фигур: круг, внутри которого вписан квадрат;
8. написать программу для расчета периметра и площади следующих геометрических фигур: треугольник, внутри которого вписан круг;
9. написать программу для расчета периметра и площади следующих геометрических фигур: треугольник, внутри которого расположен квадрат;
10. написать программу для расчета периметра и площади следующих геометрических фигур: трапеция, внутри которой расположен прямоугольник;
11. написать программу для расчета периметра и площади следующих геометрических фигур: ромб, внутри которого расположен круг.

# Часть 2.

Написать программу, используя абстрактный класс, с помощью которого формируются списки, подобные рассмотренным выше:

1. написать программу, которая организует поиск по списку учеников класса по фамилии/имени;
2. написать программу, которая организует поиск по списку магазинов по названию/профилю;
3. написать программу, которая организует поиск по списку автомобилей по номеру гос. регистрации/ФИО владельца;
4. написать программу, которая организует поиск по списку городов по названию города/количеству жителей;
5. написать программу, которая организует поиск по списку студентов вуза по номеру студенческого билета/ФИО;
6. написать программу, которая организует поиск по списку компьютеров, установленных в школе по инвентарному номеру/номеру кабинета, где он установлен;
7. написать программу, которая организует поиск по списку квартир дома по номеру квартиры/фамилии квартиросъемщика;
8. написать программу, которая организует поиск по списку рейсов автобусов по номеру рейса/названии пункта назначения;
9. написать программу, которая организует поиск по списку книг по фамилии автора/названию книги;
10. написать программу, которая организует поиск по списку телефонов по номеру телефона/адресу;
11. написать программу, которая организует поиск по списку пациентов больницы по фамилии/диагнозу;
12. написать программу, которая организует поиск по списку избирателей по ФИО/адресу;
13. написать программу, которая организует поиск по списку сотрудников фирмы по фамилии/должности;
14. написать программу, которая организует поиск по списку сотрудников фирмы по должности/величине оклада;
15. написать программу, которая организует поиск по списку товаров

магазина по номеру товара/цене;

1. написать программу, которая организует поиск по списку учеников класса по фамилии мамы/домашнему телефону.