Uitwerking Derde deeltentamen Imperatief programmeren - versie 1
Vrijdag 11 november 2016, 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 antwoorden komen binnenkort op de website.
- Opgave 1 t/m 10 zijn meerkeuzevragen, die meetellen voor $10 \times 4 = 40$ punten.
  Opgave 11 en 12 zijn programmeer- en tekstvragen, die meetellen voor 20 en 40 punten.

Meerkeuzevragen: de letter van het goede antwoord volstaat.
Belangrijk: dit is versie **1** van het tentamen, vermeld dat boven je antwoorden.

1. De methode `Main` mag gedefinieerd worden met een parameter.
   Deze parameter moet dan het volgende type hebben:

   (a) `string`
   (b) • `string[]`
   (c) `List<string>`
   (d) `Stream`

   Toelichting op het antwoord: De parameter van `Main` bevat de op de commandoregel ingetikte woorden, elk woord in een apart element van een array van strings.

2. Een lambda-expressie kun je *niet* gebruiken

   (a) Als parameter van de constructor van `Thread`
   (b) Bij de registratie van een event-handler
   (c) Als parameter van een zelfgemaakte methode
   (d) Voor het opvangen van een exception

   Toelichting op het antwoord: Een lambda-expressie is een anonieme methode, die je kunt opschrijven in plats van de methode-naam als je die moet meegeven als parameter. Dit gebeurt o.a. in de constructor van `Thread`, maar kan ook in eigen methodes (zoals in het diktaat in de Bitmap-editor gebeurt bij aanroep van `combineer`), en rechts van de `+=` van event-handler registraties. Met exceptions heeft dit echter niets te maken.

3. Wat is een verschil tussen een *interface* en een *abstracte klasse* ?

   (a) Je kunt geen variabele declareren met een interface als type
   (b) Je kunt geen `new` object aanmaken van een interface
   (c) • Je kunt geen membervariabelen declareren in een interface
   (d) In een interface staan geen properties

Toelichting op het antwoord: Variabelen declareren kan wel (ook bij abstracte klassen trouwens, dus dat is sowieso geen verschil). Een `new` object aanmaken kan niet (wel van een implementatie van de interface, of van een subklasse van de abstracte klasse), maar dit is dus ook geen verschil. In een interface staan wel properties (zoals `Count` in `ICollection`), ook bij abstracte klasse kan dit trouwens. Membervariabelen declareren kan niet in een interface, dat doe je pas bij de implementatie. In een abstracte klasse kan dit wel, dus dit is een echt verschil.

4. Een abstracte methode

   (a) mag de membervariabelen niet gebruiken
   (b) kan niet overridden worden
   (c) • moet in een abstracte klasse staan
   (d) heeft geen opdrachten in zijn body

   Toelichting op het antwoord: Membervariabelen mag je niet gebruiken in een *static* methode; in een abstracte methode mag dit wel. Abstracte methoden zijn juist speciaal bedoeld om *wel* te overriden. Een abstracte methode heeft *helemaal geen* body, dus je kunt niet ervan spreken of daar opdrachten in staan. Als er abstracte methoden zijn, wordt daarmee de klasse ook abstract, dus dat was het goede antwoord.

5. Welk van onderstaande uitspraken is *niet* waar?

   (a) UTF8 kan dezelfde tekens opslaan als Unicode
   (b) UTF8 gebruikt een variabel aantal bytes per teken
   (c) • UTF8 kost altijd minder ruimte dan Unicode
   (d) UTF8 codeert tekens met code 0 t/m 127 hetzelfde als Latin1

   Toelichting op het antwoord: UTF8 kost soms (bij Westeuropese teksten) minder bytes dan Unicode, maar soms (bij Chinese teksten) juist meer. De andere uitspraken zijn *wel* waar.

6. Een verschil tussen een *list* en een *collection* is

   (a) • Een collection heeft geen indexer-property
   (b) In een collection kun je het aantal elementen niet bepalen
   (c) Een collection kun je niet met `foreach` doorlopen
   (d) In een collection zitten geen dubbele elementen

   Toelichting op het antwoord: De indexer-property indexeert met een *volgnummer*, en dat is specifiek voor List. Collection heeft wel een Count. Collection is een subinterface van Enumerable, en mag je dus gebruiken in `foreach`. Geen dubbele elementen zitten in een `Set`.

7. In de header van een foreach-opdracht waarmee je een `List<T>` doorloopt
   declareer je een variabele

   (a) van het type `List<T>`
   (b) • van het type `T`
   (c) van het type `IEnumerable<T>`
   (d) van het type `IEnumerator<T>`

   Toelichting op het antwoord: Kijk bijvoorbeeld naar de foreach-opdracht in opgave 12: je declareert een variabele met als type het element-type van de lijst.

8. In een MDI-programma...

   (a) wordt het menu aangemaakt in de klasse van de container,
   maar wordt een deel alleen getoond als er een child-window aktief is

   (b) wordt het menu aangemaakt in de klasse van het child,
   maar wordt dit getoond in het container-window

   (c) worden menu's aangemaakt in de klasse van de container én die van het kind,
   maar worden ze gecombineerd getoond in het aktieve child-window

   (d) • worden menu's aangemaakt in de klasse van de container én die van het kind,
   maar worden ze gecombineerd getoond in het container-window

9. Is het mogelijk om in een klasse onder andere een member-variabele te declareren met diezelfde klasse als type?

   (a) Nee, een object van dit type zou oneindig veel geheugen vragen

   (b) Nee, ophalen van de waarde van deze member zou oneindig veel tijd vragen

   (c) Ja, maar alleen als je zorgt dat deze variabele niet verwijst naar
   het object waar het deel van is

   (d) • Ja, dit kan altijd

10. Met de opdracht `Console.Out.WriteLine("hallo");` wordt aangeroepen:

    (a) de static methode `WriteLine` in de klasse `Console.Out`

    (b) • de niet-static methode `WriteLine` in de klasse `TextWriter`

    (c) de static methode `WriteLine` in de klasse `Console`

    (d) de niet-static methode `WriteLine` in de klasse `string`

    Toelichting op het antwoord: `Out` is een static property van de klasse `Console`. Deze property is een object van het type `TextWriter`. Dit object wordt onder handen genomen door de methode `WriteLine`, die dus in de klasse `TextWriter` moet staan.

11. *(telt voor 20%)*

    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 na de eerste keer 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. Die punten hebben een oneindig groot mandelgetal.

(a) Schrijf een methode `mandel` die het mandelgetal oplevert van het punt waarvan de coordinaten als parameter worden meegegeven, tenzij het mandelgetal groter dan 100 is of oneindig: in dat geval levert de methode 100 als resultaat op.

(b) Schrijf het ontbrekende stuk van de methode `teken`, die de punten op het scherm zwart kleurt die een *even* 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;
    }
}
```

Antwoord: (Zie de uitwerking van de eerste practicumopgave).

12. *(telt voor 40%: onderdeel a t/m e elk voor 4%, f voor 8%, en g en h voor 6%)*

Bekijk het gegeven programma op pagina 5 (de klasse `Sterrenhemel` en pagina 6 (de klassen `Ster` en `Program`). Onderaan deze pagina staat een screenshot van het programma in werking.

De gebruiker kan plaatsen aanklikken in een window. Gecentreerd op die plaatsen verschijnt een ster met een oneven aantal uiteinden: 3, 5 enzovoorts tot maximaal 25. Het aantal uiteinden van een ster wordt bepaald door de stand van een schuifregelaar op het moment van klikken. Het aantal sterren is praktisch onbegrensd.

Als de gebruiker op de button 'Leeg' drukt, verdwijnen alle sterren en kan hij/zij opnieuw beginnen. Als de gebruiker op de button 'Opslaan' drukt, wordt de huidige situatie opgeslagen in een tekstbestand. Als de gebruiker op de button 'Inlezen' drukt, verdwijnt het huidige plaatje, en wordt de situatie hersteld van de laatste keer 'Opslaan'.

(a) In de klasse `Ster` worden de eigenschappen van een ster gemodelleerd: de positie van het midden en het aantal uiteinden (`tips`). De methode `LaatZien` laat zo'n ster zien op een graphics. Leg uit hoe de toekenningen aan `a` en `b` bijdragen in de vorming van de ster. (Je hoeft de wiskunde in de regels er boven niet uit te leggen).

Antwoord: de waarden van `a` en `b` zijn de volgnummers van de tips die verbonden moeten worden. In zon ster is dat steeds de bijna tegenoverliggende tip. Daarom wordt `b` de naar beneden afgeronde `tips/2`. In de loop schuift het steeds door: `a` wordt de oude `b`, de volgende `b` wordt vermeerderd met `tips/2`. Omdat de ster rond is, moet de nummering modulo `tips` worden gedaan.

(b) In de methode `MaakKnop` staan *twee* toekenningen aan `b.Click`. Leg uit waarom dat handig is in een programma zoals dit (en eventuele toekomstige uitbreidingen ervan).

Antwoord: Elke knop dit met `MaakKnop` wordt gemaakt kan nu een eigen eventhandler krijgen, maar zal daarnaast ook altijd automatisch `Invalidate` aanroepen. Dat hoeft dus niet meer in de aparte eventhandler van elke knop te gebeuren.

(c) Geef de drie ontbrekende declaraties in de klasse `Sterrenhemel`, compleet met hun initialisaties.

Antwoord:

```
List<Ster> sterren = new List<Ster>();
TrackBar aantal = new TrackBar();
TrackBar kleur = new TrackBar();
```

(d) Schrijf het ontbrekende opdracht aan het eind de constructormethode van `Sterrenhemel`, *zonder* daarbij nieuwe methodes in de klasse te definiëren.

Antwoord: Dit kan mooi met een lambda-expressie, vooral ook omdat na Clear geen Invalidate meer nodig is (zie opgave b):

```
this.Controls.Add(MaakKnop("Leeg", 210, (object o, EventArgs ea) => sterren.Clear()));
```

of, omdat je in het parameter-gedeelte van de lambda-expressie de types mag weglaten:

```
this.Controls.Add(MaakKnop("Leeg", 210, (o,ea) => sterren.Clear()));
```

(e) Schrijf de ontbrekende methode in de klasse `Ster`.

Antwoord:

```
public override string ToString()
{   return plek.X + " " + plek.Y + " " + tips;
}
```

We willen nu dat de gebruiker behalve sterren ook zonnen kan tekenen, door met de *rechter* muisknop te klikken. Een zon ziet er bijna hetzelfde uit als een ster, maar het verschil is dat de stralen van de ster nu een rood/oranje/gele cirkel als achtergrond krijgen. De kleur van de zon wordt bepaald door de tweede schuifregelaar: naar links is roder, naar rechts is geler.

(f) Schrijf een klasse `Zon`. Vermijd hierbij zo veel mogelijk overbodig schrijfwerk.

Antwoord:

```
public class Zon : Ster
{
    int kleur;

    public Zon(Point p0, int t0, int k0)
    {   plek = p0;
        tips = t0;
        kleur = k0;
    }
    public override string ToString()
    {   return base.ToString() + " " + kleur;
    }
    public override void LaatZien(Graphics g)
    {   Brush br = new SolidBrush(Color.FromArgb(255, kleur, 0));
        g.FillEllipse(br, plek.X - straal, plek.Y - straal, 2 * straal, 2 * straal);
        base.LaatZien(g);
    }
}
```

Deze alternatieve versie van de constructor is eigenlijk mooier, maar de benodigde notatie voor het aanroepen van de constructor van de base class is niet besproken in het college:

```
    public Zon(Point p0, int t0, int k0) : base(p0,t0)
    {   kleur = k0;
    }
```

(g) Schrijf de methode `MuisKlik` in de klasse `Sterrenhemel`.

Antwoord:

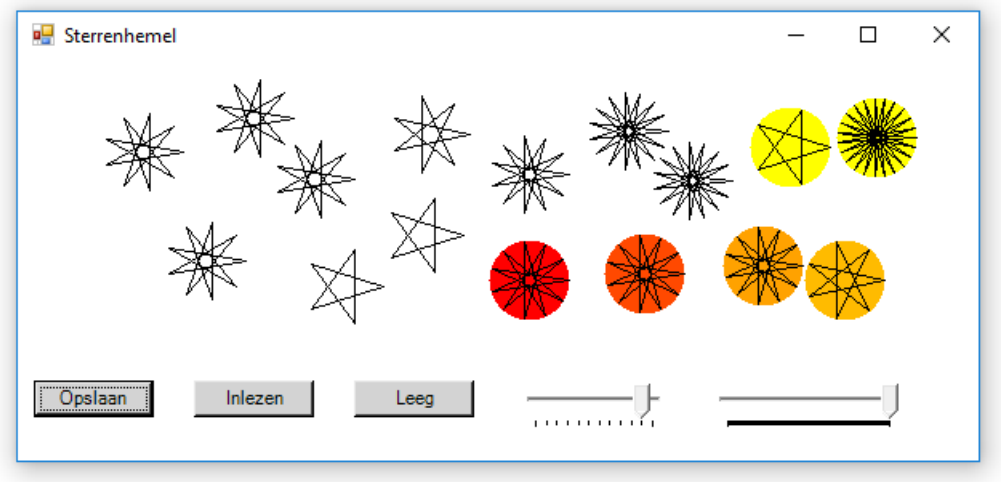
```
public void MuisKlik(object o, MouseEventArgs mea)
{
    int a = aantal.Value * 2 + 1;
```

```
        if (mea.Button == MouseButtons.Right)
            sterren.Add(new Zon(mea.Location, a, kleur.Value));
        else
            sterren.Add(new Ster(mea.Location, a));
        this.Invalidate();
    }
```

(h) Schrijf de methode `Inlezen` in de klasse `Sterrenhemel`.

Antwoord:

```
public void Inlezen(object o, EventArgs ea)
{   sterren.Clear();
    StreamReader r = new StreamReader(filenaam);
    string s;
    while ((s=r.ReadLine())!=null)
    {
        string[] a = s.Split();
        Point p = new Point(int.Parse(a[0]), int.Parse(a[1]));
        int t = int.Parse(a[2]);
        if (a.Length == 4) sterren.Add( new Zon(p, t, int.Parse(a[3])));
        else sterren.Add(new Ster(p, t));
    }
    r.Close();
}
```



Links in het voorbeeld-plaatje zie je vier 9-puntige sterren. Daarnaast heeft de gebruiker ook 5-, 7-, 11- en 15-puntige sterren gemaakt, en zes zonnen met verschillende kleur en aantal uiteinden.

```
// Bijlage bij opgave 12
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;

public class Sterrenhemel : Form
{
    // TODO opgavr c: declaraties

    private Button MaakKnop(string s, int x, EventHandler eh)
    {   Button b = new Button();
        b.Text      = s;
        b.BackColor = Color.LightGray;
        b.Location  = new Point(x, 200);
        b.Click    += eh;
        b.Click    += Herstellen;
        return b;
    }

    public Sterrenhemel()
    {   this.ClientSize = new Size(600, 250);
        this.Text = "Sterrenhemel";   this.BackColor = Color.White;
        this.Paint      += this.Teken;
        this.MouseClick += this.MuisKlik;

        aantal.Location = new Point(310, 200);
        aantal.Size = new Size(100, 20);
        aantal.Minimum = 1; aantal.Maximum = 12; aantal.Value = 4;
        kleur.Location = new Point(430, 200);
        kleur.Size = new Size(128, 20);
        kleur.Maximum = 255; kleur.Value = 128;

        this.Controls.Add(aantal);
        this.Controls.Add(kleur);
        this.Controls.Add(MaakKnop("Opslaan",  10, this.Opslaan));
        this.Controls.Add(MaakKnop("Inlezen", 110, this.Inlezen));
        // TODO opgave d: derde knop
    }

    public void Teken(object o, PaintEventArgs pea)
    {   foreach (Ster s in sterren)
            s.LaatZien(pea.Graphics);
    }
    public void Herstellen(object o, EventArgs e)
    {   this.Invalidate();
    }
    // TODO opgave g: methode MuisKlik

    const string filenaam = "sterren.txt";

    public void Opslaan(object o, EventArgs ea)
    {   StreamWriter w = new StreamWriter(filenaam);
        foreach (Ster s in sterren)
            w.WriteLine(s.ToString());
        w.Close();
    }
    // TODO opgave h: methode Inlezen
}
```

```
// vervolg bijlage bij opgave 12

public class Ster
{
    protected const int straal = 25;
    protected Point plek;
    protected int tips;

    public Ster()
    {
    }

    public Ster(Point p0, int t0)
    {   plek = p0;
        tips = t0;
    }

    // TODO opgave e: ontbrekende methode

    public virtual void LaatZien(Graphics g)
    {   int a = 0;
        int b = tips / 2;

        for (int t = 0; t < tips; t++)
        {   int dx1 = (int)(Math.Cos(2*Math.PI*a/tips)*straal);
            int dy1 = (int)(Math.Sin(2*Math.PI*a/tips)*straal);
            int dx2 = (int)(Math.Cos(2*Math.PI*b/tips)*straal);
            int dy2 = (int)(Math.Sin(2*Math.PI*b/tips)*straal);

            g.DrawLine(Pens.Black, plek.X+dx1, plek.Y+dy1, plek.X+dx2, plek.Y+dy2);
            a = b;
            b = (b + tips / 2) % tips;
        }
    }
}

class Program
{   static void Main()
    {   Application.Run(new Sterrenhemel());
    }
}

// TODO opgave f: klasse Zon
```