Uitwerking Eerste deeltentamen Imperatief programmeren
Vrijdag 30 september 2011, 11.00-13.00 uur

---

1. Deze opgave bestaat uit een aantal tekstvragen.
   Houd het antwoord kort: een of twee zinnen per onderdeel kan al genoeg zijn.

   (a) Wat is een *methode*?

   Wat is er bijzonder aan een methode die `static` is gedefinieerd?

   Antwoord: Een methode is een groepje opdrachten met een naam. Een `static` methode heeft *geen* object onder handen.

   (b) Beschrijf de syntax van de return-opdracht.

   Beschrijf daarna ook de semantiek van de return-opdracht.

   Antwoord: Syntax: een return-opdracht bestaat uit het woord `return`, gevolgd door een expressie, en tenslotte een puntkomma.

   Semantiek: als een return-opdracht in de body van een methode wordt uitgevoerd, wordt de waarde van de expressie teruggegeven aan degene die de methode heeft aangeroepen. Uitvoering van de methode is hiermee beëindigd.

   (c) Variabelen kunnen onder andere worden gedeclareerd in de *header* van een methode, en in de *body* van een methode.

   Hoe krijgt een variabele die is gedeclareerd in de *header* van een methode zijn waarde, en waar wordt zo'n variabele voor gebruikt?

   En hoe krijgt een variabele die is gedeclareerd in de *body* van een methode zijn waarde, en waar wordt zo'n variabele voor gebruikt?

   Antwoord: Een variabele in de header van een methode krijgt zijn waarde door de expressie op de corresponderende plaats in de aanroep van de methode. Hij wordt gebruikt om extra informatie door te geven aan de methode.

   Een variabele in de body van een methode krijgt zijn waarde door een toekenningsopdracht in die body. Hij wordt gebruikt voor de opslag van tussenresultaten die alleen tijdens het uitvoeren van de methode van belang zijn.

   (d) Wat is de betekenis van de operator `%` ?

   Geef een expressie die voor positieve getallen `x` en `y` dezelfde waarde heeft als `x%y`, zonder daarbij de operator `%` te gebruiken.

   Antwoord: De operator `%` geeft de rest bij deling.

   Een equivalente expressie voor `x%y` is `x-(x/y)*y`.

   (e) In een programma heeft iemand gedeclareerd `Label uitvoer;` en er voor gezorgd dat deze op het scherm zichtbaar is.
   Geef de opdracht die nodig is om op deze label de twee-regelige tekst

   ```
   "to be or not to be",
   zei Hamlet.
   ```

   te laten zien.

   Antwoord:

   ```
   uitvoer.Text = "\"to be or not to be\",\nzei Hamlet.";
   ```

2. Hieronder staat 16 fragmenten uit een programma. Schrijf op je antwoordblad een blok van 4 bij 4 vakjes en zet in elk vakje een letter passend bij het overeenkomstige fragment:

   - **T** als het programmafragment een **type** is
   - **E** als het programmafragment een **expressie**
   - **O** als het programmafragment een **opdracht** is
   - **D** als het programmafragment een **declaratie** is
   - **H** als het programmafragment een **methode-header** is
   - **X** als het programmafragment geen van bovenstaande dingen is

`Brush blue;`	`Brushes blue;`	`void a(Brush b)`	`Brush`
`(int) 1.2`	`1.2=x`	`return(1.2);`	`const float x=1.2;`
`{int x;}`	`x<0`	`while {x<0} x++;`	`true`
`using System;`	`Console.WriteLine("");`	`Console.ReadLine()`	`string s()`

Antwoord:

D E H T
E X O D
O E X E
X O E H

Opgave 3 en 4 vragen een stukje programma. Kleine schrijffoutjes (hoofdletters, puntkomma's enz.) worden niet streng afgerekend, maar de elementen die de structuur van het programma bepalen (haakjes, accolades, aanhalingstekens enz.) zijn wel belangrijk. Schrijf die dus duidelijk en op de goede plaats op! Het is toegestaan (maar niet nodig) om C#-constructies die (nog) niet zijn behandeld toch te gebruiken. Je hoeft niet aan te geven welke `using`*-directieven nodig zijn om de klassen te kunnen gebruiken.*

3. Alle methodes in deze opgave maken deel uit van de klasse `Prog`.

   (a) De 'faculteit' van een natuurlijk getal is de uitkomst van alle getallen vanaf 1 tot en met dat getal met elkaar vermenigvuldigd. Bijvoorbeeld: de faculteit van 3 is $1 \times 2 \times 3 = 6$. Schrijf een statische methode `faculteit` die de faculteit van zijn parameter uitrekent. Je mag er zonder controle van uitgaan dat de parameter ≥ 1 is.

   Antwoord:

```
static int faculteit(int n)
{
        int t, res;
        res = 1;
        for (t=1; t<=n; t++)
            res *= t;
        return res;
}
```

   (b) Een benadering van 'sinus hyperbolicus' van een reel getal x kun je berekenen door:

$$x + x^3/3! + x^5/5! + x^7/7! + x^9/9! + x^{11}/11! + \cdots$$

   De notatie 5! betekent hierin de faculteit van 5. Schrijf een statische methode `sinhyp` die deze benadering berekent door 20 van deze termen bij elkaar op te tellen, en dat als resultaat oplevert. Je mag (maar hoeft niet) zelf extra hulp-methoden definiëren. (Het is niet de bedoeling om alle termen helemaal uit te schrijven!)

   Antwoord:

```
static double sinhyp(double x)
{
        int t; double res;
        res = 0;
        for (t=1; t<40; t+=2)
                res += Math.Pow(x,t) / Prog.faculteit(t);
        return res;
}
```

4. Gegeven is de volgende klasse:

```
class Program
{   public static void Main()
    {   Roos r = new Roos();
        r.Text = "Roos";
        Application.Run(r);
    }
}
```

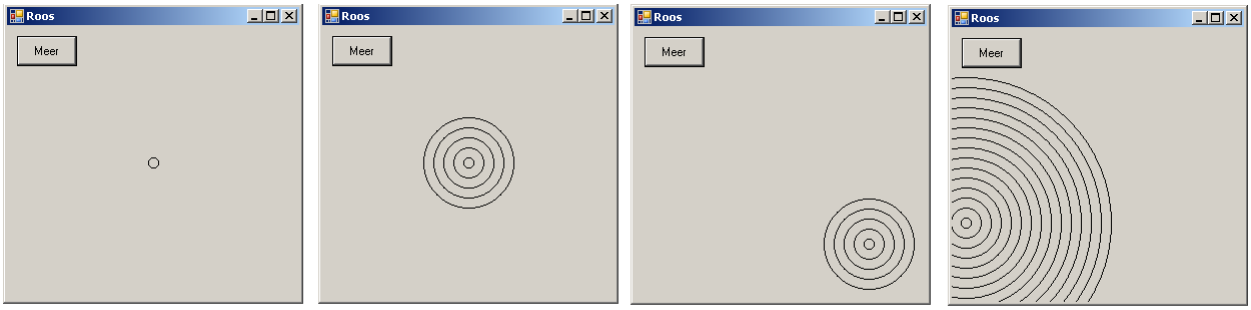
Schrijf de klasse `Roos`, zo dat het programma zich als volgt gaat gedragen.

In het midden van het scherm is een cirkel met een diameter van 10 pixels te zien. Bovenin het scherm is een knop met het opschrift 'Meer' aanwezig (zie plaatje 1).

Elke keer dat de gebruiker op de knop drukt verschijnt er een extra cirkel om de vorige heen. De afstand tussen de lijnen van opeenvolgende cirkels in 10 pixels. (In plaatje 2 is er 4 keer op de knop gedrukt).

Als de gebruiker met de muis ergens in het window klikt, wordt dat het nieuwe middelpunt van de cirkels. (In plaatje 3 heeft de gebruiker rechtsonder in het window geklikt).

De gebruiker kan steeds opnieuw op de knop drukken en in het window klikken om meer cirkels te maken en die te verplaatsen (zoals in plaatje 4).



Antwoord:

```
public  class Roos : Form
{
    int n=1, x=0, y=0;

    public Roos()
    {   Button b = new Button();
        b.Text = "Meer";
        b.Location = new Point(10, 10);
        b.Size = new Size(60, 30);
        this.Controls.Add(b);

        this.Paint += teken;
        this.MouseClick += klik;
        b.Click += meer;

        x = this.ClientSize.Width / 2;
        y = this.ClientSize.Height / 2;
    }

    public void teken(object o, PaintEventArgs pea)
    {   for (int t = 0; t < n; t++)
        {   int r = 5 + 10 * t;
            pea.Graphics.DrawEllipse( Pens.Black, x-r, y-r, 2*r, 2*r);
        }
    }
    public void klik(object o, MouseEventArgs mea)
    {   x = mea.X;
        y = mea.Y;
        this.Invalidate();
    }
    public void meer(object o, EventArgs ea)
    {   n++;
        this.Invalidate();
    }
}
```