Лабораторная работа №10

Разработка программ, реализующих механизм наследования классов

Цель: научиться применять различные механизмы наследования при решении задач.

Оборудование: персональный компьютер, практикум, тетради для лабораторных работ.

Правила по технике безопасности: общие (приложение).

Литература:

1. А. Пол, Объектно-ориентированное программирование на C++.

Время выполнения: 4 часа.

Краткие теоретические сведения

- **public**: публичный, общедоступный класс или член класса. Такой член класса доступен из любого места в коде, а также из других программ и сборок.
- **private**: закрытый класс или член класса. Представляет полную противоположность модификатору public. Такой закрытый класс или член класса доступен только из кода в том же классе или контексте.
- **protected**: такой член класса доступен из любого места в текущем классе или в производных классах.
- **internal**: класс и члены класса с подобным модификатором доступны из любого места кода в той же сборке, однако он недоступен для других программ и сборок (как в случае с модификатором public).
- **protected internal**: совмещает функционал двух модификаторов. Классы и члены класса с таким модификатором доступны из текущей сборки и из производных классов.

Объявление *полей* класса без модификатора доступа равнозначно их объявлению с модификатором **private**. *Классы*, объявленные без модификатора, по умолчанию имеют доступ **internal**.

```
public class State
    {
        int a; // все равно, что private int a;
        private int b; // поле доступно только из текущего класса
        protected int c; // доступно из текущего класса и производных классов
        internal int d; // доступно в любом месте программы
        protected internal int e; // доступно в любом месте программы и из
классов-наследников
        public int f; // доступно в любом месте программы, а также для других
программ и сборок

        private void Show_f()
        {
            Console.WriteLine("Переменная f = {0}", f);
        }

        public void Show_a()
        {
```

```
            Console.WriteLine("Переменная a = {0}", a);
        }

        internal void Show_b()
        {
            Console.WriteLine("Переменная b = {0}", b);
        }

        protected void Show_e()
        {
            Console.WriteLine("Переменная e = {0}", e);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            State state1 = new State();

            // присвоить значение переменной a у нас не получиться,
            // так как она закрытая и класс Program ее не видит
            // И данную строку среда подчеркнет как неправильную

            state1.a = 4; //Ошибка, получить доступ нельзя

            // то же самое относится и к переменной b
            state1.b = 3; // Ошибка, получить доступ нельзя

            // присвоить значение переменной c то же не получиться,
            // так как класс Program не является классом-наследником класса
State
            state1.c = 1; // Ошибка, получить доступ нельзя

            // переменная d с модификатором internal доступна из любого места
программы
            // поэтому спокойно присваиваем ей значение
            state1.d = 5;

            // переменная e так же доступна из любого места программы
            state1.e = 8;

            // переменная f общедоступна
            state1.f = 8;

            // Попробуем вывести значения переменных

            // Так как этот метод объявлен как private, мы можем использовать
его только внутри класса State
            state1.Show_f();  // Ошибка, получить доступ нельзя

            // Так как этот метод объявлен как protected, а класс Program не
является наследником класса State
            state1.Show_e();  // Ошибка, получить доступ нельзя

            // Общедоступный метод
            state1.Show_a();

            // Метод доступен из любого места программы
            state1.Show_b();

            Console.ReadLine();
        }
    }
```

НАСЛЕДОВАНИЕ

Наследование является одним из трех основополагающих принципов объектно-ориентированного программирования, поскольку оно допускает создание иерархических классификаций. Благодаря наследованию можно создать общий класс, в котором определяются характерные особенности, присущие множеству связанных элементов. От этого класса могут затем наследовать другие, более конкретные классы, добавляя в него свои индивидуальные особенности.

В языке C# класс, который наследуется, называется ***базовым***, а класс, который наследует, — ***производным***. Следовательно, производный класс представляет собой специализированный вариант базового класса. Он наследует все переменные, методы, свойства и индексаторы, определяемые в базовом классе, добавляя к ним свои собственные элементы.

Синтаксис наследования

```
class имя_класса_производного_класса : имя_базового_класса
{
    // тело класса
}
```

Обратите внимание на следующие отличия в механизме наследования C# от C++:

1. В качестве базового класса при наследовании может быть указан только один класс. Это означает, что в C# нет механизма множественного наследования. Если же необходимо всё-таки выполнить его, то тогда можно воспользоваться понятием интерфейса, которое будет рассмотрено позже.
2. При наследовании не указываются спецификаторы доступа как в C++.

Наследование (inheritance) является одним из ключевых моментов ООП. Его смысл состоит в том, что мы можем расширить функциональность уже существующих классов за счет добавления нового функционала или изменения старого. Пусть у нас есть следующий класс Person, описывающий отдельного человека:

```
class Person
    {
        private string _firstName;
        private string _lastName;

        public string FirstName
        {
            get { return _firstName; }
            set { _firstName = value; }
        }
        public string LastName
        {
            get { return _lastName; }
            set { _lastName = value; }
        }

        public void Show()
        {
            Console.WriteLine(FirstName + " " + LastName);
        }
    }
```

Но вдруг нам потребовался класс, описывающий сотрудника предприятия - класс Employee. Поскольку этот класс будет реализовывать тот же функционал, что и класс Person, так как сотрудник - это также и человек, то было бы рационально сделать класс Employee производным (или наследником, или подклассом) от класса Person, который, в свою очередь, называется базовым классом или родителем (или суперклассом):

```
class Employee : Person
{    }
class Program
{
    static void Main(string[] args)
    {
       Person p = new Person { FirstName = "Антон", LastName = "Заремба" };
        p.Show();
        p = new Employee { FirstName = "Сергей", LastName = "Науменко" };
        p.Show();
        Console.ReadKey();
    }
      }
```

Все классы по умолчанию могут наследоваться. Однако здесь есть ряд ограничений:

- Не поддерживается множественное наследование, класс может наследоваться только от одного класса. Хотя проблема множественного наследования реализуется с помощью концепции интерфейсов, о которых мы поговорим позже.
- При создании производного класса надо учитывать тип доступа к базовому классу - тип доступа к производному классу должен быть таким же, как и у базового класса, или более строгим. То есть, если базовый класс у нас имеет тип доступа internal, то производный класс может иметь тип доступа internal или private, но не public.
- Если класс объявлен с модификатором sealed, то от этого класса нельзя наследовать и создавать производные классы. Например, следующий класс не допускает создание наследников:

```
sealed class Admin {    }
```

Доступ к членам базового класса из класса-наследника

Вернемся к нашим классам Person и Employee. Хотя Employee наследует весь функционал от класса Person, посмотрим, что будет в следующем случае:

```
class Employee : Person
    {
        public void Show()
        {
            Console.WriteLine(_firstName);
        }
    }
```

Этот код не сработает и выдаст ошибку, так как переменная _firstName объявлена с модификатором private и поэтому к ней доступ имеет только класс Person. Но зато в классе Person определено общедоступное свойство FirstName, которое мы можем использовать, поэтому следующий код у нас будет работать нормально:

```
class Employee : Person
    {
        public void Show()
        {
            Console.WriteLine(FirstName);
        }
    }
```

Таким образом, производный класс может иметь доступ только к тем членам базового класса, которые определены с модификаторами public, internal, protected и protected internal.

Ключевое слово base

Теперь добавим в наши классы конструкторы:

```
class Person
    {
        protected string FirstName { get; set; }
        protected string LastName  { get; set; }
        public Person(string Name, string Familiya)
        {
            FirstName = Name;
            LastName = Familiya;
        }

        public virtual void Show()
        {
            Console.WriteLine(FirstName + " " + LastName);
        }
    }

    class Employee : Person
    {
        public string Company { get; set; }
        public Employee(string Name, string Familiya, string comp) :
base (Name,Familiya)
        {
            Company = comp;
        }
        public override void Show()
        {
            Console.WriteLine(FirstName + " " + LastName + " " +
Company);
        }

    }
    class Program
    {
        static void Main(string[] args)
        {
            Person p = new Person ("Антон", "Заремба" );
            p.Show();
            Employee obj = new Employee ( "Сергей", "Науменко", "НГАЭК"
);
            obj.Show();
            Console.ReadKey();

        }
    }
```

Класс Person имеет стандартный конструктор, который устанавливает два свойства. Поскольку класс Employee наследует и устанавливает те же свойства, что и класс Person, то логично было бы не писать по сто раз код установки, а как-то вызвать соответствующий код класса Person. К тому же свойств, которые надо установить, и параметров может быть гораздо больше.

С помощью ключевого слова base мы можем обратиться к базому классу. В нашем случае в конструкторе класса Employee нам надо установить имя, фамилию и компанию. Но имя и фамилию мы передаем на установку в

конструктор базового класса, то есть в конструктор класса Person, с помощью выражения base(fName, lName).

Конструкторы в производных классах

Конструкторы не передаются производному классу при наследовании. И если в базовом классе не опредлен конструктор по умолчанию без параметров, а только конструкторы с параметрами (как в случае с базовым классом Person), то в производном классе мы обязательно должны вызвать один из этих конструкторов через ключевое слово base. Например, из класса Employee уберем определение конструктора:

```
class Employee : Person
{
    public string Company { get; set; }
}
```

В данном случае мы получим ошибку, так как класс Employee не соответствует классу Person, а именно не вызывает конструктор базового класса. Даже если бы мы добавили какой-нибудь конструктор, который бы устанавливал все те же свойства, то мы все равно бы получили ошибку:

```
public Employee(string Name, string Familiya, string comp)
{
    FirstName = Name;
    LastName = Familiya;
    Company = comp;
}
```

То есть в классе Employee через ключевое слово base надо явным образом вызвать конструктор класса Person:

```
public Employee(string Name, string Familiya, string comp)
        : base(Name, Familiya)
{
    Company = comp;
}
```

Либо в качестве альтернативы мы могли бы определить в базовом классе конструктор без параметров:

```
class Person
{
    // остальной код класса
    // конструктор по умолчанию
    public Person()
    {
        FirstName = "Антон";
```

```
            LastName = "Зарембо";
            Console.WriteLine("Вызов конструктора без параметров");
        }
    }
```

Тогда в любом конструкторе производного класса, где нет обращения конструктору базового класса, все равно неявно вызывался бы этот конструктор по умолчанию. Например, следующий конструктор

```
public Employee(string comp)
{
    Company = comp;
}
Фактически был бы эквивалентен следующему конструктору:

public Employee(string comp)
    :base()
{
    Company = comp;
}
```

Еще один пример

Создать базовый класс «Транспортное средство» и производные классы «Автомобиль», «Велосипед», «Повозка». Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Primer
{
    class Transport
    {
        protected double time;
        protected double Rastoyanie;
        protected double cenaP;
        protected double cenaG;

        public Transport(double time, double Rastoyanie)
        {
            this.time = time;
            this.Rastoyanie = Rastoyanie;
        }

        public virtual void schet()    {        }
        public virtual void ShowInformation()  {      }
    }

    class Avto : Transport
    {
        double Ast1kmP = 12;
        double Ast1kmG = 15;
```

```
        public Avto(double time, double Rastoyanie) : base(time, Rastoyanie) {    }

        public override void schet()
        {
            cenaP = Rastoyanie * Ast1kmP;
            cenaG = Rastoyanie * Ast1kmG;

        }
        public override void ShowInformation()
        {
            Console.WriteLine("Информация по перевозке на АВТОМОБИЛЕ ");
            Console.WriteLine("Время поездки: {0} минут ", time);
            Console.WriteLine("Стоимость перевозки пассажиров за один км: {0} рублей ",
Ast1kmP);
            Console.WriteLine("Стоимость перевозки груза за один км: {0} рублей ", Ast1kmG);
            Console.WriteLine("Расстояние которое необходимо преодолеть: {0} км",
Rastoyanie);
            Console.BackgroundColor = ConsoleColor.Black;
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("Итого за услугу перевозки пассажиров {0} рублей, грузов {1}
рублей", cenaP, cenaG);
            Console.BackgroundColor = ConsoleColor.Black;
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("------------------------------------------------------------
");
        }
    }

    class Velosiped : Transport
    {
        double Vst1kmP = 25;
        double Vst1kmG = 40;
        public Velosiped(double time, double Rastoyanie) : base(time, Rastoyanie) { }

        public override void schet()
        {
            cenaP = Rastoyanie * Vst1kmP;
            cenaG = Rastoyanie * Vst1kmG;

        }
        public override void ShowInformation()
        {
            Console.WriteLine("Информация по перевозке на ВЕЛОСИПЕДЕ ");
            Console.WriteLine("Время поездки: {0} минут ", time);
            Console.WriteLine("Стоимость перевозки пассажиров за один км: {0} рублей ",
Vst1kmP);
            Console.WriteLine("Стоимость перевозки груза за один км: {0} рублей ", Vst1kmG);
            Console.WriteLine("Расстояние которое необходимо преодолеть: {0} км",
Rastoyanie);
            Console.BackgroundColor = ConsoleColor.Black;
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("Итого за услугу перевозки пассажиров {0} рублей, грузов {1}
рублей", cenaP, cenaG);
            Console.BackgroundColor = ConsoleColor.Black;
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("------------------------------------------------------------
");
        }
    }

    class Povozka : Transport
    {
        double Pst1kmP = 25;
        double Pst1kmG = 40;
        public Povozka(double time, double Rastoyanie) : base(time, Rastoyanie) { }
```

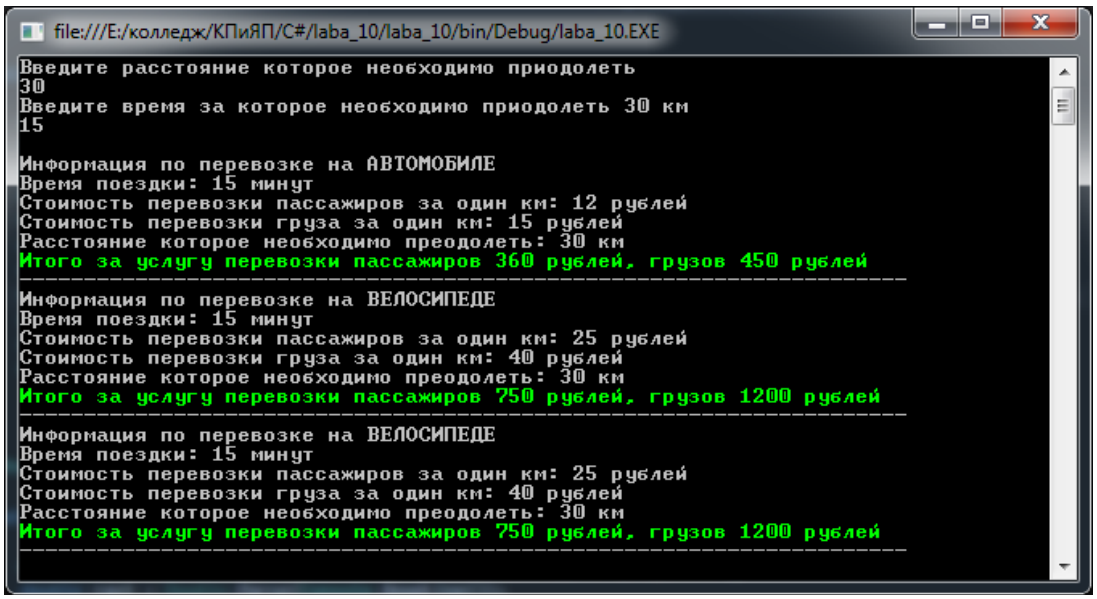
```
        public override void schet()
        {
            cenaP = Rastoyanie * Pst1kmP;
            cenaG = Rastoyanie * Pst1kmG;

        }
        public override void ShowInformation()
        {
            Console.WriteLine("Информация по перевозке на ПОВОЗКЕ");
            Console.WriteLine("Время поездки: {0} минут ", time);
            Console.WriteLine("Стоимость перевозки пассажиров за один км: {0} рублей ",
Pst1kmP);
            Console.WriteLine("Стоимость перевозки груза за один км: {0} рублей ", Pst1kmG);
            Console.WriteLine("Расстояние которое необходимо преодолеть: {0} км",
Rastoyanie);
            Console.BackgroundColor = ConsoleColor.Black;
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("Итого за услугу перевозки пассажиров {0} рублей, грузов {1}
рублей", cenaP, cenaG);
            Console.BackgroundColor = ConsoleColor.Black;
            Console.ForegroundColor = ConsoleColor.Gray;
            Console.WriteLine("-----------------------------------------------------------------
");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Введите расстояние которое необходимо приодолеть");
            double rast = Double.Parse(Console.ReadLine());
            Console.WriteLine("Введите время за которое необходимо приодолеть {0} км",
rast);
            double vremya = Double.Parse(Console.ReadLine());
            Console.WriteLine();
            Avto obj1 = new Avto(vremya, rast);
            obj1.schet();
            obj1.ShowInformation();
            Velosiped obj2 = new Velosiped(vremya, rast);
            obj2.schet();
            obj2.ShowInformation();
            Povozka obj3 = new Povozka(vremya, rast);
            obj2.schet();
            obj2.ShowInformation();

            Console.ReadKey();

        }
    }
}
```



Варианты индивидуальных заданий

Индивидуальное задание 1

Вариант	Задание
1	Создать класс **Автомобиль** со свойствами: Название, Максимальная скорость (в км/ч). Определить 2 виртуальных метода: метод «Стоимость» –стоимость автомобиля, рассчитываемую по формуле. Максимальная скорость * 100 и метод «Обновление модели», увеличивающий максимальную скорость на 10. Определить также метод «Информация», который возвращает строку, содержащую информацию об объекте: Название, Максимальную скорость и Стоимость. Создать также класс наследник **Представительский автомобиль**, в котором переопределить методы: метод «Стоимость» возвращает число, равное. Максимальная скорость * 250, а метод «Обновление модели» увеличивает скорость на 5 км/ч. В главной программе (либо по нажатию на кнопку) создать объект класса **Автомобиль** с максимальной скоростью 140 км/ч и класса **Представительский автомобиль** с максимальной скоростью 160 км/ч. Вывести на экран (или форму) информацию об автомобилях. Обновить модели автомобилей и снова вывести информацию о них
2	Создать класс **Треугольник**, заданный значениями длин трех сторон (a, b, c), с методами «Периметр» и «Площадь». Определить также метод «Информация», который возвращает строку, содержащую информацию о треугольнике: длины сторон, периметр и площадь. Создать также класс наследник **Четырехугольник**, с дополнительными параметрами – длиной четвертой стороны (d) и длинами диагоналей (e, f) и переопределить методы «Периметр» (сумма всех сторон) и «Площадь». Площадь вычислять по следующей формуле. $\sqrt{\frac{4e^2f^2-(b^2+d^2-a^2-c^2)^2}{16}}$ В главной программе создать объект класса **Треугольник** и объект класса **Четырехугольник** и вывести информацию о них. Для упрощения проверки рекомендуется в качестве конкретного объекта класса четырехугольник взять квадрат.
3	Создать класс **Компьютер** со свойствами: Частота процессора (в МГц), количество ядер, объем памяти (в МБ), объем жесткого диска (в ГБ). Определить два виртуальных метода: «Стоимость», возвращающую примерную расчетную стоимость компьютера, рассчитываемую по формуле. Частота процессора * количество ядер / 100 + количество памяти / 80 + объем жесткого диска / 20 и логический метод «Пригод-ность», возвращающий истину (true), если частота процессора не менее 2000 МГц,

	количество ядер не менее 2, объем памяти не менее 2048 МБ, и объем жесткого диска не менее 320 Гб. Определить также метод «Информация», который возвращает строку, содержащую информацию о компьютере: частоту процессора, количество ядер, объем памяти, объем жесткого диска, стоимость и пригодность для наших нужд. Создать также класс наследник **Ноутбук**, с дополнительным свойством. Продолжительность автономной работы (в минутах) и переопределить методы: метод «Стоимость» возвращает число, равное стоимости обычного компьютера + количество минут автономной работы / 10, а метод «Пригодность» возвращает истину, тогда когда и ноутбук пригоден как обычный компьютер, и Продолжительность автономной работы не меньше 60 минут. В главной программе (либо по нажатию на кнопку) создать обычный **компьютер** и **ноутбук** и вывести информацию о них.
4	Создать класс **Прямоугольник**, заданный значениями длин двух сторон (a и b), с виртуальными методами «Периметр» и «Площадь», возвращающими периметр и площадь соответственно, а также виртуальный метод «Увеличить в два раза», увеличивающий в два раза каждую из сторон. Определить также метод «Информация», который возвращает строку, содержащую информацию об треугольнике: длины сторон, периметр и площадь . Создать также класс наследник **Прямоугольник со скругленными углами**, с дополнительным параметром радиус скругления (r). Для него переопределить. Периметр по формуле $p - 8 \cdot r + 2 \cdot \pi \cdot r$, где p – периметр обычного прямоугольника с теми же сторонами, а Площадь по формуле $S - 4 \cdot r^2 + \pi \cdot r^2$, где S – площадь обычного прямоугольника. Также переопределить метод «Увеличить в два раза» так, чтобы он также увеличивал в два раза радиус скругления (по-прежнему увеличивая стороны в два раза). В главной программе создать обычный прямоугольник и **прямоугольник со скругленными углами** и вывести информацию о них. После этого увеличить оба прямоугольника в два раза и выдать обновленную информацию.
5	Создать класс **Фотоаппарат** со свойствами: Модель, Оптическое увеличение (Zoom, вещественное число от 1 до 35) и материал корпуса (металл либо пластик). Определить виртуальный метод: метод «Стоимость» – возвращает число – стоимость фотоаппарата (в $), рассчитываемую по формуле (Zoom+2)·10, если корпус пластиковый и (Zoom+2)·15, если материал металлический. Определить также метод «Информация», который возвращает строку, содержащую информацию об объекте: Модель, Zoom и Стоимость. Также определить логический метод «Дорогой», который будет возвращать истину (true), если стоимость фотоаппарата больше 200$. Создать также класс наследник **Цифровой фотоаппарат**, в котором будет дополнительный целый параметр – количество мегапикселей и переопределить метод «Стоимость», который будет возвращать число, равное стоимости обычного фотоаппарата умножить на количество мегапикселей, а также определить новый метод «Обновление модели», который увеличивает количество мегапикселей на 2. В главной программе (либо по нажатию на кнопку) создать объект класса **Фотоаппарат** с 4-ми кратным оптическим увеличением (Zoom=4) и пластиковым корпусом, а также **Цифровой фотоаппарат** с металлическим корпусом, 8-ю мегапикселями и 3-кратным оптическим увеличением. Вывести на экран (или форму) информацию о фотоаппаратах и о том, являются ли они дорогими. Обновить модели цифрового фотоаппарата и снова вывести информацию о нем.
6	оздать класс **Студент** со свойствами: ФИО, факультет, курс, минимальная оценка по экзаменам за последнюю сессию (по 5-ти бальной системе). Определить виртуальные методы: «Перевести на следующий курс», увеличивающий курс на 1, если минимальная оценка не менее 3, иначе не делающий ничего, а также «Стипендия», возвращающий стипендию (в грн): 0 грн, если минимальная оценка не выше 3, 200 грн, если минимальная оценка равна 4 и 300 грн, если минимальная оценка равна 5. Определить также метод «Информация», который возвращает строку, содержащую информацию о студенте: ФИО, факультет, курс, минимальная оценка по экзаменам и

	начисленную стипендию. Создать также класс наследник **Студент-контрактник**, в котором будет дополнительный логический параметр – уплачен ли контракт и переопределены методы «Перевести на следующий курс», увеличивающий курс на 1, если минимальная оценка не менее 3 и за контракт уплачено, а также «Стипендия» возвращающий всегда 0 грн. В главной программе (либо по нажатию на кнопку) создать объект класса **Студент** и 2 объекта класса **Студент-контрактник** (один из которых уплатил за контракт, а другой нет). Выдать информацию о студентах, затем применить к ним метод «Перевести на следующий курс» и снова выдать информацию о них.
7	Создать класс **Круг** заданный своим радиусом (r), с виртуальным методом «Площадь», возвращающим площадь круга, а также виртуальный метод «Увеличить» с одним вещественным параметром – во сколько раз увеличить, увеличивающий радиус в заданное число раз. Определить также метод «Информация», который возвращает строку, содержащую информацию о круге: радиус и площадь. Создать также класс наследник **Кольцо**, с дополнительным параметром — внутренним радиусом (rin), при этом унаследованный от родителя радиус будет обозначать внешний радиус. Переопределить метод «Площадь», как разницу между площадью внешнего круга минус площадь внутреннего круга. Также доопределить метод «Увеличить», чтобы он увеличивал также и внутренний радиус. В главной программе (либо по нажатию на кнопку) создать обычный **круг** и **кольцо** и вывести информацию о них. После этого увеличить оба объекта в полтора раза и выдать обновленную информацию.
8	Создать класс **Табуретка** со свойствами: Высота (h, в см), Качество изделия (низкое, среднее, высокое). Определить два виртуальных метода: «количество древесины», которое требует табуретка, по формуле 4·h+12, если качество низкое, и 5·h+14, если качество среднее или высокое, а также «стоимость», равная d·2, для низкого качества, d·3, для среднего качества, d·4, для высокого качества, где d – количество древесины, которое требует данный объект. Определить также метод «Информация», который возвращает строку, содержащую информацию об объекте: Высоту, качество материала, количество древесины и стоимость. Создать также класс наследник **Стул** с дополнительным свойством: высота спинки (h2, в см), и переопределить метод «количество древесины», по формуле d+2h2+5, где d – количество древесины, которые требует табуретка с такими же параметрами (Метод «стоимость» не переопределять). В главной программе создать экземпляры классов **Табуретка** и **Стул**, и напечатать информацию в таком виде: «табуретка» + информация о табуретке и «стул» + информация о стуле.
9	Создать класс **Фильм** со свойствами: Название, Режиссер, длительность (в минутах), количество актеров. Определить виртуальный метод: «Стоимость», возвращающую примерную расчетную стоимость фильма (в тыс. $), рассчитываемую по формуле длительность*20 + количество актеров*30, но если режиссер = «Стивен Спилберг» или «Джеймс Кэмерон», то стоимость в два раза выше (по сравнению с вышеуказанной формулой). Определить также метод «Информация», который возвращает строку, содержащую информацию о фильме: Название, режиссера, длительность, количество актеров и стоимость. Создать также класс наследник **Мультфильм**, в котором переопределить метод «Стоимость» по формуле длительность*25 + количество актеров*10 (вне зависимости от режиссера). В главной программе создать 2 **фильма** с режиссерами: «Стивен Спилберг» и «Ежи Гофман», а также **мультфильм** и вывести информацию о них.
10	Создать класс **Самолет** со свойствами: Марка, Модель, Максимальная скорость (в км/ч), Максимальная высота (в метрах). Определить виртуальный метод «Стоимость» – стоимость самолета, рассчитываемую по формуле Максимальная скорость * 1000 + Максимальная высота * 100 Определить также метод «Информация», который возвращает строку, содержащую информацию об объекте: Марка, Модель, Максимальную скорость, Максимальную высоту и Стоимость. Создать также класс

	наследник **Бомбардировщик**, в котором переопределить метод «**Стоимость**», который вернет удвоенную стоимость относительно формулы для класса **Самолет**. Также создать класс **Истребитель** – наследник класса Самолет, для которого переопределить метод «Стоимость» как утроенную стоимость, относительно формулы стоимости для Самолета. В главной программе создать объект класса **Самолет**, класса **Бомбардировщик**, класса **Истребитель**. Вывести на экран (или форму) информацию о самолетах.

Индивидуальное задание 2

1	Программно промоделировать разговор людей. Всего есть 5 людей. Каждый человек имеет имя (строку) и возраст (число). Установите возраст каждого человека - случайное число от 20 до 40, а имена установите случайным образом из списка «Александр», «Андрей», «Анастасия», «Ирина», «Наталья», «Павел», «Роман», «Светлана», «Сергей», «Татьяна». Любой человек способен выполнять два действия: - здороваться с другим человеком; - рассказывать о себе. Люди делятся на 3 типа (разные классы): 1 (Формалисты) Здороваются со всеми так: «Здравствуй, <имя>», где <имя> – имя человека, с которым он здоровается. 2 (Неформалы) Со всеми здороваются: «Привет, <имя>!» 3 (Реалисты) Если возраст собеседника меньше или равен или больше не более чем на 5 лет, говорит «Привет, <имя>!», иначе «Здравствуй, <имя>». В программной реализации приветствие должно быть реализовано как полиморфный метод, принимающий параметр – человек и возвращающий строку. Рассказ о человеке является строкой вида «Меня зовут Вася, мой возраст 21 лет, я неформал» (вместо Вася имя человека, вместо 21 его возраст, как видите у людей с грамматикой не все в порядке и они говорят «лет» после любого числа, реализовать правильную грамматику, вместо неформал может быть формалист, либо реалист). Программа должна показать информацию обо всех людях . Затем все люди должны поздороваться друг с другом в таком порядке: первый здоровается со вторым, потом второй с первым, потом первый с третьим, третий с первым, и так далее первый со всеми и все с первым, потом второй с третьим, третий со вторым и т.д. Нужно выдать в отдельных строках имя человека, который здоровается, двоеточие, приветствие Петя: Привет, Вася! Вася: Здравствуй, Петя Петя: Привет, Женя!
2	Программно промоделировать стрельбу по мишени группой человек. Каждый человек имеет свое имя, возраст (в годах) и стаж обучения стрельбе (в годах). Люди делятся на новичков, опытных и ветеранов (потомки класса человек). Для каждого человека определите полиморфный метод. «Стрелять» без параметров, возвращающих логическое значение (попал – true, не попал – false). Попадание определяется случайным образом, причем для новичка вероятность попасть равна 0,01*стаж обучения; для опытного = 0,05*стаж обучения стрельбе; для ветерана = 0,9 – 0,01*возраст. Люди стреляют по очереди, начиная с первого, пока кто-то не попадет в мишень. Стрельба прекращается после того, как кто-то попал или все выстрелили по одному разу. После каждого выстрела нужно выводить на экран всю информацию о стреляющем и результат стрельбы. В главной программе создайте массив из 7 людей в таком порядке: новичок, опытный, ветеран, опытный, новичок и произведите стрельбу с выводом ее результатов.
3	Программно промоделировать сдачу зачета студентами. Каждый студент характеризуется ФИО и количеством посещенных занятий. Также известно количество общих занятий = 20. Студенты делятся на обычных, сообразительных и гениев (потомки класса студент). Для каждого

	человека определите полиморфный метод «Сдать зачет» без параметров, возвращающих логическое значение (сдал – true, не сдал – false). Обычные студенты точно сдают зачет, если посетили все занятия, если были более чем на половине занятий, то сдают с вероятностью 0,5; иначе не сдают. Сообразительные студенты тоже точно сдают зачет, если посетили все занятия, если были более чем на половине занятий, то сдают с вероятностью 0,7; иначе не сдают зачет Гении точно сдают зачет, если были хотя бы на одном занятии, иначе не сдают. Создайте массив из 10 студентов (5 обычных, 4 сообразительных и 1 гения), задайте их ФИО и количество посещенных занятий константами, указанными в конструкторах объектов и промоделируйте сдачу зачета с выводом подробных результатов (всех сведений о студентах, а также результате сдачи).
4	Промоделировать отливку листов стали. Листы стали характеризуются толщиной (в мм), и плотностью стали (в кг/м3). Листы делятся на квадратные (дополнительно задаются одним числом – шириной и длиной одновременно, в мм), прямоугольные (задаются шириной и длиной, в мм) и треугольные (в виде прямоугольного треугольника, задаются двумя катетами, в мм). Для каждого типа стали определите виртуальный метод «Площадь» – возвращающий площадь листа. Также определите метод «Вес», который вычисляет вес листа, умножая площадь листа на его толщину и плотность стали. Также задайте метод «Информация», который будет выдавать информацию об листе. В главной программе создайте массив из 15 листов стали, создав 5 квадратных, 7 прямоугольных и 3 треугольных листа случайных размеров. Выведите информацию о листах и посчитайте суммарную площадь и суммарный вес всех листов.

ВОПРОСЫ ВЫХОДНОГО КОНТРОЛЯ:

1. Дайте определение наследования.
2. Перечислите методы доступа.
3. Перечислите механизмы наследования.

Домашнее задание: закрепить навыки применять различные механизмы наследования.