Tweede deeltentamen **Imperatief programmeren**
Vrijdag 15 oktober 2010, 11.00-13.00 uur

- Schrijf op elk ingeleverd blad je naam. Schrijf op het eerste blad ook je studentnummer en het aantal ingeleverde bladen.
- De opgaven en de lijst met standaardfuncties mag je houden (behalve als je heel vroeg vertrekt).
- Het tentamen bestaat uit 4 opgaven. Elke opgave levert 10 punten op. Je cijfer is het totaal aantal punten gedeeld door 4. Als je een deel van een opgave niet weet, probeer dan toch zo veel mogelijk op te schrijven!

Veel succes!

---

1. *(2 punten per deelvraag)* Deze opgave bestaat uit een aantal tekstvragen. Houd het antwoord kort: een of twee zinnen per onderdeel kan al genoeg zijn.

   (a) Definieer een property `Getal` die de waarde van een private `int`-variabele `getal` uit dezelfde klasse opvraagt/wijzigt, maar daarbij garandeert dat de nieuwe waarde niet groter dan 100 wordt.

   (b) Een `string` is in feite gewoon een array van waarden van het type `char`. Geef twee redenen waarom het toch handig is dat er een klasse `string` bestaat.

   (c) Wat betekent het dat een object *immutable* is? Geef een voorbeeld van een type object dat immutable is.

   (d) Wat is de semantiek van een opdracht van de volgende vorm: `try` A `catch (`E`)` B ?

   (e) Er zijn twee manieren om een twee-dimensionale array te declareren:

   ```
   int [,] eerste;
   int [][] tweede;
   ```

   Wat is het verschil? In welke situatie is het handig om de `tweede` te gebruiken?

2. In de klasse `String` zitten onder andere de volgende methoden:

   ```
   static bool IsNullOrWhiteSpace(string)
   static int Compare(string, string)
   ```

   (a) *(4 punten)* De naam van `IsNullOrWhiteSpace` spreekt voor zichzelf. Onder 'whitespace' verstaan we spaties, tab-tekens en newline-tekens.

   Schrijf deze methode, *zonder* gebruik te maken van de bestaande `IsNullOrEmpty` en `IsNullOrWhiteSpace` methoden.

   (b) *(6 punten)* De methode `Compare` levert 0 op als de twee parameters precies gelijk zijn. Hij levert een negatief getal op (bijvoorbeeld -1, maar iets anders mag ook) als de eerste parameter kleiner is dan de tweede, en een positief getal (bijvoorbeeld 1) als die groter is. Met kleiner en groter wordt hier de woordenboek-ordening bedoeld: de eerste letter waar de strings verschillen bepaalt de ordening (volgens de Unicodes van die letters). Is de ene string een beginstuk van de andere, dan is de kortste de kleinste. Spaties en leestekens tellen gewoon mee, die hoeven dus niet speciaal behandeld te worden.

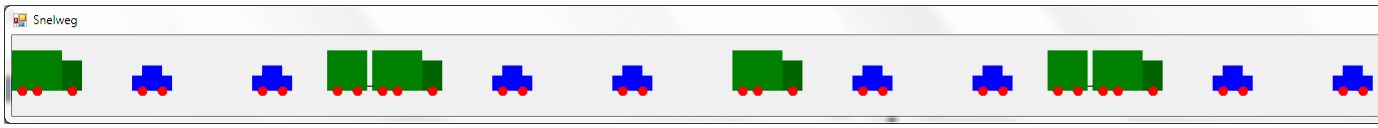
   Voorbeelden:

`String.Compare("aap", "noot")`	geeft een negatief getal, want `'a'<'n'`
`String.Compare("noot", "nieten")`	geeft een positief getal, want `'o'>'i'`
`String.Compare("niet", "nietmachine")`	geeft een negatief getal vanwege de lengte
`String.Compare("noot", "noot")`	geeft 0, want precies gelijk
`String.Compare("noot", "NOOT")`	geeft een positief getal, want `'n'>'N'`

   De methode neemt aan dat de parameters niet `null` zijn (en controleert dat ook niet).

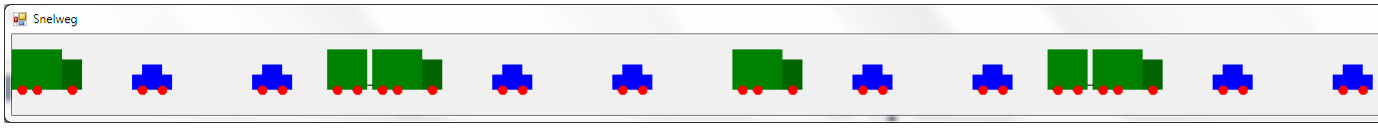
   Schrijf deze methode, *zonder* gebruik te maken van de bestaande `Compare` en `CompareTo` methoden.

3. Bekijk het onderstaande programma. Het moet een file auto's op de snelweg tekenen, zoals in de screendump hieronder. Elke derde auto is een vrachtwagen, en elke tweede vrachtwagen is een combinatie met aanhanger.




```
public class Snelweg : Form
{   public Snelweg()
    {   this.Text = "Snelweg";
        this.ClientSize = new Size(1800, 80);
        this.Paint += this.tekenSnelweg;
        // TODO: ontbrekend deel van de constructor
    }
    public void tekenSnelweg(object o, PaintEventArgs pea)
    {   for (int t = 0; t < rijbaan.Length; t++)
            rijbaan[t].Teken(pea.Graphics, t*120, 60);
    }
    static void Main()
    {   Application.Run(new Snelweg());
    }
}
class MotorVoertuig
{   public void Teken(Graphics gr, int x, int y)
    {
    }
}
class PersonenAuto : MotorVoertuig
{   public void Teken(Graphics gr, int x, int y)
    {   gr.FillRectangle(Brushes.Blue, x, y - 20, 40, 15);
        gr.FillRectangle(Brushes.Blue, x+10, y - 30, 20, 10);
        gr.FillEllipse(Brushes.Red, x + 5, y - 10, 10, 10);
        gr.FillEllipse(Brushes.Red, x + 25, y - 10, 10, 10);
    }
}
class Vrachtwagen : MotorVoertuig
{   public void Teken(Graphics gr, int x, int y)
    {   gr.FillRectangle(Brushes.Green, x, y - 45, 50, 40);
        gr.FillRectangle(Brushes.DarkGreen, x+50, y - 35, 20, 30);
        gr.FillEllipse(Brushes.Red, x + 5, y - 10, 10, 10);
        gr.FillEllipse(Brushes.Red, x + 20, y - 10, 10, 10);
        gr.FillEllipse(Brushes.Red, x + 55, y - 10, 10, 10);
    }
}
class Combinatie : Vrachtwagen
{   public void Teken(Graphics gr, int x, int y)
    {   // de vrachtwagen
        gr.FillRectangle(Brushes.Green, x, y - 45, 50, 40);
        gr.FillRectangle(Brushes.DarkGreen, x + 50, y - 35, 20, 30);
        gr.FillEllipse(Brushes.Red, x + 5, y - 10, 10, 10);
        gr.FillEllipse(Brushes.Red, x + 20, y - 10, 10, 10);
        gr.FillEllipse(Brushes.Red, x + 55, y - 10, 10, 10);
        // de aanhanger
        gr.DrawLine(Pens.Black, x - 5, y - 10, x, y - 10);
        gr.FillRectangle(Brushes.Green, x-45, y - 45, 40, 40);
        gr.FillEllipse(Brushes.Red, x -40, y - 10, 10, 10);
        gr.FillEllipse(Brushes.Red, x -20, y - 10, 10, 10);
    }
}
```

(zie volgende blad voor de opgaven)



(a) *(1 punt)* Er ontbreekt nog een declaratie. Schrijf deze declaratie, en geef aan waar die moet staan.

(b) *(2 punten)* Schrijf het ontbrekende deel van de constructor van `Snelweg`.

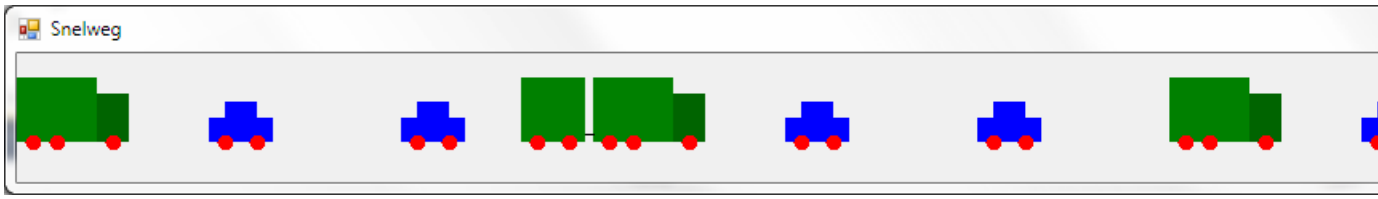
(c) *(2 punten)* In het gegeven programma zit nog een fout, waardoor er helemaal niets zichtbaar wordt. Hoe komt dat, en hoe kan de fout worden verbeterd?

(d) *(1 punt)* De programmeur heeft een flink stuk code met copy&paste gedupliceerd. Waarom is dat geen goed idee?

(e) *(2 punten)* Hoe had het dupliceren van de code het beste vermeden kunnen worden?

(f) *(2 punten)* We willen de object-georiënteerde opzet van het programma nog verder doorvoeren, zo dat ook de concepten 'wiel' en 'aanhanger' met klassen worden gemodelleerd. Hoe kan dat netjes worden aangepakt? Zorg ervoor dat code-duplicatie zo veel mogelijk vermeden kan worden, en dat er nooit door een programmeerfout een losse aanhanger op de weg terecht kan komen.

Je hoeft dit niet helemaal uit te programmeren; geef alleen aan welke klassen er komen, hoe hun subklasse-relatie is, en welke declaraties er in (bestaande en/of nieuwe) klassen komen te staan.



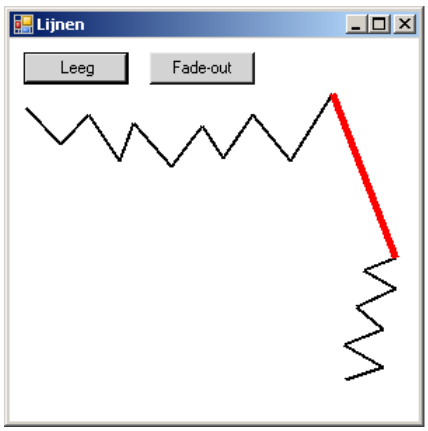
op de achterkant staat nog een opgave!

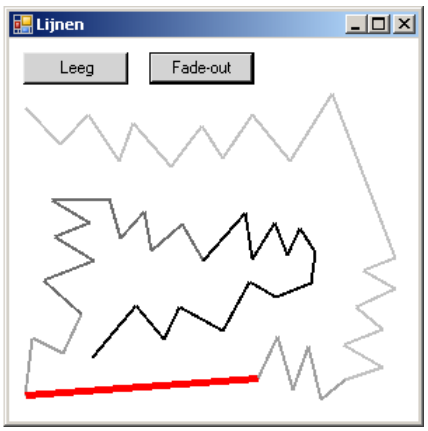
4. Gegeven is de volgende klasse, met daarin de methode `Main` en twee andere handige methoden.

```
public class Help
{   public static void Main()
    {   Lijnen lijnen = new Lijnen();
        lijnen.Text = "Lijnen";
        lijnen.BackColor = Color.White;
        Application.Run(lijnen);
    }
    public static Button MaakKnop(string tekst, int x, int y, EventHandler eh)
    {   Button b = new Button();
        b.Text = tekst;
        b.BackColor = Color.LightGray;
        b.Location = new Point(x, y);
        b.Click += eh;
        return b;
    }
    public static int Kwadraat(int x)
    {   return x * x;
    }
}
```

Schrijf nu de klasse `Lijnen`, zo dat het programma zich als volgt gaat gedragen:

- De gebruiker ziet twee knoppen met het opschrift 'Leeg' en 'Fade out'.
- De gebruiker kan verder overal in het window klikken. De aangeklikte punten worden verbonden door lijnen. (Na de eerste klik ziet de gebruiker nog niets, bij de tweede klik verschijnt er een lijn, bij de derde klik en tweede lijn, enz).
- Er zijn maximaal 100 lijnen zichtbaar. Als de gebruiker daarna toch meer punten aanklikt, gebeurt er niets (ook geen foutmelding!).
- Alle lijnen zijn zwart met dikte 2, behalve de langste lijn: die is rood met dikte 5. (Als er meerdere lijnen de langste zijn, mag je er daar een van kiezen, of ze allemaal rood maken).
- Na het indrukken van de knop 'Leeg' verdwijnen alle lijnen. De gebruiker kan dan weer met 100 nieuwe lijnen beginnen.
- Na het indrukken van de knop 'Fade out' beginnen de lijnen langzaam te vervagen: elke seconde worden ze 10% grijzer. Theoretisch gesproken worden ze dus nooit helemaal wit, maar in de praktijk zijn ze na een halve minuut onzichtbaar geworden op de witte achtergrond. Nieuw aangeklikte lijnen beginnen wel weer zwart. De langste lijn blijft rood.
- Na nogmaals indrukken van de knop 'Fade out' stopt het vervagen, na een derde keer indrukken gaat het weer verder waar het gebleven was, enz.





In het voorbeeld links heeft de gebruiker 19 punten aangeklikt. De langste lijn is dikker en rood.

In het voorbeeld rechts is de 'Fade out' knop gebruikt. Later gemaakte lijnen zijn nog donkerder. Inmiddels is ook een andere lijn de langste.

EINDE TENTAMEN