Uitwerking Eerste deeltentamen Imperatief programmeren
Vrijdag 30 september 2016, 16.00–18.00 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 er in een `static` methode anders dan in methoden die dat niet zijn?
   Wat is er in een `void` methode anders dan in methoden die dat niet zijn?
   Antwoord: Een `static` methode heeft geen object onder handen. Of: een `static` methode mag niet `this` gebruiken in zijn body.
   Een void-methode geeft geen resultaat terug aan de aanroeper. Of: een void-methode hoeft geen return-opdracht te bevatten.

   (b) Beschrijf de syntax van de aanroep van een static void-methode met één parameter.
   Beschrijf daarna ook de semantiek daarvan.
   Antwoord: Syntax: de naam van de klasse waarin de methode staat, een punt, de naam van de methode, haakje openen, expressie, haakje sluiten, puntkomma.
   Semantiek: bepaal de waarde van de expressie, ken deze waarde toe aan de variabele die in de header van de methode staat gedeclareerd, en voer de opdrachten in de body van de methode uit.

   (c) Geef één opdracht om in een console-application de twee-regelige tekst

   ```
   Cesar zei:
   "veni vidi vici!"
   ```

   te laten zien (inclusief de leestekens).
   Antwoord:

   ```
   Console.WriteLine( "Cesar zei:\n\"veni vidi vici!\"" );
   ```

   (d) Waaraan herken je de aanroep van een *constructormethode* ?
   Wat gebeurt er bij de aanroep van een constructormethode meer dan bij de aanroep van een gewone methode?
   Antwoord: Bij aanroep van een constructormethode schrijf je `new` voor de naam van de methode.
   Voordat de opdrachten in de body worden uitgevoerd, wordt er eeerst een nieuw object aangemaakt.

   (e) Welke twee rollen heeft het begrip *klasse* in C# ?
   Hoe hangen deze twee rollen samen?
   Antwoord: Een klasse is een groepje methoden, maar ook het type van een object.
   Methoden uit de klasse nemen een object onder handen die diezelfde klasse als type heeft.

2. Hieronder staan 16 fragmenten uit een programma. Maak 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

```
true        while(false);    Graphics      string s = "//";

t+1=t;      t!=t             t*=1;         static s(int i)

float       return int;      (int)1.2      float f()

Pen p;      Pens.Red         { }           g.DrawLine(p,0,0,0,0);
```

   Antwoord:

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

3. In de wiskunde wordt vaak de *exponentiële functie*, of kortweg `exp` gebruikt. Deze kan worden berekend door:

$$exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots$$

   Hierin betekent 4! de *faculteit* van 4, dat is alle gehele getallen van 1 t/m 4 met elkaar vermenigvuldigd.

   Schrijf een methode `exp` die een benadering van deze functie geeft door 20 termen van deze reeks op te tellen. Het is hierbij niet toegestaan om de methoden `Math.Exp` of `Math.Pow` of de operator `^` te gebruiken. Het is ook niet toegestaan om de 20 termen helemaal uit te schrijven. Wel mag je zelf extra hulp-methoden schrijven en aanroepen.

   Antwoord: De kortste (en efficiëntste) manier is om de achtereenvolgende waardes van de macht en de faculteit uit te rekenen terwijl je bezig bent om de waarden te sommeren:

```
static double exp(double x)
{
    double res = 1, macht = 1, fac = 1;
    for (int t = 1; t < 20; t++)
    {
        macht *= x;
        fac *= t;
        res += macht / fac;
    }
    return res;
}
```

Maar een meer voor de hand liggende aanpak is om eerst hulp-methoden te schrijven voor de berekening van macht en faculteit (zoals op het werkcollege), en die vervolgens aan te roepen:

```
static double macht(double x, int n)
{
    double res = 1;
    for (int t = 0; t < n; t++)
        res *= x;
    return res;
}
static double fac(int n)
{
    double res = 1;
    for (int t = 1; t <= n; t++)
        res *= t;
    return res;
}
static double exp(double x)
{
    double res = 0;
    for (int t = 0; t < 20; t++)
        res += macht(x, t) / fac(t);
    return res;
}
```

4. Gegeven is de volgende klasse:

```
class Program
{   public static void Main()
    {   Kralen t = new Kralen();
        t.Text = "Kralen";
        t.ClientSize = new Size(500, 500);
        Application.Run(t);
    }
}
```

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

Er zijn twee kralen (gekleurde cirkels met een doorsnede van 30 beeldpunten) te zien:

- een rode, die gecentreerd is op de linker bovenhoek van het window (en die je dus maar voor een kwart ziet)
- een blauwe, die gecentreerd is op de positie van de muis

De twee kralen zijn verbonden door een lijn.

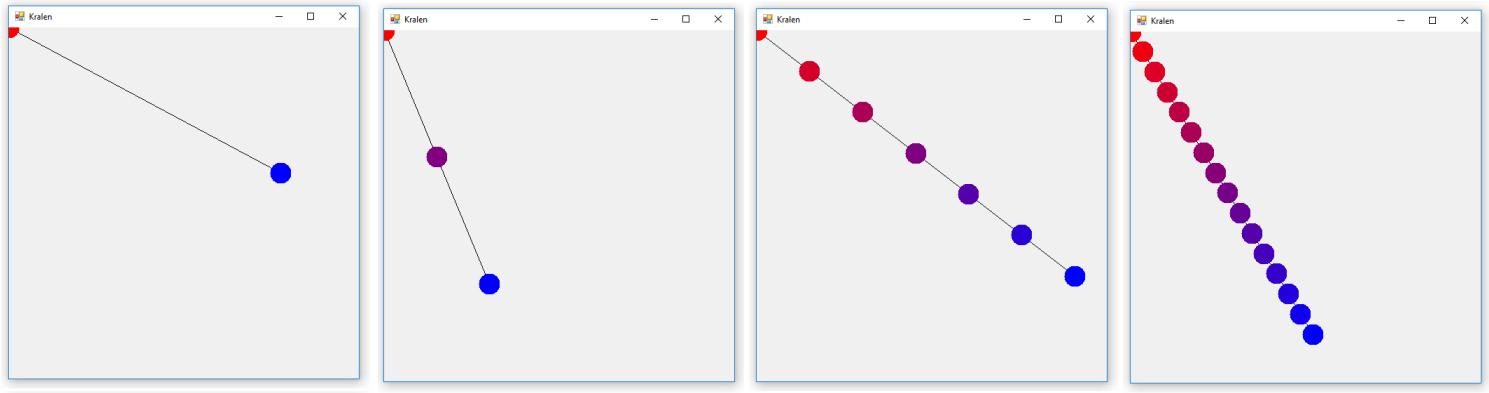
Als de muis beweegt, beweegt de blauwe kraal mee. De kralen blijven verbonden door de lijn.

Elke keer als de gebruiker met de muis klikt, onstaat er een nieuwe kraal op de lijn. De kralen verdelen de ruimte op de lijn gelijkmatig.

De kleur van elke kraal op de lijn is verschillend: ze verkleuren langzaam van rood (in de bovenhoek) naar blauw (bij de muis). Een kraal in het midden bijvoorbeeld is dus paars.

Als de gebruiker heel vaak klikt, gaan de kralen overlappen. Dat is niet erg.

De illustraties hieronder tonen achtereenvolgens: de startsituatie, na 1 keer klikken, na 5 keer klikken, en na 14 keer klikken, waarbij ook de muis steeds ergens anders staat.



Antwoord:

```
class Kralen : Form
{
    int x, y, n = 1;

    public Kralen()
    {
        this.Paint += this.teken;
        this.MouseMove += muis;
        this.MouseClick += klik;
    }
    void muis(object obj, MouseEventArgs mea)
    {
        this.x = mea.X;
        this.y = mea.Y;
        this.Invalidate();
    }
    void klik(object obj, MouseEventArgs mea)
    {
        this.n = this.n + 1;
        this.Invalidate();
    }
    void teken(object obj, PaintEventArgs pea)
    {
        pea.Graphics.DrawLine(Pens.Black, 0, 0, x, y);

        int t = 0;
        while (t<=n)
        {
            int cx = x * t / n;
            int cy = y * t / n;
            int k = 255 * t / n;
            Brush b = new SolidBrush(Color.FromArgb(255 - k, 0, k));
            pea.Graphics.FillEllipse(b, cx - 15, cy - 15, 30, 30);
            t = t + 1;
        }
    }
}
```