Лабораторная работа №12.

Тема: Разработка классов.

Задание: Разработать и реализовать классы, которые являются основными в задании. Для сдачи лабораторной работы необходимо создать проект, в котором демонстрируется работа объектов созданных классов, а так же создать отчет по выполнению курсового проекта, в котором будут описаны классы и продемонстрирована работа программы. Должны быть продемонстрированы выполнение конструктора, всех доступных методов и деструктора.

Контрольные вопросы:

1. Дайте определение *класса*.
2. Дайте определение *объекта*.
3. Что понимается под *состоянием и поведением* объекта, как они представляются в классе?
4. Объясните достоинства и недостатки языков C++ и C#?
5. Поясните понятие *инкапсуляция*.
6. Поясните понятие *наследование*.
7. Поясните понятие *полиморфизм*
8. Что такое *пространство имён*?
9. Какие существуют особенности при *передаче значений параметров* в функциях языка C#?
10. Охарактеризуйте каждый *тип доступа* к данным и функциям класса.

Литература:

1. Марченко А. Л. C#. Введение в программирование - учебное пособие, МГУ, 2005 (L:\TASKS\BOOKS\C#\).
2. Герберт Шилдт - C# Учебный Курс - учебное пособие, Санкт-Петербург, 2003(L:\TASKS\BOOKS\C#\).
3. Информационный портал дистанционного образования каф ИСЭ www.lms.mvtom.ru

4. ВИДЕО-курс по дисциплине «Инструментальные средства разработчика»

ВЫПОЛНЕНИЕ РАБОТЫ

Разработать программу, имитирующую движение городского транспорта. Время процесса дискретно. В целях упрощения предполагается, что улицы города пересекаются под прямыми углами. Транспортные средства различных моделей двигаются с различными скоростями, меняя направление движения на перекрестках случайным образом. Любое транспортное средство имеет следующие характеристики – максимальную скорость, уровень загрязнения (г/м, считается, что за один шаг по времени загрязнение полностью разлагается), уровень шума (в децибелах), обратно пропорциональный расстоянию до транспортного средства (считается, что шум распространяется только вдоль проезжей части). Транспортные средства могут иметь внутренний или внешний (например, троллейбусы) источник энергии. Все транспортные средства делятся на пассажирские и грузовые. Пассажирские средства имеют максимальное количество пассажирских мест, грузовые – максимальный вес перевозимого груза. Транспортное средство может производить обгон, если оно имеет внутренний источник энергии, иначе оно должно двигаться со скоростью впереди идущей машины. Машины могут случайным образом покидать город, а также въезжать в город. Количество пассажиров, вес груза и начальная скорость задается случайно, так чтобы они не превышали свои максимальные значения. На каждом шаге по времени пользователь может вывести следующую информацию:

- Состояние любого транспортного средства;
- уровень шума в каждой точке проезжей части;
- уровень загрязнения в каждой точке проезжей части;
- количество пассажиров на любой улице;
- вес перевозимого груза на любой улице;
- количество пассажирских транспортных средств;
- количество грузовых транспортных средств;
- количество транспортных средств, не загрязняющих окружающую среду.

Запустите Microsoft Visual Studio 2008.

После запуска откроется стартовая страница – рисунок 1.

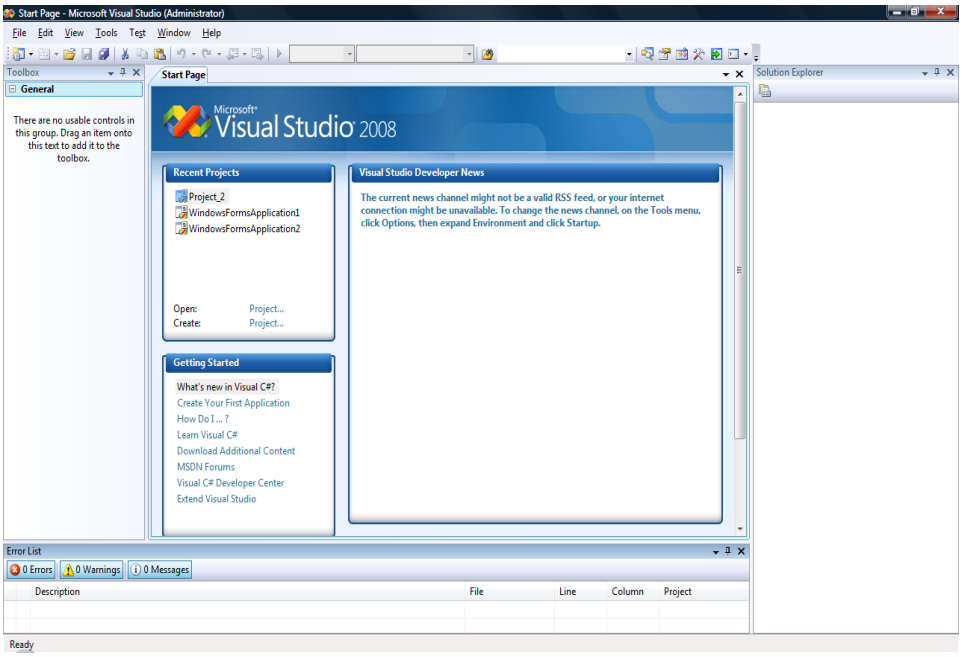

Рисунок 1 – Стартовая страница Microsoft Visual Studio 2008

Для создания нового проекта, выберете в меню File - пункт New, затем Project.

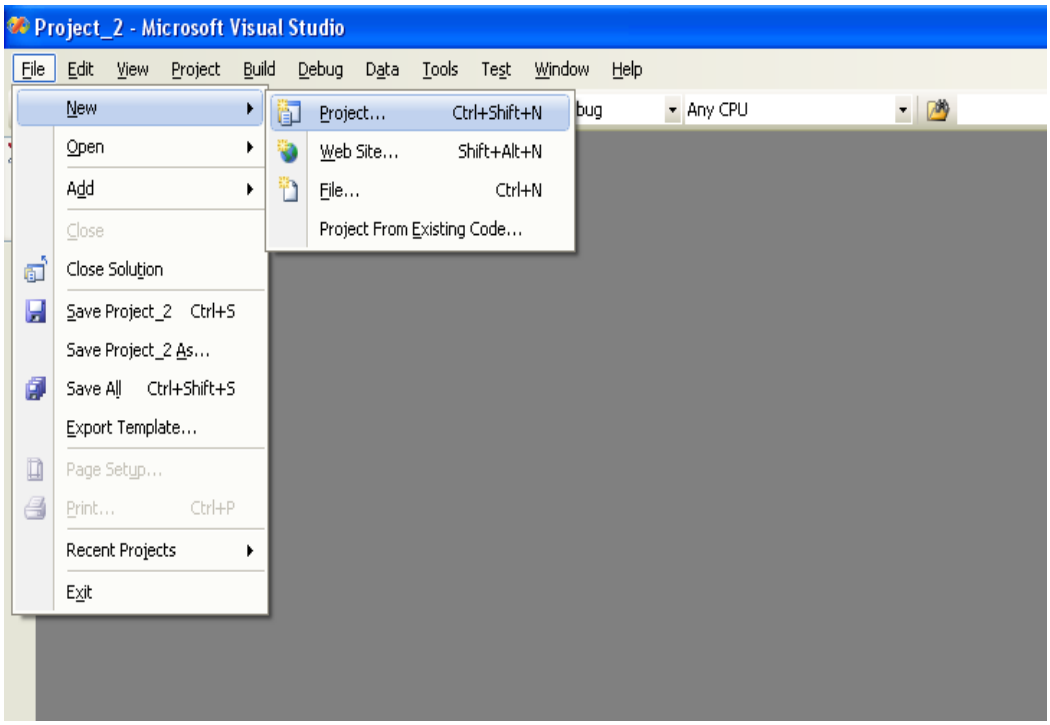

Рисунок 2 - Создание проекта с помощью меню

Так как вам необходимо создать проект, выберите Windows Forms Application. Также желательно изменить имя проекта.

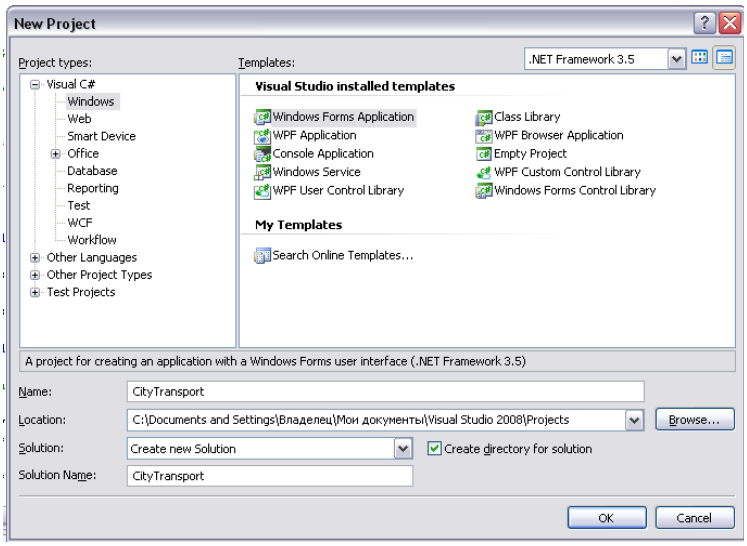

Рисунок 3 – Создание проекта Windows Forms Application

Для удобства работы с Microsoft Visual Studio 2008 необходимо добавить некоторые окна, работа с которыми вам предстоит. Для этого в меню View выберите пункты Task List, TollBox (для закрепления этого окна нажмите второй значок из ) и Properties Window, а также Error List.

Новый проект создан. После создания проекта открывается вкладка с редактированием формы.

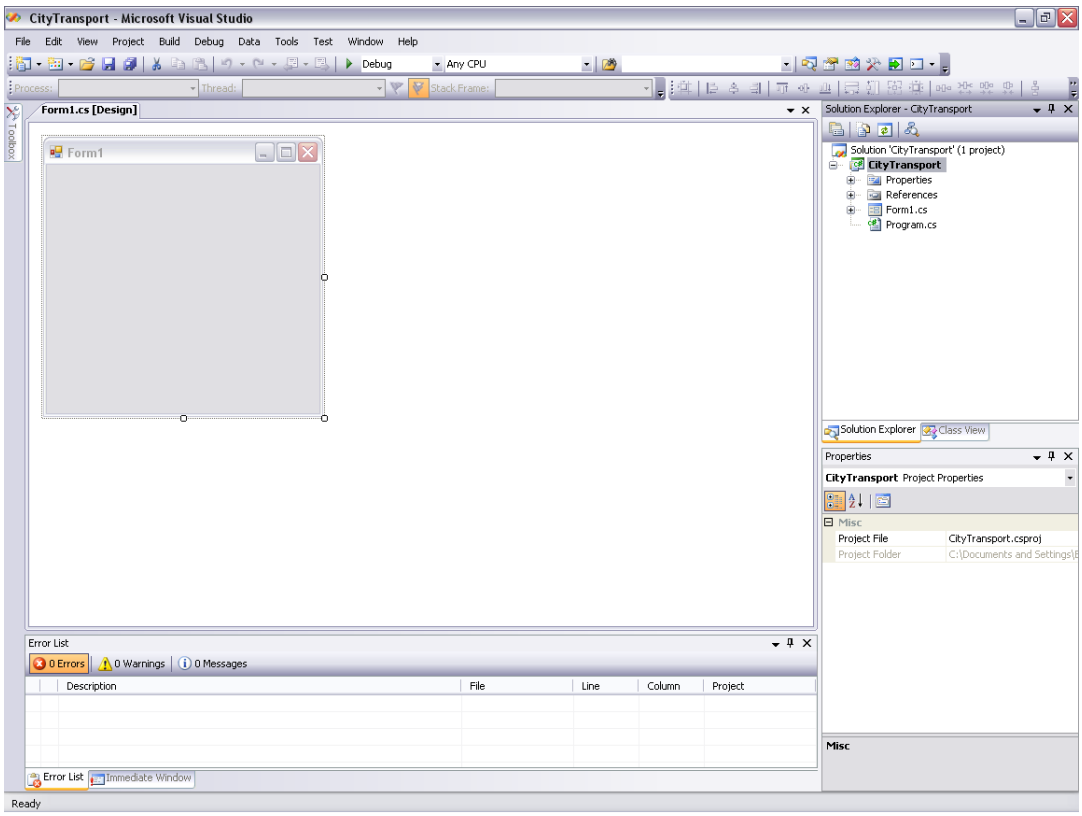

Рисунок 4 – Окно с редактированием формы

Если вам необходимо создать кнопки на форме, необходимо выбрать пункт Button и создать кнопку необходимого размера.

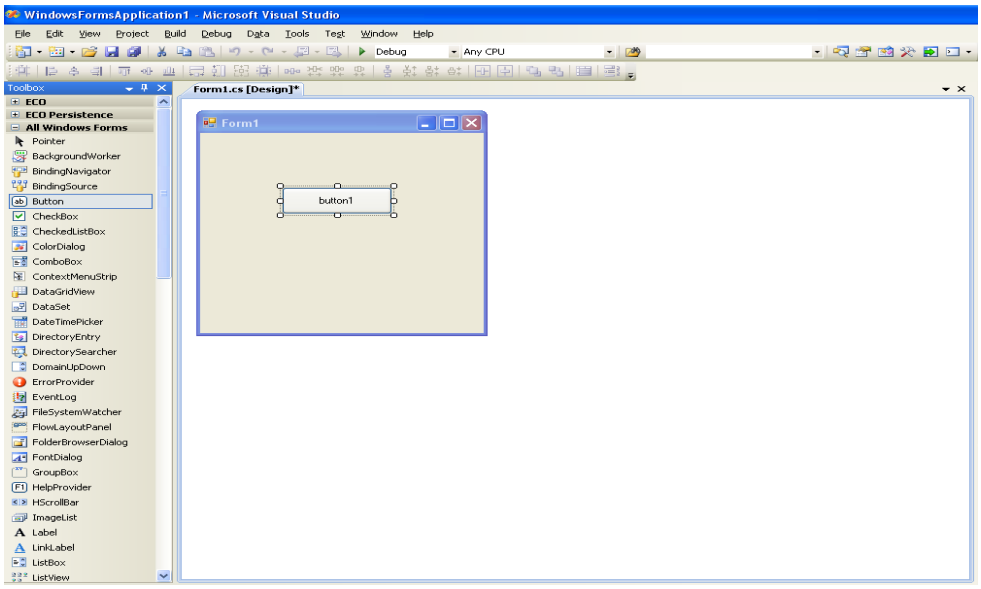

Рисунок 5 – Создание кнопки на форме

Для того, чтобы изменить текст, отображаемый на кнопке, изменить его в окне Properties, строка Text.

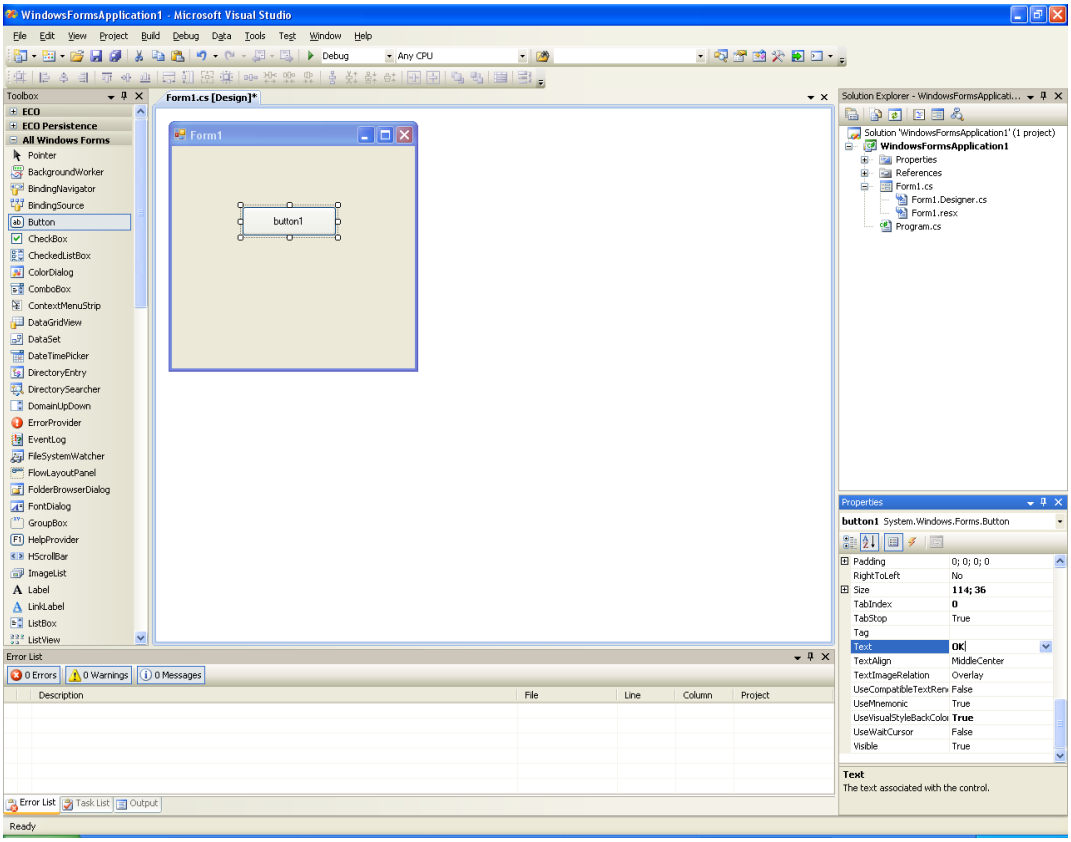

Рисунок 6 – Изменение текста, отображаемого на кнопке

После редактирования формы необходимо создать класс, который будет базовым для вашей программы. При определении класса объявляются данные, которые он содержит, и код, работающий с этими данными. Самые простые классы могут содержать только код или только данные, но в реальных программах классы включают обе эти составляющие. Данные содержатся в переменных экземпляра, которые определены классом, а код содержится в методах. При создании (определении) класса вначале указывается ключевое слово class.

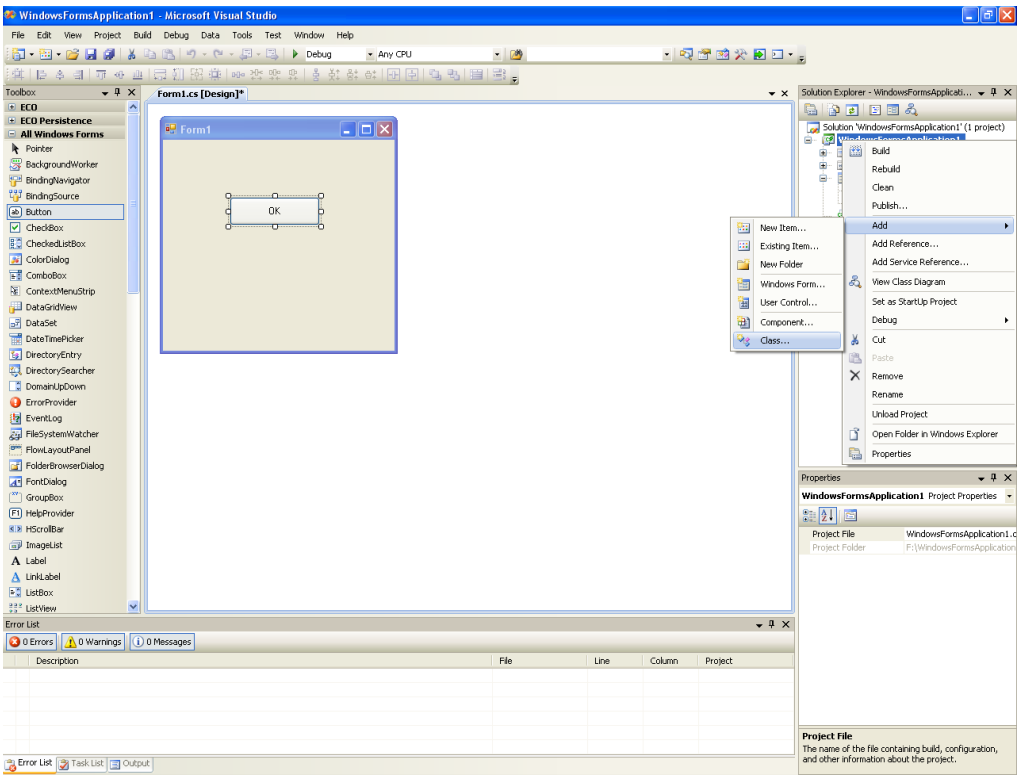

Рисунок 7 – Добавление класса из окна Solution Explorer

Также это можно сделать, открыв меню Project:

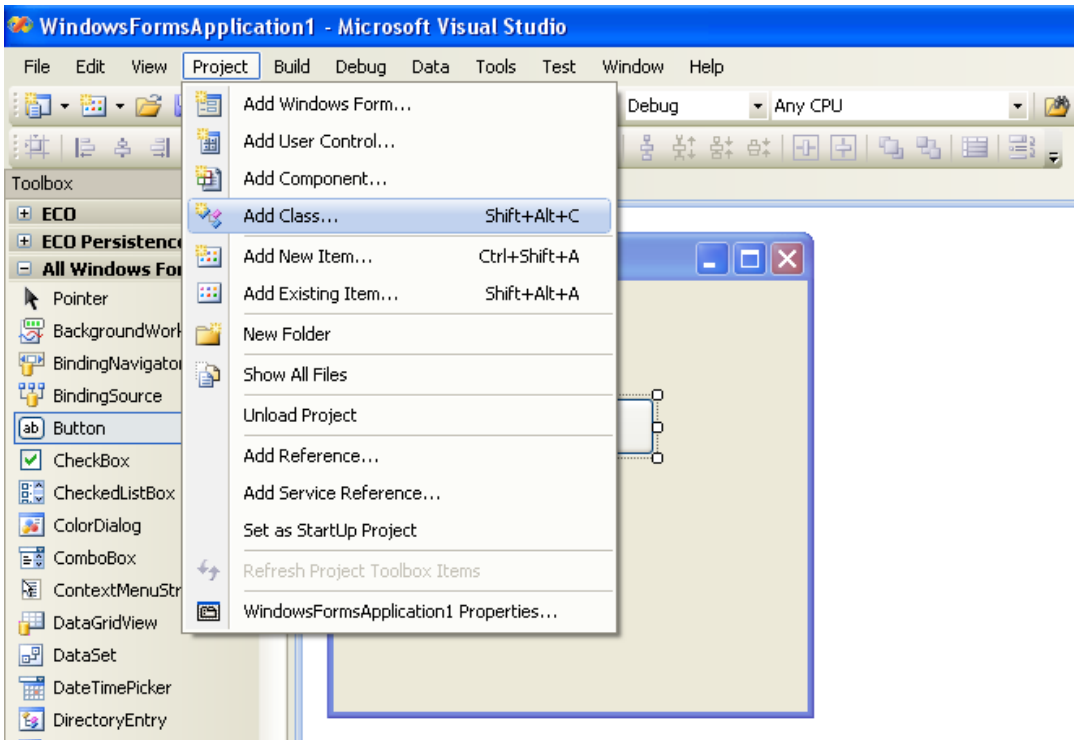

Рисунок 8 – Добавление класса из меню Project

Затем нужно изменить имя класса.

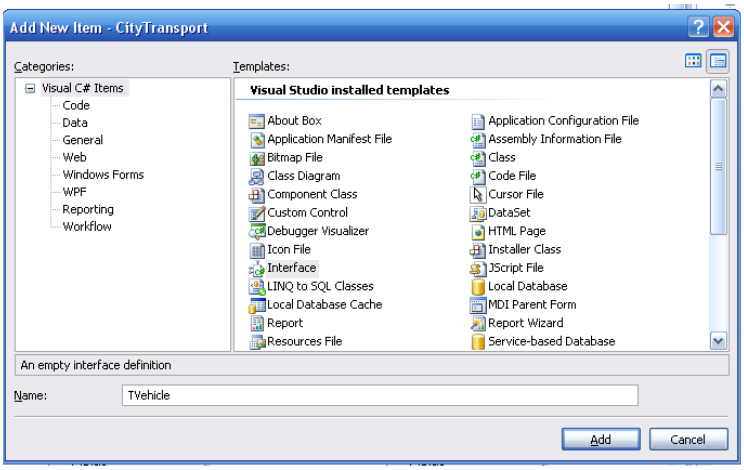

Рисунок 9 – Изменение имени добавляемого класса

Откроется окно с кодом класса.

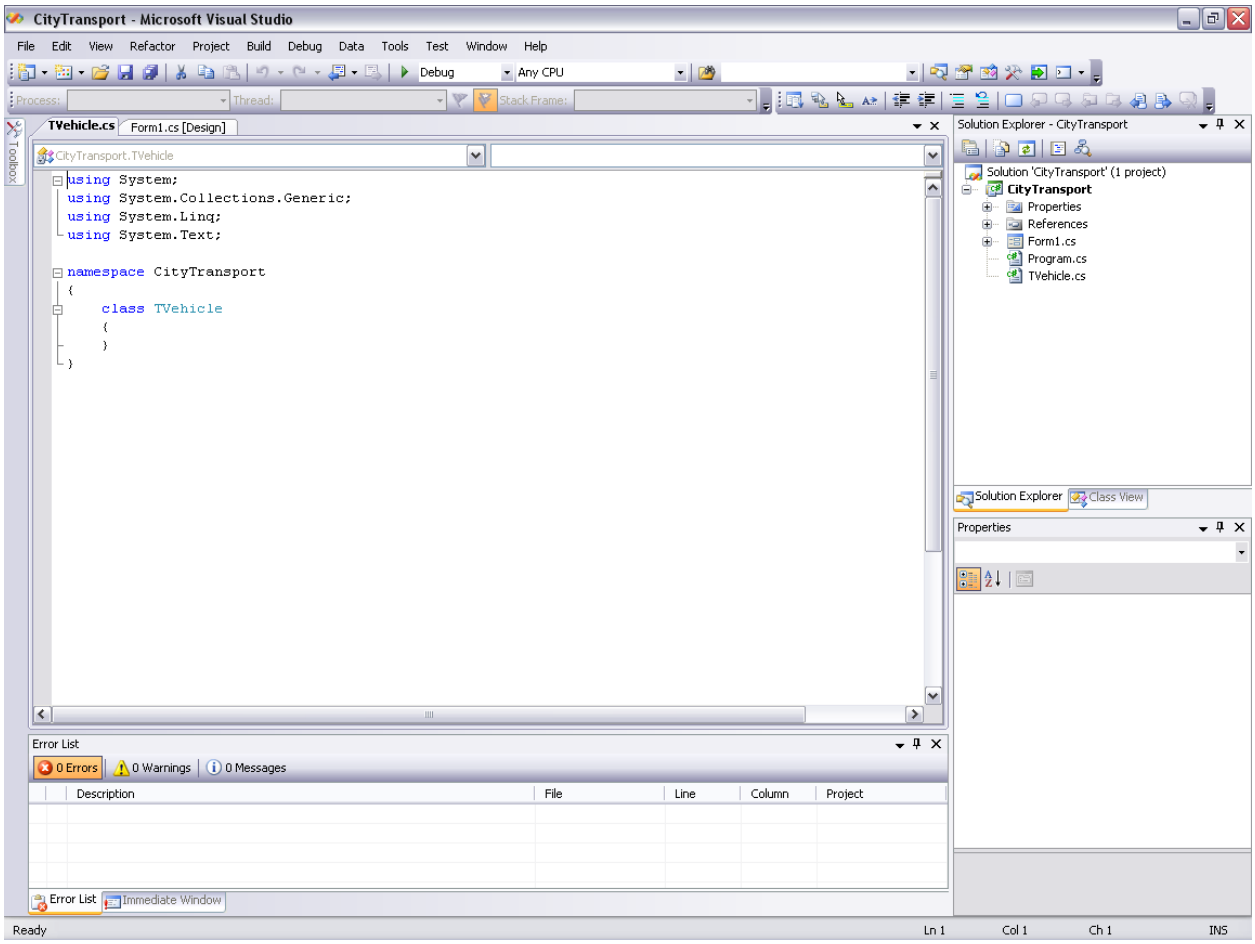

Рисунок 10 – Вкладка с кодом класса

Для редактирования класса удобно использовать диаграмму класса. Для ее создания необходимо открыть пункт View Class Diagram, щелкнув правой кнопкой мыши на названии класса.

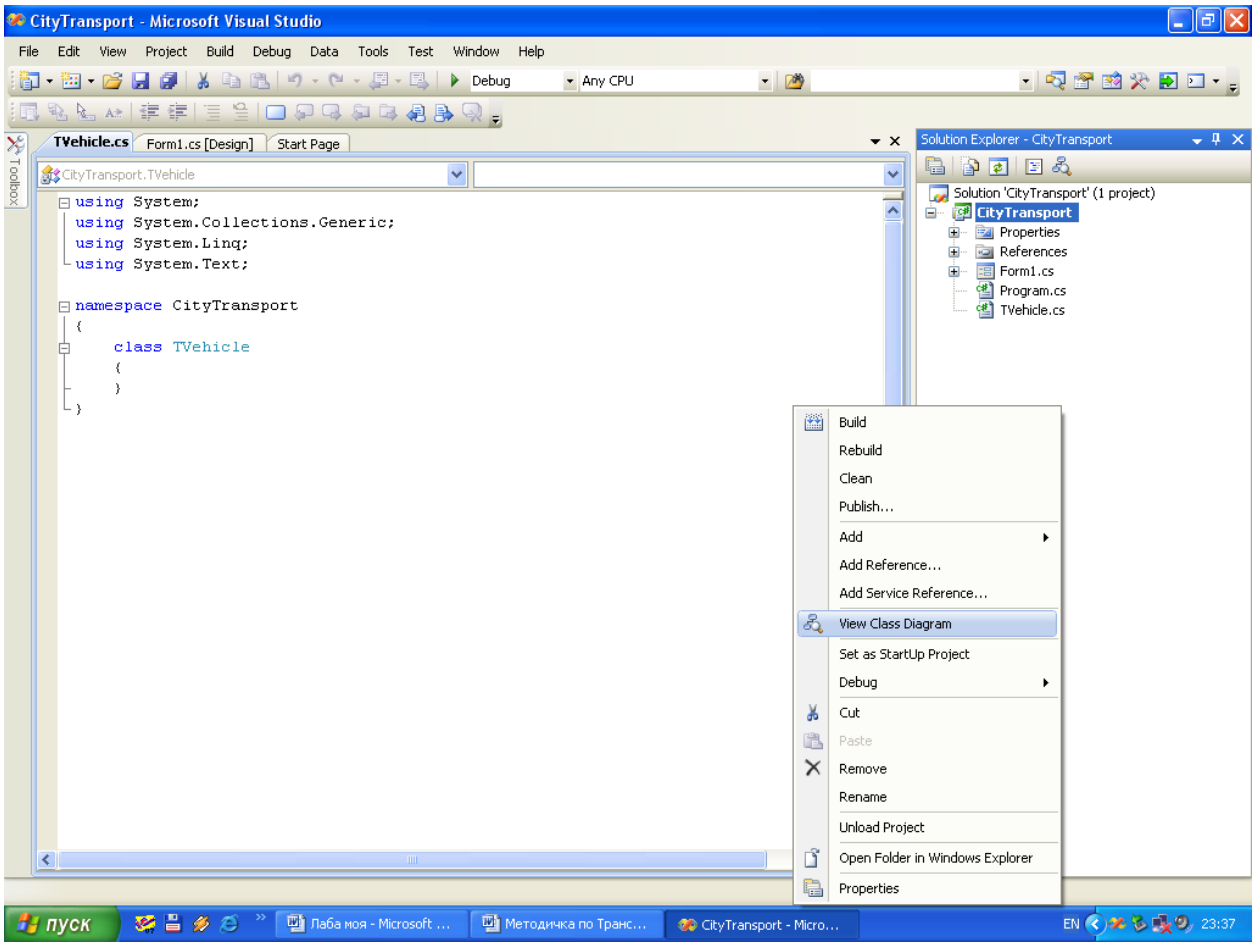

Рисунок 11 – Создание диаграммы класса

Для добавления методов, полей или конструктор можно использовать два способа.

Первый – щелкнуть правой кнопкой мыши и выбрать пункт Add.

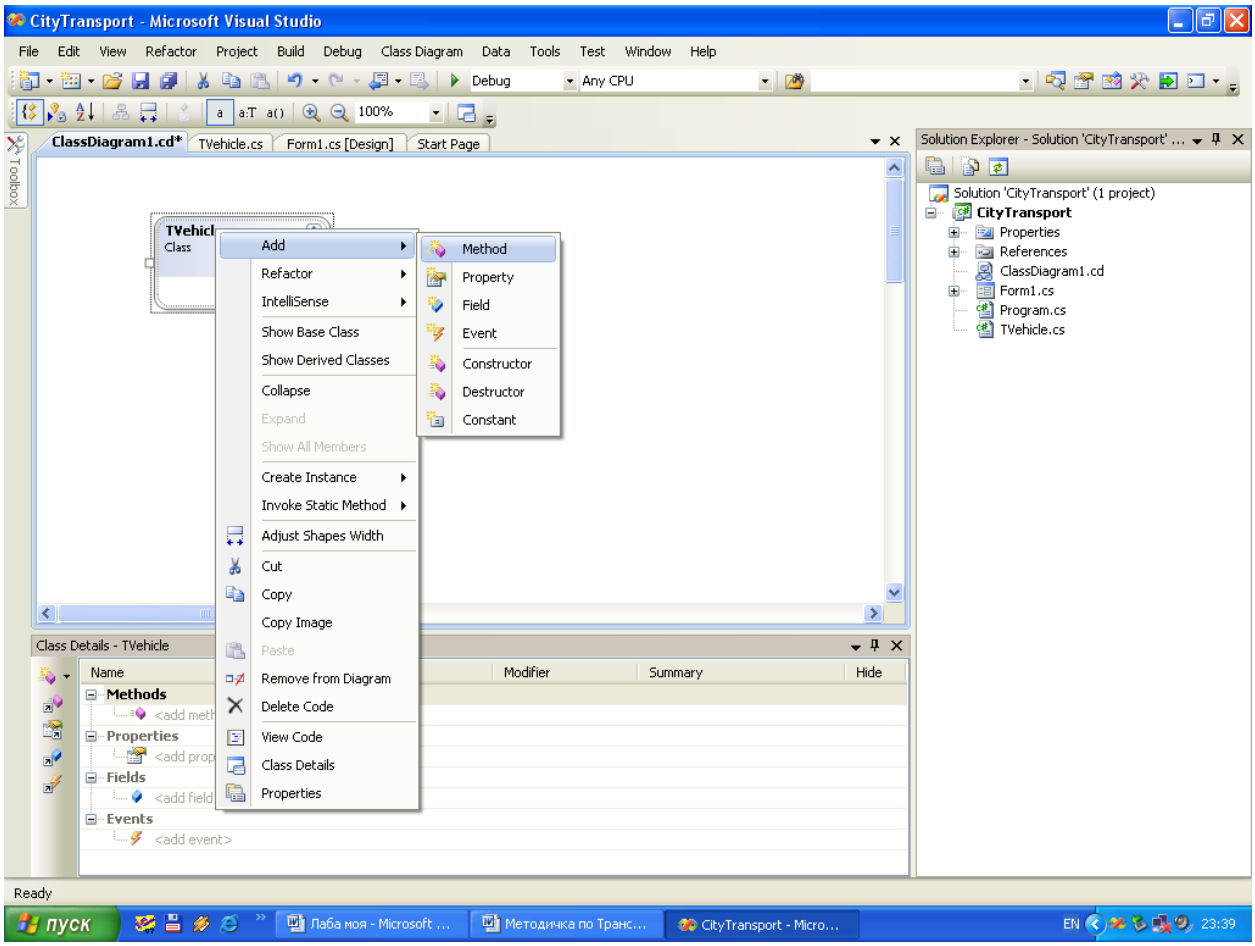

Рисунок 12 – Добавление методов, полей и конструкторов

Второй – в окне Class Details. Для открытия этого окна необходимо щелкнуть правой кнопкой мыши по блоку класса и выбрать пункт Class Details.

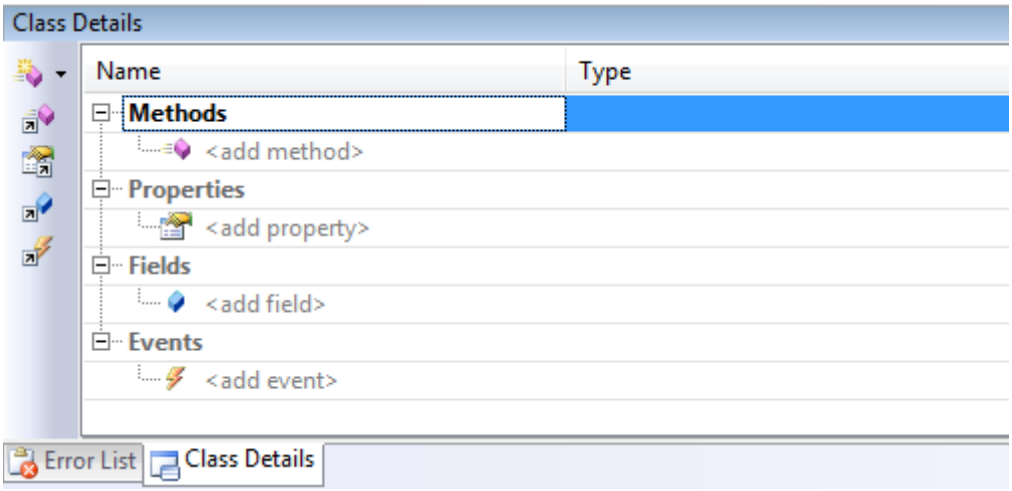

Рисунок 13 – Добавление методов, полей и конструкторов с помощью Class Details

В соответствии с заданием создадим **класс TVehicle** с полями:

```
        MaxSpeed; //максимальная скорость
        PolutionLevel; //уровень загрязнения
        NoiseLevel; //уровень шума
        EnergySource;     // источник энергии 1- внутренний, 0- внешний
        Speed, NormalSpeed; // текущая скорость и скорость движения без
помех
        Direction;     // направление движения (в радианах, 0-восток, pi -
запад, pi/2 - север)
        X, Y; //координаты
        Street; // улица
```

и методами:

```
        TVehicle // конструктор
        Type //тип т/с
        CanObgon//возможность обгона
```

После редактирования окно Class Details будет выглядеть следующим образом:

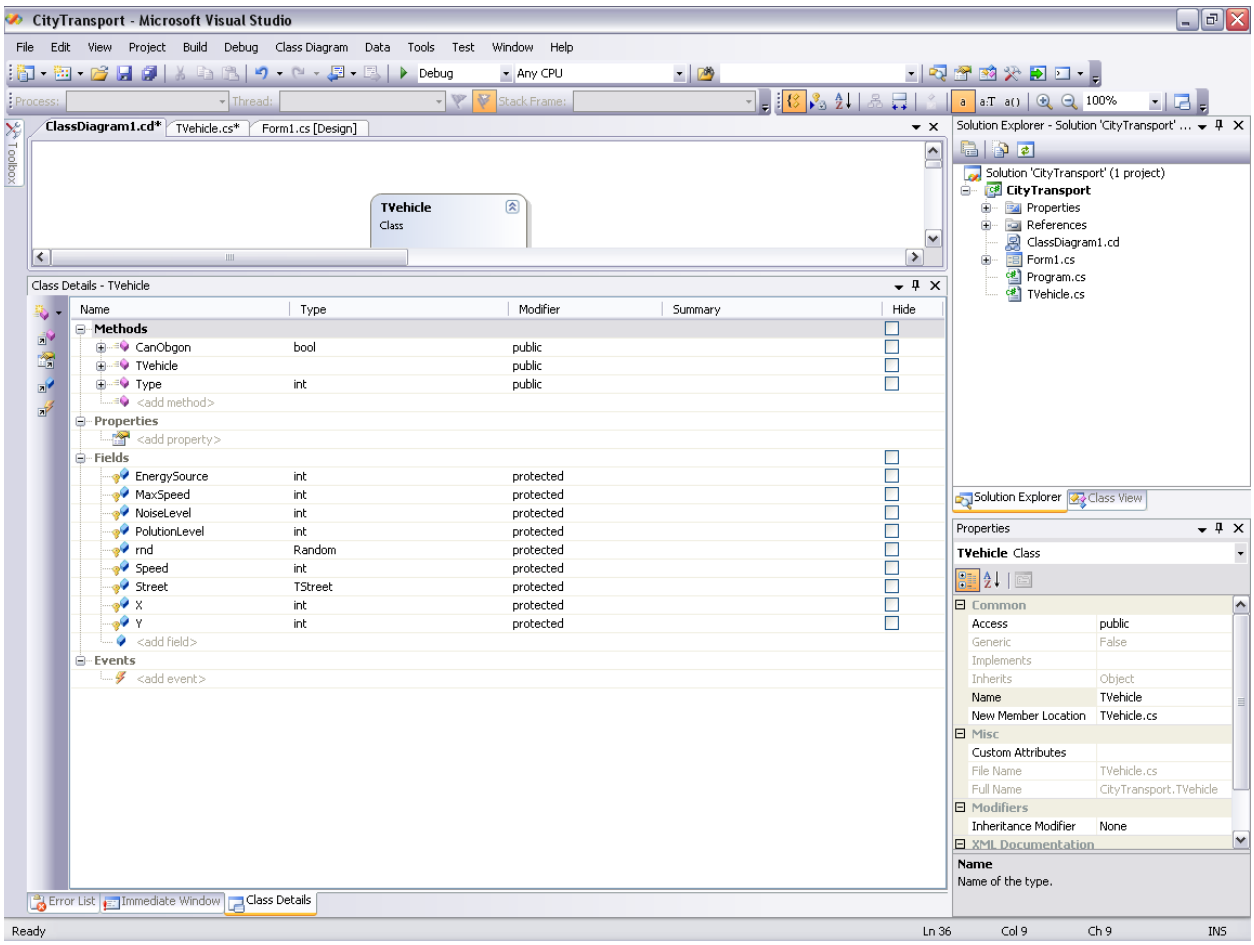

Рисунок 14 – Окно Class Details после редактирования класса

Методы и поля, записанные в Class Details, будут отображены и в диаграмме класса. Также они будут оформлены и в коде класса.

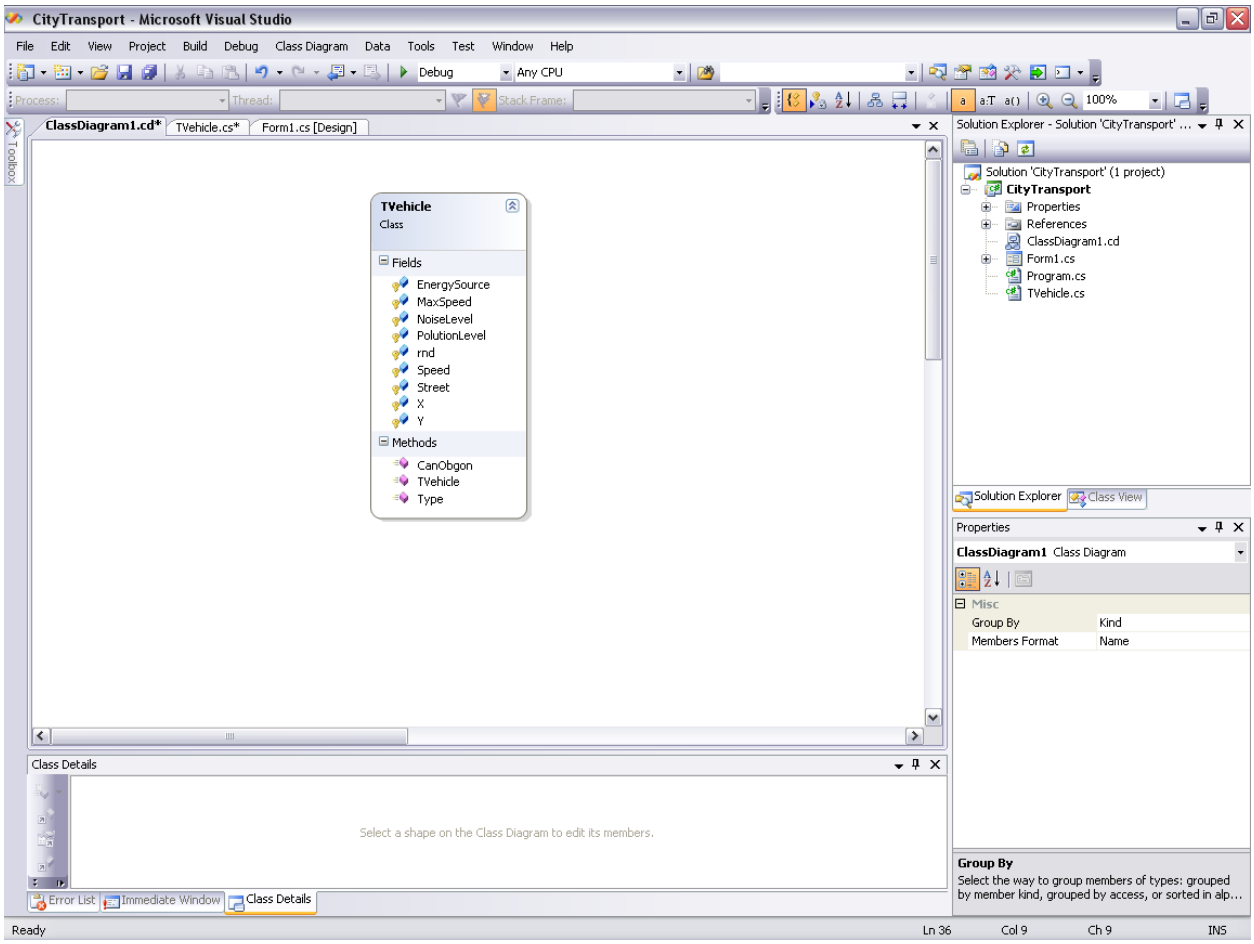

Рисунок 15 – Диаграмма класса после его редактирования

Класс TVehicle (базовый класс для всего транспорта)

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

namespace CityTransport
{
    public class TVehicle
    {
        protected int MaxSpeed;//максимальная скорость
        protected int PolutionLevel;//уровень загрязнения
        protected int NoiseLevel;//уровень шума
        protected int EnergySource;     // источник энергии 1- внутренний, 0-
внешний
        protected int Speed;//скорость
        protected int X, Y;//координаты
        protected TStreet Street;//улица

        protected Random rnd = new Random();

        public TVehicle()//конструктор класса
        {
        }
```

```
//методы класса
        public int Type ()  {}// тип т/с
        public bool CanObgon() {} // возможность обгона
    }
}
```

НАСЛЕДОВАНИЕ – свойство, с помощью которого один объект может приобретать свойства другого. Без использования наследования каждый объект должен явно определять все свои характеристики. Использую наследование, объект должен определить только те качества, которые делают его уникальным в пределах своего класса.

Для создания классов наследников необходимо создать новые классы и в коде указать родительский класс, например,

```
public class TCivilTransport: TVehicle
```

На диаграмме класс-наследник отобразится после выбора пункта Show Derived Classes.

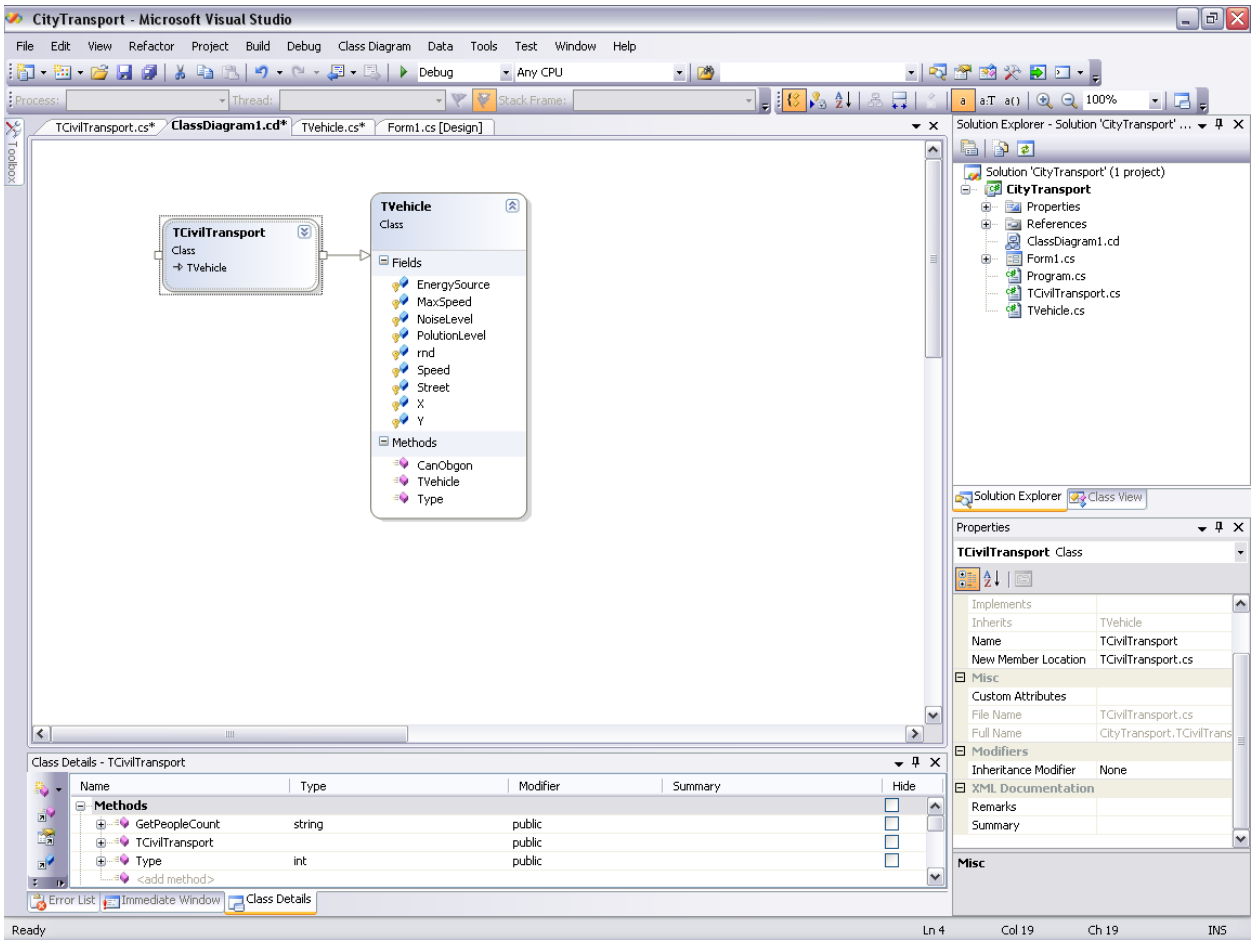

Рисунок 16 – Отображение классов-наследников

Редактируются классы-наследники аналогично. Однако могут возникнуть проблемы с объявлением конструктора, если в базовом классе он имеет свои параметры. Для решения этой задачи можно в базовом классе объявить еще один конструктор, но уже

без параметров. Тогда у классов-наследников могут быть свои конструкторы со своими параметрами.

Класс TCivilTransport (Пассажирский транспорт)

Поля:

```
MaxPeopleCount; // кол-во пассажирских мест
PeopleCount;    // кол-во пассажиров
```

Методы:

```
TCivilTransport//конструктор класса
GetPeopleCount ()//возвращает кол-во пассажиров
Type() //возвращает тип т/с
```

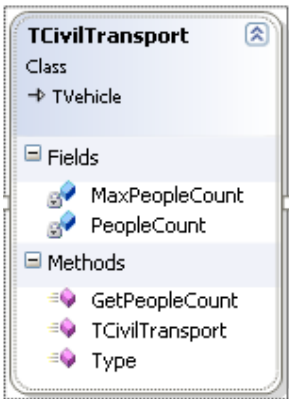

Рисунок 17 – Диаграмма класса TCivilTransport

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public class TCivilTransport: TVehicle // наследуемый класс от TVehicle
    {
        private int MaxPeopleCount; //  переменныя кол-во пассажирских мест
        private int PeopleCount;    // переменная кол-во пассажиров

        public TCivilTransport(int aMaxPeopleCount)//конструктор класса
TCivilTransport
        {
            MaxPeopleCount = aMaxPeopleCount;
            PeopleCount = rnd.Next(1, MaxPeopleCount);//кол-во пассажиров
равно случайному числу от 1 до кол-ва пассажирских мест
        }
//методы класса
        public string GetPeopleCount ()//возвращает кол-во пассажиров

        public int Type() //возвращает тип т/с
    }
}
```

Аналогично создадим наследуемый класс TLorryTransport и определим его методы и поля.

Класс TLorryTransport (Грузовой транспорт)

Поля:

```
MaxGruzMassa;    // переменная грузоподъемность
GruzMassa;       //  переменная масса груза
```

Методы:

```
TLorryTransport(int aMaxGruzMassa)//конструктор класса
TLorryTransport
GetGruzMassa ()//возвращает массу груза
Type()//возвращает тип т/с
```

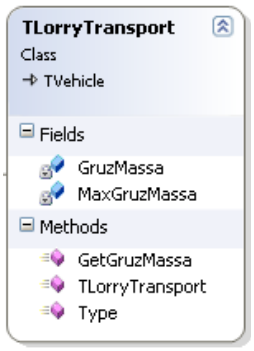

Рисунок 18 – Диаграмма класса TLorryTransport

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public class TLorryTransport: TVehicle // грузовые транспортные
средства,наследуемый класс от TVehicle
    {
        private int MaxGruzMassa;    // переменная грузоподъемность
        private int GruzMassa;       //  переменная масса груза

        public TLorryTransport(int aMaxGruzMassa)//конструктор класса
TLorryTransport
        {
            MaxGruzMassa = aMaxGruzMassa;
            GruzMassa = rnd.Next(1, MaxGruzMassa);//масса перевозимого груза
равна случайному числу от 1 до грузоподъемности
        }
//методы класса
        public string GetGruzMassa ()//возвращает массу груза


        public override int Type()//возвращает тип т/с
    }
```

Создадим еще один класс **TStreet** (Улицы)

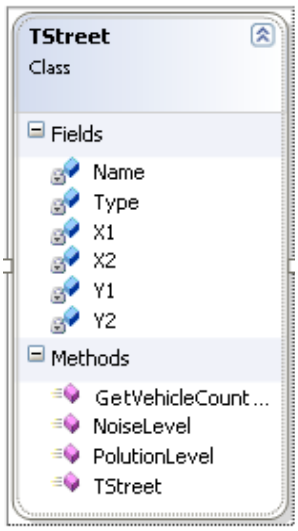

Рисунок 19 – Диаграмма класса TSreet

Класс TStreet (Улицы)

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{


    public class TStreet
    {
        private int Type; // 1- вертикальная (улица), 2- горизонтальная
(проспект)
        private int X1, X2, Y1, Y2; // координаты улицы
        private string Name; // название улицы

        public TStreet (string aName, int aX, int aY) // конструктор класса
        {
            Name = aName;
            X1 = aX; Y1 = aY;
            if (aX == 0) // проспект
            {
                Type = 2;
                X2 = 500; Y2 = Y1;
            }
            else // улица
            {
                Type = 1;
                X2 = X1; Y2 = 400;

            }
        }
// методы класса
        public double NoiseLevel(int L) // уровень шума
        {
            return 0;
        }
        public double PolutionLevel(int L) // уровень загрязнения
        {
```

```
            return 0;
        }
        public int GetVehicleCountOnStreet(int aVehicleType) // кол-во т/с на
улице
        {

            return 0;
        }


    }
}
```

После редактирования классов, создания новых классов, не связанных с базовым, диаграмма может выглядеть следующим образом:

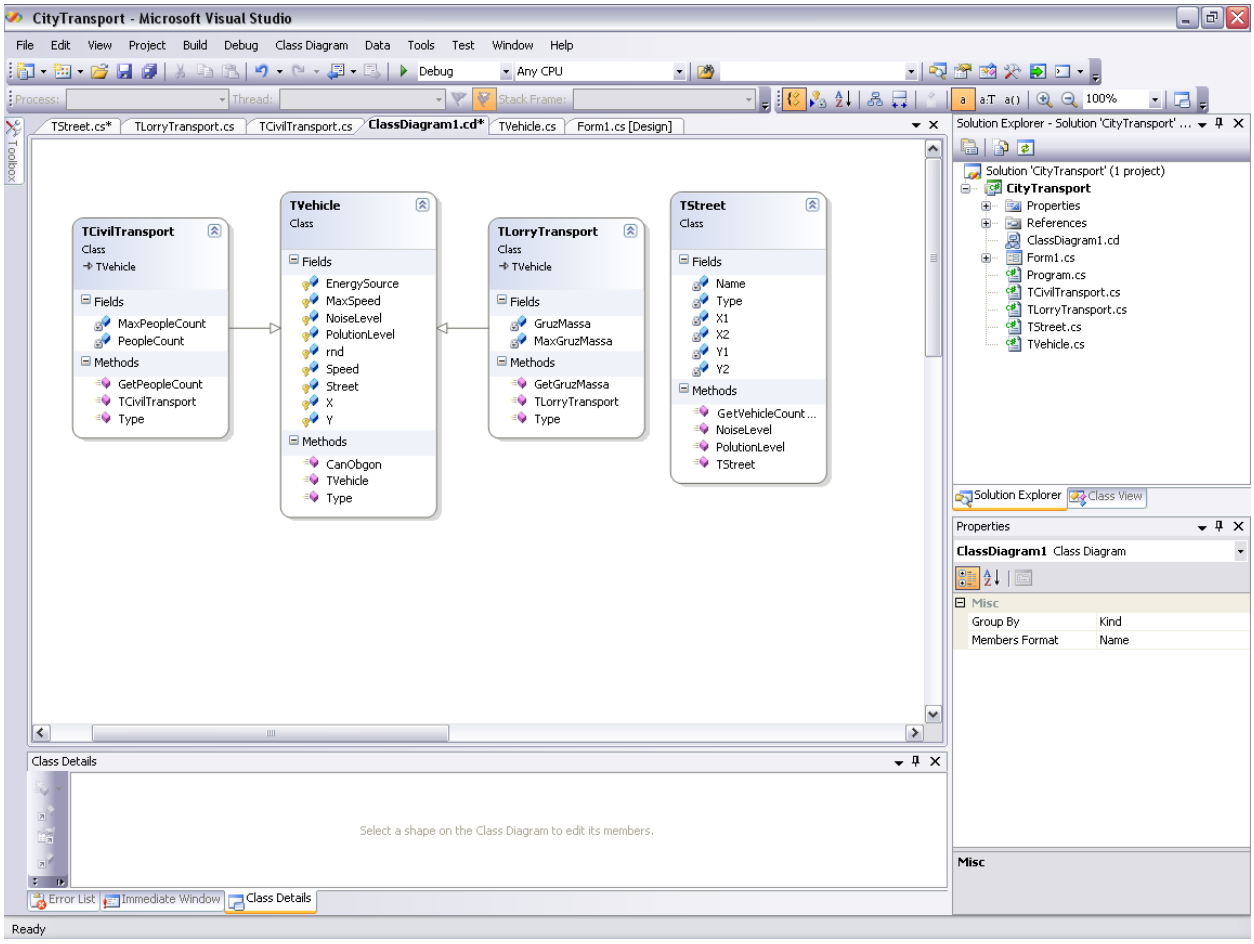

Рисунок 20 – Итоговый вариант диаграммы

Двойной щелчок по блоку класса переведет вас во вкладку с кодом программы этого класса.

После разработки базового класса TVehicle, производных классов TCityTransport, TlorryTransport реализуем их, добавив в базовый класс виртуальные методы.

Класс TVehicle

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

namespace CityTransport
{
    public class TVehicle
    {
        protected int MaxSpeed;//максимальныя скорость
        protected int PolutionLevel;//уровень загрязнения
        protected int NoiseLevel;//уровень шума
        protected int EnergySource;     // источник энергии 1- внутренний, 0-
внешний
        protected int Speed;//скорость
        protected int X, Y;//координаты
        protected TStreet Street;//улица

        protected Random rnd = new Random();

        public TVehicle()//конструктор класса TVehicle
        {
            EnergySource = rnd.Next(0, 2);//источник энергии - случайным
образом выбирается 0 или 1
            MaxSpeed = rnd.Next(1,60); //максимальная скорость  - случайным
образом выбирается от 1 до 60
            NoiseLevel = rnd.Next(1, 20); //уровень шума - случайным образом
выбирается число от 1 до 20
            if (EnergySource == 0) //если источник энергии внешний,то уровень
загрязнения равен 0
                PolutionLevel = 0;
            else PolutionLevel = rnd.Next(1, 10);//иначе уровень загрязнения
выбирается случайным образом от 1 до 10


            Speed = rnd.Next(1,MaxSpeed); // скорость генерируется случайным
образом от 1 до максимальной скорости
        }


        public virtual int Type () //виртуальный метод Type в базовом классе
        {
            return 0;
        }

        public bool CanObgon()// возможность обгона
        {
            return true;
        }

    }
}
```

Класс TCivilTransport

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public class TCivilTransport: TVehicle // пассажирские транспортные
средства
    {
        private int MaxPeopleCount; // кол-во пассажирских мест
        private int PeopleCount;    // кол-во пассажиров

        public TCivilTransport(int aMaxPeopleCount)
        {
            MaxPeopleCount = aMaxPeopleCount;
            PeopleCount = rnd.Next(1, MaxPeopleCount);

        }

        public string GetPeopleCount ()
        {
            string res;
            res = PeopleCount.ToString("F0");
            return res;
        }

        public override int Type()/*виртуальный метод, переопределенный в
наследуемом классе*/
        {
            return 1; // пассажирские т/с
        }
    }
}
```

Класс TLorryTransport

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public class TLorryTransport: TVehicle // грузовые транспортные средства
    {
        private int MaxGruzMassa;   // грузоподъемность
        private int GruzMassa;      // масса груза

        public TLorryTransport(int aMaxGruzMassa)
        {

            MaxGruzMassa = aMaxGruzMassa;
            GruzMassa = rnd.Next(1, MaxGruzMassa);
        }

        public string GetGruzMassa ()
        {
            string res;
            res = GruzMassa.ToString("F0");
            return res;
```

```
        }

        public override int Type()/*виртуальный метод, переопределенный в
наследуемом классе*/
        {
            return 2; // грузовое транспортное средство
        }
    }
}
```

Также в программу были добавлены новые классы для дальнейшей работы с программой.

Класс Init

```
//класс для инициализации массивов транспортных средств и улиц

using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public class Init
    {
        static public TCivilTransport[] CivilTransportArray = new
TCivilTransport[20]; //массив пассажирских средств
        static public TLorryTransport[] LorryTransportArray = new
TLorryTransport[20]; //массив грузовых средств
        static public TStreet[] StreetArray = new TStreet[4]; //массив улиц
        static public TStreet[] ProspektArray = new TStreet[3]; //массив
проспектов


        public List<TVehicle> VehicleMassiv = new List<TVehicle>();
//Представляет строго типизированный список объектов, доступных по индексу.


        int i;
        public Random rnd = new Random();
        public void FillTransportArray()//Заполнение массивов пассажирского и
грузового транспорта
        {
            for (i = 0; i <= 19; i++)
            {
                CivilTransportArray[i] = new TCivilTransport(rnd.Next(1,
40));
                LorryTransportArray[i] = new TLorryTransport(rnd.Next(1,
20));
            }
        }

        public void FillStreetArray()//Заполнение массивов улиц и проспектов
        {
            for (i = 0; i <= 3; i++)
                StreetArray[i] = new TStreet("Улица-"+(i+1), (i+1)*100, 0);
            for (i = 0; i <= 2; i++)
                ProspektArray[i] = new TStreet("Проспект-" + (i + 1), 0, (i +
1) * 100);
        }
```

```
    }
}
```

Класс CityMovement

```
//класс для реализации движения транспортных средств

using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public class CityMovement: Init
    {
        int i;
        private int VehicleCount;

        public int GetVehicleCount ()
        {
            VehicleCount = 0;
            VehicleCount = VehicleCount + CivilTransportArray.GetLength(0) +
LorryTransportArray.GetLength(0);

            return VehicleCount;

        }

        public int GetVehicleCount(int aType) // кол-во транспорта
определенного типа в городе
        {
            int res = 0;
            //TVehicle vvv = new TVehicle();

            foreach (TVehicle vvv in VehicleMassiv)
            {
                if (vvv.Type() == aType)
                    res = res + 1;
            }
            return res;
        }

        public void test()
        {
            VehicleMassiv.Add(new TCivilTransport(rnd.Next(1, 40)));
            VehicleMassiv.Add(new TCivilTransport(rnd.Next(1, 40)));
            VehicleMassiv.Add(new TLorryTransport(rnd.Next(1, 20)));
        }
        public CityMovement()
        {

        }
    }
}
```

Диаграмма классов стала выглядеть следующим образом:

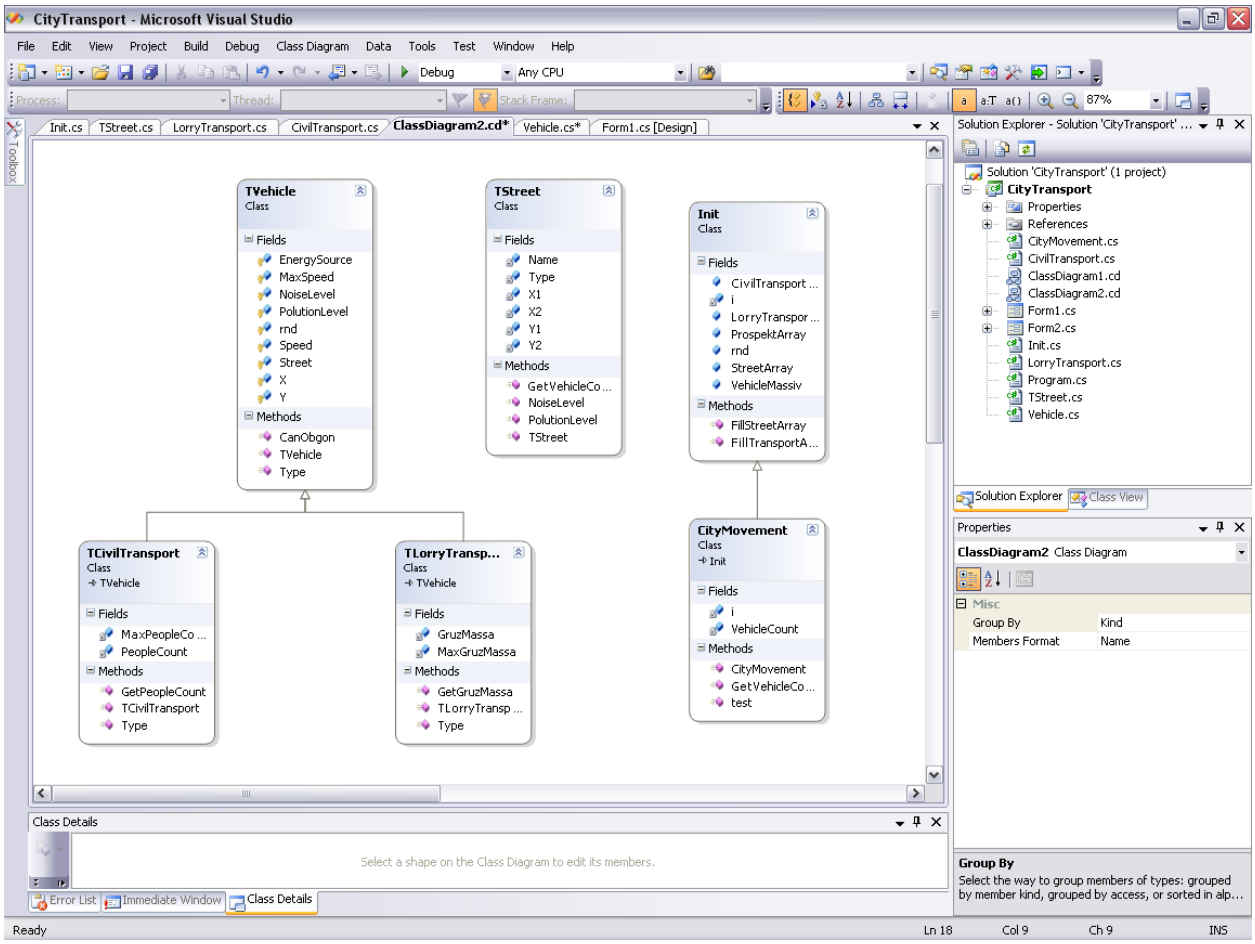

Рисунок 21 – Итоговый вариант диаграммы

Создадим интерфейс IObjectInfo для получения информации по состоянию движущегося объекта (т/с).

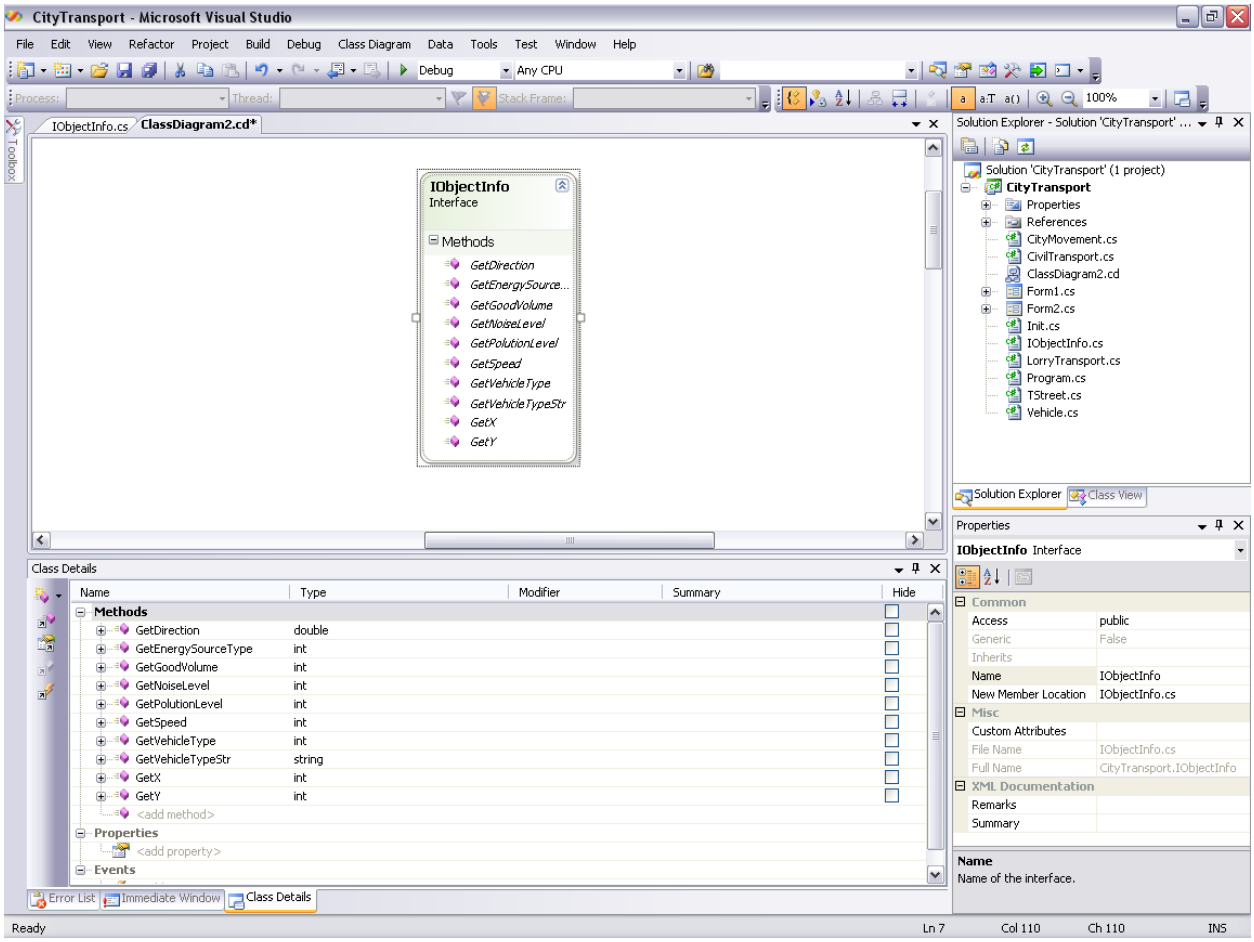

Рисунок 22 – Диаграмма и методы интерфейса IobjectInfo

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public interface IObjectInfo // интерфейс для получения информации по
состоянию движущегося объекта (т/с)

    {
        int GetX(); // координаты
        int GetY();
        int GetSpeed(); // текущая скорость
        double GetDirection();// направление движения
        int GetNoiseLevel();// уровень шума
        int GetPolutionLevel();  // уровень загрязнения
        int GetVehicleType();   // тип т/с
        string GetVehicleTypeStr();// тип т/с в строковом виде
        int GetEnergySourceType();  // Тип источника энергии
        int GetGoodVolume();        // полезная нагрузка (пассажиры или груз)
    }
                                                    }
```

Реализуем интерфейс для классов TCivilTransport и TLorryTransport.

Класс TCivilTransport

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public class TCivilTransport: TVehicle, IObjectInfo // пассажирские
транспортные средства
    {
        private int MaxPeopleCount; // кол-во пассажирских мест
        private int PeopleCount;    // кол-во пассажиров

        public TCivilTransport(int aMaxPeopleCount)
        {
            MaxPeopleCount = aMaxPeopleCount;
            PeopleCount = rnd.Next(1, MaxPeopleCount);

        }


        public override string GetInfo()
        {
            string res = base.GetInfo();
            res = res + " пассажирский";
            return res;
        }

        // реализация интерфейса IObjectInfo
        public int GetX() { return X; }
        public int GetY() { return Y; }
        public int GetEnergySourceType() { return EnergySource;}
        public int GetNoiseLevel() { return NoiseLevel; }
        public int GetPolutionLevel() { return PolutionLevel; }
        public int GetVehicleType() { return 1; }
        public string GetVehicleTypeStr() { return "пассажирский"; }
        public int GetGoodVolume() { return PeopleCount; }
        public int GetSpeed() { return Speed; }
        public double GetDirection() { return Direction;}
    }
}
```

Класс TLorryTransport

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public class TLorryTransport: TVehicle, IObjectInfo // грузовые
транспортные средства
    {
        private int MaxGruzMassa;   // грузоподъемность
        private int GruzMassa;      // масса груза
```

```
        public TLorryTransport(int aMaxGruzMassa)
        {

            MaxGruzMassa = aMaxGruzMassa;
            GruzMassa = rnd.Next(1, MaxGruzMassa);
        }


        public override string GetInfo()
        {
            string res = base.GetInfo();
            res = res + " грузовой";
            return res;
        }
        // реализация интерфейса IObjectInfo
        public int GetX() {return X;}
        public int GetY() { return Y; }
        public int GetEnergySourceType() { return EnergySource; }
        public int GetNoiseLevel() { return NoiseLevel; }
        public int GetVehicleType() { return 2; }
        public string GetVehicleTypeStr() { return "грузовой"; }
        public int GetGoodVolume() { return GruzMassa; }
        public double GetDirection() { return Direction; }
        public int GetSpeed() { return Speed; }
        public int GetPolutionLevel() { return PolutionLevel; }
    }
}
```

Диаграмма классов после преобразований выглядит следующим образом:

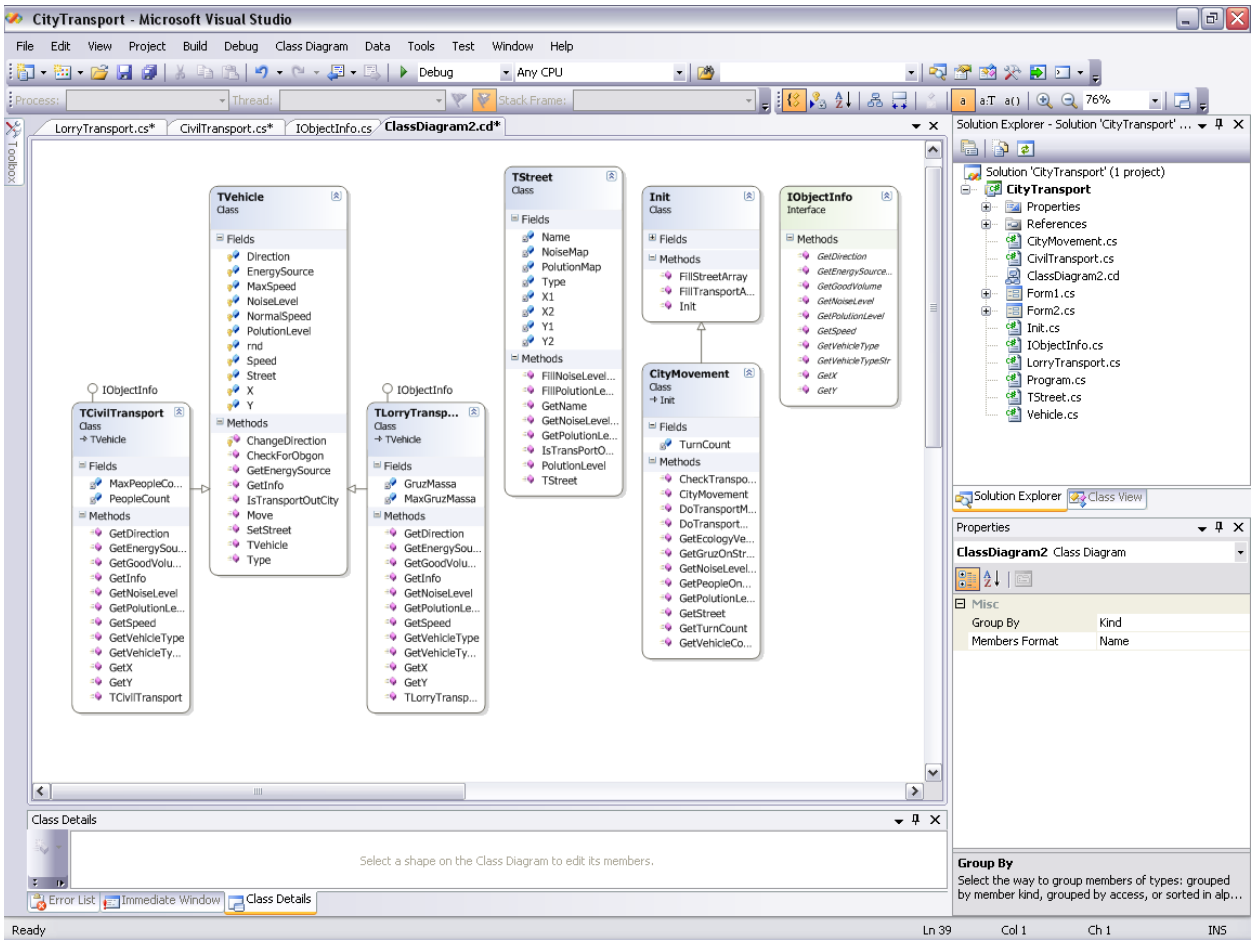

Рисунок 23 – Диаграмма классов

Добавим методы в классы для отрисовки объектов.

В класс TStreet добавим метод Draw для отрисовки улиц. А также новые методы:

FillNoiseLevelMap // определение характера распространения шума по всей длине улицы

FillPolutionLevelMap // определение характера распространения загрязнения по всей длине улицы

GetName //название улицы

GetNoiseLevelView//показывает уровень шума на улице

GetPolutionLevelView//показывает уровень загрязниния на улице

IsTransPortOnStreet//нахождение т/с на улице

Класс Tstreet

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;


namespace CityTransport
{


    public class TStreet
    {
        private int Type; // 1- вертикальная (улица), 2- горизонтальная
(проспект)
        private int X1, X2, Y1, Y2;//координаты улицы
        private string Name;//название улицы
        private int[] NoiseMap;//карта уровня шума
        private int[] PolutionMap;//карта уровня загрязнения

        public TStreet (int aX, int aY)//конструктор класса
        {

            X1 = aX; Y1 = aY;
            if (aX == 0) // проспект
            {
                Type = 2;
                X2 = 500; Y2 = Y1;
                Name = "Проспект-" + Y1;
                NoiseMap = new int[500];
                PolutionMap = new int[500];
            }
            else // улица
            {
                Type = 1;
                X2 = X1; Y2 = 400;
                Name = "Улица-" + X1;
```

```
                NoiseMap = new int[400];
                PolutionMap = new int[400];
            }
        }

        public void FillNoiseLevelMap(List<TVehicle> aList) // определение
характера распространения шума по всей длине улицы
        {
            double NoiseLevel = 0; // уровень шума в точке

            int L, N; // текущая позиция и максимальная длина улицы
            if (Type == 2) // горизонтальная улица
            { L = X1; N = X2; }
            else
            { L = Y1; N = Y2; }

            while (L < N)
            {
                foreach (IObjectInfo vvv in aList)
                    if (IsTransPortOnStreet(vvv))
                        if (Type == 2)
                            NoiseLevel = NoiseLevel + vvv.GetNoiseLevel() /
Math.Max(1, Math.Abs(L - vvv.GetX()));
                        else
                            NoiseLevel = NoiseLevel + vvv.GetNoiseLevel() /
Math.Max(1, Math.Abs(L - vvv.GetY()));

                if (NoiseLevel < 1)
                    NoiseMap[L]=0;
                else
                    NoiseMap[L] = Convert.ToInt32 ( NoiseLevel);

                NoiseLevel = 0;
                L++;
            }
        }

        public string GetNoiseLevelView()
        {
            string result="";
            for (int i = 0; i < NoiseMap.Length; i++)
            {
                if (NoiseMap[i] == 0) result = result + ".";
                else result = result + "[" + NoiseMap[i].ToString() + "]";
            }
            return result;
        }
        public void FillPolutionLevelMap (List<TVehicle> aList) //
определение характера распространения загрязнений по всей длине улицы
        {
            double PL = 0; // уровень загрязнения в точке

            int L, N; // текущая позиция и максимальная длина улицы
            if (Type == 2) // горизонтальная улица
            { L = X1; N = X2; }
            else
            { L = Y1; N = Y2; }

            while (L < N)
            {
                foreach (IObjectInfo vvv in aList)
```

```
                    if ((vvv.GetPolutionLevel() > 0) &
(IsTransPortOnStreet(vvv)))
                        if (Type == 2)
                        {
                            if (
                                    ((Math.Cos(vvv.GetDirection()) == 1) &
((vvv.GetX() - L) <= vvv.GetSpeed()) & (vvv.GetX() >= L)) ||
                                    ((Math.Cos(vvv.GetDirection()) == -1) &
((L - vvv.GetX()) <= vvv.GetSpeed()) & (vvv.GetX() <= L))
                                )
                                PL = PL + vvv.GetPolutionLevel();
                        }
                        else
                            if (
                                ((Math.Sin(vvv.GetDirection()) == 1) &
(((vvv.GetY() - L)) <= vvv.GetSpeed()) & (vvv.GetY() >= L)) ||
                                ((Math.Sin(vvv.GetDirection()) == -1) & (((L
- vvv.GetY())) <= vvv.GetSpeed()) & (vvv.GetY() <= L))
                                )
                                PL = PL + vvv.GetPolutionLevel();

                if (PL < 1)
                    PolutionMap[L]=0;
                else
                    PolutionMap[L]= Convert.ToInt32 (PL);

                PL = 0;
                L++;
            }
        }
        public string GetPolutionLevelView()
        {
            string result = "";
            for (int i = 0; i < PolutionMap.Length; i++)
            {
                if (PolutionMap[i] == 0) result = result + ".";
                else result = result + "[" + PolutionMap[i].ToString() + "]";
            }
            return result;

        }

        public double PolutionLevel(int L)
        {
            return 0;
        }
        public bool IsTransPortOnStreet(IObjectInfo aTransport)//проверка
нахождения т/с на улице
        {
            if ((aTransport.GetX()==X1)||(aTransport.GetY()==Y1))
                return true;
            else return false;
        }
        public string GetName()
        {
            return Name;
        }

        public void Draw(Graphics G, float sfx, float sfy, bool DrawNoise,
bool DrawPolution)//отрисовка улиц с уровнем шума и загрязнением
        {
```

```
            G.DrawLine(new Pen(Color.Blue), X1 * sfx, Y1 * sfy, X2 * sfx, Y2
* sfy);
            Pen NoisePen = new Pen(Color.LemonChiffon);
            Pen PolutionPen = new Pen(Color.LimeGreen);

            if (DrawNoise)
            {
                if (Type == 1)
                    for (int i = 0; i < 400; i++)
                        G.DrawLine(NoisePen, (X1 - NoiseMap[i] * 3) * sfx, i
* sfy, X1 * sfx, i * sfy);
                else
                    for (int i = 0; i < 500; i++)
                        G.DrawLine(NoisePen, i * sfx, (Y1 - NoiseMap[i] * 3)
* sfy, i * sfx, Y1 * sfy);
            }
            if (DrawPolution)
            {
                if (Type == 1)
                    for (int i = 0; i < 400; i++)
                        G.DrawLine(PolutionPen, (X1 + PolutionMap[i] * 3) *
sfx, i * sfy, X1 * sfx, i * sfy);
                else
                    for (int i = 0; i < 500; i++)
                        G.DrawLine(PolutionPen, i * sfx, (Y1 + PolutionMap[i]
* 3) * sfy, i * sfx, Y1 * sfy);
            }



        }

    }
}
```

В класс TVehicle добавим метод Draw для отрисовки улиц. А также новые методы:

ChangeDirection `// изменение направления движения`

CheckForObgon `// проверка на возможность обгона между двумя т/с`

GetEnergySource `//источник энергии`

GetInfo `// информация по состоянию т/с`

IsTransportOutCity `//проверка на выезд т/с за пределы города`

Move `// перемещение транспортного средства`

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;

namespace CityTransport
{
    public class TVehicle
```

```
    {
        protected int MaxSpeed;// максимальная скорость
        protected int PolutionLevel;//уровень загрязнения
        protected int NoiseLevel;//уровень шума
        protected int EnergySource;     // источник энергии 1- внутренний, 0-
внешний
        protected int Speed, NormalSpeed; // текущая скорость и скорость
движения без помех
        protected double Direction;     // направление движения (в радианах,
0-восток, pi -запад, pi/2 - север)
        protected int X, Y; //координаты
        protected TStreet Street;//улица

        protected Random rnd = new Random();

        public TVehicle()//конструктор класса
        {

            MaxSpeed = rnd.Next(20,40); //максимальная скорость выбирается
случайным образом от 20 до 40
            NoiseLevel = rnd.Next(1, 20);//уровень шума выбирается случайным
образом от 1 до 20

            double rrr = rnd.NextDouble();

            if (rrr < 0.5)//проверка на источник энергии
            {
                EnergySource = 0;//источник энергии внешний
                PolutionLevel = 0;//уровень шума равен 0
            }
            else
            {
                EnergySource = 1;//источник энергии внутренний
                PolutionLevel = rnd.Next(1, 10);//уровень шума выбирается
случайным образом от 1 до 10
            }

            Speed = rnd.Next(20,MaxSpeed);//текущая скорость выбирается
случайным образом от 20 до максимальной скорости
            NormalSpeed = Speed;//скорость движения без помех равна текущей
скорости

            X=0;
            Y = rnd.Next(1, 4) * 100; // въезд в город слева по одному из
проспектов
        }

        public virtual int Type () //тип т/с
        {
            return 0;
        }
        public int GetEnergySource ()//источник энергии
        {
            return EnergySource;
        }

        public void CheckForObgon(TVehicle aTransport2) // проверка на
возможность обгона между двумя т/с
        {
            if ((Street==aTransport2.Street) & (EnergySource==0))
            {
                int X1 = X+Convert.ToInt32(Speed * Math.Cos(Direction));
```

```
                int Y1 = Y+Convert.ToInt32(Speed * Math.Sin(Direction));
                int X2 = aTransport2.X+ Convert.ToInt32(aTransport2.Speed *
Math.Cos(aTransport2.Direction));
                int Y2 = aTransport2.Y+Convert.ToInt32(aTransport2.Speed *
Math.Sin(aTransport2.Direction));

                if (((X<aTransport2.X)&(X1>=X2))||
((X>aTransport2.X)&(X1<=X2))||
                   ((Y<aTransport2.Y)&(Y1>=Y2))||
((Y>aTransport2.Y)&(Y1<=Y2)))
                {
                    Speed = aTransport2.Speed;
                }
                else
                    Speed = NormalSpeed;
            }
            else
                Speed = NormalSpeed;
        }

        public void Move() // перемещение транспортного средства
        {
            int dX = Convert.ToInt32(Speed * Math.Cos(Direction));
            int dY = Convert.ToInt32(Speed * Math.Sin(Direction));

            int r;
            if (dX< 0)
                Math.DivRem((1000-X) -dX, 100, out r); // разворачиваем
систему координат на 180
            else
                Math.DivRem (X+ dX,100, out r);

            if ((r != 0) & (r < Speed)) // возможен поворот на улицу
                ChangeDirection(r);
            else
            {
                if (dY < 0)
                    Math.DivRem((1000 - Y) - dY, 100, out r);// разворачиваем
систему координат на 180
                else
                    Math.DivRem(Y + dY, 100, out r);
                if (((r != 0) & (r < Speed))) // возможен поворот на проспект
                    ChangeDirection(r);
                else // поворот не возможен
                {
                        { X = X + dX; Y = Y + dY; }
                }
            }


        }
        protected void ChangeDirection ( // изменение направления движения
            int aDelta) // aDelta - остаток движения после поворота
        {
            /*switch (Math.Cos(Direction)) {
                case 1:
            }*/

            X = X + Convert.ToInt32((Speed-aDelta) * Math.Cos(Direction));
            Y = Y + Convert.ToInt32((Speed-aDelta) * Math.Sin(Direction));

            int tmp = rnd.Next(-1, 2);
```

```
            Direction = Direction + Math.PI / 2 * Math.Sign(tmp);
            X = X + Convert.ToInt32(aDelta * Math.Cos(Direction));
            Y = Y + Convert.ToInt32(aDelta * Math.Sin(Direction));

        }

        public bool IsTransportOutCity()//проверка на выезд т/с за пределы
города
        {
            if ((X >= 500) || (X <= 0))
                return true;
            else
                if ((Y >= 400) || (Y <= 0))
                    return true;
                else
                    return false;

        }
        public virtual string GetInfo() // информация по состоянию т/с
        {
            string res = "X=" + X + " Y=" + Y;
            res = res + " Скорость=" + Speed;
            if (EnergySource==1)
                res = res + " Источник внутренний";
            else
                res = res + " Источник внешний";

            res = res + " Шум=" + NoiseLevel;
            res = res + " Загрязнение=" + PolutionLevel;

            return res;
        }
        public void SetStreet(TStreet aStreet)
        {
            Street = aStreet;
        }
        public virtual void Draw(Graphics G, float sfx, float sfy)
{ }//отрисовка графики т/с


    }

}
```

Также переопределим классы TCivilTransport и TlorryTransport.

Класс TCivilTransport

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;

namespace CityTransport
{
    public class TCivilTransport: TVehicle, IObjectInfo // пассажирские
транспортные средства
```

```
    {
        private int MaxPeopleCount; // кол-во пассажирских мест
        private int PeopleCount;    // кол-во пассажиров

        public TCivilTransport(int aMaxPeopleCount)//конструктор класса
        {
            MaxPeopleCount = aMaxPeopleCount;
            PeopleCount = rnd.Next(1, MaxPeopleCount);

        }

        public override string GetInfo()//возвращает информацию о т/с
        {
            string res = base.GetInfo();
            res = res + " пассажирский";
            return res;
        }
        public override void Draw(System.Drawing.Graphics G, float sfx, float
sfy)//отрисовка пассажирского т/с
        {

            Color clr;
            if (EnergySource == 0) //если источник энергии внешний, то т/с
зеленое
            {clr = Color.Green;}
            else
                {clr = Color.Red;}//иначе красное

            G.DrawEllipse(new Pen(clr), (X - 5) * sfx, (Y - 5) * sfy, 10 *
sfx, 10*sfy);//рисуем эллипс с помощью пера по координатам
        }

        // реализация интерфейса IObjectInfo
        public int GetX() { return X; }
        public int GetY() { return Y; }
        public int GetEnergySourceType() { return EnergySource;}
        public int GetNoiseLevel() { return NoiseLevel; }
        public int GetPolutionLevel() { return PolutionLevel; }
        public int GetVehicleType() { return 1; }
        public string GetVehicleTypeStr() { return "пассажирский"; }
        public int GetGoodVolume() { return PeopleCount; }
        public int GetSpeed() { return Speed; }
        public double GetDirection() { return Direction;}
    }
      }
```

Класс TLorryTransport

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;

namespace CityTransport
{
    public class TLorryTransport: TVehicle, IObjectInfo // грузовые
транспортные средства
    {
        private int MaxGruzMassa;   // грузоподъемность
        private int GruzMassa;      // масса груза
```

```
        public TLorryTransport(int aMaxGruzMassa)//конструктор класса
        {

            MaxGruzMassa = aMaxGruzMassa;
            GruzMassa = rnd.Next(1, MaxGruzMassa);
        }

        public override void  Draw(System.Drawing.Graphics G, float sfx,
float sfy)//отрисовка грузовых т/с
        {
            Color clr;
            if (EnergySource == 0)
                { clr = Color.Green; }
            else
                { clr = Color.Red; }

            G.DrawRectangle (new Pen(clr), (X - 5) * sfx, (Y - 5) * sfy, 10 *
sfx, 10 * sfy);

        }

        public override string GetInfo()//информация о т/с
        {
            string res = base.GetInfo();
            res = res + " грузовой";
            return res;
        }
        // реализация интерфейса IObjectInfo
        public int GetX() {return X;}
        public int GetY() { return Y; }
        public int GetEnergySourceType() { return EnergySource; }
        public int GetNoiseLevel() { return NoiseLevel; }
        public int GetVehicleType() { return 2; }
        public string GetVehicleTypeStr() { return "грузовой"; }
        public int GetGoodVolume() { return GruzMassa; }
        public double GetDirection() { return Direction; }
        public int GetSpeed() { return Speed; }
        public int GetPolutionLevel() { return PolutionLevel; }
    }
}
```

Изменим классы Init и CityMovement, добавив новые методы.

Класс Init

```
using System;
using System.Collections.Generic;
using System.Text;

namespace CityTransport
{
    public class Init
    {

        public List<TVehicle> VehicleList = new List<TVehicle>();
```

```
        static public TStreet[] StreetArray = new TStreet[7];


        int i;
        public Random rnd = new Random();
        public void FillTransportArray()
        {
            for (i = 0; i <= 9; i++)
            {
                VehicleList.Add (new TCivilTransport(rnd.Next(1, 40)));
                VehicleList.Add (new TLorryTransport(rnd.Next(1, 20)));
            }
        }

        public void FillStreetArray()
        {
            for (i = 0; i <= 3; i++)
            {
                StreetArray[i] = new TStreet((i + 1) * 100, 0);
            }
            for (i = 4; i <= 6; i++)
                StreetArray[i] = new TStreet(0, (i - 3) * 100);
        }
        public Init()
        {
            FillStreetArray();
        }
    }
}
```

Класс CityMovement

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;

namespace CityTransport
{
    public class CityMovement: Init
    {
        private int TurnCount; // кол-во прошедших еденицы времени

        public int GetVehicleCount ()
        {
            int VehicleCount;
            VehicleCount = 0;
            VehicleCount = VehicleCount + VehicleList.Count;

            return VehicleCount;

        }
        public int GetVehicleCount(int aType) // кол-во транспорта
определенного типа в городе
        {
            int res = 0;
            foreach (IObjectInfo vvv in VehicleList)
            {
                if (vvv.GetVehicleType() == aType)
                    res = res + 1;
            }
```

```
            return res;
        }
        public int GetEcologyVehicleCount()   // кол-во транспорта, не
загрязняющего окружающую среду
        {
            int res = 0;
            foreach (TVehicle vvv in VehicleList)
            {
                if (vvv.GetEnergySource() == 0)
                    res = res + 1;
            }
            return res;
        }
        public CityMovement() { }

        public int GetTurnCount() { return TurnCount; }//возвращает
количество прощедшего времени
        public void DoTransportMove()    // движение транспорта за еденичный
промежуток времени
        {
            foreach (TVehicle vvv in VehicleList)
            {
                foreach (TVehicle vvv_2 in VehicleList)
                    vvv.CheckForObgon(vvv_2);

                vvv.Move();
                foreach (TStreet sss in StreetArray)
                    if (sss.IsTransPortOnStreet(vvv as IObjectInfo))
                    {
                        vvv.SetStreet(sss);
                    }
            }

            foreach (TStreet sss in StreetArray)
            {
                sss.FillNoiseLevelMap(VehicleList);
                sss.FillPolutionLevelMap(VehicleList);
            }

            TurnCount++;
        }
        public void CheckTransportOut()  // обработка транспорта выехавшего
за пределы города
        {
            for (int i=VehicleList.Count-1; i>=0; i--)
            {

                if (VehicleList[i].IsTransportOutCity())
                {
                    VehicleList.RemoveAt(i);
                }
            }
        }
        public void DoTransportWelcome() // въезд транспорта в город
        {
            Random rnd = new Random();
            if (rnd.NextDouble()<0.5) // вероятность появления нового
транспорта
                if (rnd.Next(1,3)==1)
                    VehicleList.Add (new TCivilTransport(rnd.Next(1,40)));
                else
                    VehicleList.Add (new TLorryTransport(rnd.Next(1,20)));
```

```
        }
        public int GetPeopleOnStreet(TStreet aStreet) // кол-во пассажиров на
заданной улице
        {
            int res=0;
            foreach (IObjectInfo vvv in VehicleList)
            {
                if ((aStreet.IsTransPortOnStreet(vvv)) & (vvv.GetVehicleType
() == 1))
                {
                    res = res + vvv.GetGoodVolume();
                }
            }
            return res;
        }
        public int GetGruzOnStreet(TStreet aStreet) // кол-во груза на
заданной улице
        {
            int res = 0;
            foreach (IObjectInfo vvv in VehicleList)
            {
                if ((aStreet.IsTransPortOnStreet(vvv)) &
(vvv.GetVehicleType() == 2))
                {
                    res = res + ((TLorryTransport)vvv).GetGoodVolume();
                }
            }
            return res;
        }
        public TStreet GetStreet(int aIndex) { return StreetArray[aIndex]; }
        public string GetNoiseLevelView(TStreet aStreet) // характер
распространения шума по улице
        {
            return aStreet.GetNoiseLevelView();
        }
        public string GetPolutionLevelView(TStreet aStreet) // характер
распространения загрязнений по улице
        {
            return aStreet.GetPolutionLevelView();
        }

        public void DrawItems (IntPtr h, bool DrawNoise, bool DrawPolution)
        {
            Graphics gr = Graphics.FromHwnd(h);//графическая переменная gr
создается методом Graphics.FromHwnd(h) из указанного дискриптора окна h
            gr.Clear(Color.FromKnownColor
(System.Drawing.KnownColor.Control)); //метод который создает структуру колор
из указанного определенного цвета
            float sfx = gr.VisibleClipBounds.Width / 500;//координаты
меняются при изменение размера окна по х
            float sfy = gr.VisibleClipBounds.Height / 400;

            foreach (TStreet sss in StreetArray) { sss.Draw(gr, sfx, sfy,
DrawNoise, DrawPolution); }
            foreach (TVehicle vvv in VehicleList) { vvv.Draw(gr, sfx, sfy); }

            gr.Dispose();//освобождает все ресурсы,используемыми данным
объектом
        }

    }
}
```

Диаграмма классов после преобразований выглядит следующим образом:

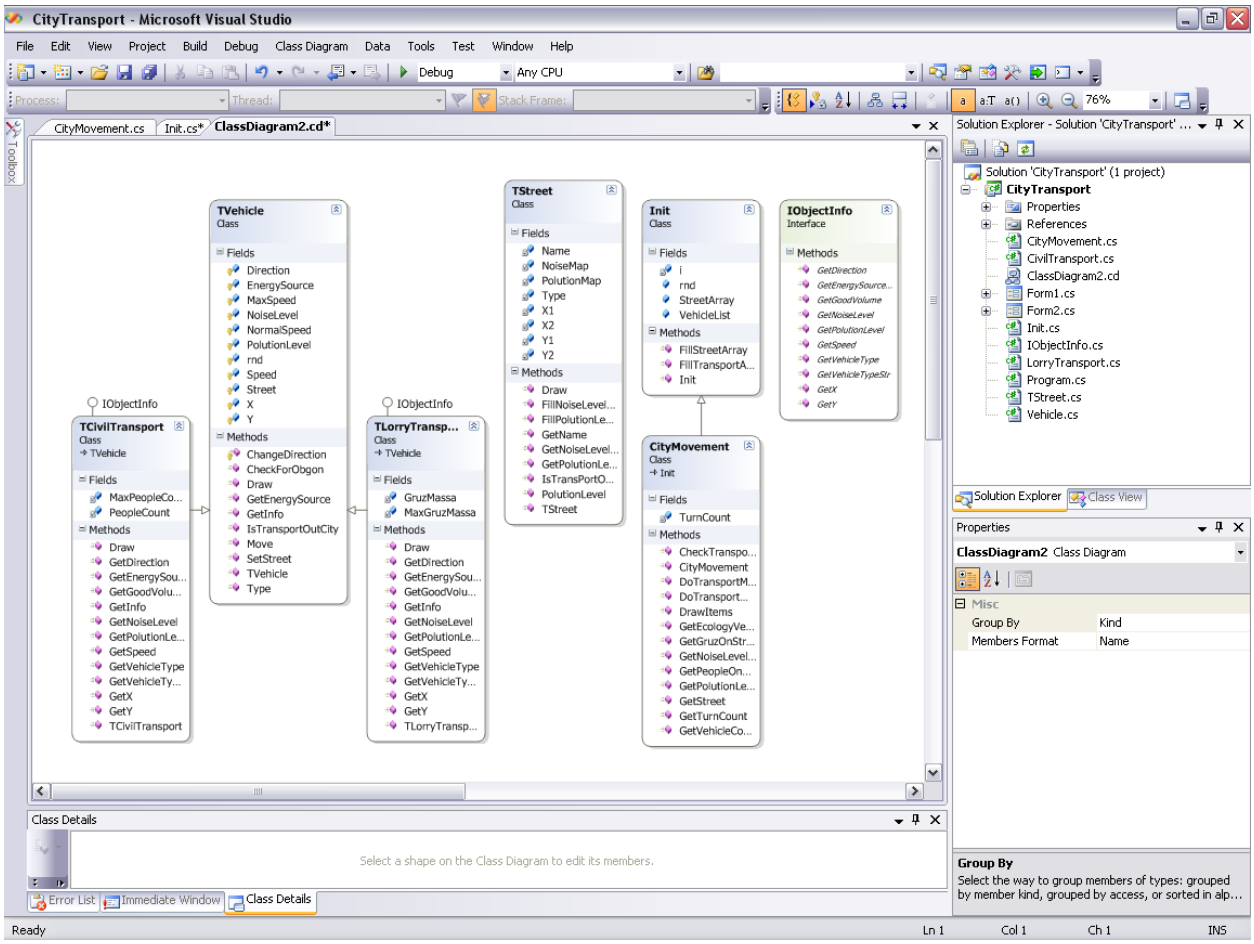

Рисунок 24 – Диаграмма классов

Варианты заданий:

1. Справочник меломана

Разработайте программу для ведения справочника произведений, полностью помещающегося в оперативной памяти. Имеются люди – фамилия, имя, дата рождения, пол. Каждый человек является либо певцом, либо поэтом, либо композитором. Певцы могут обладать голосом (мужчины - бас, баритон, тенор, женщины – сопрано, контральто). Сведения о произведении – наименование, жанр, список поэтов, список композиторов, дата создания. Жанр – джаз, поп, рок (популярная музыка), классическая музыка. Если жанр – классическая музыка, то в произведении указываются требования к голосам исполнителей. Имеется также список записей произведений. Для каждой записи указываются – произведение, список исполнителей, дата исполнения. Программа должна загружать справочник с жесткого диска, править все его данные, сохранять на диск. В процессе работы пользователь программы должен иметь возможность просмотра, как минимум, следующих параметров:

- Список певцов с возможностью сортировки по фамилии, дате рождения, полу, голосу, количеству произведений, в исполнении которых он участвовал.
- Список композиторов с возможностью сортировки по фамилии, дате рождения, полу, жанру произведений, наименованию произведений.
- Список записей заданного жанра с сортировкой по наименованию, дате создания, последней дате записи. Должна быть реализована возможность отбора записей, которые исполнял заданный певец или в списках авторов имеется заданные поэт и/или композитор.
- Список произведений популярной или классической музыки с возможностью сортировки по дате создания или жанру.

2. Склад

Разработать программу для имитации движения товаров, хранящихся на складе. Каждый вид товара характеризуется наименованием и базовую единицу измерения. На складе могут быть товары только следующих типов – краски, разбавители, гвозди. Краски характеризуются цветом. Кроме того, краски делятся на масляные и нитроэмали. Нитроэмали характеризуются тем, что имеют список разбавителей, которыми их можно разбавлять. Гвозди определяются их размером. Имеется также список единиц измерения и всевозможные коэффициенты перевода из одной единицы в другую. Партия товара характеризуются датами запроса, поступления или выдачи, товаром, единицей измерения и количеством. Поступление и выдача партий осуществляется случайным образом. Поступление наступает на следующий день после запроса, выдача осуществляется немедленно при наличии товара. При отсутствии товара выдача задерживается до поступления товара или истечении срока ожидания, после чего заявка на выдачу отменяется. Программа должна загружать файл с состоянием склада с жесткого диска и в паузе или по окончании процесса имитации сохранять файл на диск.
Процесс имитации может быть остановлен пользователем программы для просмотра параметров:

- Список всех товаров на складе с указанием их количества и единиц измерения. У всех товаров, единица измерения которых приводится к базовой, количество должно быть выведено в базовых единицах измерения. Список может быть отсортирован по наименованию товаров, по типам товаров
- Количество краски заданного цвета, в том числе с разбивкой на масляные и нитроэмали.
- Список цветов красок, имеющихся на складе, в том числе с разбивкой на масляные и нитроэмали.

- Список нитроэмалей, для которых на складе есть нужные разбавители.
- Список разбавителей, которые не используются в красках, находящихся на складе.
- Количество гвоздей нужного размера.

3. Городской транспорт

Разработать программу, имитирующую движение городского транспорта. Время процесса дискретно. В целях упрощения предполагается, что улицы города пересекаются под прямыми углами. Транспортные средства различных моделей двигаются с различными скоростями, меняя направление движения на перекрестках случайным образом. Любое транспортное средство имеет следующие характеристики – максимальную скорость, уровень загрязнения (г/м, считается, что за один шаг по времени загрязнение полностью разлагается), уровень шума (в децибелах), обратно пропорциональный расстоянию до транспортного средства (считается, что шум распространяется только вдоль проезжей части). Транспортные средства могут иметь внутренний или внешний (например, троллейбусы) источник энергии. Все транспортные средства делятся на пассажирские и грузовые. Пассажирские средства имеют максимальное количество пассажирских мест, грузовые – максимальный вес перевозимого груза. Транспортное средство может производить обгон, если оно имеет внутренний источник энергии, иначе оно должно двигаться со скоростью впереди идущей машины. Машины могут случайным образом покидать город, а также въезжать в город. Количество пассажиров, вес груза и начальная скорость задается случайно, так чтобы они не превышали свои максимальные значения. На каждом шаге по времени пользователь может вывести следующую информацию:

- Состояние любого транспортного средства;
- уровень шума в каждой точке проезжей части;
- уровень загрязнения в каждой точке проезжей части;
- количество пассажиров на любой улице;
- вес перевозимого груза на любой улице;
- количество пассажирских транспортных средств;
- количество грузовых транспортных средств;
- количество транспортных средств, не загрязняющих окружающую среду.

4. Отдел кадров в университете

Разработать программу, реализующую справочник сотрудника отдела кадров. Университет состоит из факультетов, факультеты имеют в своем составе кафедры и студенческие группы. На каждой кафедра есть заведующий кафедрой. Некоторые кафедры являются профилирующими. Каждая группа имеет студента - старосту и профилирующую кафедру. Будем полагать, что в университете все люди являются преподавателями и/или студентами. Сведения о каждом человеке должны содержать – фамилия, имя, отчество, пол, паспортные данные, место проживания, государство и субъект РФ (для иностранных граждан «За пределами Российской Федерации»). Для студентов дополнительно должна быть информация о родителях, направлении подготовки и группе. Для преподавателей дополнительно должна быть информация о кафедре, должности и ученого звания. Предусмотреть возможную ситуацию, когда один и тот же человек может быть одновременно студентом, родителем и преподавателем. Один родитель может иметь несколько детей-студентов. Программа должна загружать справочник с жесткого диска, править все его данные, сохранять на диск. В процессе работы пользователь программы должен иметь возможность просмотра, как минимум, следующих параметров:

- Список всех студентов с возможностью сортировки по ФИО, факультету, направлению подготовки, группе, профилирующей кафедре.

- Список студентов, не имеющих родителей с возможностью сортировки по ФИО, факультету, группе, профилирующей кафедре.
- Список преподавателей с возможностью сортировки по ФИО, факультету, кафедре.
- Список всех заведующих кафедрами.
- Список всех групп без старост и кафедр без заведующих.
- Поиск у заданного родителя всех его детей – студентов.
- Список всех преподавателей, имеющих детей – студентов.

5. Очередь печати

Разработать программную систему, реализующую имитацию очереди печати на принтерах.

Каждый принтер имеет следующие параметры: название, максимальный ширина и высота печатного листа, максимальное разрешение, скорость печати (количество листов в единицу времени (такт)), емкость лотка подачи листов. Дополнительно принтеры делятся на принтеры, поддерживающие печать на «бесконечной» ленте в текстовом режиме (матричные) и другие. Другие делятся на струйные и лазерные. Струйные принтеры имеют возможность цветной печати, лазерные могут иметь «бесконечный» лоток подачи.

Каждое задание на печать может иметь следующие признаки: количество листов, разрешение печати, признак цветной печати, признак печати в текстовом режиме на бесконечной ленте. Кроме того, может быть задан либо признак печати на определенном принтере, либо признак печати на определенном типе принтера. Считается, что задание может быть распечатано на принтере только в случае, когда количество листов задания меньше или равно емкости лотка принтера, у принтера есть подходящее разрешение и принтер имеет необходимые дополнительные возможности (цвет, лента).

Очередь должна быть способна принимать задания на печать, располагать задания в надлежащем порядке в соответствии с их признаками и временем поступления (время дискретно). Система должна уметь выполнять следующие действия на каждом шаге по времени:

- добавить задание печати в очередь;
- удалить задание печати из очереди;
- найти подходящий принтер для задания в соответствии с его признаками, в случае отсутствия подходящего принтера – удалить задание из очереди;
- имитировать процесс печати на задания на принтере;
- имитировать поломку принтеры (удаление его из пула принтеров);
- добавление нового принтера;
- вывод информации о состоянии принтеров (у какого задания сколько страниц напечатано, какие задания распределены для этого принтера)
- вывод информации о состоянии заданий (с возможностью отбора по следующим признакам – по типам принтеров, количеству листов в задании, задания для печати на ленте, задания для печати в цвете), количество печатающих заданий и количество ожидающих печать.
- выход.

По итогам работы в очереди печати необходимо получать отчет о количестве и объеме прошедших через очередь печати заданий и суммарное количество заданий и листов по каждому принтеру, типу принтеров.

6. Банкомат

Разработать программное обеспечение для банкомата, описание которого дано ниже.

Банкомат – автомат, выполняющий финансовые операции для клиентов банка. Пользовательский интерфейс банкомата состоит из устройства чтения банковских

карточек (УЧК), дисплея, числовой клавиатуры, специальных клавиш, устройства выдачи денег (УВД), устройство приема денег (УПД) и печатающего устройства (ПУ).
В состоянии ожидания банкомат показывает на дисплее некоторое приветствующее сообщение. Клавиатура и устройство выдачи денег находятся в неактивном состоянии до момента, пока пользователь не вставит карточку в устройство чтения карточек. После вставки карточки УЧК пытается прочесть ее. Если при этом происходит ошибка, на дисплей выдается соответствующее сообщение и карточка возвращается.
Каждая карточка имеет PIN-код и сумму, имеющуюся на карточке. Карточки делятся на дебетовые и кредитные. Дебетовые карточки не позволяют иметь отрицательную сумму на счету карточки. Кредитные – позволяют, но при отрицательной сумме автоматически рассчитывают пени с заданным процентом. Некоторые карточки позволяют выводить историю карточки за последний месяц. Некоторые карточки позволяют переводить только всю сумму со счета на карточку, а некоторые позволяют дополнительно запрашивать сумму на счете и переводить заданную пользователем сумму.
В случае успешного чтения карточки, банкомат запрашивает у пользователя ввод PIN-кода (personal identification number), с использованием числовой клавиатуры. Ввод каждой цифры PIN-кода сопровождается индикацией на дисплее факта нажатия клавиши, но цифры кода не отображаются.
Если пользователь не смог ввести корректный PIN-код, ему предоставляется еще две дополнительные попытки. Если за три попытки правильный код не был введен (в силу, например, разного рода повреждений карточки), банкомат забирает карточку, и она может быть извлечена из него только банковским служащим.
Если пользователь ввел правильный PIN-код, банкомат отображает главное меню, содержащее следующие пункты:

- снять (если возможно) заданную сумму с карточки;
- перевести сумму со счета в банке на карточку с учетом возможностей данной карточки;
- напечатать баланс заданного счета;
- вывести (если возможно) историю карточки.

Пользователь может выбрать желаемое действие и указать необходимую информацию.
После завершения операции банкоматом, он возвращается в главное меню.
В любой момент времени до окончания операции пользователь может ее прервать нажатием специальной кнопки <Отмена>. В этом случае банкомат прерывает все выполняемые операции, печатает отчет о результатах законченных операций, возвращает карточку и переходит в состояние ожидания.

7. Сеть аэропортов

Разработать программу, моделирующую "жизнь" сети аэропортов, регламентируемую следующими правилами. Время дискретно. Каждый аэропорт имеет следующие характеристики: пространственные координаты, одну взлетно-посадочную полосу, количество стоянок для самолетов. За один шаг по времени аэропорт может осуществлять либо посадку, либо взлет одного самолета. Некоторые аэропорты могут принимать только военные самолеты, некоторые – и военные, и гражданские. Любой самолет имеет следующие характеристики: вес заправленного самолета, скорость, максимальное время продолжительности полета, время, необходимое для обслуживания самолета на стоянке. Полет происходит всегда из одного аэропорта в другой. Самолеты делятся на самолеты с полезным грузом и самолеты без груза (военные, прогулочные). Время полета самолета с грузом уменьшается пропорционально фактическому полному весу самолета. Самолеты с грузом бывают пассажирские и транспортные. У пассажирских самолетов имеется дополнительный параметр – количество мест. У транспортных – максимальный вес груза. Через случайные промежутки времени самолеты прибывают из других аэропортов и

запрашивают разрешение на посадку у диспетчера. Диспетчер выдает разрешение на посадку, если взлетно-посадочная полоса в настоящий момент не занята садящимся или взлетающим самолетом и количество приземлившихся самолетов не превышает вместимость аэропорта. Самолет терпит крушение в случае, если время полета превышено, а диспетчер так и не выдал разрешения на посадку. Разрешения на посадку выдаются в следующем порядке – сначала самолеты с минимальным остатком времени полета, затем пассажирские, затем все остальные. Разрешение на посадку не выдается, если все стоянки заняты. У самолета на стоянке аэропорт назначения задается случайно. Масса груза или количество занятых мест также задается случайно непосредственно перед вылетом.

На каждом шаге по времени пользователь может вывести следующую информацию:

- список самолетов в воздухе;
- список самолетов, остаток времени полета которых не превышает заданную величину;
- список самолетов на стоянке любого аэропорта;
- информацию всех предыдущих пунктов можно получить для любого типа самолета;
- количество пассажиров в полете;
- общая масса перевозимого груза;
- общее количество самолетов, потерпевших крушение, количество погибших пассажиров и количество потерянного груза.

8. Магазин

Разработать программу, имитирующую процесс обслуживания покупателей в магазине с несколькими отделами и кассами. Время дискретно. Каждый отдел торгует несколькими видами товаров, однако, каждый вид товаров продается только в одном отделе. В каждом отделе имеется несколько касс. Запас каждого товара пополняется через случайный период времени. Каждый товар принадлежит виду товара (например, фрукты, хлебобулочные изделия), имеет цену и название (например, яблоко, персик). Кроме того, некоторые товары могут быть скоропортящиеся. У таких товаров есть срок хранения, по истечении которого они автоматически изымаются из продажи. Некоторые товары имеют сезонный спрос. У них цена случайным образом меняется по времени на заданную для этого товара величину. Покупатели появляются в магазине через случайные промежутки времени со случайно задаваемой целью закупки заданного количества определенного вида товара и располагая необходимой для этого суммой. При наличии необходимого количества товара в отделе покупатель заполняет корзину, отправляется в наименее загруженную кассу и рассчитывается. За единицу времени каждая касса может обслужить только одного покупателя. Если какого-то товара не хватает, покупатель кладет в корзину фактически имеющееся количество товара.

У каждого покупателя есть пороговое значение длины очереди. Если длина очереди более этого порогового значения покупатель покидает магазин, не дождавшись обслуживания. В этом случае товары возвращаются в отдел к следующему шагу по времени.

Процесс имитации может быть остановлен пользователем в произвольный момент времени с целью просмотра характеристик:

- Сколько и на какую сумму любой выбранный товар был продан.
- Сколько и на какую сумму товар имеется в магазине.
- Сколько и на какую сумму любой выбранный вид товара был продан.
- Сколько и на какую сумму вид товара имеется в магазине.
- Сколько и на какую сумму имеется в магазине скоропортящегося товара с остатком срока хранения не более заданного пользователем количества шагов по времени.

- Какие сезонные товары сейчас имеют нестандартную цену.
- Среднее количество покупателей и средняя сумма покупки за шаг по времени.
- Количество покупателей, не дождавшихся обслуживания.
- Сумма выручки по каждой кассе, отделу и магазину в целом.

9. Компьютерные классы

Разработать программу имитации процесса взаимодействия пользователей сетевых компьютерных классов. Время процесса дискретно. Каждый компьютер имеет следующие параметры:

- тип процессора;
- объем оперативной памяти;
- вместимость жесткого диска;
- вероятность возникновения сбоев в течении часа.

Компьютеры подразделяются на рабочие станции и серверы. Каждая рабочая станция имеет дисплей с заданным максимальным разрешением и принадлежит одному компьютерному классу. Каждый сервер может обслуживать несколько компьютерных классов. Некоторые серверы имеют СУБД (система управления базами данных). Некоторые серверы имеют средства работы с ГИС (географическая информационная система). Некоторые серверы имеют средства резервирования, которые позволяют сохранять работоспособность при одиночном сбое.

Компьютерный класс работоспособен, если в рабочем состоянии находится хотя бы один из способных его обслуживать серверов. Любой компьютер может подвергнуться сбою и потребовать ремонта в течение случайного промежутка времени.

Каждый пользователь следующими параметрами:

- сетевое имя;
- сетевой пароль.

Некоторым пользователям для работы нужен дисплей заданным максимальным разрешением. Некоторым – доступ к СУБД, некоторым – доступ к ГИС. Некоторым – произвольная комбинация вышеперечисленных параметров. На вход в сеть и выход из сети требуется один шаг по времени. Свободная рабочая станция может быть случайно переставлена из одного компьютерного класса в другой, на что требуется один шаг по времени. Сервер может быть случайно подключен или отключен от компьютерного класса.

Пользователи появляются в классе через случайные промежутки времени с целью поработать случайное количество времени. Если нет свободных рабочих станций в этом классе, пользователь пытается найти их в других классах. В случае совершения 3-х неудачных попыток отыскать свободную рабочую станцию пользователь покидает область видимости. Пользователям, которым нужен дисплей с заданным разрешением, могут определить это до входа в сеть. Определить – есть ли сервер с СУБД или ГИС в случае такой необходимости можно только после входа в сеть, проверяя серверы, к которым подключен данный класс. При выходе рабочей станции из строя пользователь предпринимает попытку пересесть на другой компьютер. При выходе из строя сервера пользователи остаются в работе тогда, когда имеются другие серверы с требуемым ПО (и имеющие связь с соответствующим компьютерным классом).

Процесс имитации может быть остановлен пользователем программы для просмотра параметров объектов:

- Сколько работает рабочих станций всего и рабочих станций с заданным разрешением дисплея.
- Сколько серверов обслуживают заданных компьютерный класс – в т.ч., поддерживающие работу СУБД и ГИС.

- Сколько имеется свободных исправных рабочих станций, в т.ч., сколько позволяющих использовать СУБД и ГИС, в т.ч. - с разбивкой по классам.
- Сколько компьютеров неисправно, в т.ч. - сколько рабочих станций и серверов.
- Есть ли серверы, поддерживающие резервирование и подвергшиеся одиночному сбою.
- Статистику по пользователям – сколько времени провел каждый и в среднем.
- Среднее количество попыток, который затратил пользователь на вход в систему, в т.ч. – по пользователям, которым нужны были СУБД и ГИС.

10. Парикмахерская

Разработать программу имитации работы парикмахерской. Всех парикмахеров можно разделить на три уровня – начинающий, опытный, мастер. Имеется общий перечень услуг. Каждая услуга имеет время обслуживания и стоимость. Каждая услуга относится к одной из четырех категорий – базовой, обычной, сложной и нестандартной. Все парикмахеры умеют выполнять все услуги базовой категории. Опытные парикмахеры дополнительно умеют выполнять некоторые услуги из обычной категории. Мастера дополнительно умеют выполнять все услуги из обычной категории и некоторые из сложной категории. Нестандартные услуги не требуют особой квалификации от парикмахера. Каждый парикмахер может выполнять некоторые нестандартные услуги. Парикмахеры работают по принципу – день работы, три дня отдыха, так что в конкретный день может и не быть парикмахеров, выполняющих определенную услугу. Каждый клиент хочет выполнить какую-то услугу, возможно имеет требования к уровню парикмахера и, возможно, при длине очереди, превышающей заданное число, откажется от услуги. Все клиенты делятся на постоянных и случайных. Постоянные клиенты, в отличие от случайных, посещают парикмахерскую регулярно и, возможно, имеют желание, чтобы какой-то вид работ выполнял определенные парикмахер. Клиенты случайно по времени заходят в парикмахерскую и в случае возможности получить услугу, занимают очередь к определенному парикмахеру (или сразу получают услугу, если очереди нет).
Процесс имитации может быть остановлен пользователем программы для просмотра параметров объектов:

- Какова выручка каждого парикмахера и парикмахерской в целом и с разбивкой по уровню парикмахеров.
- Сколько времени в среднем проводит клиент в очереди.
- Сколько клиентов ушли не обслуженными, в т.ч. - сколько постоянных и сколько случайных.
- Какова выручка с разбивкой по типам услуг.
- Насколько выручка парикмахеров, владеющими нестандартными услугами отличается от выручки парикмахеров, не владеющими таковыми.
- Сколько времени в среднем простаивают парикмахеры с разбивкой по их уровню.

11. Железная дорога

Разработать программу, имитирующую процесс пассажирских перевозок по железной дороге. Имеются несколько станций, соединенных железнодорожной сетью. Каждая станция имеет координаты. Между станциями курсируют поезда из нескольких вагонов. Каждый поезд имеет номер, станцию отправления, станцию назначения, время убытия со станции отправления и время прибытия на станцию назначения. Каждый вагон имеет номер. Все вагоны делятся на пассажирские и служебные. Все служебные делятся почтовые, вагон-ресторан, вагон-буфет. Все пассажирские вагоны имеют количество мест и цену места на километр расстояния. Пассажирские вагоны подразделяются на сидячие, плацкартные (дополнительно имеется возможность лежать) и купейные (дополнительно имеется возможность лежать и заказать за отдельную плату постель). Кроме того,

некоторые вагоны оборудованы телевизорами, некоторые – телефонной связью. Эти опции увеличивают стоимость проезда. Пассажиры, желающие уехать, покупают билеты на станциях, называя номер поезда, дату отъезда и пункт назначения и, возможно, условия: тип вагона, наличие ресторана или буфета, требования к дополнительному оборудованию вагонов. В зависимости от наличия таких поездов и мест кассир обеспечивает пассажира билетом.
Процесс имитации может быть остановлен пользователем программы для просмотра параметров объектов:

- Состояние каждого поезда, в т.ч.: типы вагонов, их состояние, количество пассажиров по вагонам и всего.
- Количество пассажиров, требующих дополнительное оборудование вагонов по станциям и всего.
- Загруженность вагонов с разбивкой по их типам.
- Загруженность маршрутов.
- Выручка с разбивкой по поездам, станциям и типам вагонов.
- Количество пассажиров, не сумевших купить билеты с указанием причин, по которым им отказано в продаже.

12. Пчелы

Разработать программу, имитирующую жизнь пчелиной семьи. Пчелиная семья состоит из матки (пчелы, производящей женские яйцеклетки), трутней (производящих мужские яйцеклетки), личинок и рабочих пчел. Рабочие пчелы выполняют два типа работ – добычу меда и уборку улья от трупов мертвых пчел. Матка регулярно производит засев, который оплодотворяется трутнями. Один трутень может оплодотворить некоторое количество яйцеклеток. Личинки появляются на свет через некоторый промежуток времени. После появления личинка некоторое время набирает вес, потребляя мед, затем превращается с какой-то вероятности либо в рабочую пчелу, либо в трутня. Каждая из пчел описывается индивидуальным номером, весом и возрастом. Каждая из пчел регулярно потребляет в пищу из общего хранилища запас меда, пропорциональный ее весу. Длительность жизни пчелы не превосходит некоторой максимальной величины. Умереть раньше этого срока пчела может от голода. Продуктивность матки зависит от количества общего запаса меда в улье. Продуктивность трутней колеблется случайным образом вокруг некоторой средней величины также в зависимости от общего запаса меда. При уборке пчела способна вымести очередной труп, если его масса меньше массы выметающей пчелы. В противном случае пчела ждет другую рабочую пчелу. Производительность матки падает с ростом количества мертвых, не выметенных пчел.
Процесс имитации может быть остановлен пользователем программы для просмотра параметров объектов:

- Количество личинок, трутней, рабочих пчел обоих видов.
- Сколько меда потребляют рабочие пчелы в сравнении с тем, сколько они его приносят.
- Какова эффективность трутней – проанализировать – избыточно или недостаточно количество трутней.
- Процент пчел, умирающих с голода, в т.ч. с разбивкой по личинкам, трутням, рабочим пчелам.
- Количество простаивающих выметающих рабочих пчел.
- Количество мертвых, не выметенных пчел.

13. Почта

Разработать программу, имитирующую работу междугородней почтовой системы в пределах города. Почтовая система города состоит из головного почтамта и сети

почтовых отделений, обслуживающих определенные районы города. Каждое почтовое отделение имеет список адресов, находящихся на ее территории. В обязанности отделений входит сбор входящей почты из ящиков, расположенных на подчиненной им территории, и доставка почтовых отправлений в персональные ящики жильцов домов, расположенных на этой территории. Если отправление адресовано абоненту за пределами обслуживаемой территории, но в пределах города, то отделение пересылает его отделению, обслуживающему указанный район. Если отправление адресовано абоненту за пределами города, то оно пересылается на городской главпочтамт. Однако некоторые почтовые отделения имеют возможность пересылать отправления в некоторые почтовые отделения других городов напрямую. Все отправления с неправильными адресами накапливаются в специальном отделении на главпочтамте и затем возвращаются отправителю. Если и адрес отправителя неправилен, то такое отправление окончательно остается в специальном отделении на главпочтамте. Главпочтамт принимает почту, приходящую из-за пределов города и перераспределяет ее по нужным отделениям. Перераспределение почты между отделениями и главпочтамтом происходит два раза в день. Отправления из-за пределов города поступают на главпочтамт раз день с задержкой по времени, требуемой на междугороднюю доставку. Поступление почты от внутригородских отправителей происходит случайным образом.

Процесс имитации может быть остановлен пользователем программы для просмотра параметров объектов:

- Количество входящих и исходящих отправлений по каждому почтовому отделению, городу и в целом.
- Среднее время доставки междугороднего отправления без прямых связей в сравнении с временем доставки между почтовыми отделениями напрямую.
- Процент отправлений с неправильным адресом получателя
- Список отправлений, окончательно помещенных в специальном отделении на главпочтамтах.
- Поиск всех отправлений для заданного адреса.
- Поиск всех почтовых отделений, имеющих возможность работы напрямую с заданным почтовым отделением.

14. Транспортное агентство

Разработайте программу, имитирующую работу трансагентства. Трансагентство имеет сеть филиалов в нескольких городах. Транспортировка грузов осуществляется между этими городами тремя видами транспорта: автомобильным, железнодорожным и воздушным. Любой вид транспортировки имеет стоимость единицы веса на единицу пути и скорость доставки. Воздушный транспорт можно использовать только между крупными городами, этот вид самый скоростной и самый дорогой. Кроме того, воздушный транспорт зависит от погоды. Доставить груз воздушным путем можно только при условии хорошей погоды одновременно в городах отправки и назначения. Хорошая или плохая погода задается случайным образом. Железнодорожный транспорт можно использовать между крупными и средними городами, этот вид самый дешевый. Автомобильный транспорт можно использовать между любыми городами. Заказчики через случайные промежутки времени обращаются в один из филиалов трансагентства с заказом на перевозку определенной массы груза и возможным пожеланием о скорости/цене доставки. Трансагентство организует отправку грузов одним из видов транспорта с учетом пожеланий клиента. Оплату трансагенство получает только после успешной доставки груза. Между некоторыми городами для железнодорожного и/или автомобильного транспорта имеются скоростные магистрали, на которых скорость соответствующего вида транспорта увеличивается с заданным коэффициентом. При перевозке грузов могут происходить аварии, при этом вероятность аварии на

автотранспорте больше, чем на железнодорожном транспорте, а авиатранспорт имеет аварийность очень низкую. На скоростных магистралях вероятность аварии меньше, чем на обычных дорогах. При аварии трансагентство возвращает заказчику двойную стоимость перевозки.
Процесс имитации может быть остановлен пользователем программы для просмотра параметров объектов:

- Доход трансагенства, в том числе с разбивкой по видам транспорта и городам.
- Среднее время доставки груза, в том числе с разбивкой по видам транспорта и городам.
- Потери, связанные с плохой погодой.
- Потери, связанные с аварийностью, в том числе с разбивкой по видам транспорта и по видам дорог.
- Доход на тонно-километр скоростных магистралей в сравнении с таким же доход на обычных дорогах.
- Список исполняемых заказов с возможность сортировки по городам, видам транспорта, стоимости перевозки.
- Список задерживаемых заказов в связи с плохой погодой.

15. Автострада

Имеется однорядный участок автострады без перекрестков. Автомобили появляются на одном конце дороги, проезжают и бесследно исчезают на другом конце. Машины стремятся двигаться с постоянными скоростями (разными для разных машин). Имеются разные типы автомобилей (разной длины, с разными интервалами типичной скорости). Автомобили можно разделить на легковые и грузовые. Легковые машины имеют повышенную начальную скорость и ограничение на максимальную длину автомобиля. Кроме того, некоторые легковые автомобили являются такси. Такси делают случайные кратковременные остановки (сажают или высаживают клиентов). Грузовые автомобили имеют пониженную начальную скорость и ограничение на минимальную длину автомобиля. Некоторые грузовые автомобили являются рейсовыми автобусами. Они делают регулярные остановки на заранее выделенных участках дорог. Кроме того, часть автомобилей (и легковые, и грузовые) являются автомобилями повышенной проходимости. Такие автомобили при остановке впереди идущего автомобиля могут съехать на обочину и на пониженной скорости (5 км/час) стараться объехать пробку, при первой же возможности пытаясь вернуться на автостраду. При сближении автомобилей на расстояние менее трех длин заднего автомобиля на каждые 15 км/час переднего автомобиля, задний автомобиль начинает тормозить, уменьшая скорость на 1 км/час каждую секунду до тех пор, пока скорости не сравняются. Если передний автомобиль тормозит, то задний снижает скорость на 15 км/час каждую секунду. Если расстояние между автомобилями превышает 10 длин заднего автомобиля, задний автомобиль увеличивает свою скорость на 5 км/час каждую секунду до тех пор, пока его скорость не сравняется с первоначальной скоростью. Программа должна имитировать заполнение автострады машинами разного типа и позволять изучать создание пробок и аварий при различной загрузке автострады. Автомобили запускаются со случайной первоначальной скоростью из допустимого интервала.
Процесс имитации может быть остановлен пользователем программы для просмотра параметров объектов:

- Средняя скорость всех, легковых, грузовых автомобилей, автобусов, автомобилей повышенной проходимости.
- Количество аварий и пробок.
- Количество всех, легковых, грузовых автомобилей, автобусов, автомобилей повышенной проходимости.

- Список всех автомобилей на трассе с выводом их состояния и возможности сортировки по скорости, пройденному расстоянию, типу автомобиля.

16. Жилищно-эксплуатационное управление

В ЖЭУ имеется несколько ремонтных бригад и несколько транспортных средств. Каждая бригада умеет выполнять ряд базовых ремонтных услуг. Некоторые бригады умеют выполнять специализированный ремонт (электрики, сантехники, штукатуры-маляры). Транспортное средство передвигается с заданной скоростью и на его восстановление в случае неисправности требуется некоторое время. Каждая бригада может иметь транспортное средство и в этом случае передвигается со скоростью этого транспортного средства, иначе – с минимальной скоростью (пешехода). Транспортное средство случайно может сломаться. В этом случае транспортное средство отправляется на восстановление, а бригаде при наличии свободных транспортных средств выдается новое. Если же свободных транспортных средств нет, бригада передвигается пешим порядком. Каждая бригада затрачивает на ремонт некоторое время. Некоторые виды ремонта требуют специального инструмента независимо от назначения ремонта. Бригада может выполнять такой ремонт только при умении пользоваться специальным инструментом и наличии транспортного средства. Имеются дома с заданным расстоянием от ЖЭУ. В каждом доме имеется некоторое количество квартир. Каждая квартира имеет срок службы оборудования различного типа и процент его износа. Каждое оборудование требует ремонта соответствующего назначения (базовый, электрический, сантехнический, строительный) и возможно, соответствующего специального инструмента. Как только в квартире процент износа оборудования становится 100-процентным - вызывается соответствующая бригада (если есть таковая свободная). Бригада затрачивает время на перемещение до дома и обратно и на ремонт. После ремонта износ считается нулевым. Написать программу для имитации работы ремонтных бригад с возможностью задания разного количества бригад, транспортных средств, специального инструмента, домов и оборудования.

Процесс имитации может быть остановлен пользователем программы для просмотра параметров:

- Среднее время, затрачиваемое на ремонт одного оборудования, в т.ч. с разбивкой по назначению оборудования.
- Среднее время, затрачиваемое на ремонт одного оборудования, требующего специального инструмента по сравнению со средним временем, затрачиваемым на ремонт одного оборудования, не требующего специального инструмента.
- Список всех бригад с указанием из состояния с возможностью сортировки по наличию транспортного средства, назначения бригады, состоянию (ожидает, едет на ремонт, ремонтирует, возвращается с ремонта).
- Список оборудования с процентом износа больше заданного с возможностью сортировки по удаленности от ЖЭУ, назначению.

17. Ломбард

Разработайте программу, имитирующую работу ломбарда. Имеются люди, которые сдают вещи в залог. Каждый человек имеет имя и паспорт. Паспорт человека содержит номер и домашний адрес. Некоторые люди проходят государственную службу, и у них вместо паспорта имеется удостоверение личности, в котором есть только номер. Каждая вещь имеет наименование и оценку. Вещи принимаются только следующих типов – золото, меха, бытовая техника. Оценка за золотые вещи определяется исходя из веса и пробы. Оценка за бытовую технику определяется, исходя из времени эксплуатации техники и типичной стоимости нового аналога этой техники. Оценка за меха определяется исходя из веса, ценности меха и степени износа. На каждую вещь выдается ссуда в

размере 50% от стоимости оценки. За один залог человек может сдать несколько вещей различного типа. За залог взимается процентная ставка в день от суммы ссуды. В случае продления залога, ломбард получает только процентную ставку и создает новый залог на тех же условиях, что и первоначальный. Возможен досрочный выкуп залога. За каждый просроченный день взимается штраф – заданная процентная ставка от суммы оценки. В случае просрочки плановой даты возврата на 30 дней – вещи залога продаются и ломбард получает сумму оценки через 90 дней после плановой даты возврата.

Процесс имитации может быть остановлен пользователем программы для просмотра параметров:

- Сумма дохода и расхода за каждый день.
- Планируемый дневной доход в начале текущего дня.
- Полное состояние любого залога.
- Список просроченных залогов с возможностью сортировки по продолжительности просрочки, имени человека, сумме залога.
- Список людей, которые задерживали выплату залога более заданного количества раз.
- Распределение доходов по типам вещей.
- Список проданных вещей с возможностью сортировки по наименованию и типу вещи.

18. Государственное унитарное предприятие

Написать программу для имитации работы государственного унитарного предприятия (ГУП). Работа ГУП заключается в получении бюджетного финансирования, заключении договоров с хозяйством, поставке по эти договорам бюджетных ресурсов (денег) хозяйству и получении от него готовой продукции. Готовая продукция имеет наименование. Будем полагать, что хозяйства поставляют всего два вида продукции – пшеницу и сыр. Сыр имеет дополнительные параметры – сорт (алтайский, радонежский, голландский и т.д.) и жирность – 30%, 40%, 50%. Пшеница имеет параметр твердости – твердая или мягкая. Кроме того, сыр может иметь упаковку (пленка или парафин), а пшеница может классифицироваться по классности – 3 класс, 4 класс, нестандартная. Договор заключается на каждый год. В нем указываются планируемое количество выдаваемых денег и получаемой продукции. На реальные поставки выписываются документы, в которых указываются – дата, поставщик, получатель, список продукции (ресурсов). Для каждого пункта этого списка дополнительно указывается количество. Программа должна загружать файл с состоянием договоров и реальных поставок с жесткого диска и в паузе или по окончании процесса имитации сохранять файл на диск.
Процесс имитации может быть остановлен пользователем программы для просмотра параметров:

- Количество ресурсов и продукции по договорным обязательствам – за все годы и в выбранном году. Список продукции может быть детальным или укрупненным (пшеница, сыр). Список продукции может быть отсортирован по наименованию или количеству.
- Количество ресурсов и продукции по реальным поставкам – за все годы и в выбранном году. Список продукции может быть детальным или укрупненным (пшеница, сыр). Список продукции может быть отсортирован по наименованию или количеству.
- Отклонение количества поставляемых ресурсов и продукции от договорных обязательств – за все годы и в выбранном году. Список продукции может быть детальным или укрупненным (пшеница, сыр). Список продукции может быть отсортирован по наименованию или количеству.

- Количество сыра заданной жирности и возможно заданной упаковкой с разбивкой по сортам и суммарно.
- Поиск всех реальных поставок, в которых имеется заданная продукция.
- Поиск всех реальных поставок, в которых отсутствует заданная продукция.

19. Справочник работника зоопарка

Разработайте программу для ведения справочника обезьян зоопарка, полностью помещающегося в оперативной памяти. Каждое животное имеет кличку, рост и вес. Любое животное классифицируется по семейству и роду. Всего есть три семейства обезьян: широконосые (игрунки, ревуны), узконосые (мартышки, макаки), человекообразные (гориллы, орангутаны, шимпанзе). Узконосые обезьяны любят купаться. Человекообразным обезьянам нужно много место в вольерах. Кроме того, некоторые виды обезьян не имеют хвоста. Некоторые животные обладают повышенной драчливостью. Имеются также вольеры, в которых живут животные. Каждый вольер имеет объем и признак наличия бассейна. Программа должна загружать справочник с жесткого диска, править все его данные, сохранять на диск. Правка данных заключается в приеме из других зоопарков новых обезьян, отсылке обезьян, перемещению обезьян по вольерам. В процессе работы пользователь программы должен иметь возможность просмотра, как минимум, следующих параметров:

- Список всех обезьян с возможность сортировки по кличке, росту, весу, роду, наличию хвоста, уровню драчливости, номеру вольера.
- Список всех вольеров с возможности сортировки по объему, наличию бассейна, общему количеству обезьян, количеству драчливых обезьян.
- Список узконосых обезьян, которые живут в вольерах без бассейна.
- Список человекообразных обезьян, отсортированный по объему вольера, приходящегося на одну обезьяну.

20. Добыча природного топлива

Разработать программу, реализующую справочник для хранения информации о поставщиках природного топлива. Имеется список поставщиков с полями: наименование топлива, единица измерения, добывается в год, год, страна, месторождение, транспортные расходы на тыс. км, удаленность от базовой точки, тыс. км., срок поставки. Справочник видов природного топлива. Справочник поставщиков. Предусмотреть возможность подбора наилучшего поставщика по критерию

Программа должна загружать файл с состоянием склада с жесткого диска и в паузе или по окончании процесса имитации сохранять файл на диск.

21. Свободная тема

Любая другая тема, обеспечивающая проект, описываемый 5-10 основными понятиями с нетривиальными связями между ними. Для утверждения свободной темы необходимо согласие преподавателя.