Eerste deeltentamen Imperatief programmeren
Woensdag 29 september 2010, 13.00-15.00 uur

- Schrijf op elk ingeleverd blad je naam. Schrijf op het eerste blad ook je studentnummer en het aantal ingeleverde bladen.
- De opgaven mag je houden (behalve als je heel vroeg vertrekt).
- Het tentamen bestaat uit 4 opgaven. Elke opgave levert 10 punten op. Je cijfer is het totaal aantal punten gedeeld door 4. Als je een deel van een opgave niet weet, probeer dan toch zo veel mogelijk op te schrijven!

Veel succes!

---

1. *(2 punten per deelvraag)* Deze opgave bestaat uit een aantal tekstvragen. Houd het antwoord kort: een of twee zinnen per onderdeel kan al genoeg zijn.

   (a) Wat wordt er in een C#-methode aangeduid met `this`? In welke situatie is het niet toegestaan om `this` te gebruiken?

   Antwoord: `this` staat voor het object dat momenteel onder handen genomen wordt. We mogen `this` niet gebruiken in static methoden omdat die geen object onder handen hebben.

   (b) Wanneer is een programmeertaal een *imperatieve* programmeertaal?

   Antwoord: Eem programmeertaal noemen we *imperatief* als de taal gebaseerd is op opdrachten om het geheugen te veranderen.

   (c) De vertaling van een C#-programma naar machinecode gebeurt in twee stappen. Eerst wordt het programma vertaald naar een *intermediate language*, en die wordt vervolgens weer vertaald naar machinecode. Noem twee voordelen van het vertalen in twee stappen.

   Antwoord: Voordelen zijn: je kunt de intermediate code van libraries vrijgeven, zonder de broncode bekend te hoeven maken; de relatief eenvoudige tweede stap kan voor verschillende processoren worden gedaan; de intermediate code kan ook gebruikt worden voor andere programmeertalen.

   (d) Wat wordt er verstaan onder de *syntax* van een (programmeer)taal-constructie? En wat is de *semantiek* van een taal-constructie?

   Antwoord: De syntax van een taalconstructie is de grammaticale opbouw. De semantiek is de betekenis ervan.

   (e) Geef een voorbeeld van een type uit een standaardlibrary dat een `struct` is, en een voorbeeld van een type dat een `class` is. Wat is het verschil tussen een struct-type en een class-type?

   Antwoord: Voorbeelden van een standaard-struct zijn `Color`, `Size` en `Point`. Voorbeelden van een standaard-class zijn `Button`, `Bitmap` en `Form`. Het verschil is dat een variabele van een struct-type het object direct bevat, terwijl een variabele van een class-type een *verwijzing* naar het object bevat.

2. *(-1 punt per fout)* 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** is
   - **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

   Antwoord:

`[T] Button`	`[E] (bool)true`	`[H] int t()`	`[X ] class A : Size`
`[D] Button b;`	`[X] const bool true;`	`[D] int t=1;`	`[E] this.Size=this.ClientSize;`
`[O] b.Text = "ok";`	`[T] bool`	`[E] t==t+1`	`[H] Size s(Size t)`
`[X] new Button b;`	`[O] while(true) t=1;`	`[O] t=t+1;`	`[E] new Size(x,y)`

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`*-opdrachten nodig zijn om de klassen te kunnen gebruiken.*

3. (a) Schrijf een methode `cijfer` met twee getallen als parameter. Als de eerste parameter 0 is, geeft de methode het *laatste cijfer* van de tweede parameter terug, als de eerste parameter 1 is, geeft de methode het *voorlaatste cijfer* van de tweede parameter terug, als de eerste parameter 2 is, geeft de methode het *op twee na laatste cijfer* van de tweede parameter terug, enzovoorts.

   Bijvoorbeeld: `cijfer(0,456)` geeft 6 terug, `cijfer(2,98765)` geeft 7 terug, en `cijfer(4,1234)` geeft 0 terug. Je mag zonder controle aannemen dat beide parameters niet negatief zijn.

   (b) Bankrekeningnummers (met uitzondering van de vroegere gironummers) bestaan uit 9 cijfers. Maar niet elk getal van 9 cijfers is een geldig rekeningnummer. Om te controleren of er geen tikfouten zijn gemaakt bij het invoeren van rekeningnummers, doet online banking software de volgende controle: Het eerste cijfer wordt vermenigvuldigd met 9, het tweede cijfer wordt vermenigvuldigd met 8, het derde cijfer met 7, enzovoort, en het laatste cijfer met 1. Alle uitkomsten worden opgeteld. Als het totaal deelbaar is door 11, is het een geldig rekeningnummer. Bijvoorbeeld: voor de controle van 839801149 wordt uitgerekend: $8\times9+3\times8+9\times7+8\times6+0\times5+1\times4+1\times3+4\times2+9\times1 = 231$, en dat is inderdaad deelbaar door 11.

   Schrijf een methode `Controleer` met een getal als parameter, dat teruggeeft of dat getal een geldig bankrekeningnummer is. Vermijd daarbij om 9 maal vrijwel dezelfde expressie op te schrijven; gebruik in plaats daarvan een C#-opdracht om de regelmaat uit te buiten.

Antwoord:

```
static int cijfer(int n, int x)
{
    int t = 0;
    while (t < n) { x = x / 10; t++; }
    return x % 10;
}
public static bool Controleer(int x)
{
    int t = 0;
    int s = 0;
    while (t < 9) { s = s + (t+1)*Program.cijfer(t, x); t++; }
    return s % 11 == 0;
}
```

4. Gegeven is de volgende klasse:

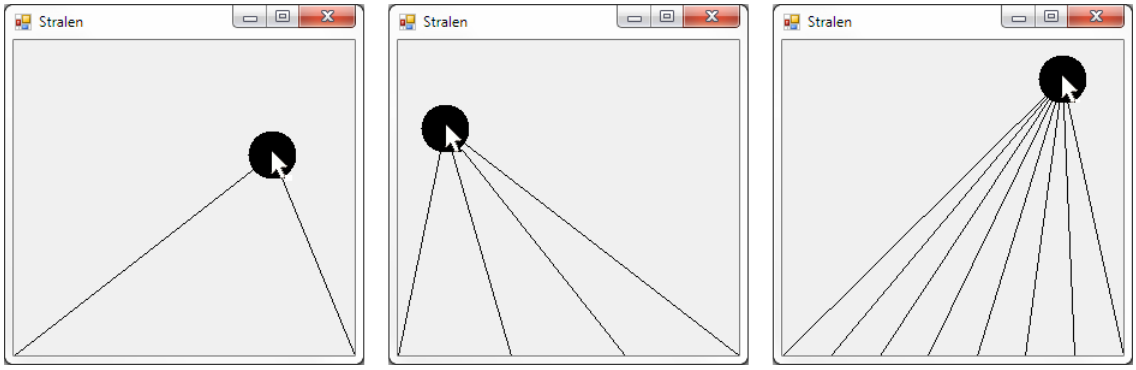
```
class Program
{   public static void Main()
    {   Stralen s = new Stralen();
        s.Text = "Stralen";
        Application.Run(s);
    }
}
```

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

Er is een zwart opgevulde cirkel met een diameter van 40 pixels in beeld. Het middelpunt van de cirkel bevindt zich op de positie van de muis; de cirkel beweegt dus mee met de muis.

Twee lijnen verbinden het midden van de cirkel met de twee onderhoeken van het window. Elke keer als de gebruiker met de muis klikt komt er een lijn bij. De lijnen monden op gelijke afstanden uit op de onderrand van het window.

Zie onderstaande figuur, met daarin: de beginstituatie, de situatie na 2 keer klikken, en de situatie na nog 4 keer klikken. (De pijl geft de muiscursor aan, deze hoef je niet te tekenen).



Antwoord:

```
public class Stralen : Form
{
    int x, y, n;

    public Stralen()
    {   n = 1;
        this.Paint += this.teken;
        this.MouseMove += this.beweeg;
        this.MouseClick += this.klik;
    }
    void teken(object o, PaintEventArgs pea)
    {   Graphics gr = pea.Graphics;
        gr.FillEllipse(Brushes.Black, this.x - 20, this.y - 20, 40, 40);
        int t = 0;
        while (t <= n)
        {   gr.DrawLine(Pens.Black, this.x, this.y, t * this.ClientSize.Width / n, this.ClientSize.Height)
            t = t+1;
        }
    }
    void beweeg(object o, MouseEventArgs mea)
    {   this.x = mea.X;
        this.y = mea.Y;
        this.Invalidate();
    }
    void klik(object o, MouseEventArgs mea)
    {   n = n+1;
        this.Invalidate();
    }
}
```

EINDE TENTAMEN