DERDE DEELTENTAMEN IMPERATIEF PROGRAMMEREN - VERSIE 1
VRIJDAG 10 NOVEMBER 2017, 11.00-13.00 UUR

- Schrijf op elk ingeleverd blad je naam. Schrijf op het eerste blad ook je studentnummer en het aantal ingeleverde bladen.
- De lijst met standaardfuncties na afloop graag weer inleveren. De antwoorden komen binnenkort op de website.
- Opgave 1 t/m 10 zijn meerkeuzevragen, die meetellen voor $10 \times 4 = 40$ punten.
  Opgave 11 en 12 zijn programmeer- en tekstvragen, die meetellen voor 20 en 40 punten.

Meerkeuzevragen: de letter van het goede antwoord volstaat.
Belangrijk: dit is versie **1** van het tentamen, vermeld dat boven je antwoorden.

1. Een `try-catch` opdracht kan worden gebruikt om

   (a) het optreden van exceptions te voorkomen
   (b) programmeerfouten op te vangen
   (c) een foutsituatie netjes af te handelen
   (d) in een methode ongeldige waarden van een parameter te detecteren

2. In C# staat het symbool `=>`

   (a) tussen de parameters en de body van een methode zonder naam
   (b) tussen de naam en de body van een methode zonder parameters
   (c) tussen het type en de naam van de parameters van een methode
   (d) tussen de parameters en het resultaattype van een methode zonder body

3. Als je in de header van een klasse de naam van een interface erbij schrijft, dan

   (a) moeten alle methoden uit de interface in de klasse aanwezig zijn
   (b) kun je alle methodes uit de interface in de klasse gebruiken
   (c) mag je sommige methoden uit de interface in de klasse herdefiniëren
   (d) erft de klasse alle member-variabelen van de interface

4. Een abstracte methode

   (a) heeft geen body en kan dus niet worden aangeroepen
   (b) heeft geen body en kan dus niet worden overridden
   (c) kan alleen maar een abstracte klasse als resultaattype hebben
   (d) kan alleen maar in een abstracte klasse staan

5. Aan het begin van een tekstfile staat vaak een header. Deze wordt gebruikt

   (a) zodat de methode `ReadToEnd` uit de klasse `TextReader` weet hoeveel characters er gelezen moeten worden

   (b) zodat de methode `ReadByte` uit de klasse `Stream` weet hoeveel characters er gelezen moeten worden

   (c) zodat de methode `Read` uit de klasse `StreamReader` weet hoeveel bytes er gelezen moeten worden

   (d) zodat de methode `ReadLine` uit de klasse `TextReader` weet met welke characters een regel wordt afgesloten

6. Een verschil tussen een `List` en een `IList` is

   (a) je kunt geen variabele declareren van het type `IList`

   (b) je kunt geen `new` object maken van het type `IList`

   (c) je kunt een `IList` niet doorlopen met `foreach`

   (d) je kunt een `IList` niet indexeren met vierkante haken

7. Het ophalen van een waarde op een bepaalde plaats kan voor een `List` met dezelfde notatie als voor een array omdat

   (a) de klasse `List` een indexer-property definieert

   (b) bij de implementatie van de klasse `List` een array is gebruikt

   (c) de notatie voor beide situaties in de taal is ingebouwd

   (d) de klasse `List` de methode `IndexOf` implementeert

8. Hoe kunnen menu-items van een typische tekstverwerker het best worden verdeeld tussen het MDI-containerwindow en het MDI-childwindow?

   (a) Save en New in de container, Find en Close in het child

   (b) Close en Exit in de container, Find en New in het child

   (c) Help en Exit in de container, New en Close in het child

   (d) New en Help in de container, Save en Close in het child

9. Na de declaratie `int[,,] a = new int[1,1,1];` heeft de array `a`

   (a) nog geen elementen

   (b) 1 element met de waarde 0

   (c) 3 elementen, elk met de waarde 1

   (d) 8 elementen met een nog niet bepaalde waarde

10. Aanroep van de methode `Start` van een `Thread`-object heeft tot gevolg dat

    (a) de methode `Run` wordt aangeroepen

    (b) de methode `Run` steeds opnieuw wordt aangeroepen

    (c) de methode die bij de constructor van `Thread` werd meegegeven wordt aangeroepen

    (d) de constructormethode van `Thread` wordt aangeroepen

11. *(telt voor 20%)*

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 gehele, positieve getallen. Het programma meldt wat het grootste van deze getallen is.

Daarna geeft het programma een overzicht hoe vaak elk van de cijfers 0 t/m 9 in alle getallen samen voorkomen, maar alleen als dat minstens één keer is.

Bijvoorbeeld:

```
$ test.exe 123 4125 229 53
grootste getal: 4125
cijfer 1 komt 2 keer voor
cijfer 2 komt 4 keer voor
cijfer 3 komt 2 keer voor
cijfer 4 komt 1 keer voor
cijfer 5 komt 2 keer voor
cijfer 9 komt 1 keer voor
```

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

Het programma zal onder andere de getallen in losse cijfers uit elkaar moeten peuteren (er zijn meerdere manieren om dat te doen, je mag zelf kiezen welke manier je gebruikt). Zorg er wel voor dat elk getal maar één keer uit elkaar hoeft te worden gepeuterd.

12. *(telt voor 40%: onderdeel a, b en e elk voor 4%; c, d, f en g elk voor 7%)*

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.

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

(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
```

(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.

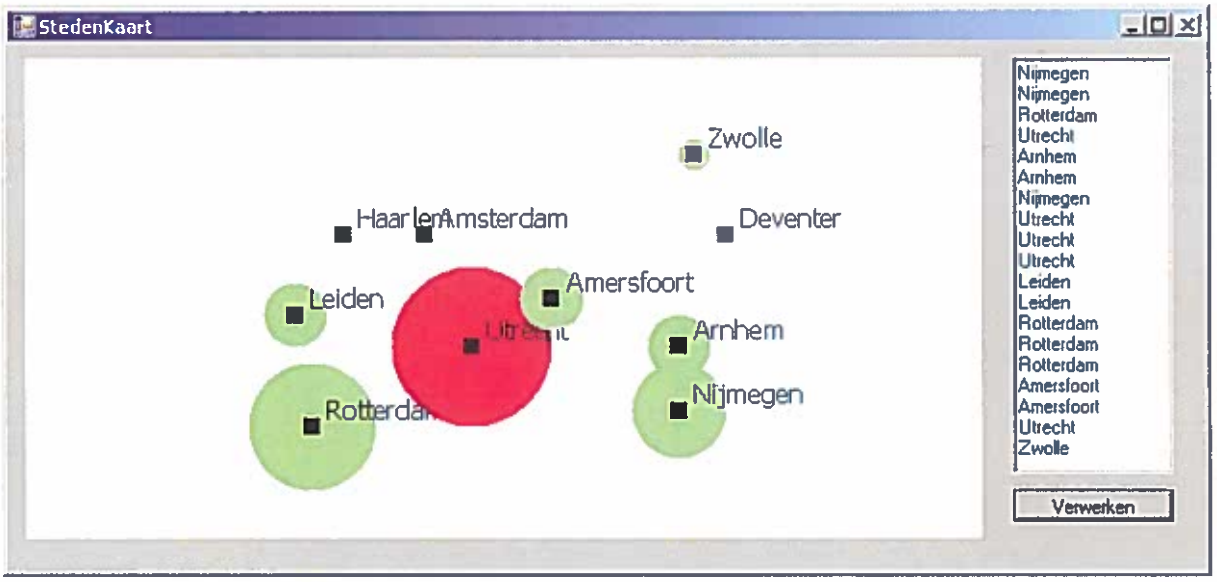
(e) Schrijf de methode `Teken` die alle steden in beeld brengt. De kleur van de cirkel rond elke stad moet zijn zoals de gegeven methode `KleurVan` dat voorschrijft (momenteel groen voor alle steden).

(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.

(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.



```
//////// Bijlage bij opgave 12 ////////////////////////////////////////
public class Scherm : Form
{   private TextBox namen;
    private 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.StaatOpScherm(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 + "\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
{   private string naam;
    private Point plek;
    static Font font = new Font("Tahoma", 12);

    public Stad(Point p, string s)
    {   this.plek = p;
        this.naam = s;
    }
    public string GeefNaam()
    {   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);
    }
}
```

```
//////////  vervolg Bijlage bij opgave 12 ////////////////////////////////////////
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 StaatOpScherm(Scherm s)
    {   this.schermWaaropIkSta = s;
    }

    public void Lees(string filenaam)
    {
       // 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)
```