Лабораторная работа №2.2: Создание геоинформационного приложения. Движение объектов на карте

Цель работы:

- освоить технологии создания геоинформационных приложений, реализовать движение объектов на карте;
- ознакомиться с принципами взаимодействия объектов в объектно-ориентированном программировании.

Агрегирование

В объектно-ориентированном программировании под агрегированием (или как его еще называют - делегированием) подразумевают методику создания нового класса из уже существующих классов путём их включения. Об агрегировании также часто говорят как об «отношении принадлежности» по принципу «у машины есть корпус, колёса и двигатель».

В качестве примера агрегирования можно привести класс треугольника, реализованный в лабораторной работе №2:

```
class Triangle
{
    private Point2D p1;
    private Point2D p2;
    private Point2D p3;

    public Triangle(Point2D p1, Point2D p2, Point2D p3)
    {
        this.p1 = p1;
        this.p2 = p2;
        this.p3 = p3;
    }

    public Point2D getPoint1()
    {
        return p1;
    }
    public Point2D getPoint2()
    {
        return p2;
    }
    public Point2D getPoint3()
    {
        return p3;
    }

    public void shiftX(double value)
    {
        p1.shiftX(value);
        p2.shiftX(value);
        p3.shiftX(value);
    }
    public void shiftY(double value)
    {
        p1.shiftY(value);
        p2.shiftY(value);
        p3.shiftY(value);
    }
}
```

Класс Triangle агрегирует значения трёх точек, которые являются объектами класса Point2D. Методы shiftX и shiftY реализуют (в данном случае делегируют) свой функционал с помощью вложенных объектов-точек.

События

Событие - это сообщение, посланное объектом, чтобы сообщить о совершении действия.

Порядок работы с событиями в языке C# выглядит следующим образом:

1. определить событие в классе-отправителе;
2. реализовать отправку событий;
3. определить обработчики событий в классах-подписчиках;
4. назначить обработчики событий (подписаться на событие);
5. отписаться по завершению работы.

Ниже приведен пример реализации счётчика, который накапливает переданные числовые значения и сообщает о переполнении с помощью событий.

```
class Counter
{
    // пороговый уровень
    private const int THRESHOLD = 10;
    // текущее значение счетчика
    private int total;

    // ссылка на событие переполнения счётчика
    public event EventHandler ThresholdReached;

    // метод увеличения счетчика
    public void Add(int value)
    {
        // увеличение текущего значения счетчика
        total += value;
        // проверка на переполнение счетчика относительно порогового уровня
        if (total >= THRESHOLD)
        {
            // отправка события переполнения счётчика
            if (ThresholdReached != null)
            {
                ThresholdReached(this, EventArgs.Empty);
            }
        }
    }
}
```

```
static void Main()
{
    // создание экземпляра счетчика
    var c = new Counter();
    // назначение обработчика на событие переполнения счётчика
    c.ThresholdReached += c_ThresholdReached;
}

// обработчик события переполнения счётчика
static void c_ThresholdReached(object sender, EventArgs e)
{
    Console.WriteLine("The threshold was reached.");
}
```

Таким образом, при достижении счётчиком значения 10 и более, класс `Counter` отправит событие переполнения `ThresholdReached`, что повлечёт вызов метода `c_ThresholdReached`.

Перемещение маркеров

Перемещение маркеров можно описать в виде блок-схемы:

С помощью встроенных в библиотеку GMap.NET средств навигации можно построить маршрут между двумя точками, который будет представлен в виде списка географических координат. Жля этого, необходимо установить ключ проверки подлинности карт при инициализации карты:

```
private void MapLoaded(object sender, RoutedEventArgs e)
{
    GMaps.Instance.Mode = AccessMode.ServerAndCache;              // настройка доступа к данным
    BingMapProvider.Instance.ClientKey = " authentication key "; // ключ проверки подлинности карт
    Map.MapProvider = BingMapProvider.Instance;                   // установка провайдера карт
    …
}
```

Инструкцию по получению ключа можно найти по ссылке: https://docs.microsoft.com/ru-ru/windows/uwp/maps-and-location/authentication-key

А затем, получить маршрут между указанными точками:

```
// определение маршрута
MapRoute route = GMap.NET.MapProviders.BingMapProvider.Instance.GetRoute(
    new PointLatLng(55.016215, 82.948772), // начальная точка маршрута
    new PointLatLng(55.016667, 82.949546), // конечная точка маршрута
    false, // поиск по шоссе (false - включен)
    false, // режим пешехода (false - выключен)
    (int)DEFAULT_ZOOM);
// получение точек маршрута
List<PointLatLng> routePoints = route.Points;
```

Стоит отметить, что не для всех точек возможно построение маршрута, поэтому ссылка `route` может быть пустой (равной `null`).

Последовательное перемещение объекта по точкам маршрута должно выполняться в отдельном потоке, чтобы не блокировать главный поток приложения, например:

```
Thread newThread = new Thread(new ThreadStart(MoveByRoute));
newThread.Start();
```

где `MoveByRoute` – метод, осуществляющий перемещение объекта. Метод `MoveByRoute` может выглядеть следующим образом:

```
private void MoveByRoute()
{
    // последовательный перебор точек маршрута
    foreach (var point in Route.Points)
    {
        // делегат, возвращающий управление в главный поток
        Application.Current.Dispatcher.Invoke(delegate {
            // изменение позиции маркера
            Marker.Position = point;
        });
        // задержка 500 мс
        Thread.Sleep(500);
    }
}
```

Чтобы маркер автомобиля при перемещении выглядел естественно, необходимо поворачивать его на угол направления движения. Ниже приведен пример приближенного вычисления угла направления движения и поворота маркера:

```
// вычисление разницы между двумя соседними точками по широте и долготе
double latDiff = nextPoint.Lat - point.Lat;
double lngDiff = nextPoint.Lng - point.Lng;
// вычисление угла направления движения
// latDiff и lngDiff - катеты прямоугольного треугольника
double angle = Math.Atan2(lngDiff, latDiff) * 180.0 / Math.PI;
// установка угла поворота маркера
CarMarker.Shape.RenderTransform = new RotateTransform { Angle = angle };
```

где `point` и `nextPoint` – две соседние точки маршрута.

На практике часто требуется отслеживать состояние мобильных объектов (например, для контроля скорости или местоположения) другими объектами. Например, человек отслеживает местоположение автомобиля при заказе такси и получает уведомление о прибытии. Такое поведение может быть реализовано при помощи событий. Ниже представлен пример кода, реализующего отправку и прием события о прибытии автомобиля:

```
// класс автомобиля
class Car : MapObject
{
    // событие прибытия
    public event EventHandler Arrived;
    // метод перемещения по маршруту
    private void MoveByRoute()
    {
        foreach (var point in Route.Points)
        {
            Application.Current.Dispatcher.Invoke(delegate
            {
                CarMarker.Position = point;
            });
            Thread.Sleep(500);
        }
        // отправка события о прибытии после достижения последней точки маршрута
        Arrived?.Invoke(this, null);
    }
}
```

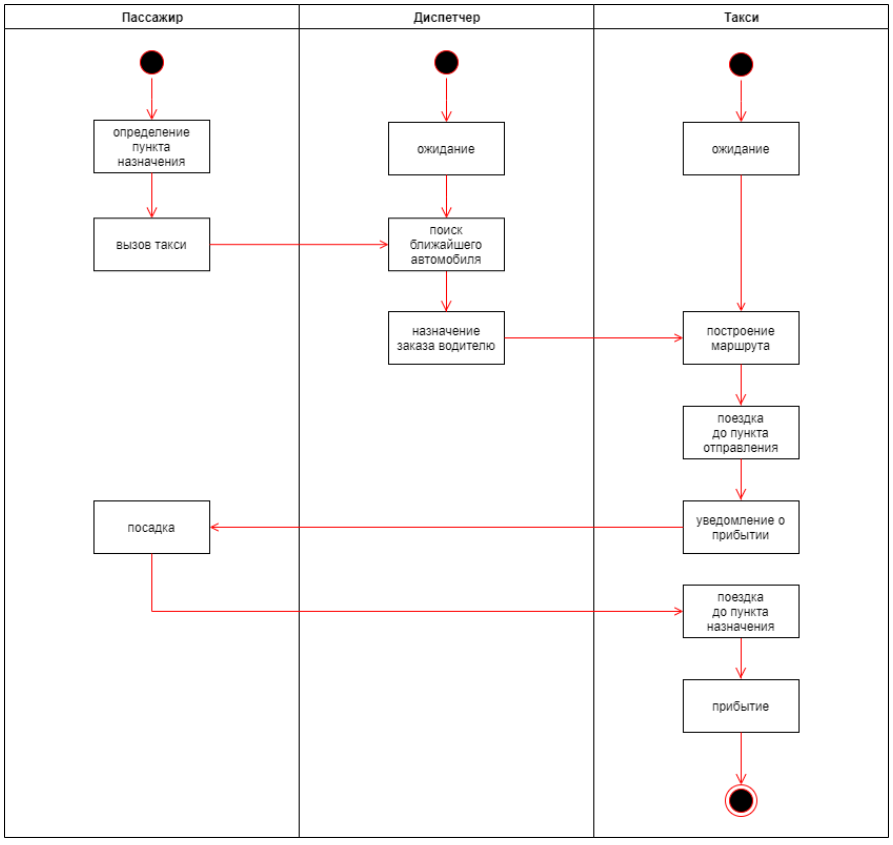
```
// класс человека
class Human : MapObject
{
    // обработчик события прибытия такси
    public void CarArrived(object sender, EventArgs e)
    {
        // TODO : сесть в машину
    }
}
```

```
Human human = new Human();
Car car = new Car();
car.Arrived += human.CarArrived;
```

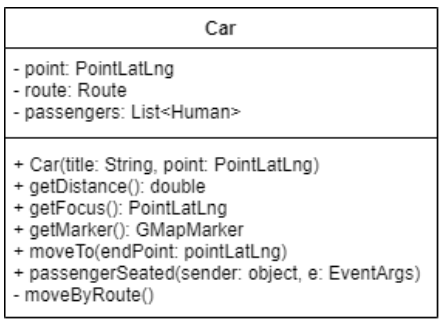
Задание:

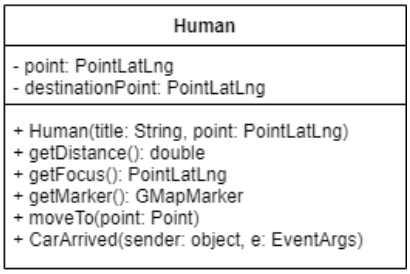
Доработать приложение из лабораторной работы №2.1 и внести следующий функционал:

1. Симулировать процесс заказа такси, описанный диаграммой деятельности:

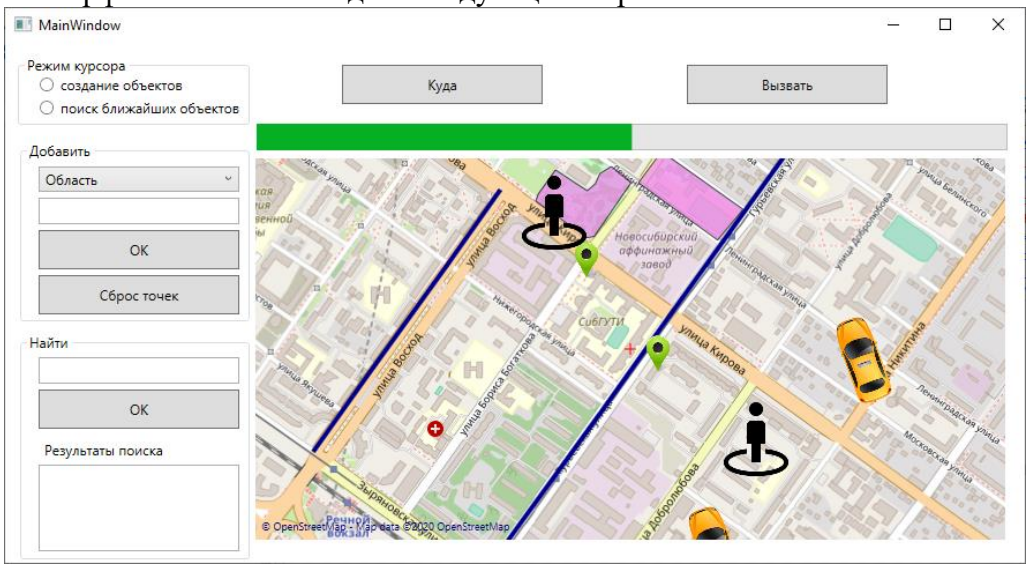


1.1. Реализовать методы перемещения для классов `Car` и `Human`. Дополненная диаграмма этих классов будет выглядеть так:





1.2. Выбор пункта отправления и пункта назначения реализовать двойным нажатием курсора на карту. Интерфейс может выглядеть следующим образом:



1.3. Реализовать взаимодействие объектов «Пассажир» и «Такси» с помощью событий. В данном случае, события соответствуют переходам от одного потока к другому в диаграмме деятельности (событие вызова такси, событие прибытия такси, событие посадки).

1.4. Перемещение пассажира после посадки реализовать с помощью агрегации (при изменении местоположения такси меняется и местоположение пассажира).

1.5. Реализовать отрисовку маршрута и индикатора прогресса.

1.6. Во время поездки карта должна фокусироваться на автомобиле, при этом маркер автомобиля должен поворачиваться согласно своему направлению движения.

Контрольные вопросы:

1) Что такое агрегирование? Приведите пример.
2) Объясните порядок работы с событиями в C#.
3) Классы EventHandler и EventArgs. Передача данных в событиях.
4) Какой метод из библиотеки GMap.NET.Windows позволяет определить маршрут между двумя точками? Какая сигнатура у этого метода?

Список литературы:

1) Герберт Шилдт "C# 4.0: полное руководство"
2) Эндрю Троелсен "Язык программирования C# 5.0 и платформа .NET 4.5"
3) Руководство по программированию на C# https://docs.microsoft.com/ruru/dotnet/csharp/programming-guide/index
4) Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7 https://metanit.com/sharp/tutorial/
5) Руководство по WPF https://metanit.com/sharp/wpf/
6) C# 5.0 и платформа .NET 4.5 http://professorweb.ru/my/csharp/charp_theory/level1/infocsharp.php
7) https://github.com/Microsoft/WPF-Samples