Лабораторная работа №2. Проектирование и реализация классов

Часть I. Поля и методы класса

Задание 1: Создайте класс **Point** для описания точки на координатной плоскости.

1. Создайте решение **Лабораторная работа 2** и в нем проект **Example_1**.
2. Добавьте новый класс:
   1) в главном меню выберите пункт: **Проект – Добавить класс…**
   2) в качестве шаблона выберите **Класс**;
   3) введите имя **Point.cs**;
   4) нажмите кнопку **Добавить**.
3. В модуле **Point.cs** создайте класс, добавив следующий программный код:

```
class Point
{
   private int x; //координата X
   private int y; //координата Y

   //Метод установки значений полей
   public void SetXY (int x, int y)
   {
      this.x = x;
      this.y = y;
   }

   //Перегрузка метода ToString класса System.Object,
   //формирующего строковое представление объекта
   public override string ToString()
   {
      return String.Format("x={0}; Y={1}", x, y);
   }
}
```

4. В модуле **Program.cs** создайте объект – экземпляр класса **Point**:

```
Point P1 = new Point();
```

5. Заполните поля объекта и выведите на экран координаты точки:

```
P1.SetXY(10, 10);                  //изменение координат точки
Console.WriteLine(P1.ToString())  //вывод информации о точке на экран
```

6. Проверьте правильность работы программы.
7. Дополните класс свойством для чтения и записи координаты **x**:

```
public int X //свойство X
{
   get //чтение
   {
         return x;  //возвращаем значение поля x
   }
   set //запись
   {
         x = value; //записываем в поле x новое значение из value
   }
}
```


8. **Самостоятельно** реализуйте аналогичное свойство для координаты **y.**
9. Создайте объект-точку **P2**.
10. В модуле **Program.cs** проверьте правильность реализации свойств:

```
//установка значений через свойства
P2.X = 15;
P2.Y = 20;

//чтение значений полей через свойства
Console.WriteLine("x = {0}, y = {1}", P2.X, P2.Y);
```

11. Дополните класс **Point** конструктором без параметров, задающим нулевые значения координат:

```
public Point() //конструктор по умолчанию
{
   x = 0;
   y = 0;
}
```

12. В модуле **Program.cs** создайте точку **P3** и сразу же вызовите ее метод вывода на экран. Оцените результат.
13. Добавьте конструктор точки с параметрами:

```
public Point(int x, int y) //конструктор точки с параметрами
{
   this.x = x;
   this.y = y;
}
```

14. Создайте точку **P4**, используя конструктор с параметрами:

```
Point P4 = new Point(10, 15);
```

15. Выведите на экран координаты точки **P4** и проверьте результат.
16. Дополните класс **Point** методом «рисования» точки на экране:

```
public void Draw()    //метод рисования точки
{
   Console.SetCursorPosition(x, y);             //установка позиции курсора
   Console.ForegroundColor = ConsoleColor.Blue; //установка цвета текста
   Console.Write("*");                          //вывод символа на экран
   Console.ForegroundColor = ConsoleColor.Gray; //установка исходного цвета текста
}
```

17. Используя метод «рисования» точек, сформируйте на экране первую букву своего имени.
18. Дополните класс **Point** методом **DistanceTo**, определяющим расстояние от текущей точки до другой:

```
public int DistanceTo(Point other)
{
   int xDiff = this.x - other.x; //вычисляем разницу между координатами текущей и
   int yDiff = this.y - other.y; //другой точками

   //вычисляем расстояние между точками
   int distance = (int) Math.Sqrt(xDiff * xDiff + yDiff * yDiff);
   return distance;              //возвращаем результат
}
```

19. **Самостоятельно** дополните класс **Point** статическим полем **Count** для подсчета количества созданных экземпляров класса. В модуле **Program.cs** приведите пример его использования.

Часть II. Композиция классов

Задание 2. Создайте класс **HorizontalLine** для описания горизонтальной линии, состоящей из точек на координатной плоскости.

1. Добавьте в проект **Example_1** файл класса **HorizontalLine.cs**:

```
class HorizontalLine
{
   List <Point> pList; //список, элементы которого - объекты класса Point

   public HorizontalLine(int leftX, int rightX, int y) //конструктор
   {
         pList = new List<Point>();          //создание списка
         for (int x=leftX; x<=rightX; x++) //перебор точек слева на право
         {
               Point p = new Point(x, y);  //вызов конструктора класса Point
               pList.Add(p);               //добавление точки в список
         }
   }

   public void Draw()                //метод прорисовки линии
   {
         foreach (Point p in pList) //для каждой точки из списка
               p.Draw();            //вызываем метод рисования
   }
}
```

2. В модуле **Program.cs** создайте экземпляр класса **HorizontalLine**, указав в вызове конструктора координаты крайней левой, крайней правой точек линии и координату по вертикали:

```
HorizontalLine Hl = new HorizontalLine(10,20,15);
Hl.Draw();
```

3. Проверьте правильность работы программы.
4. **Самостоятельно** создайте горизонтальную линию **H2** с другими значениями и выведите ее на экран.
5. **Самостоятельно** дополните класс Горизонтальная линия методом **IsInLine**, определяющим, принадлежит ли заданная точка выбранной линии. В модуле **Program.cs** приведите пример его использования.

Часть III. Абстрактный класс и наследование

Задание 3: Создайте иерархию классов Фигура (**Figure**) – Окружность (**Circle**).

1. Добавьте новый проект **Example_2**.
2. В проекте создайте абстрактный класс **Figure** (файл **Figure.cs)**:

```
abstract class Figure
{
   public abstract double Perimetr(); //нахождение периметра
   public abstract double Square(); //нахождение площади фигуры
}
```

3. Используя механизм наследования, создайте новый класс **Circle** (файл **Circle.cs**), для которого в качестве базового укажите класс **Figure**:

```
class Circle: Figure
{
   private int radius;

   public Circle() //конструктор по умолчанию
   {
   }

   public Circle(int _radius) //конструктор с параметрами
   {
         this.radius = radius;
   }

   public int Radius
   {
         set
         {
             if (value > 0) radius = value; //радиус не может быть отрицательным
         }
         get
         {
             return radius;
         }
   }

   public override double Square() //переопределенный метод базового класса
   {
         return Math.PI * Math.Pow(radius, 2);
   }
}
```

4. **Самостоятельно** реализуйте метод нахождения **периметра** (длины окружности), а так же перегрузите метод **ToString()**.
5. В модуле **Program.cs** попробуйте создать экземпляр **F1** абстрактного класса **Figure**. Проанализируйте сообщение, сгенерированное редактором. Закомментируйте этот программный код и добавьте в комментарий описание ошибки.
6. Создайте объект **C1** класса **Circle** и выведите на экран информацию о нем и величину площади.
7. Откройте диаграмму классов:
   1) в **Обозревателе решений** выделите все файлы классов;
   2) с помощью контекстного меню выберите пункт **Перейти к диаграмме классов…**;

3) раскройте описания классов и проанализируйте из значения.

8. Сохраните схему классов в папке с проектом: **Схема классов – Экспорт схемы как изображения**.
9. **Самостоятельно** реализуйте класс Прямоугольник (**Rectangle**), наследуемый от **Figure**. В модуле **Program.cs** создайте объекты нового класса и приведите примеры использования его методов. Обновите и сохраните в файл схему классов.

Часть IV. Перегрузка операторов и методов

Задание 4: Реализуйте класс Матрица (**Matrix**) для выполнения операций с квадратными матрицами.

1. Добавьте в решение новый проект **Example_3**.
2. Создайте класс **Matrix** (**Matrix.cs**) и определите в нем следующие поля и методы:

```
class Matrix
{
   private int a11, a12, a21, a22; //поля матрицы

   public Matrix()         //конструктор по умолчанию
   {
   }

   public Matrix (int a11, int a12, int a21, int a22) //конструктор с параметрами
   {
         this.a11 = a11;
         this.a12 = a12;
         this.a21 = a21;
         this.a22 = a22;
   }

   public override string ToString() //перегруженный метод класса System.Object
   {
         return String.Format("{0} {1}\n{2} {3}", a11, a12, a21, a22);
   }
}
```

3. Дополните описание Матрицы перегрузкой оператора сложения:

```
public static Matrix operator + (Matrix x, Matrix y) //перегрузка оператора +
{
   return new Matrix(x.a11 + y.a11, x.a12 + y.a12, x.a21 + y.a21, x.a22 + y.a22);
}
```

4. В модуле **Program.cs** создайте два объекта **M1** и **M2** класса **Matrix**, используя конструктор с параметрами, и объект **M3** с конструктором по умолчанию.
5. Выведите на экран значения **M1**, **M2** и **M3**, используя метод преобразования матрицы в строку:

```
Console.WriteLine("Матрица M1\n" + M1.ToString());
```

6. В **M3** присвойте результат сложения **M1** и **M2**:

```
m3 = m1 + m2;
```

7. Выведите на экран новое значение **M3**.
8. **Самостоятельно** выполните перегрузку оператора вычитания.

Часть V. Задания для самостоятельного решения

Задание: Разработайте консольное приложение в среде MS Visual Studio на языке C# для решения задачи согласно своему варианту.

В программе должны быть предусмотрены:

1) конструкторы класса;
2) защита полей класса от присваивания некорректных значений;
3) примеры работы всех методов класса;
4) комментарии к основным блокам.

Вариант 1:

Класс	Поля	Методы
Треугольник	Длины сторон: a, b и c.	• вывод информации об объекте на экран; • нахождение площади; • нахождение периметра; • проверка, является ли равносторонним.

Вариант 2:

Класс	Поля	Методы
Компьютер	• тактовая частота процессора (ГГц); • количество ядер (шт.); • объем оперативной памяти (Гб); • объем жесткого диска (Гб).	• вывод информации об объекте на экран; • проверка, является ли многоядерным; • рейтинг (вычисляется по формуле: Тактовая частота * количество ядер + объем оперативной памяти + объем жесткого диска / 100).

Вариант 3:

Класс	Поля	Методы
Сотрудник	• ФИО; • образование; • год приема на работу; • оклад; • премия (% от оклада).	• вывод информации об объекте на экран; • вычисление стажа работы; • вычисление заработной платы (Оклад + Оклад * Премия).

Вариант 4:

Класс	Поля	Методы
Обыкновенная дробь	• числитель; • знаменатель.	• вывод информации об объекте на экран; • представление в десятичной форме; • оператор сложения; • оператор вычитания.

Вариант 5:

Класс	Поля	Методы
Равнобедренный треугольник	• длина основания; • длина высоты.	• вывод информации об объекте на экран; • нахождение площади; • нахождение периметра; • проверка, является ли равносторонним.

Вариант 6:

Класс	Поля	Методы
Время	• часы; • минуты; • секунды.	• вывод времени в 24-часовом формате (например, 13:05:00); • вывод времени в 12-часовом формате (например, 1:05:00 p.m.); • операторы сравнения (равно, неравно).

Вариант 7:

Класс	Поля	Методы
Дата	• день; • месяц; • год.	• вывод даты в формате полном формате (например, 12 декабря 2016 года); • вывод даты в коротком формате (например, 12.12.2016); • проверка, является ли год високосным.

Вариант 8:

Класс	Поля	Методы
Параллелепипед	• a, b, c – длины сторон • количество созданных объектов (статическое поле)	• вывод информации об объекте на экран; • нахождение площади поверхности; • нахождение объема; • проверка, является ли кубом.