DERDE DEELTENTAMEN IMPERATIEF PROGRAMMEREN - VERSIE 2
VRIJDAG 7 NOVEMBER 2014, 8.30-10.30 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 12 en 13 zijn programmeervragen, die meetellen voor 20 en 40 punten.

Meerkeuzevragen: de letter van het goede antwoord volstaat.
Belangrijk: dit is versie **2** van het tentamen, vermeld dat boven je antwoorden.

1. Bij het opstarten van een programma vanaf een commandoregel kun je achter de naam van het programma nog extra gegevens tikken, die je in het programma kunt gebruiken. Hoe krijg je deze gegevens in handen?

   (a) Door aanroep van de methode `Console.ReadLine()`
   (b) Via een string-parameter in de header van `Main`
   (c) Door gebruik van de property `System.Argv`
   (d) Via een string-array-parameter in de header van `Main`

2. Met een `delegate`-declaratie beschrijf je

   (a) Welke methoden er moeten zijn in een klasse die deze delegate implementeert
   (b) Welke members in een klasse worden geërfd
   (c) Het type van een bepaald soort methoden, zodat die als parameter gebruikt kunnen worden
   (d) Dat een member een verwijzing bevat naar een bepaald type object

3. Wat is het belangrijkste voordeel van een *abstract* methode boven een *virtual* methode?

   (a) Je hoeft hem in een subklasse niet te overriden
   (b) Het is minder tikwerk in de sourcecode
   (c) Je kunt niet vergeten om hem in een subklasse te overriden
   (d) Je kunt hem niet per ongeluk aanroepen terwijl dat nog geen zin heeft

4. Een abstracte klasse

   (a) Heeft geen membervariabelen
   (b) Heeft geen constructormethode
   (c) Heeft geen subklassen
   (d) Heeft geen superklasse

5. Welk van onderstaande uitspraken is *niet* waar?

   (a) UTF8 kost voor sommige teksten meer ruimte dan Unicode
   (b) UTF8 kan sommige teksten niet opslaan die Unicode wel kan opslaan
   (c) UTF8 kost voor sommige teksten minder ruimte dan Unicode
   (d) UTF8 kan sommige teksten wel opslaan die Latin1 niet kan opslaan

6. Een klasse *implementeert* een interface als

   (a) De in de interface gegenereerde events worden afgehandeld
   (b) De in de interface opgeworpen exceptions worden afgevangen
   (c) De in de interface beschreven methodes aanwezig zijn
   (d) De in de interface gedefinieerde methodes worden geërfd

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

   (a) Open en Help in de container, Save en Find in het child
   (b) Open en Close in de container, Save en Help in het child
   (c) Open en Save in de container, Help en Find in het child
   (d) Save en Close in de container, Open en Find in het child

8. Het keyword `value` kan worden gebruikt binnen de definitie van een property

   (a) in de `set`-minimethode, en heeft dan de waarde van de rechterkant van een toekenning aan de property
   (b) in de `get`-minimethode, en heeft dan de waarde van het resultaat van de property
   (c) in de `set`-minimethode, en heeft dan de waarde van de parameter van de property
   (d) in de `get`-minimethode, en heeft dan de waarde van de `private` variabele die door de property wordt afgeschermd

9. Na de declaratie `int[,,] a = new int[2,3,5];` wijst de variabele `a` naar een array

   (a) met 10 elementen
   (b) met 30 elementen
   (c) met 3 elementen
   (d) met een vrijwel onbeperkt aantal elementen

10. Het verschil tussen een *Collection* en een *List* is

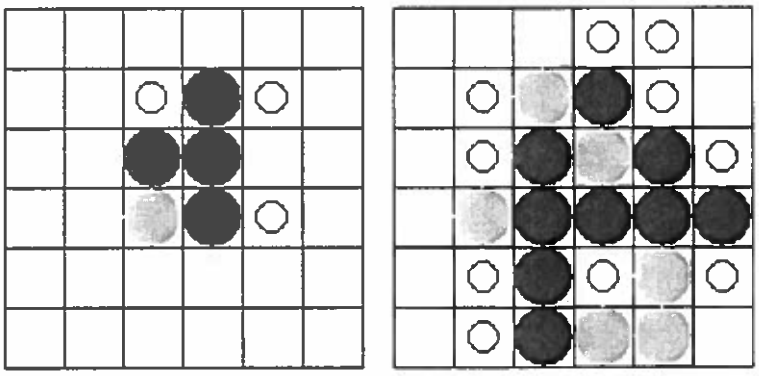
    (a) In een Collection is het aantal elementen niet bekend
    (b) In een Collection zitten geen dubbele elementen
    (c) In een Collection is het type van de elementen niet bekend
    (d) In een Collection hebben de elementen geen volgnummer

11. 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.

Gegeven zijn verder volgende methoden (die hoef je dus niet te schrijven):

```
public bool mag  (int kleur, int x, int y)
public bool sluit(int kleur, int x, int y, int dx, int dy)
```

De methode `sluit` test of speler `kleur`, door te zetten op veld (x, y), één of meer stenen van de tegenstander insluit in de richting (dx, dy), waarbij dx en dy 1, 0 of -1 zijn. De methode `mag` bepaalt of speler kleur mag zetten op veld (x, y).

De opgave: schrijf nu een methode

```
public void doeZet (int kleur, int x, int y)
```

die, als dat is toegestaan, de zet voor speler `kleur` op veld (x, y) uitvoert.

12. Met het programma in deze opgave kan de gebruiker een blokkenwereld tekenen. Op het scherm is een groot gedeelte, de 'wereld' zichtbaar waarin de gebruiker kan klikken. Ernaast staan een schuifregelaar, een tekstveld, en twee knoppen.

Als de gebruiker in de wereld klikt, ontstaat er gecentreerd op dat punt een vierkant. De diameter van het vierkant kan de gebruiker als getal intikken in het tekstveld.

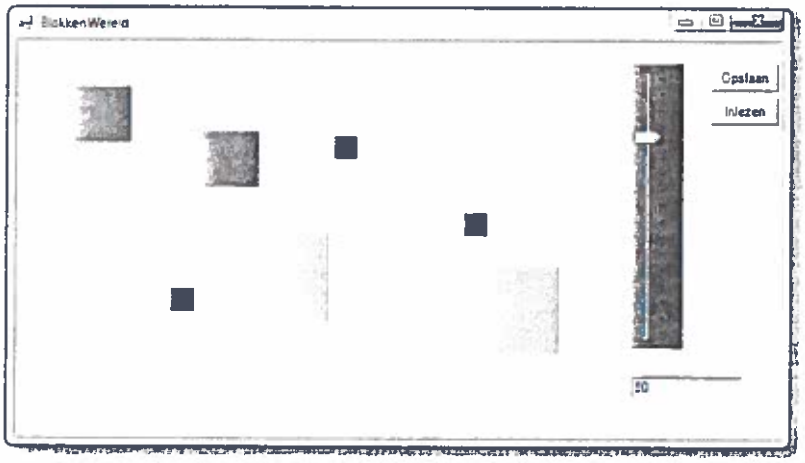
De vierkanten kunnen allerlei grijstonen hebben, variërend van wit tot zwart. Met de schuifregelaar kan de gebruiker de kleur van het vierkant instellen.

In de bijlage aan het eind van dit tentamen staat een deel van het programma gegeven. Het bestaat uit vier klassen: `Program`, `Scherm`, `Wereld`, en `Blok`. Een aantal onderdelen moet nog worden ingevuld.

(a) Schrijf de klasse `Blok`, zodat in een object van dit type alle relevante gegevens voor één blok worden opgeslagen. Zet daarin:
- de benodigde member-variabelen
- een constructormethode waarmee de members een waarde kunnen krijgen
- een read-only string-property, die de members samengevat als string teruggeeft
- een tweede constructormethode, die zo'n string weer uitpakt in de afzonderlijke members

(b) Schrijf de klasse `Wereld`. Zet daarin:
- de benodigde member-variabelen
- een constructormethode, die aangeroepen kan worden zoals dat in klasse `Scherm` gebeurt.
  Hint: let op, de methode heeft een parameter. Bewaar de waarde daarvan, want je gaat die later nog nodig hebben!
- overige methoden die nodig zijn om ervoor te zorgen dat op de aangeklikte plaatsen inderdaad blokken getoond worden

(c) Met de button 'opslaan' kan de hele situatie worden opgeslagen in een tekstbestand. Met de button 'inlezen' verdwijnt de huidige situatie, en wordt die vervangen door de eerder opgeslagen situatie. De te gebruiken filenaam staat gedeclareerd in de klasse `Scherm`. Schrijf de twee benodigde methoden, en geef aan in welke klasse die moeten staan.



Einde tentamen

// Bijlage bij opgave 12

```
static class Program
{   static void Main()
    {   Application.Run(new Scherm());
    }
}

public class Scherm : Form
{
    public TrackBar tb;
    public TextBox box;
    public string filenaam = "test.txt";

    public Scherm()
    {   this.Size = new Size(700, 400);
        this.Text = "BlokkenWereld";

        Wereld w = new Wereld(this);
        w.Location = new Point(20, 20);  w.Size = new Size(500, 300);
        w.BackColor = Color.White;
        this.Controls.Add(w);

        tb = new TrackBar();
        tb.Location = new Point(550, 20); tb.Size = new Size(40, 255);
        tb.Orientation = Orientation.Vertical;
        tb.BackColor = Color.Green;
        tb.Maximum = 255;
        tb.Value = 128;
        tb.TickFrequency = 16;
        this.Controls.Add(tb);

        box = new TextBox();
        box.Location = new Point(550, 300); box.Size = new Size(100, 24);
        box.Text = "50";
        this.Controls.Add(box);

        Button b1 = new Button();
        b1.Location = new Point(620, 20); b1.Size = new Size(60, 24);
        b1.Text = "Opslaan";
        b1.Click += w.opslaan;
        this.Controls.Add(b1);

        Button b2 = new Button();
        b2.Location = new Point(620, 50); b2.Size = new Size(60, 24);
        b2.Text = "Inlezen";
        b2.Click += w.inlezen;
        this.Controls.Add(b2);
    }
}

public class Wereld : UserControl
{
    // TODO
}

public class Blok
{
    // TODO
}
```