Uitwerkingen Derde deeltentamen **Imperatief programmeren**
Vrijdag 12 november 2010, 8.30-10.30 uur

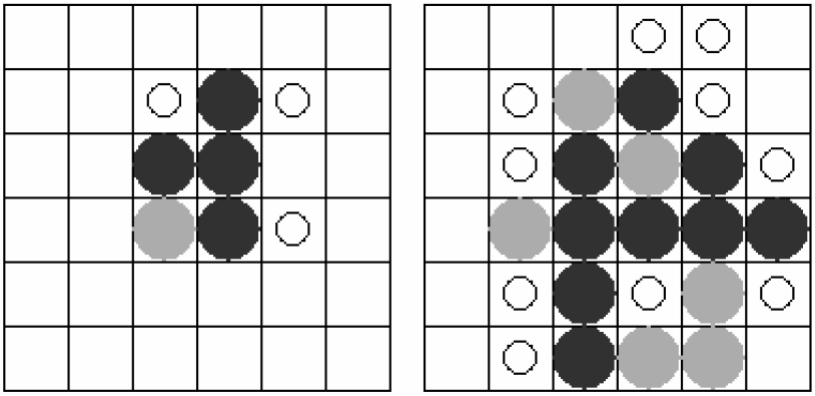
1. *(telt voor 20%)*

   Bij het spel 'Reversi' leggen twee spelers om de beurt een gekleurde steen op een veld van een rechthoekig speelbord.

   Een steen mag alleen maar worden neergelegd op een veld als het veld nog leeg is, en met deze zet een rij van een of meer stenen van de andere kleur wordt ingesloten tussen de nieuwe steen en een al op het bord liggende steen van de eigen kleur.

   De ingesloten stenen kunnen in acht mogelijke richtingen naast de nieuwe steen liggen: horizontaal, verticaal of diagonaal. Stenen insluiten in meerdere richtingen mag ook.

   In de figuur is voor twee voorbeelden met open cirkels aangegeven op welke velden de speler met de lichte stenen mag zetten.



   Als gevolg van een zet veranderen alle ingesloten stenen van kleur. In een programma wordt de situatie opgeslagen in een twee-dimensionale array:

```
int bord[,] = new int[6,6];
```

   Lege velden zijn gecodeerd met 0, gevulde velden met 1 of -1 voor de twee kleuren.

   De opgave: Schrijf een methode

```
bool mag (int kleur, int x, int y)
```

   die controleert of speler kleur een steen op veld (x, y) mag zetten. (De zet wordt dus nog niet uitgevoerd!). Hint: het is toegestaan (en handig) om een extra methode te schrijven die het insluiten in één van de 8 richtingen controleert.

2. *(telt voor 45%)*

Het programma in deze opgave leest een bestand in waarin de ligging van een aantal steden wordt vastgelegd. Het programma toont deze steden op het scherm als zwarte vierkantjes met de naam ernaast.

De gebruiker kan deze vierkantjes aanklikken. De namen van de aangeklikte steden verschijnen in een lijst rechts van het plaatje. Bovendien zie je rond de steden een cirkel, die bij elke klik groter wordt.

De gebruiker kan ook de lijst rechts op het scherm editten. Als hij/zij daarna op de knop onderaan het scherm drukt, worden de cirkeltjes aangepast op grond van het aantal keer dat de naam van de stad in het lijstje voorkomt.

In onderdeel (g) passen we het programma aan, zo dat de grootste cirkel (of cirkels, als er meerdere evengroot zijn) in een andere kleur worden getoond.

In de bijlage aan het eind van dit tentamen staat een deel van het programma gegeven. Het bestaat uit vier klassen: `Scherm`, `Teller`, `Stad`, en `Kaart`. Een aantal onderdelen moet nog worden ingevuld. Let op dat sommige variabelen `private` zijn gedeclareerd: die mogen van buiten de klasse niet worden gebruikt.

(a) Schrijf de methode `Raak` in de klasse `Stad`, die oplevert of de parameter een punt binnen het zwarte vierkantje van de betreffende stad aanduidt.

Antwoord:

```
public bool Raak(Point p)
{
    return Math.Abs(this.plek.X - p.X) < 5 && Math.Abs(this.plek.Y - p.Y) < 5;
}
```

(b) Schrijf de declaraties en initialisaties van de member-variabelen die nodig zijn in de klasse `Kaart`.

Antwoord:

```
protected ICollection<Stad> steden = new List<Stad>();
Scherm parent;
```

(c) Schrijf het ontbrekende deel van de methode `Lees` van de klasse `Kaart`, die de steden inleest vanuit een bestand. De regels van dat bestand bevatten steeds twee getallen (de x- en y-coordinaat) en precies één woord, dus regels zoals

```
250 110 Amsterdam
280 180 Utrecht
```

Antwoord:

```
public void Lees(string naam)
{
    TextReader invoer = new StreamReader(naam);
    string regel;
    while ((regel = invoer.ReadLine()) != null)
    {
        string[] woorden = regel.Split();
        this.steden.Add(new Stad(new Point(int.Parse(woorden[0]), int.Parse(woorden[1])), woorden[2]));
    }
}
```

(d) Als met de muis een vierkantje wordt aangeklikt, moeten er twee dingen gebeuren:
- de naam van de stad wordt toegevoegd aan het lijstje rechts op het scherm
- de cirkel die om de stad wordt getekend wordt groter

Schrijf de methodes `WelkeStad` en `StadPlus` die daarvoor nodig zijn.

Antwoord:

```
public Stad WelkeStad(Point p)
{
    foreach (Stad s in steden)
        if (s.Raak(p))
            return s;
    return null;
}
public void StadPlus(Stad s)
{
```

```
    s.Increment();
    this.parent.NaamErbij(s.GetNaam());
}
```

(e) Schrijf de methode `Teken` die alle steden in beeld brengt. De kleur van de cirkel rond elke stad moet zijn zoals de gegeven methode `KleurVan` dat voorschrijft (momenteel groen voor alle steden).

Antwoord:

```
public void Teken(object o, PaintEventArgs pea)
{
    Graphics g = pea.Graphics;
    foreach (Stad s in steden)
        s.Teken(g, this.KleurVan(s));
}
```

(f) De gebruiker kan ook met het toetsenbord de lijst van steden aanpassen. Als hij/zij daarna op de knop 'Verwerken' onderin beeld klikt, dan worden de cirkels rond de steden aangepast op grond van het aantal keer dat de stad dan in de lijst voorkomt. Namen met een spelfout erin worden genegeerd. Schrijf de methode `Verwerk` die daarvoor nodig is.

Antwoord:

```
public void Verwerk(string namen)
{   foreach (Stad s in steden)
        s.Reset();
    foreach (string naam in namen.Split())
        foreach (Stad s in steden)
            if (s.GetNaam() == naam)
                s.Increment();
    this.Invalidate();
}
```

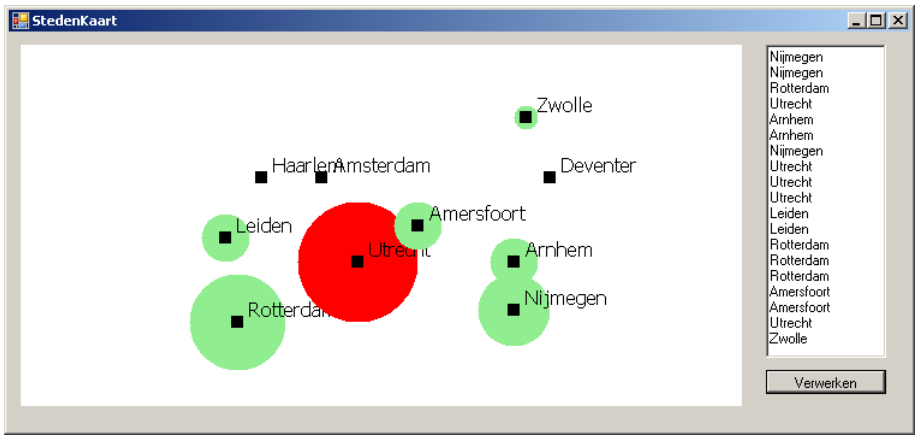
(g) We gaan het programma nu aanpassen. In de klasse `Scherm` wordt de initialisatie van variabele `kaart` vervangen door

```
    kaart = new RodeKaart();
```

Schrijf de klasse `RodeKaart` zodat het programma zich als volgt gaat gedragen: bij de stad met de grootste cirkel eromheen (of steden, als er meerdere steden de grootste zijn) wordt die cirkel niet groen maar rood. In de klasse `RodeKaart` moet zo veel mogelijk van het werk dat al in de klasse `Kaart` is gedaan worden hergebruikt; het mag dus niet opnieuw worden opgeschreven.

Antwoord:

```
public class RodeKaart : Kaart
{
    public override Color KleurVan(Stad s)
    {
        int max = 0;
        foreach (Stad a in this.steden)
            if (a.Waarde > max)
                max = a.Waarde;
        if (s.Waarde == max)
            return Color.Red;
        return base.KleurVan(s);
    }
}
```



3. *(telt voor 35%)*

(a) Sommige programma's maken geen gebruik van een window, maar communiceren met de gebruiker via een commandoregel.
- Hoe kun je in het programma de woorden (bijvoorbeeld filenamen) te pakken krijgen die de gebruiker bij het opstarten van het programma heeft meegegeven?
- Hoe kun je een string tonen aan de gebruiker van zo'n programma?

Antwoord:

Je kunt bij de methode `Main` een arrays van strings als parameter declararen. Die wordt gevuld met de woorden die de gebruiker op de commandoregel intikt.

Een console-applicatie kan door aanroep van methode `Console.WriteLine` een string tonen aan de gebruiker.

(b) Vergelijk de volgende drie *encodings* met elkaar, voor wat betreft de benodigde opslagruimte, en de flexibiliteit in het gebruik:
- `Encoding.ASCII`
- `Encoding.Unicode`
- `Encoding.UTF8`

Antwoord: Ascii gebruikt 1 byte per character, maar kan alleen de eerste 128 tekens weergeven. Unicode kost 2 bytes per character, maar kan alle 65536 tekens weergeven. UTF8 kan ook alle 65536 tekens weergeven, en kost een variabel aantal bytes per character: 1 byte voor Ascii-tekens, en 2 of 3 bytes voor andere tekens. Voor teksten met veel Ascii-tekens is dat dus efficiënter.

(c) Geef een korte beschrijving van
- de syntax van `switch`-opdrachten
- de semantiek van `switch`-opdrachten

Antwoord:

Een switch-opdracht begint met het woord `switch`, en dan tussen haakjes een expressie. Daaronder een *blok* dat de body van de opdracht vormt. De opdrachten in dat blok kunnen zijn gelabeld met `case` *constante* `:` of met `default:` .

Bij het uitvoeren van de switch-opdracht wordt de expressie uitgerekend. Daarna worden de opdrachten in de body uitgevoerd, te beginnen met de opdracht die is gemarkeerd met de waarde van de expressie. Als de waarde van de expressie niet aanwezig is als één van de cases, wordt de default-opdracht gebruikt.

(d) Bekijk de volgende klasse `Raster`:

```
class Raster
{
    public bool[,] vakjes = new bool[10,10];

    public void Combineer( Raster ander, Func<bool,bool,bool> comb )
    {
        for (int x=0; x<vakjes.GetLength(0); x++)
            for (int y=0; y<vakjes.GetLength(1); y++)
                vakjes[x,y] = comb( vakjes[x,y], ander.vakjes[x,y] );
    }
}
```

Stel dat we nu twee `Raster`-objecten hebben:

```
Raster r1 = new Raster();
Raster r2 = new Raster();
```

We gaan `Combineer` aanroepen op zo'n manier, dat in `r1` alleen die vakjes `true` blijven, waarbij het vakje op dezelfde plek in `r2` ook `true` is.

Geef twee manieren om `Combineer` aan te roepen:

- met behulp van een apart gedefinieerde extra hulpfunctie
- met behulp van een anonieme functie die direct wordt meegegeven

Antwoord:

Aanroep: `r1.Combineer(r2, hulpfunctie);` met een aparte hulpfunctie:

```
bool hulpfunctie(bool a, bool b)
{   return a && b;
}
```

Het alternatief is om de hulpfunctie direct op te schrijven als parameter in de aanroep van `Combineer`:

```
r1.Combineer(r2, (a,b)=>a&&b );
```

Bijlage bij opgave 2

```
public class Scherm : Form
{   TextBox namen;
    Kaart kaart;

    public Scherm()
    {   this.Text = "StedenKaart";
        this.ClientSize = new Size(750, 330);
        kaart = new Kaart();   // later te vervangen door   new RodeKaart()
        kaart.Location = new Point(10,10); kaart.Size = new Size(600,300);
        kaart.SetParent(this);
        namen = new TextBox();
        namen.Location = new Point(630,10); namen.Size = new Size(100,260); namen.Multiline = true;
        Button knop = new Button(); knop.Text = "Verwerken";
        knop.Location = new Point(630, 280); knop.Size = new Size(100, 20);
        knop.Click += verwerken;
        this.Controls.Add(kaart); this.Controls.Add(namen); this.Controls.Add(knop);
    }
    public void NaamErbij(string naam)
    {   namen.Text += (naam + "\r\n");
    }
    public void verwerken(object o, EventArgs ea)
    {   kaart.Verwerk(namen.Text);
    }
}
```

```
public class Teller
{   private int x;
    public void Reset()
    {   x = 0;
    }
    public void Increment()
    {   x++;
    }
    public int Waarde
    {   get { return x; }
    }
}
```

```
public class Stad : Teller
{   string naam;
    Point plek;
    static Font font = new Font("Tahoma", 12);

    public Stad(Point p, string s)
    {   this.plek = p;
        this.naam = s;
    }
    public string GetNaam()
    {   return naam;
    }
    public bool Raak(Point p)
    {   // TODO (opgave a)
    }
    public void Teken(Graphics g, Color c)
    {   int d = this.Waarde;
        g.FillEllipse(new SolidBrush(c), plek.X - 10 * d, plek.Y - 10 * d, 20 * d, 20 * d);
        g.FillRectangle(Brushes.Black, plek.X - 5, plek.Y - 5, 10, 10);
        g.DrawString(naam, font, Brushes.Black, plek.X + 7, plek.Y - 20);
    }
}
```

```
public class Kaart : UserControl
{
    // TODO: Membervariabelen declareren (opgave b)

    public Kaart()
    {   this.BackColor = Color.White;
        this.Lees("../../steden.txt");
        this.Paint      += this.Teken;
        this.MouseClick += this.Muisklik;
    }

    public void SetParent(Scherm s)
    {   this.parent = s;
    }

    public void Lees(string naam)
    {
       // TODO (opgave c)
    }

    public void Muisklik(object o, MouseEventArgs mea)
    {
        Stad s = this.WelkeStad(mea.Location);
        if (s != null)
            this.StadPlus(s);
        this.Invalidate();
    }

    public Stad WelkeStad(Point p)
    {
        // TODO (opgave d)
    }

    public void StadPlus(Stad s)
    {
        // TODO (opgave d)
    }

    public virtual Color KleurVan(Stad s)
    {
        return Color.LightGreen;
    }

    public void Teken(object o, PaintEventArgs pea)
    {
        // TODO (opgave e)
    }

    public void Verwerk(string namen)
    {
        // TODO (opgave f)
        this.Invalidate();
    }
}
```

```
public class RodeKaart  // TODO (opgave g)
```