Uitwerking Aanvullende tentamen Imperatief programmeren
Woensdag 5 januari 2011, 8.30–10.30 uur

---

1. (a) Met een `Thread` object kun je een tekenfilmpje maken. Hoe doe je dat?

Antwoord:

Delcareer een Thread-object, en geef aan de constructor een methode mee die in een loop steeds een animatie-stap doet, het scherm invalideert, en eventjes pauzeert. Roep de `Start`-methode aan van de thread.

```
Thread t = new Thread(this.run);
t.Start();
// met de volgende methode run:
void run()
{   while (true)
    {   this.doeEenStap();
        scherm.Invalidate();
        Thread.Sleep(50);
    }
}
```

(b) Iemand schrijft de volgende opdracht om het gemiddelde van eerder uit een file ingelezen getallen op het label `lab` op het scherm te tonen:

```
lab.Text = "Gemiddelde: " + totaal / aantal;
```

In sommige situaties wordt het programma echter afgebroken met een foutmelding.

In plaats daarvan willen we liever dat de foutmelding op de label verschijnt.
Je kunt dit op twee manieren voor elkaar krijgen:

- Vooraf controleren of de foutsituatie zich zou gaan voordoen
- Als de foutsituatie optreedt deze afhandelen

Geef voor beide aanpakken aan hoe de opdracht er dan uit komt te zien.

Antwoord:

```
// aanpak 1: vooraf controleren
if (aantal==0)
     lab.Text = "geen getallen";
else lab.Text = "Gemiddelde: " + totaal / aantal;

// aanpak 2: foutsituatie afvangen
try
{   lab.Text = "Gemiddelde: " + totaal / aantal;
}
catch (Exception e)
{   lab.Text = "geen getallen";
}
```

(c) In C# kan een methode worden aangeroepen met het speciale object `base` voor de punt.
(In Java heet dat speciale object `super`).

- In welke situatie is dat zinvol? (geef een abstracte beschrijving van de situatie)
- Geef een voorbeeld van een praktijkgeval waar deze situatie zich voordoet.

Antwoord:

Als je in een subklasse een methode herdefinieert, en darbij de oorspronkelijke versie wilt gebruiken, kun je die aanroepen met `base` voor de punt.

Dit kan bijvoorbeeld gebruikt worden als de subklasse een extra geldigheids-check doet alvorens de oorspronkelijke methode aan te roepen, zoals de casus in het boek waar in een frequentie-analyse de methode `turf` een karakter turft, maar in de subklasse dat alleen gebeurt als de karakter gene onderdeel uitmaakt van een html-tag.

(d) Bij het schrijven van een tekstfile kun je kiezen uit een aantal verschillende *encodings*.
Mogelijk zijn onder andere Ascii, Latin1 (ook bekend als iso-8859-1), Unicode, en UTF8.

- Noem een voordeel en een nadeel van Latin1 in vergelijking met Unicode.
- Noem een voordeel en een nadeel van UTF8 in vergelijking met Unicode.

Antwoord:

Latin1 gebruikt slechts 1 byte per character tegen Unicode 2 bytes, maar Latin1 kan dan ook slechts 256 verschillende tekens weergeven, tegen Unicode 65536.

UTF8 gebruikt voor de eerste 128 characters slechts 1 byte, en kan westerse teksten dus compacter opslaan. Nadeel is dat de (de)codering ingewikkelder is, en dat chinese teksten juist 3 bytes per karakter kosten tegen 2 in Unicode. (UTF8 kan precies dezelfde 65536 characters representeren als Unicode, dus dat is geen voordeel of nadeel).

(e) Je kunt een object van het type `A` declareren en een waarde geven met `A a = new A()`.
Het maakt daarbij uit of `A` is gedefinieerd als `class A` of als `struct A`.

Wat is het verschil?

Antwoord:

Bij ene class wordt een *verwijzing* naar het object opgeslagen, bij een struct wordt het object zelf opgeslagen.

(f) Sommige arrays zijn *drie-dimensionaal*.
- Hoe ziet de declaratie van een drie-dimensionale array van strings er uit, en hoe kan daarbij het aantal elementen worden vastgelegd?
- Geef een voorbeeld van een praktijksituatie waarin deze array zinvol gebruikt kan worden.

Antwoord:

```
// declaratie
string [,,] woorden;
// aantal elementen vastleggen
woorden = new string[10,20,30]; // maakt een array van 10*20*30=6000 strings
```

Zo'n array zou je bijvoorbeeld kunnen gebruiken om in een reserveringssysteem voor elke dag van de week (1 t/m 7) voor elke uur van de dag (0 t/m 23) en voor elk van de drie beschikbare zalen, een gebruiker vast te leggen.

2. In onderstaand programma wordt een array gedeclareerd met daarin de uitslagen van een tentamen op een schaal van 0 tot en met 10:

```
using System.Windows.Forms;
using System.Drawing;

public class StaafDiagram : Form
{
    double [] cijfers
    = { 10, 8, 9, 7, 10, 6.5, 8, 7.5, 9, 7.5, 8, 8.5, 4, 6.5, 8.5, 8, 8, 8, 7, 8.5, 9, 8.5, 4, 5.5, 9, 9
      , 8.5, 9.5, 4, 9, 9, 8, 7.5, 6.5, 8, 7, 8, 9.5, 8, 8, 8, 6, 7.5, 9, 8, 9, 6, 8.5, 7, 9, 6, 8.5, 8.5
      , 9, 8, 6, 8, 7, 8.5, 7.5, 8, 7.5, 8, 8, 9, 9, 8.5, 8.5, 9.5, 9, 8.5, 0, 5.5, 8.5, 8, 7
      };

    public StaafDiagram()
    {   this.Text   = "Staafdiagram";  this.Size = new Size(500, 280);
        this.Paint += this.teken;
    }
    static void Main()
    {   Application.Run(new StaafDiagram());
    }
    // TODO: methode teken
}
```

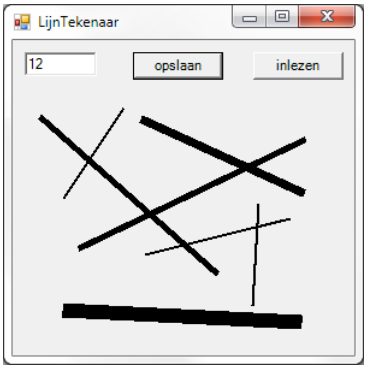
Schrijf de ontbrekende methode `teken` van dit programma, waarmee een staafdiagram van deze uitslag op het scherm wordt getekend, zoals in de afbeelding. De uitslagen worden daarbij naar beneden afgerond, dus bijvoorbeeld een 7.5 wordt meegeteld in de staaf behorend bij 7.
Het programma moet ook gebruikt kunnen worden als, later, de array wordt aangepast (waarna het programma opnieuw wordt gecompileerd).
Hint: het ankerpunt bij het tekenen van een tekst ligt linksboven. De staven liggen 20 beeldpunten uit elkaar en gebruiken een breedte van 10 beeldpunten voor elke uitslag.

Antwoord:

```
public void teken(object o, PaintEventArgs pea)
{
    Graphics g = pea.Graphics;

    int []turf = new int[11];
    int t;
    for (t = 0; t < cijfers.Length; t++)
        turf[(int)cijfers[t]]++;
    for (t = 0; t <= 10; t++)
    {
        g.FillRectangle(Brushes.Black, 50, 10+20 *t, 10 * turf[t], 18);
        g.DrawString(t.ToString(), new Font("Arial", 12), Brushes.Black, 10, 10+20*t);
    }
}
```

3. Bekijk het programma `LijnTekenaar`, waarvan hiernaast een screenshots staat. In het window zijn een tekstveld en twee buttons zichtbaar. De gebruiker kan met de muis steeds twee punten aanklikken, die dan verbonden worden door een lijn. Het aantal lijnen dat de gebruiker mag tekenen is niet aan een maximum gebonden.
De dikte van de lijn wordt gespecificeerd door het getal dat in het tekstveld staat ingevuld op het moment van de tweede klik. Je mag zonder controle aannemen dat in het tekstveld alleen maar cijfertekens zijn ingevuld.



Hieronder is al een deel van het programma gegeven. Vul hierop het volgende aan:

(a) Schrijf een hulpklasse `Lijn`, zo dat in een object van die klasse alle gegevens die nodig zijn om een lijn te kunnen tekenen beschikbaar zijn. Maak een constructormethode die zo'n object van beginwaardes voorziet, een methode `ToString` die de waarden 'inpakt' in een string, en een tweede constructormethode die zo'n string weer 'uitpakt'.

(b) Schrijf de methoden `muisklik` en `teken` in de klasse `LijnTekenaar`, en geef de declaraties van de daarvoor benodigde member-variabelen.

(c) Als de gebruiker op de knop 'opslaan' drukt, wordt de toestand van de tekening opgeslagen in een file waarvan de naam in de constante `naam` staat. Als de gebruiker op de knop 'inlezen' drukt, wordt de huidige tekening vervangen door de opgeslagen tekening. Schrijf de methoden `opslaan` en `inlezen` die daarvoor nodig zijn.

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Collections.Generic;
using System.IO;
```

```
namespace Opgave3
{   public class Program
    {   static void Main()
        {   Application.Run(new LijnTekenaar());
        }
    }
    public class Lijn
    {   // OPGAVE a
    }
    public class LijnTekenaar : Form
    {   const string naam = "tekening.txt";
        TextBox tb;

        public LijnTekenaar()
        {   this.Text = "LijnTekenaar";
            Button b1, b2;
            tb = new TextBox(); tb.Text="5";       tb.Location=new Point(10,10); tb.Size=new Size(60,30);
            b1 = new Button();  b1.Text="opslaan"; b1.Location=new Point(100,10);
            b2 = new Button();  b2.Text="inlezen"; b2.Location=new Point(200,10);
            this.Controls.Add(tb); this.Controls.Add(b1); this.Controls.Add(b2);
            this.MouseClick += this.muisklik;
            this.Paint      += this.teken;
            b1.Click        += this.opslaan;
            b2.Click        += this.inlezen;
        }
        // OPGAVE b en c
    }
}
```

Antwoord:

(a) De klasse `Lijn`:

```
public class Lijn
{
    public Point start, eind;
    public int dikte;

    public Lijn(Point p1, Point p2, int d)
    {   start = p1;
        eind  = p2;
        dikte = d;
    }
    public Lijn(string s)
    {   string[] v = s.Split();
        start = new Point( int.Parse(v[0]), int.Parse(v[1]));
        eind  = new Point( int.Parse(v[2]), int.Parse(v[3]));
        dikte = int.Parse(v[4]);
    }
    public override string ToString()
    {   return start.X + " " + start.Y + " " + eind.X + " " + eind.Y + " " + dikte;
    }
}
```

(b) De methoden `klik` en `teken` met membervariabelen:

```
List<Lijn> lijnen = new List<Lijn>();
Point punt;
int n = 0;

void klik(object o, MouseEventArgs mea)
{   if (n%2==0)
        punt = mea.Location;
    else
        lijnen.Add(new Lijn(punt, mea.Location, int.Parse(tb.Text)));
    n++;
    this.Invalidate();
```

```
}
void teken(object o, PaintEventArgs pea)
{   foreach (Lijn lijn in lijnen)
        pea.Graphics.DrawLine(new Pen(new SolidBrush(Color.Black), lijn.dikte), lijn.start, lijn.eind
}
```

(c) De methoden `opslaan` en `inlezen`:

```
void opslaan(object o, EventArgs ea)
{   StreamWriter w = new StreamWriter(naam);
    foreach (Lijn lijn in lijnen)
        w.WriteLine(lijn.ToString());
    w.Close();
}
void inlezen(object o, EventArgs ea)
{   lijnen.Clear();
    StreamReader r = new StreamReader(naam);
    string regel;
    while ((regel = r.ReadLine()) != null)
        lijnen.Add(new Lijn(regel));
    r.Close();
    this.Invalidate();
}
```