Uitwerking Tweede deeltentamen Imperatief programmeren
Woensdag 22 december 2010, 8.30–10.30 uur

---

1. (a) Gegeven zijn de volgende twee declaraties van variabelen:

```
Label lab;
double waarde;
```

Iemand schrijft de volgende opdracht om de wortel van de waarde te berekenen en aan de gebruiker te tonen:

```
lab.Text = "De wortel is " + Math.Sqrt(waarde);
```

Maar als `waarde` negatief is wordt het programma afgebroken met een foutmelding.

In plaats daarvan willen we liever dat de tekst `onmogelijk` op de label verschijnt. Je kunt dit op twee manieren voor elkaar krijgen:

- Vooraf controleren of de foutsituatie zich gaat voordoen
- De wortel gewoon maar uitrekenen en de foutsituatie opvangen

Geef voor beide aanpakken aan hoe de opdracht er dan uit komt te zien.

Antwoord:

```
// aanpak 1: vooraf controleren
if (waarde<0)
     lab.Text = "onmogelijk";
else lab.Text = "De wortel is " + Math.Sqrt(waarde);

// aanpak 2: foutsituatie afvangen
try
{   lab.Text = "De wortel is " + Math.Sqrt(waarde);
}
catch (Exception e)
{   lab.Text = "onmogelijk";
}
```

(b) Wat is een drie-dimensionale array?

Hoe declareer je zo'n drie-dimensionale array, en hoe kun je in het programma aangeven hoeveel waarden de array kan bevatten?

Antwoord: Een drie-dimensionale array is een hoeveelheid variabelen van hetzelfde type waarvan de individuele elementen zijn op te vragen door drie int-waarden als index op te geven.

Declaratie: *type* `[,,]` *naam* ;

Initialisatie: *naam* `= new` *type* `[` a `,` b `,` c `];`
geeft een array met $a \times b \times c$ elementen.

(c) Gegeven is een klasse `Data` met daarin een member-variabele `getallen`, en methoden die deze variabele een zinvolle waarde geven:

```
class Data
{
    private double [] getallen;

    // hier staan de bestaande methoden
    // hier komt een nog te schrijven property
}
```

Schrijf een read-only property `Totaal` waarmee het totaal van de opgeslagen waarden bepaald kan worden.

Antwoord:

```
public double Totaal
{   get
    {  double res = 0.0;
       for (int t=0; t<getallen.Length; t++)
           res += getallen[t];
       return res;
    }
}
```

(d) Hieronder staan vijf programma-fragmenten. Geef in elk van de gevallen aan *hoe vaak* de methode `iets` wordt aangeroepen. Licht het antwoord kort toe.

1	`for (x=0; x<5; x++)`
	`  this.iets();`
	`for (y=0; y<5; y++)`
	`  this.iets();`

2	`for (x=0; x<5; x++)`
	`for (y=0; y<x; y++)`
	`  this.iets();`

3	`for (x=0; x<5; x++)`
	`{  this.iets();`
	`  x=x+1;`
	`}`

4	`x=0;`
	`while (x<0) ;`
	`  this.iets();`
	`  x=x+1;`

5	`for (x=0; x<x; x++)` `  this.iets();`

Antwoord:
fragment 1: 10 keer (5 keer voor elke `x`, en dan nog eens 5 keer voor elke `y`)
fragment 2: 10 keer (0+1+2+3+4 keer)
fragment 3: 3 keer (voor `x` is 0, 2 en 4)
fragment 4: 1 keer (de while-opdracht heeft een lege body, daarna is `iets` aan de beurt)
fragment 5: 0 keer (de voorwaarde is meteen `false`)

2. In de klasse String zitten onder andere de volgende methodes:

```
int IndexOf(char c)
int IndexOf(String s)
string Substring(int)
```

Deze eerste methode `IndexOf` levert het nummer van de eerste positie op waar `c` in de string voorkomt. Als `c` nergens in de string voorkomt, is het resultaat -1. Voorbeelden:

`"Utrecht".IndexOf('h')` geeft 5
`"Utrecht".IndexOf('t')` geeft 1
`"Utrecht".IndexOf('x')` geeft -1

De tweede variant van `IndexOf` levert de eerste plaats op waar `s` een deel is van het totaal (of -1 als `s` nergens voorkomt). Voorbeelden:

`"Utrecht".IndexOf("ech")` geeft 3
`"Utrecht".IndexOf("trh")` geeft -1
`"Utrecht".IndexOf("Utrecht")` geeft 0

De hier bedoelde versie van `Substring` levert het deel van de string op, dat op een bepaalde plek begint. Voorbeelden:

`"Utrecht".Substring(2)` geeft `"recht"`
`"Utrecht".Substring(7)` geeft `""`

Stel dat je de auteur van de klasse `String` bent. Sommige methoden van die klasse zijn al geschreven (die mag je dus gebruiken), maar de methoden met de naam `Substring`, en die met de naam `IndexOf` ontbreken nog (die mag je dus niet aanroepen).

De opgave: Schrijf de drie hierboven beschreven methoden.

(De zelf-geschreven methodes mogen wel elkaar aanroepen).

Antwoord:

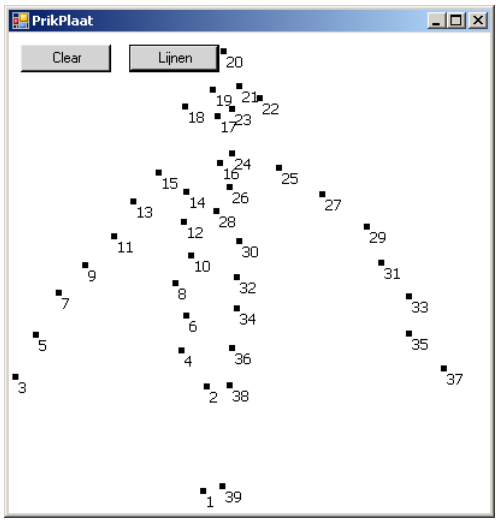
```
int IndexOf(char c)
{   for(int t=0; t<this.Length; t++)
        if (this[t]==c)
            return t;
    return 0;
}
string Substring(int b)
{   string res = "";
    for (int t=b; t<this.Length; t++)
        res += this[t];
    return res;
}
string Substring(int b, int e)
{   string res = "";
    for (int t=b; t<this.Length && t<e; t++)
        res += this[t];
    return res;
}
int IndexOf(string s)
{   for (int t=0; t<this.Length; t++)
        if (this.Substring(t, t+s.Length)==s)
            return t;
    return -1;
}
```

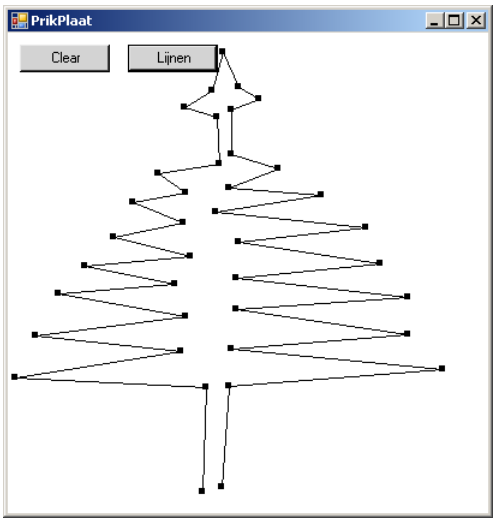
3. Gegeven is de volgende klasse, met daarin de methode `Main` en nog een andere handige methode.

```
class Program
{
    public static void Main()
    {
        PrikPlaat plaat = new PrikPlaat();
        plaat.ClientSize = new Size(400, 400);
        plaat.Text = "PrikPlaat";
        plaat.BackColor = Color.White;
        Application.Run(plaat);
    }
    public static Button MaakKnop(string tekst, int x, int y, EventHandler eh)
    {
        Button b = new Button();
        b.Text = tekst;
        b.BackColor = Color.LightGray;
        b.Location = new Point(x, y);
        b.Click += eh;
        return b;
    }
}
```

Schrijf nu de klasse `PrikPlaat`, zo dat het programma zich als volgt gaat gedragen:

- De gebruiker ziet twee knoppen met het opschrift 'Clear' en 'Lijnen'.
- De gebruiker kan verder overal in het window klikken. Op de aangeklikte punten verschijnen kleine vierkantjes, gecentreerd rond het aangeklikte punt.
- De punten zijn genummerd; naast elk aangeklikt punt is een getal zichtbaar: '1', '2', '3' enz. in volgorde van aanklikken.
- Er zijn maximaal 100 punten zichtbaar. Als de gebruiker daarna toch meer punten aanklikt, gebeurt er niets (ook geen foutmelding!).
- Na het indrukken van de knop 'Clear' verdwijnen alle punten. De gebruiker kan dan weer met 100 nieuwe punten beginnen.
- Na het indrukken van de knop 'Lijnen' verdwijnen de getallen, en worden de punten (die zelf wel zichtbaar blijven) verbonden door lijnen.
- Door nogmaals op de knop 'Lijnen' te drukken krijgt de gebruiker weer het beeld met getallen maar zonder lijnen te zien. Bij een derde keer klikken worden de lijnen weer zichtbaar, enz.





In het voorbeeld links heeft de gebruiker 39 punten aangeklikt.
In het voorbeeld rechts is er daarna op de 'Lijnen' knop gedrukt.

Antwoord:

```
public class PrikPlaat : Form
{
    Point[] ps = new Point[100];
    int n = 0;
    bool met = false;

    public PrikPlaat()
    {   this.Controls.Add(Program.MaakKnop("Clear", 10, 10, this.schoon));
        this.Controls.Add(Program.MaakKnop("Lijnen", 100, 10, this.lijnen));
        this.Paint += this.teken;
        this.MouseClick += this.klik;
    }
    public void teken(object o, PaintEventArgs pea)
    {   Graphics gr = pea.Graphics;
        for (int t=0; t < n; t++)
        {
            gr.FillRectangle(Brushes.Black, ps[t].X-3, ps[t].Y-3, 5, 5);
            if (!met)
                gr.DrawString((t+1).ToString(), new Font("Tahoma", 10)
                             , Brushes.Black, ps[t].X, ps[t].Y );
            if (met && t>0)
                gr.DrawLine(Pens.Black, ps[t-1], ps[t]);
        }
    }
    public void klik(object o, MouseEventArgs mea)
    {   if (n < 100)
        {   ps[n] = mea.Location;
            n++;
            this.Invalidate();
        }
    }
    public void schoon(object o, EventArgs ea)
    {   n = 0;
        this.Invalidate();
    }
    public void lijnen(object o, EventArgs ea)
    {   met = !met;
        this.Invalidate();
    }
}
```