DERDE DEELTENTAMEN IMPERATIEF PROGRAMMEREN
VRIJDAG 8 NOVEMBER 2013, 8.30–10.30 UUR

- Schrijf op elk ingeleverd blad je naam.
  Schrijf op het eerste blad ook je studentnummer en het aantal ingeleverde bladen.
- De lijst met standaardfuncties na afloop graag weer inleveren.
  De opgaven mag je houden (behalve als je heel vroeg vertrekt).
- Het tentamen bestaat uit 3 opgaven. Opgave 1 telt voor 20% mee, opgave 2 en 3 elk voor 40%.
  Als je een deel van een opgave niet weet, probeer dan toch zo veel mogelijk op te schrijven!

Veel succes!

---

1. Voor elk punt (x, y) van het platte vlak, waarbij x en y reële getallen zijn, kan een bijbehorend getal worden bepaald – laten we dit het 'mandelgetal' noemen. Om het mandelgetal te kunnen uitrekenen, bekijken we eerst de volgende functie, die punten (a, b) van het vlak transformeert naar andere punten:

   $$f\ (a, b) = (a * a - b * b + x,\ \ 2 * a * b + y)$$

   Let op: deze functie transformeert het punt (a, b), maar in de berekening speelt ook de waarde van x en y, dat is het punt waarvan we het mandelgetal willen bepalen, een rol.

   Deze functie f nu, passen we toe op het punt $(a, b) = (0, 0)$. Op het punt dat daar uitkomt, passen we nog eens de functie f toe. Op het punt dat daar weer het resultaat van is, passen we opnieuw f toe, enzovoorts. We stoppen pas met toepassen van f als het resultaat-punt een afstand van meer dan 2 tot het punt $(0, 0)$ heeft. Het mandelgetal is nu gelijk aan het *aantal keren* dat f is toegepast.

   Voor sommige punten (x, y) is dat meteen al zo, en is het mandelgetal dus gelijk aan 1. Voor andere punten duurt het langer: die hebben een groter mandelgetal. Er zijn ook punten waarbij je f kan blijven toepassen, zonder dat de afstand tot de oorsprong ooit meer dan 2 wordt. Voor die punten stellen we het mandelgetal op 100.

   (a) Schrijf een methode `mandel` die het mandelgetal uitrekent van het punt waarvan de coordinaten als parameter worden meegegeven.

   (b) Schrijf het ontbrekende stuk van de methode `teken`, die de punten op het scherm zwart kleurt die een *oneven* mandelgetal hebben. De gedeclareerde `schaal` moet worden gebruikt zo dat het plaatje wordt getoond voor x en y tussen 0 en 4.

```
class Mandelbrot : Form
{
    double schaal = 0.01;

    // TODO opgave a: methode mandel

    public void teken(object obj, PaintEventArgs pea)
    {   Graphics gr = pea.Graphics;
        for (int x=0; x<400; x++)
        {   for (int y=0; y<400; y++)
            {    // TODO opgave b: body
            }
        }
    }
    public Mandelbrot
    {   this.Paint += this.teken;
    }
}
```

2. *Van onderstaande zes onderdelen mag je er één overslaan. De andere vijf tellen elk voor 2 punten.*
Geef duidelijk aan welke je overslaat.
Maak je ze toch alle zes, dan tellen ze elk voor 1.67 punten
(als je er eentje fout beantwoordt, had je hem dus beter kunnen overslaan!)

(a) Wat houdt het in als een programma een *MDI-interface* heeft?
Wat gebeurt er met de menu-balk in dat soort programma's?

(b) 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.

(c) Wat is het verschil tussen een *abstracte methode* en een *virtuele methode*?
Wat is er het voordeel van om een methode *abstract* te maken in plaats van *virtual*?

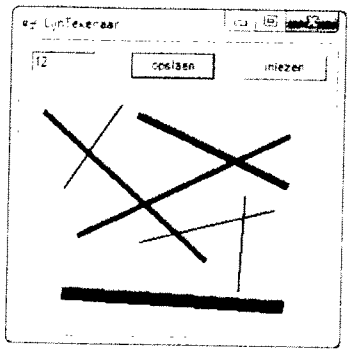
(d) Hoe ziet de body van een `interface`-declaratie eruit?
Stel dat er een `interface A` is gedeclareerd. Noem twee plaatsen in de syntax van een programma waar een interface-naam, zoals `A`, gebruikt kan worden.

(e) Wat is het essentiële verschil tussen een `FileStream` en een `SteamReader` ?
De klasse `FileStream` is een voorbeeld van een *store*. Daarnaast kennen we ook zogeheten *decorators*. Wat is het verschil tussen een store en een decorator?

(f) Beschrijf de syntax en de semantiek van een `foreach`-opdracht.
In welke situatie is het noodzakelijk om een `foreach`-opdracht te gebruiken en is het niet mogelijk om een `for`-opdracht met een tellertje te gebruiken?
In welke situatie is het noodzakelijk om een `for`-opdracht met een tellertje te gebruiken en is het niet mogelijk om een `foreach`-opdracht te gebruiken?

zie vervolgblad

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
    }
}
```