Uitwerking Aanvullend tentamen Imperatief programmeren
Maandag 14 maart 2011, 9.00–12.00 uur

---

1. Voor de implementatie van het spel 'Reversi' schrijft iemand de volgende klasse. Het speelbord wordt gerepresenteerd met een 2-dimensionale array `bord`. Lege velden worden gesymboliseerd door een 0, de stenen van de twee spelers met 1 en -1.

   De methode `IngeslotenInRichting` moet testen of de speler met kleur `kleur`, door te zetten op positie (x, y), een reeks stenen van de tegenstander in de richting (dx, dy) insluit.

```
class Reversi
{
    const int size = 8;
    int [,] bord = new int[size,size];

    bool IngeslotenInRichting(int kleur, int x, int y, int dx, int dy)
    {   for (int t=1; t<size; t++)
        {   int px = x + t*dx;
            int py = y + t*dy;
            if (bord[px,py] == 0)     // leeg vakje:
                return false;         // niet ingesloten
            if (bord[px,py] == kleur) // eigen kleur:
                return t>1;           // ingesloten, tenzij dit de eerste is
        }
        return false;
    }
    // ... en andere methoden ...
}
```

   Deze methode werkt een tijdlang goed, maar na enige tijd spelen wordt het programma afgebroken met een foutmelding.

   (a) Welke fout is dat, en in welke spelsituatie treedt die op?
   Antwoord:
   Er treedt een arraybound error op als `px` of `py` niet tussen 0 en 8 ligt. Dat gebeurt als het rijtje stenen van de tegenstander dat gescand wordt tot de rand van het bord doorloopt.

   (b) Het programma kan worden verbeterd door met behulp van een extra test te voorkomen dat de foutsituatie optreedt. Hoe luidt die test, en waar in de methode voeg je die in?
   Antwoord:

```
if (px<0 || px>=size || py<0 || py>=size)
    return false;
```

   invoegen op de regel na de declaratie van `py`.

   (c) Een andere manier om het programma te verbeteren is de fout gewoon te laten optreden, maar de foutsituatie op te vangen zodat het programma toch verder kan gaan. Hoe pak je dat aan?
   Antwoord:

```
try
{
    // hier de originele code
}
catch (Exception e)
{
    return false;
}
```

2. (a) Definieer een klasse `Begrensd` met de volgende eigenschappen. Een object van deze klasse bevat een geheel getal, maar dat kan van buiten de klasse niet direct worden bekeken of veranderd. Wel is er een property `Getal`, waarmee de waarde van het getal kan worden opgevraagd. Ook kan via deze property de waarde van het getal worden veranderd, maar alleen als de nieuwe waarde kleiner dan 100 is. Bij een poging om het getal toch groter te maken blijft het getal onveranderd.
   Antwoord:

```
class Begrensd
{
    private int x;
    public int Getal
    {
        get { return x; }
        set { if (value<100) x=value; }
    }
}
```

(b) Bespreek voor elk van de onderstaande *encodings* hoeveel geheugenruimte het kost om een character in een file op te slaan, en hoeveel verschillende characters er opgeslagen kunnen worden.

- `Encoding.ASCII`
- `Encoding.GetEncoding("iso-8859-1")`, ook wel bekend als 'Latin1'
- `Encoding.Unicode`
- `Encoding.UTF8`

Antwoord:

Ascii: 1 byte (eigenlijk 7 bits); 128 mogelijkheden

Latin1: 1 byte (alle 8 bits); 256 mogelijkheden (waarvan er 224 echt gebruikt worden)

Unicode: 2 bytes, 65536 mogelijkheden

UTF8: 1 byte (voor de ascii-tekens), 2 bytes (voor Europese tekens) of 3 bytes (voor Aziatische tekens); 65536 mogelijkheden

(c) In sommige libraries komen klasses voor met methoden die `virtual` zijn gedeclareerd. Daarmee wordt geanticipeerd op een bepaald soort gebruik van de library door applicatie-programmeurs.

- Welk gebruik is dat?
- Geef een voorbeeld van een situatie waarin dat handig is.

Antwoord:

Klassen met virtuele methoden anticiperen erop dat een applicatieprogrammeur een of meer *subklassen* van de klasse maakt waarin de methode wordt *hergedefinieerd*.

Dat is handig als je een array van objecten van die klasse hebt, waarbij er objecten van de verschillende subklassen door elkaar voorkomen. Je kunt dan de methode voor alle elementen van de array aangeroepen, maar wat er dan precies gebeurt hangt af van het type van elk afzonderlijk object.

3. In de klasse `String` zitten onder andere de volgende methoden:

```
static int Compare(String, String)
int CompareTo(String)
String ToLower()
```

De methode `Compare` levert 0 op als de twee parameters precies gelijk zijn. Hij levert een negatief getal op (bijvoorbeeld -1, maar iets anders mag ook) als de eerste parameter kleiner is dan de tweede, en een positief getal (bijvoorbeeld 1) als die groter is. Met kleiner en groter wordt hier de woordenboekordening bedoeld: de eerste letter waar de strings verschillen bepaalt de ordening (volgens de Unicodes van die letters). Is de ene string een beginstuk van de andere, dan is de kortste de kleinste. Spaties en leestekens tellen gewoon mee, die hoeven dus niet speciaal behandeld te worden.

Voorbeelden:

`String.Compare("aap", "noot")`	geeft een negatief getal, want `'a'<'n'`
`String.Compare("noot", "nieten")`	geeft een positief getal, want `'o'>'i'`
`String.Compare("niet", "nietmachine")`	geeft een negatief getal vanwege de lengte
`String.Compare("noot", "noot")`	geeft 0, want precies gelijk
`String.Compare("noot", "NOOT")`	geeft een positief getal, want `'n'>'N'`

De methode `CompareTo` doet hetzelfde, maar is niet `static` en heeft dus een parameter minder nodig.

Stel dat je de auteur van de klasse `String` bent. Alle andere methodes zijn al geschreven, behalve de methoden met de naam `Compare`, `CompareTo` en `ToUpper`.

(a) Schrijf de methode `Compare`.

Antwoord:

```
static int Compare(string a, string b)
{
    for (int t=0; t<a.Length && t<b.Length; t++)
    {
        int d = a[t] - b[t];
        if (d!=0) return d;
    }
    return a.Length - b.Length;
}
```

(b) Schrijf de methode `CompareTo`. Vermijd dubbel schrijfwerk.

Antwoord:

```
int CompareTo(string s)
{
    return string.Compare(this, s);
}
```

(c) Schrijf de methode `ToUpper`. Voor het gemak mag je er van uitgaan dat er alleen Ascii-tekens in de string voorkomen.

Antwoord:

```
string ToUpper()
{
    string res;
    for (int t=0; t<this.Length; t++)
    {
        char c = this[t];
        if (c>='a' && c<='z')
            c = (char)(c - 'a' + 'A');
        res += c;
    }
    return res;
}
```

(d) Schrijf een compleet programma, dat door de gebruiker vanaf een console kan worden opgestart. Achter de naam van het programma specificeert de gebruiker een of meer woorden. Het programma schrijft het woord dat volgens `Compare` het grootste is op de console. Bijvoorbeeld:

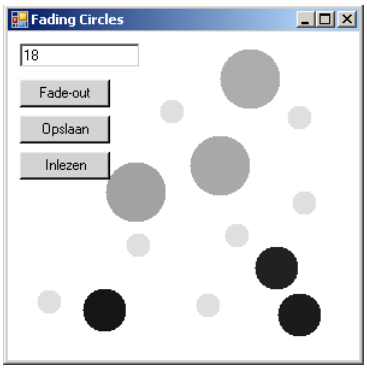
```
$ test.exe aap noot niet nietmachine
noot
```

Als de gebruiker vergeet om woorden te specificeren moet het programma hem daar netjes op wijzen.

Antwoord:

```
void main(string [] ws)
{
    if (ws.Length==0)
        Console.WriteLine("Usage: test.exe woord1 woord2 ...");
    else
    {   string max = ws[0];
        for (int t=1; t<ws.Length; t++)
            if (string.Compare(ws[t],max)>0)
                max = ws[t];
        Console.WriteLine(max);
    }
}
```

4. Bekijk het programma `FadingCircles`, waarvan hiernaast een screenshots staat. In het window zijn een tekstveld en vier buttons zichtbaar. De gebruiker kan met de muis steeds een punt aanklikken. Op dat punt verschijnt, gecentreerd, een zwart opgevulde cirkel.
De straal van de cirkel wordt gespecificeerd door het getal dat in het tekstveld staat ingevuld op het moment van de klik. Je mag zonder controle aannemen dat in het tekstveld alleen maar cijfertekens zijn ingevuld.
Als de gebruiker op de knop 'Fade-out' drukt, dan worden de cirkels langzaam lichter. Na ongeveer een minuut zijn de cirkels helemaal wit geworden. Nieuwe cirkels die nog worden bijgeklikt beginnen wel weer zwart, maar vervagen ook met de tijd.



Hieronder is al een deel van het programma gegeven.

```
using System;     using System.Windows.Forms; using System.Collections.Generic;
using System.IO;  using System.Drawing;       using System.Threading;

public class Cirkel
{   // opgave a
}
public class FadingCircles : Form
{   TextBox tb;
    const string naam = "cirkels.txt";

    public FadingCircles()
    {     this.tb = new TextBox(); tb.Text = "10";       tb.Location = new Point(10, 10);
        Button b1 = new Button();  b1.Text = "Fade-out"; b1.Location = new Point(10, 40);
        Button b2 = new Button();  b2.Text = "Opslaan";  b2.Location = new Point(10, 70);
        Button b3 = new Button();  b3.Text = "Inlezen";  b3.Location = new Point(10, 100);
        this.Controls.Add(tb);
        this.Controls.Add(b1); b1.BackColor = Color.LightGray; b1.Click += this.fadeout;
        this.Controls.Add(b2); b2.BackColor = Color.LightGray; b2.Click += this.inlezen;
        this.Controls.Add(b3); b3.BackColor = Color.LightGray; b3.Click += this.opslaan;
        this.Paint += this.teken;  this.MouseClick += this.muisklik;
    }
    // ontbrekende member-variabelen en methoden: opgave b, c en d
}
public class Program
{   static void Main()
    {   FadingCircles f = new FadingCircles();
        f.Text = "Fading Circles"; f.BackColor = Color.White;
        Application.Run(f);
    }
}
```

Vul hierop het volgende aan:

(a) Schrijf een hulpklasse `Cirkel`, zo dat in een object van die klasse alle gegevens die nodig zijn om een cirkel te kunnen tekenen beschikbaar zijn. Maak een constructormethode die zo'n object van beginwaardes voorziet, een methode `ToString` die de waarden 'inpakt' in een string, en een tweede constructormethode die zo'n string weer 'uitpakt'. Maak ook een methode `Fade` die de kleur van de cirkel lichter maakt.

Antwoord:

```
public class Cirkel
{
    public int x, y, r, c;
    public Cirkel(int ax, int ay, int ar, int ac)
    {   x = ax; y = ay; r = ar; c = ac;
    }
    public Cirkel(string s)
    {   string[] veld = s.Split();
```

```
        x = int.Parse(veld[0]);
        y = int.Parse(veld[1]);
        r = int.Parse(veld[2]);
        c = int.Parse(veld[3]);
    }
    public override string ToString()
    {   return x + " " + y + " " + r + " " + c;
    }
    public void fade()
    {   if (c < 255) c++;
    }
}
```

(b) Schrijf de methoden `muisklik` en `teken` in de klasse `FadingCircles`, en geef de declaraties van de daarvoor benodigde member-variabelen.

Antwoord:

```
List<Cirkel> lijst = new List<Cirkel>();
public void teken(object o, PaintEventArgs pea)
{   foreach (Cirkel c in lijst)
    {   Brush b = new SolidBrush( Color.FromArgb(c.c,c.c,c.c) );
        pea.Graphics.FillEllipse(b, c.x-c.r, c.y-c.r, 2*c.r, 2*c.r);
    }
}
public void muisklik(object o, MouseEventArgs mea)
{   Cirkel c = new Cirkel(mea.X, mea.Y, int.Parse(tb.Text), 0);
    lijst.Add(c);
    this.Invalidate();
}
```

(c) Schrijf de methoden `fadeout` en eventuele hulpmethodes, zo dat dat de knop 'Fade-out' werkt zoals hierboven beschreven.

Antwoord:

```
public void fadeout(object o, EventArgs ea)
{   new Thread(this.run).Start();
}
public void run()
{   while (true)
    {   foreach (Cirkel c in lijst)
            c.fade();
        Thread.Sleep(100);
        this.Invalidate();
    }
}
```

(d) Als de gebruiker op de knop 'Opslaan' drukt, wordt de toestand van de tekening opgeslagen in een file waarvan de naam in de constante `naam` staat. Als de gebruiker op de knop 'Inlezen' drukt, wordt de huidige tekening vervangen door de opgeslagen tekening. Schrijf de methoden `opslaan` en `inlezen` die daarvoor nodig zijn.

Antwoord:

```
void opslaan(object o, EventArgs ea)
{   StreamWriter w = new StreamWriter(naam);
    foreach (Cirkel c in lijst)
        w.WriteLine(c.ToString());
    w.Close();
}
void inlezen(object o, EventArgs ea)
{   lijst.Clear();
    StreamReader r = new StreamReader(naam);
    string regel;
    while ((regel = r.ReadLine()) != null)
        lijst.Add(new Cirkel(regel));
    r.Close();
    this.Invalidate();
}
```