Лабораторно упражнение № 9

Наследяване на класове

I. Теоретична част

1. Механизъм „наследяване на класове”

В обектно-ориентираното програмиране се използва един мощен механизъм за построяване на отношения между класовете, известен като **наследяване** (***inheritance***). Този механизъм позволява даден клас да наследи друг. Класът, който наследява се нарича **наследник** или **производен клас**. Класът, от който се наследява се нарича **предшественик** или **базов (основен) клас**.

Процесът на наследяване по отношение на базовите и производните класове се изразява в следното:

- производният клас наследява структурата на основния т.е. неговите полета методи и свойства;
- производният получава достъп до наследените от основния клас членове;
- освен наследените, производният клас може да има свои собствени членове (полета, свойства и методи);
- достъпът на методите на производния клас до наследените членове от основния се регламентират от модификаторите за достъп ***public***, ***private*** и ***prortected, internal***;
- производният клас, от своя страна може да е базов за други класове;
- един клас може да е основен за няколко производни. По този начин се получава йерархична структура, породена от механизма на наследяване;
- в C#, за разлика от C++, се допуска един производен клас да наследи само един основен клас т.е. език C# не поддържа **множествено** наследяване на класове.

в език C# всеки клас може да бъде наследен, стига това да не е изрично забранено с ключова дума ***sealed***.

2. Деклариране на производни класове в C#

Декларацията на производен клас в C# е следната:

```
class ИмеПроизводенКлас : ИмеБазовКлас
{
        // тяло на производен клас
}
```

Пример:

```
class Point
  {
```

```
    private int x, y;
    public void Input()
    {
        Console.Write("X=");
        string temp = Console.ReadLine();
        x = Int32.Parse(temp);
        Console.Write("Y=");
        temp = Console.ReadLine();
        y = Int32.Parse(temp);
    }
    public void Output()
    {
        Console.WriteLine("( {0}, {1} )",x,y);
    }

  }

class Circle : Point
  {
    private int radius;

    public int Radius
    {
        get { return radius; }
        set { value = radius; }
    }
    public double Area
    {
        get
        {
            return radius * radius * Math.PI;
        }
    }
    public void InputRadius()
    {
        string temp;
        Console.Write("Radius = ");
        temp = Console.ReadLine();
        radius = Int32.Parse(temp);
    }
  }
```

Когато се дефинира обект от производен клас, той съдържа полета както от наследената част на базовия клас, така и собствени полета за производния клас:

```
Circle cir= new Circle();
```

3. Конструктори на базови и производни класове

Конструкторите на класовете отговарят за инициализацията на полетата (данните в класа). Производният клас автоматично получава всички елементи на базовия. Всеки един клас има поне един конструктор. Ако няма такъв компилаторът генерира служебен конструктор по подразбиране.

Конструкторите на базовите и на производните класове отговарят за инициализацията на собствените си членове т.е. конструкторът на производния

клас инициализира собствените си полета, а конструктора на базовия – наследените.

Когато са дефинирани конструктори с параметри, тогава програмистът трябва да се погрижи за явното и извикване като използва ключова дума ***base***. Извикването на конструктор на базов клас е чрез следния синтаксис:

public ИмеПроизводенКлас(списък формални параметри) : base(списък фактически параметри)
{
    // тяло на конструктора на производния клас
}

Пример:

```
class Point
  {
    private int x, y;
    public Point()
    {
      x = y = 0;
    }
    public Point(int x, int y)
    {
      this.x = x;
      this.y = y;
    }


  }

class Circle : Point
  {
    private int radius;
    public Circle():base()
    {
      radius = 0;
    }
    public Circle(int x, int y, int radius)
      : base(x, y)
    {
      this.radius = radius;
    }

    }
  }
```

II. Задачи за изпълнение:

1. Да се създаде йерархия от класове, описващи окръжност, кръгов конус и цилиндър. Всеки един от класовете да съдържа конструктор (по подразбиране) за инициализация стойностите на полетата. Класът, описващ окръжност да съдържа метод за пресмятане лице на кръг, а класовете описващи цилиндър и конус – методи за пресмятане обема на фигурите. Да се създаде демонстрационна програма за работа с тези класове. (да се дефинират обекти от класовете като се използват и двата конструктора).

2. Да се дефинира клас **Person**, описващ лични данни на персона: имена, ЕГН, лична карта и производен клас **Student**, описващ студент с данни – специалност, група, факултетен номер. Класовете да съдържат методи за въвеждане и извеждане стойностите на полетата.