Uitwerking Derde deeltentamen Imperatief programmeren
Woensdag 2 februari 2011, 8.30–10.30 uur

---

1. Voor elk punt (x, y) van het platte vlak, waarbij x en y reële getallen zijn, kan een bijbehorend getal worden bepaald – laten we dit het 'mandelgetal' noemen. Om het mandelgetal te kunnen uitrekenen, bekijken we eerst de volgende functie, die punten (a, b) van het vlak transformeert naar andere punten:

   $$f\ (a, b) = (a * a - b * b + x,\ \ 2 * a * b + y)$$

   Let op: deze functie transformeert het punt (a, b), maar in de berekening speelt ook de waarde van x en y, dat is het punt waarvan we het mandelgetal willen bepalen, een rol.

   Deze functie f nu, passen we toe op het punt $(a, b) = (0, 0)$. Op het punt dat daar uitkomt, passen we nog eens de functie f toe. Op het punt dat daar weer het resultaat van is, passen we opnieuw f toe, enzovoorts. We stoppen pas met toepassen van f als het resultaat-punt een afstand van meer dan 2 tot het punt $(0, 0)$ heeft. Het mandelgetal is nu gelijk aan het *aantal keren* dat f is toegepast.

   Voor sommige punten (x, y) is dat meteen al zo, en is het mandelgetal dus gelijk aan 1. Voor andere punten duurt het langer: die hebben een groter mandelgetal. Er zijn ook punten waarbij je f kan blijven toepassen, zonder dat de afstand tot de oorsprong ooit meer dan 2 wordt. Voor die punten stellen we het mandelgetal op 100.

   (a) Schrijf een methode `mandel` die het mandelgetal uitrekent van het punt waarvan de coordinaten als parameter worden meegegeven.

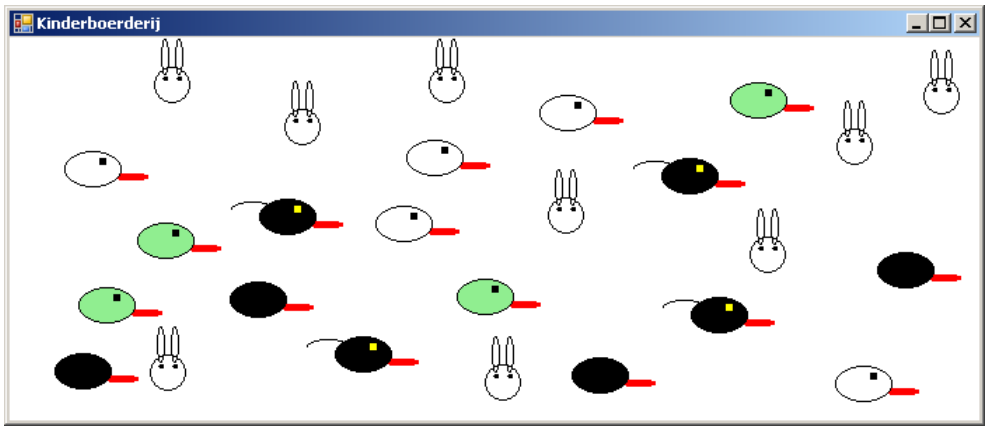
   (b) Schrijf het ontbrekende stuk van de methode `teken`, die de punten op het scherm zwart kleurt die een *oneven* mandelgetal hebben. De gedeclareerde `schaal` moet worden gebruikt zo dat het plaatje wordt getoond voor x en y tussen 0 en 4.

```
class Mandelbrot : Form
{
    double schaal = 0.01;

    // TODO opgave a: methode mandel

    public void teken(object obj, PaintEventArgs pea)
    {   Graphics gr = pea.Graphics;
        for (int x=0; x<400; x++)
        {   for (int y=0; y<400; y++)
            {    // TODO opgave b: body
            }
        }
    }
    public Mandelbrot
    {   this.Paint += this.teken;
    }
}
```

2. Bekijk het gegeven programma op pagina 4 (als je het losscheurt kun je het naast de opgave leggen). De gebruiker kan er een vrijwel onbeperkt aantal dierenkoppen mee tekenen: overal waar de gebruiker klikt met de linker muisknop onstaat de kop van een eend, en met de rechter muisknop de kop van een konijntje. Na een aantal kliks zou het scherm er zo uit kunnen zien:



Er zijn vier soorten eenden: de gewone eend heeft een witte kop en een zwart oog, de wilde eend heeft een groene kop en een zwart oog, de bergeend heeft een zwarte kop waarop het zwarte oog niet meer te zien is, en de kuifeend heeft een zwarte kop met een geel oog en bovendien een sierlijke kuif achter op z'n kop.

(a) Er ontbreken nog declaraties in dit programma. Schrijf deze declaraties, en geef aan waar die moeten staan.

Antwoord:

```
List<Dier> dieren = new List<Dier>();
int k = 0;  // nodig in opgave c
```

(b) In de methode `teken` staat een `foreach`-opdracht. Kun je in plaats daarvan ook een gewone `for`-opdracht gebruiken? Zo ja: hoe ziet die er dan uit? Zo nee: waarom kan dat niet?

Antwoord:

als je bij opgave a een List hebt gebruikt: Ja, dit kan als volgt:

```
for (int t=0; t<dieren.Count; t++)
    dieren[t].LaatZien(pea.Graphics);
```

als je bij opgave a een Collection hebt gebruikt: Nee, althans niet een for-opdracht met een tellertje, omdat de elementen van een Collection niet via een index te benaderen zijn. (Het kan eventueeel wel door in de for-opdracht expliciet een Iterator-object aan te maken).

(c) Schrijf de ontbrekende methode `klik`. Als de gebruiker met de rechter muisknop klikt, moet er een konijntje verschijnen. Met de linker muisknop verschijnen er afwisselend de vier soorten eenden. Je kunt daarbij gebruik maken van de bestaande methode `eendKeuze`. Hint: gebruik de bijlage om te zien hoe je de knoppen kunt onderscheiden.

Antwoord:

```
public void klik(object obj, MouseEventArgs mea)
{
    Dier dier;
    if (mea.Button == MouseButtons.Right)
        dier = new Konijn();
    else{  dier = this.eendKeuze(k%4); k++; }
    dier.plek = mea.Location;
    dieren.Add(dier);
    this.Invalidate();
}
```

(d) Schrijf de ontbrekende klasse `Dier`. Welke members en/of methoden daarin nodig zijn blijkt uit de rest van het programma. Zorg ervoor dat er in het programma niet per ongeluk dieren kunnen worden gemaakt waarvan het niet duidelijk is of het een konijn of een eend is.

Antwoord:

```
abstract class Dier
{
    public Point plek;
    public abstract void LaatZien(Graphics g);
}
```

(e) In de klasse `Eend` worden twee properties gebruikt die nog niet zijn gedefinieerd. Definieer deze properties, zo dat de methode `LaatZien` een gewone eend laat zien. Houd daarbij wel alvast rekening met de uitbreidingen in de opgave f en g.

Antwoord:

```
public virtual Brush Kop
{   get { return Brushes.White; }
}
public virtual Brush Oog
{   get { return Brushes.Black; }
}
```

(f) Schrijf de ontbrekende klassen `WildeEend` en `BergEend`. Vermijd daarbij zo veel mogelijk het dupliceren van code.

Antwoord:

```
class WildeEend : Eend
{   public override Brush Kop
    {   get { return Brushes.LightGreen; }
    }
}
class BergEend : Eend
{   public override Brush Kop
    {   get { return Brushes.Black; }
    }
}
```

(g) Schrijf de ontbrekende klasse `KuifEend`. Vermijd ook hier weer zo veel mogelijk het dupliceren van code.

Antwoord:

```
class KuifEend : BergEend
{   public override Brush Oog
    {   get { return Brushes.Yellow; }
    }
    public override void LaatZien(Graphics g)
    {   base.LaatZien(g);
        g.DrawArc(Pens.Black, plek.X - 20, plek.Y + 2, 30, 10, 180, 180);  // kuif
    }
}
```

3. (a) Schrijf een programma `LangsteRegels` dat de gebruiker vanaf een commando-regel kan starten met vermelding van een aantal namen van tekstfiles. Bijvoorbeeld:

```
$ LangsteRegels aap.txt note.txt mies.txt
```

Het programma moet op dezelfde console waarvan het gestart werd voor elke file aangeven wat de langste regel is, zoals in het voorbeeld hieronder. De regel uit de tekst moet daarbij tussen aanhalingstekens getoond worden. Als er een probleem is met het lezen van een van de files moet dat gemeld worden, maar gaat de verwerking van de andere files wel door.
De output zou bijvoorbeeld kunnen zijn:

```
Langste regel van aap.txt: "De gorilla is een herbivoor die leeft in de Afrikaanse regenwouden"
Probleem bij het lezen van note.txt
Langste regel van mies.txt: "Omstreeks 1910 bedacht Jetzes de aap-noot-mies leesmethode."
```

Als er meerdere regels precies even lang zijn, mag je zelf kiezen welke je laat zien.

Antwoord:

```
public void Main(string[] namen)
{
    for (int t=0; t<namen.Length; t++)
    {   try
        {   StreamReader sr = new StreamReader(namen[t]);
            string regel, langste="";
            while ((regel=sr.ReadLine())!=null)
                if (regel.Length > langste.Length)
                    langste = regel;
            Console.WriteLine("Langste regel van " + namen[t] + ": \"" + langste + "\"");
        }
        catch (Exception e)
        {   Console.Writeline("Probleem bij het lezen van " + namen[t]);
        }
    }
}
```

(b) *In deze vraag hoef je niet te programmeren. Je kunt het in woorden uitleggen.*

Beschrijf welke klassen je moet definiëren als je een programma met een MDI-gebruikersinterface wilt maken, en van welke klassen deze klassen een subklasse moeten zijn. In welke klassen kunnen de menu-keuzes 'New', 'Open', 'Save' en 'Find' het beste worden ondergebracht? In welke methoden worden er nieuwe objecten van de genoemde klassen aangemaakt?

Antwoord:

Een klasse voor het parentwindow en een klasse voor het childwindow, allebei subklasse van `Form`. Menu-keuzes 'New' en 'Open' horen in het parentwindow, menu-keuzes 'Save' en 'Find' horen in het childwindow. Een object van het parentwindow wordt aangemaakt in methode `Main`. Objecten van het childwindow worden aangemaakt in de click-event-handlers van menuopties `New` en `Open`.

(c) Op welke manier worden characters in een file weggeschreven als je daarbij `Encoding.UTF8` gebruikt?

Wat is het voordeel van UTF8 boven de Latin1 encoding?

Wat is het voordeel van UTF8 boven de Unicode encoding?

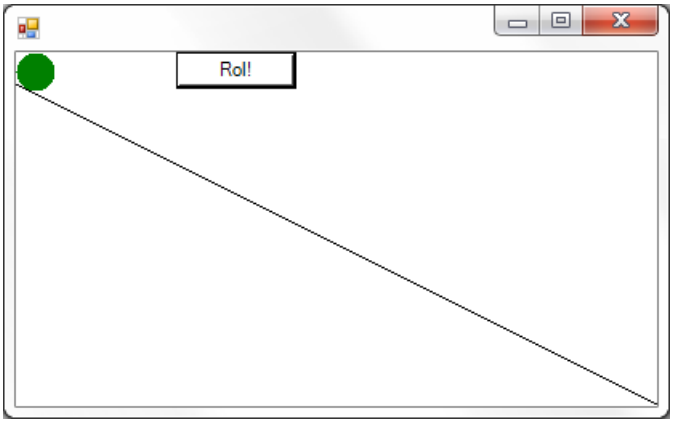
Antwoord:

Het voordeel van UTF8 boven Latin1 is dat ook niet-Westeuropese symbolen kunnen worden opgeslagen. Het voordeel van UTF8 boven Unicode is dat Ascii-symbolen met 1 byte in plaats van 2 bytes kunnen worden opgeslagen.

(d) Dit programma tekent een balletje bovenaan een helling. Als de gebruiker op de knop drukt, moet het balletje in een 'tekenfilmpje' van circa 10 seconden van de berg af (en daarna uit beeld) rollen. Schrijf de daarvoor benodigde methode `rol` en eventuele extra hulp-methoden.

```
public class HellendVlak : Form
{
    static void Main()
    {    Application.Run(new HellendVlak());
    }
    public HellendVlak()
    {   Button b = new Button(); this.Controls.Add(b);
        b.Text  = "Rol!";        b.Location = new Point(100, 0);
        b.Click += this.rol;     this.Paint += this.teken;
```

```
    }
    int x = 0, y = 0;
    public void teken(object obj, PaintEventArgs pea)
    {   pea.Graphics.DrawLine   (Pens.Black,    0, 20, 400, 220); // helling
        pea.Graphics.FillEllipse(Brushes.Green, x,  y,  24,  24); // balletje
    }
}
```



Antwoord:

```
public void rol(object obj, EventArgs ea)
{   Thread t = new Thread(run);
    t.Start();
}
public void run()
{   while (true)
    {   x+=2; y+=1;
        this.Invalidate();
        Thread.Sleep(50);
    }
}
```

Bijlage bij opgave 2

```
public class Kinderboerderij : Form
{
    public Kinderboerderij()
    {   this.Text = "Kinderboerderij";
        this.Size = new Size(700, 300);
        this.BackColor   = Color.White;
        this.MouseClick += this.klik;
        this.Paint      += this.teken;
    }
    public void teken(object obj, PaintEventArgs pea)
    {   foreach (Dier dier in dieren)
            dier.LaatZien(pea.Graphics);
    }
    private Eend eendKeuze(int n)
    {   switch (n)
        {   case 1:  return new WildeEend();
            case 2:  return new BergEend();
            case 3:  return new KuifEend();
            default: return new Eend();
        }
    }
    static void Main()
    {   Application.Run(new Kinderboerderij());
    }
    // TODO: methode klik
}

// TODO: klasse Dier

class Konijn : Dier
{
    public override void LaatZien(Graphics g)
    {   g.DrawEllipse(Pens.Black,    plek.X+5 , plek.Y-20,  5, 25);  // linkeroor
        g.DrawEllipse(Pens.Black,    plek.X+15, plek.Y-20,  5, 25);  // rechteroor
        g.DrawEllipse(Pens.Black,    plek.X,    plek.Y,    25, 25);  // kop
        g.FillEllipse(Brushes.Black, plek.X+6 , plek.Y+6,   4,  4);  // linkeroog
        g.FillEllipse(Brushes.Black, plek.X+16, plek.Y+6,   4,  4);  // rechteroog
    }
}

class Eend : Dier
{
    public override void LaatZien(Graphics g)
    {   g.FillEllipse  (Brushes.Red, plek.X+35, plek.Y+15, 25,  6);  // snavel
        g.FillEllipse  (this.Kop,    plek.X,    plek.Y,    40, 25);  // kop
        g.DrawEllipse  (Pens.Black,  plek.X,    plek.Y,    40, 25);  // rand van de kop
        g.FillRectangle(this.Oog,    plek.X+25, plek.Y+5,   5,  5);  // oog
    }
    // TODO: properties
}

// TODO: klasse WildeEend, BergEend en KuifEend
```