Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра технічної кібернетики

Лабораторна робота № 1

по дисципліні «Об'єктно-орієнтоване програмування»

Тема: Створення програми взаємодії класів «Транспорт2»

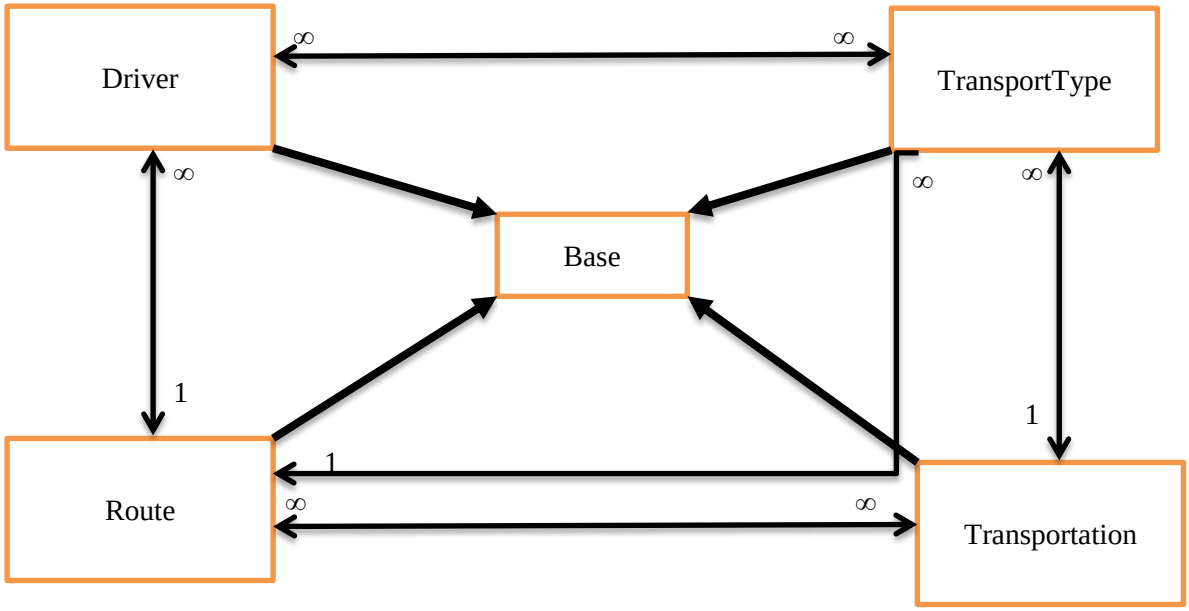
Виконав:
студент групи ІТ-62
ПІБ Любченко П.І.
Дата здачі_____
Захищено з балом _____

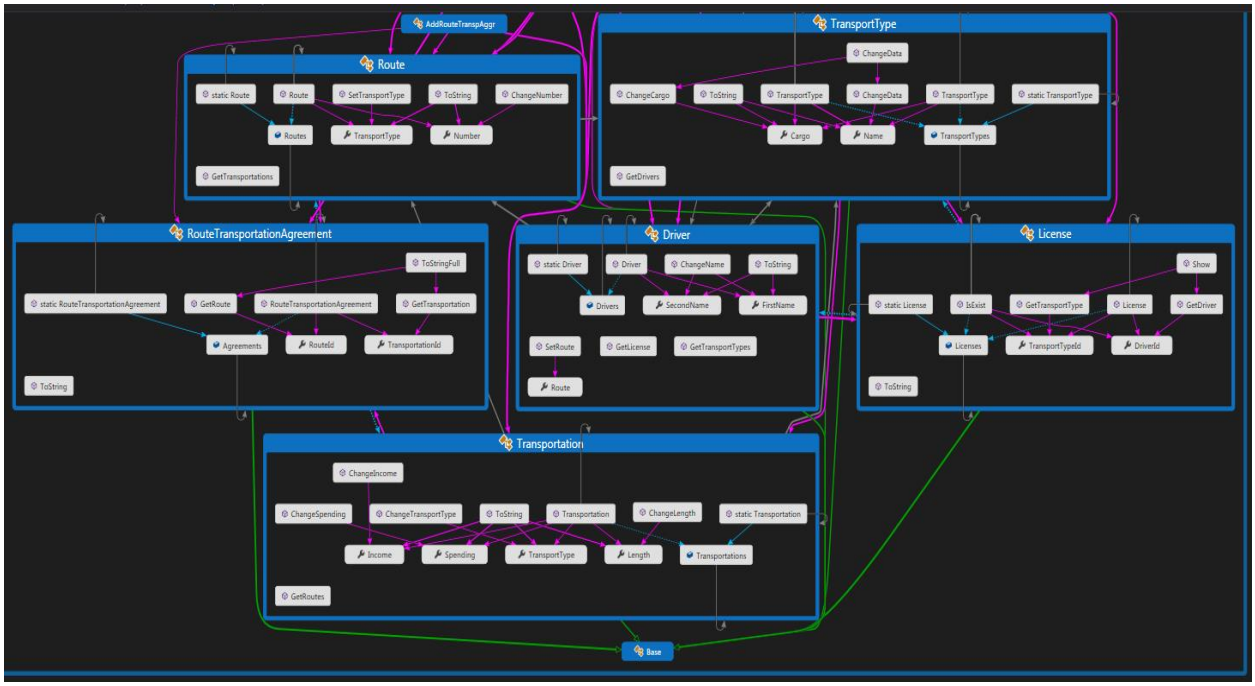
Перевірено:
ст.вик. кафедри АУТС
Хмелюк Марина Сергіївна

Київ 2018

Завдання:
Транспорт2 (водитель, перевозки, маршрут, тип транспорта)
Согласно теме:
- создать 4 класса;
- в каждом из классов описать ВСЕ основные поля (разнотипные);
- с помощью форм создания, редактирования и удаления записей объектов предметной области наполнить массивы соответствующих объектов;
- в соответствующих формах добавить необходимые элементы управления для ввода полей объектов соответствующих классов

Схема класів (UMl-діаграма)





Код Класів:
Base

```
public class Base
    {
        private Guid Id { get; set; }
        public Base()
```

```
        {
            this.Id = Guid.NewGuid();
        }

        public Guid GetId()
        {
            return Id;
        }
    }
```

Driver

```
{
        public static List<Driver> Drivers = new List<Driver>();

        public string FirstName { get; private set; }
        public string SecondName { get; private set; }
        public Route Route { get; private set; }

        public Driver(string firstname, string secondname)
        {
            FirstName = firstname;
            SecondName = secondname;
            Drivers.Add(this);
        }

        public void ChangeName(string firstname, string secondname)
        {
            FirstName = firstname;
            SecondName = secondname;
        }
        public void SetRoute(Route route)
        {
            Route = route;
        }
        public List<TransportType> GetTransportTypes()
        {
            List<TransportType> types = new List<TransportType>();
            foreach (var lic in License.Licenses)
            {
                if (lic.DriverId == this.GetId())
                    types.Add(lic.GetTransportType());
            }
            return types;
        }
        public List<License> GetLicense()
        {
            List<License> lics = new List<License>();
            foreach (var lic in License.Licenses)
            {
                if (lic.DriverId == this.GetId())
                    lics.Add(lic);
            }
            return lics;
        }
        public override string ToString()
        {
            return String.Format("{0} {1}", FirstName, SecondName);
        }
    }
```

Route

```
public class Route : Base
    {
```

```
        public static List<Route> Routes = new List<Route>();
        public int Number { get; private set; }
        public TransportType TransportType { get; private set; }

        public Route(TransportType transporttype)
        {
            TransportType = transporttype;
            Number = Routes.Count >= 1 ? Routes.Count + 1 : 1;

            Routes.Add(this);
        }
        public List<RouteTransportationAgreement> GetTransportations()
        {
            List<RouteTransportationAgreement> transp = new
List<RouteTransportationAgreement>();
            foreach (var rta in RouteTransportationAgreement.Agreements)
            {
                if (rta.RouteId == this.GetId())
                    transp.Add(rta);
            }
            return transp;
        }
        public void ChangeNumber(int number)
        {
            Number = number;
        }
        public void SetTransportType(TransportType transporttype)
        {
            TransportType = transporttype;
        }

        public override string ToString()
        {
            if (TransportType != null)
                return String.Format("№ {0} - {1}", Number.ToString(),
TransportType.ToString());
            return String.Format("№ {0}", Number.ToString());
        }
    }
```

TransportType

```
public class TransportType : Base
    {
        public static List<TransportType> TransportTypes = new List<TransportType>();

        public string Name { get; private set; }
        public string Cargo { get; private set; }



        public TransportType(string name, string cargo)
        {
            Name = name;
            Cargo = cargo;
            bool isExist = false;
            foreach (var tt in TransportTypes)
                if (tt.Name == name && tt.Cargo == cargo)
                {
                    isExist = true;
                    foreach (Transportation tr in Transportation.Transportations)
                        if (tr.TransportType == this)
                            tr.ChangeTransportType(tt);
                    break;
                }
            if (!isExist)
```

```
            TransportTypes.Add(this);
        }
        public TransportType()
        {
            Name = String.Empty;
            Cargo = String.Empty;

            TransportTypes.Add(this);
        }
        public List<Driver> GetDrivers()
        {
            List<Driver> drivers = new List<Driver>();
            foreach(var lic in License.Licenses)
            {
                if (lic.TransportTypeId == this.GetId())
                    drivers.Add(lic.GetDriver());
            }
            return drivers;
        }
        public void ChangeData(string name)
        {
            Name = name;
        }
        public void ChangeCargo(string cargo)
        {
            Cargo = cargo;
        }
        public void ChangeData(string name, string cargo)
        {
            ChangeData(name);
            ChangeCargo(cargo);
        }

        public override string ToString()
        {
            return String.Format("{0} - {1}", Name, Cargo);
        }
       }
```

Transportation

```
public class Transportation : Base
    {
        public static List<Transportation> Transportations = new List<Transportation>();
        public decimal Income { get; private set; }
        public decimal Spending { get; private set; }
        public int Length { get; private set; }
        public TransportType TransportType { get; set; }

        public Transportation(int length, decimal income, decimal spending, TransportType
transporttype = null)
        {
            Income = income;
            Spending = spending;
            Length = length;
            if (transporttype != null)
                TransportType = transporttype;
            Transportations.Add(this);
        }
        public List<Route> GetRoutes()
        {
            List<Route> routes = new List<Route>();
            foreach (var rta in RouteTransportationAgreement.Agreements)
            {
                if (rta.TransportationId == this.GetId())
                    routes.Add(rta.GetRoute());
```

```
            }
            return routes;
        }
        public void ChangeTransportType(TransportType transporttype)
        {
            TransportType = transporttype;
        }

        public override string ToString()
        {
            if (TransportType != null)
                return String.Format("{0} - {1} km. | ${2}", TransportType.ToString(),
Length.ToString(), (Income - Spending).ToString());
            return String.Format("No Type - {0} km. | ${1}", Length.ToString(), (Income -
Spending).ToString());
        }
        public void ChangeLength(int distance) {
            Length = distance;
        }
        public void ChangeIncome(decimal income)
        {
            Income = income;
        }
        public void ChangeSpending(decimal spending)
        {
            Spending = spending;
        }
    }
```

License (розвязочний клас Driver - Route)

```
public class License : Base
    {
        public static List<License> Licenses = new List<License>();

        public Guid DriverId { get; set; }
        public Guid TransportTypeId { get; set; }

        public License(Driver driver, TransportType transporttype)
        {
            DriverId = driver.GetId();
            TransportTypeId = transporttype.GetId();
            Licenses.Add(this);
        }
        public Driver GetDriver()
        {
            Driver driver = null;
            foreach (var dr in Driver.Drivers)
            {
                if (DriverId == dr.GetId())
                {
                    driver = dr;
                    break;
                }
            }
            return driver;
        }
        public TransportType GetTransportType()
        {
            TransportType transporttype = null;
            foreach (var tt in TransportType.TransportTypes)
            {
                if (TransportTypeId == tt.GetId())
                {
                    transporttype = tt;
```

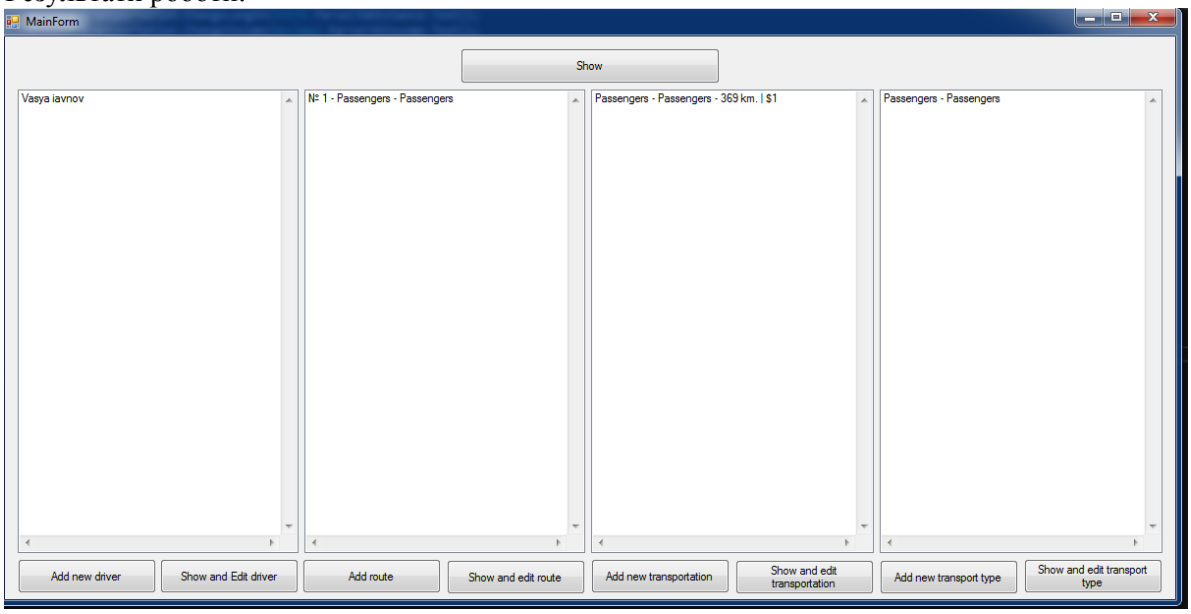
```
                    break;
                }
            }
            return transporttype;
        }
        public string Show()
        {
            Driver driver = GetDriver();
            TransportType transporttype = GetTransportType();
            return String.Format("Driver: {0}\nTransport type: {1}", driver.ToString(),
transporttype.ToString());
        }
        public override string ToString()
        {
            int length = this.GetId().ToString().Length;
            return String.Format("{0}...{1}", this.GetId().ToString().Substring(0, 4),
this.GetId().ToString().Substring(length - 6, 5));
        }
        public static bool IsExist(Driver driver, TransportType transporttype)
        {
            foreach (var lic in Licenses)
            {
                if (lic.DriverId == driver.GetId() && lic.TransportTypeId ==
transporttype.GetId())
                    return true;
            }
            return false;
        }
    }
```

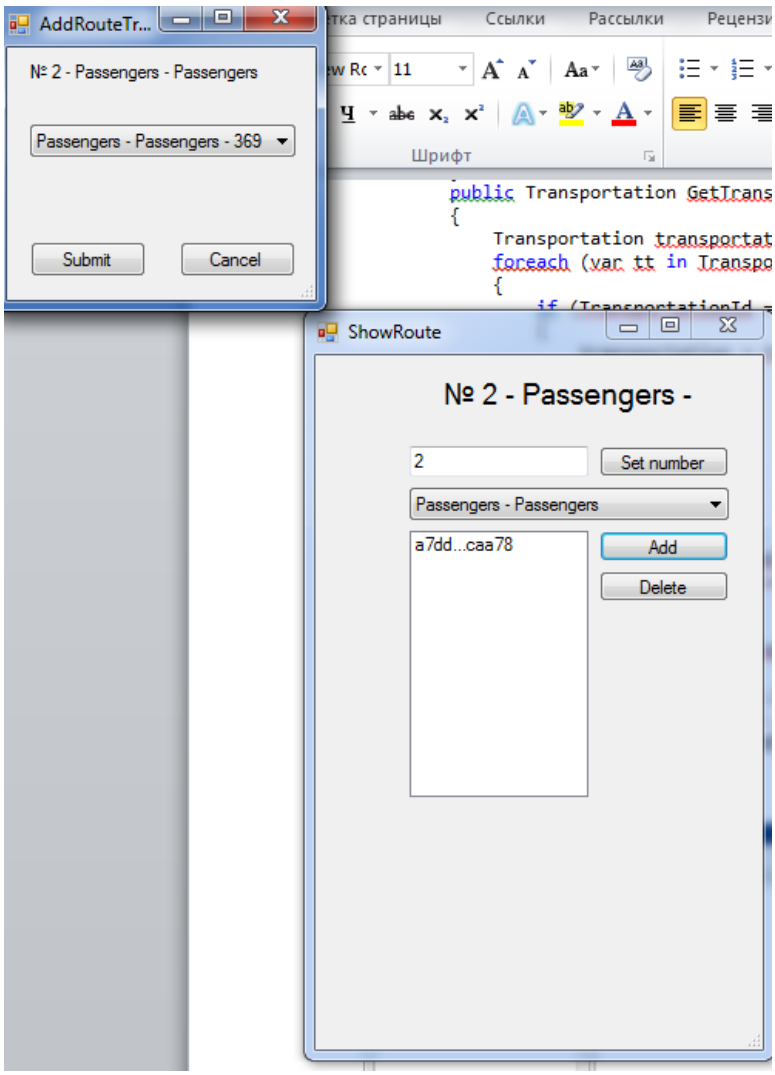
RouteTransportationAgreement

```
public class RouteTransportationAgreement : Base
    {
        public static List<RouteTransportationAgreement> Agreements = new
List<RouteTransportationAgreement>();

        public Guid RouteId { get; private set; }
        public Guid TransportationId { get; private set; }

        public RouteTransportationAgreement(Route route, Transportation transportation)
        {
            RouteId = route.GetId();
            TransportationId = transportation.GetId();
            Agreements.Add(this);
        }
        public Route GetRoute()
        {
            Route route = null;
            foreach (var rt in Route.Routes)
            {
                if (RouteId == rt.GetId())
                {
                    route = rt;
                    break;
                }
            }
            return route;
        }
        public Transportation GetTransportation()
        {
            Transportation transportation = null;
            foreach (var tt in Transportation.Transportations)
            {
                if (TransportationId == tt.GetId())
                {
```

```
                    transportation = tt;
                    break;
                }
            }
            return transportation;
        }
        public string ToStringFull()
        {
            Route route = GetRoute();
            Transportation transportation = GetTransportation();
            return String.Format("Driver: {0}\nTransport type: {1}", route.ToString(),
transportation.ToString());
        }
        public override string ToString()
        {
            int length = this.GetId().ToString().Length;
            return String.Format("{0}...{1}", this.GetId().ToString().Substring(0, 4),
this.GetId().ToString().Substring(length - 6, 5));
        }
    }
```

Результати роботи:





Висновок: я навчився налаштовувати звязки між класами та їх взаємодію; покращив вміння працювати з формами.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра технічної кібернетики

Лабораторна робота № 2

по дисципліні «Об'єктно-орієнтоване програмування»

Тема: Робота з методами класів «Транспорт2»

Виконав:
студент групи ІТ-62
ПІБ Любченко П.І.
Дата здачі_____
Захищено з балом _____

Перевірено:
ст.вик. кафедри АУТС
Хмелюк Марина Сергіївна

Київ 2018

Завдання:
- в каждом классе реализовать по 5 методов с разнотипными параметрами и значением, что возвращается;
- в двух классах реализовать, как минимум, по одному перегруженному методу.

Необв'язкове завдання:
Напишите рекурсивную программу, печатающую n-ое число Фибоначчи

Методи:
Driver

```
public void ChangeName(string firstname, string secondname)
        {
            FirstName = firstname;
            SecondName = secondname;
        }
        public void SetRoute(Route route)
        {
            Route = route;
        }
        public List<TransportType> GetTransportTypes()
        {
            List<TransportType> types = new List<TransportType>();
            foreach (var lic in License.Licenses)
            {
                if (lic.DriverId == this.GetId())
                    types.Add(lic.GetTransportType());
            }
            return types;
        }
        public List<License> GetLicense()
        {
            List<License> lics = new List<License>();
            foreach (var lic in License.Licenses)
            {
                if (lic.DriverId == this.GetId())
                    lics.Add(lic);
            }
            return lics;
        }
        public override string ToString()
        {
            return String.Format("{0} {1}", FirstName, SecondName);
        }
```

Route

```
public Route(TransportType transporttype)
        {
            TransportType = transporttype;
            Number = Routes.Count >= 1 ? Routes.Count + 1 : 1;

            Routes.Add(this);
        }
        public List<RouteTransportationAgreement> GetTransportations()
        {
            List<RouteTransportationAgreement> transp = new
List<RouteTransportationAgreement>();
            foreach (var rta in RouteTransportationAgreement.Agreements)
            {
                if (rta.RouteId == this.GetId())
                    transp.Add(rta);
            }
            return transp;
```

```
        }
        public void ChangeNumber(int number)
        {
            Number = number;
        }
        public void SetTransportType(TransportType transporttype)
        {
            TransportType = transporttype;
        }

        public static void Remove(Route route)
        {
            Routes.Remove(route);
        }

        public override string ToString()
        {
            if (TransportType != null)
                return String.Format("№ {0} - {1}", Number.ToString(),
TransportType.ToString());
            return String.Format("№ {0}", Number.ToString());
        }
```

TransportType

```
public List<Driver> GetDrivers()
        {
            List<Driver> drivers = new List<Driver>();
            foreach(var lic in License.Licenses)
            {
                if (lic.TransportTypeId == this.GetId())
                    drivers.Add(lic.GetDriver());
            }
            return drivers;
        }
        public void ChangeData(string name)
        {
            Name = name;
        }
        public void ChangeCargo(string cargo)
        {
            Cargo = cargo;
        }
        public void ChangeData(string name, string cargo)
        {
            ChangeData(name);
            ChangeCargo(cargo);
        }

        public override string ToString()
        {
            return String.Format("{0} - {1}", Name, Cargo);
        }
```

Transportation

```
public List<Route> GetRoutes()
        {
            List<Route> routes = new List<Route>();
            foreach (var rta in RouteTransportationAgreement.Agreements)
            {
                if (rta.TransportationId == this.GetId())
```

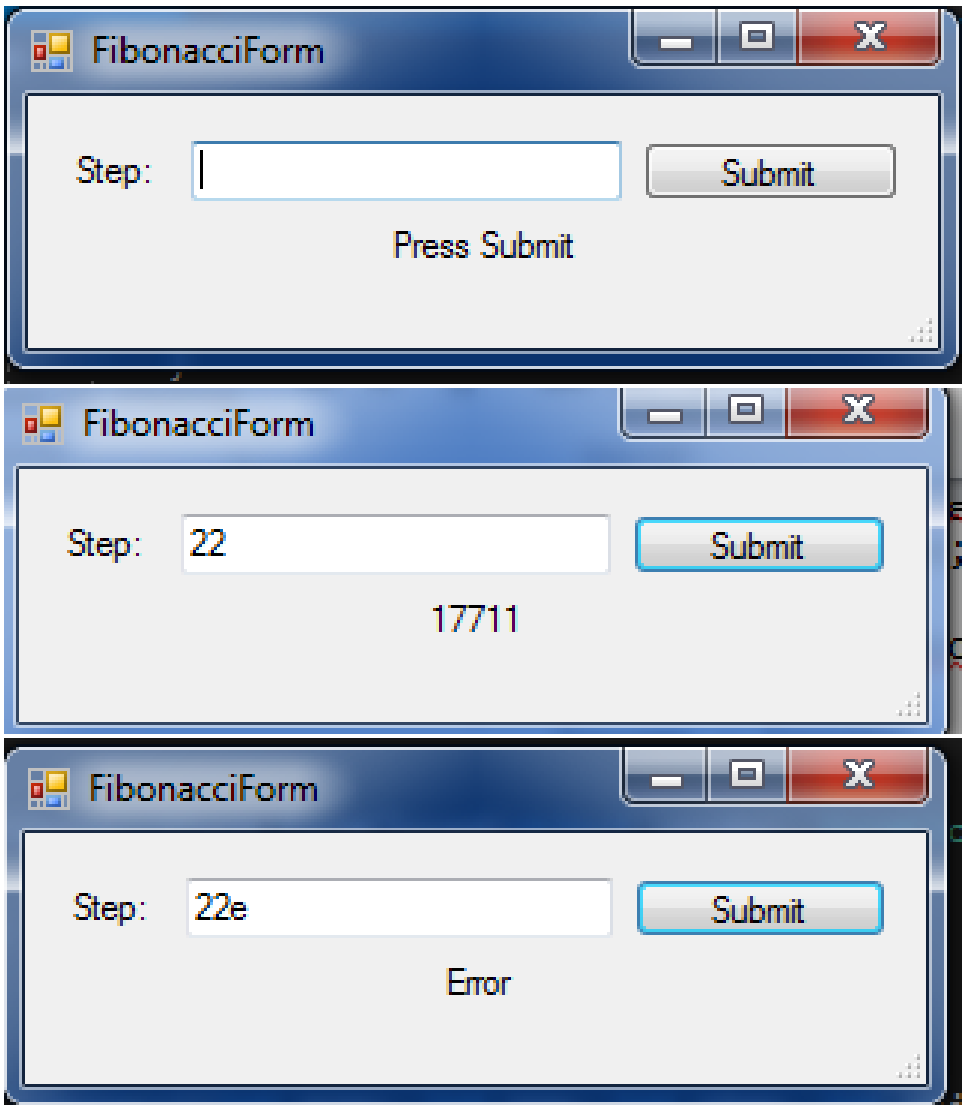
```
                routes.Add(rta.GetRoute());
            }
            return routes;
        }
        public void ChangeTransportType(TransportType transporttype)
        {
            TransportType = transporttype;
        }

        public override string ToString()
        {
            if (TransportType != null)
                return String.Format("{0} - {1} km. | ${2}", TransportType.ToString(),
Length.ToString(), (Income - Spending).ToString());
            return String.Format("No Type - {0} km. | ${1}", Length.ToString(), (Income -
Spending).ToString());
        }
        public void ChangeLength(int distance) {
            Length = distance;
        }
        public void ChangeIncome(decimal income)
        {
            Income = income;
        }
        public void ChangeSpending(decimal spending)
        {
            Spending = spending;
        }
```

У кожному класі реалізовано по 1 перевантаженому методу ToString()

Реалізація рекурсивного вирахування числа Фібоначчі на певному кроці:

```
private Int32 Fibonacci(int prev, int now, int step, int waitedstep)
        {
            if (step == waitedstep)
                return prev + now;
            return Fibonacci(now, prev + now, step + 1, waitedstep);
        }
        private Int32 Fibonacci(int waitedstep)
        {
            if (waitedstep < 0)
                return -1;
            if (waitedstep == 0)
                return 0;
            if (waitedstep <= 2)
                return 1;
            return Fibonacci(1, 1, 3, waitedstep);
        }
```

Результати роботи:



Висновок: Я покращив вміння працювати з методами класів і об’єктів, а також використовувати рекурсію для вирішення завдань.