Aanvullende toets **Gameprogrammeren**
Woensdag 5 januari 2011, 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 opgaven en de lijst met standaardfuncties mag je houden (behalve als je heel vroeg vertrekt).
- Het tentamen bestaat uit 2 opgaven. Elke opgave telt voor 50% mee. Als je een deel van een opgave niet weet, probeer dan toch zo veel mogelijk op te schrijven!

Veel succes!

---

1. (a) *(2 punten)* Gegeven zijn de volgende twee declaraties van variabelen:

```
string s;
double waarde;
```

Iemand schrijft de volgende opdracht om de wortel van de waarde te berekenen en aan de gebruiker te tonen:

```
s = "De wortel is " + Math.Sqrt(waarde);
```

Maar als `waarde` negatief is wordt het programma afgebroken met een foutmelding.

In plaats daarvan willen we liever dat de tekst `onmogelijk` in de string bewaard wordt. Je kunt dit op twee manieren voor elkaar krijgen:

- Vooraf controleren of de foutsituatie zich gaat voordoen
- De wortel gewoon maar uitrekenen en de foutsituatie opvangen

Geef voor beide aanpakken aan hoe de opdracht er dan uit komt te zien.

(b) *(1 punt)* Wat is een drie-dimensionale array? Hoe declareer je zo'n drie-dimensionale array, en hoe kun je in het programma aangeven hoeveel waarden de array kan bevatten?

(c) *(2 punten)* Gegeven is een klasse `Data` met daarin een member-variabele `getallen`, en methoden die deze variabele een zinvolle waarde geeft:

```
class Data
{
    private double [] getallen;

    // hier staan de bestaande methoden
    // hier komt een nog te schrijven property
}
```

Schrijf een read-only property `Totaal` waarmee het totaal van de opgeslagen waarden bepaald kan worden.

(d) *(2 punten)* Hieronder staan vijf programma-fragmenten. Geef in elk van de gevallen aan *hoe vaak* de methode `iets` wordt aangeroepen. Licht het antwoord kort toe.

1	`for (x=0; x<5; x++)`
	`  this.iets();`
	`for (y=0; y<5; y++)`
	`  this.iets();`
		2	`for (x=0; x<5; x++)`
			`for (y=0; y<x; y++)`
			`  this.iets();`

3	`for (x=0; x<5; x++)`
	`{  this.iets();`
	`  x=x+1;`
	`}`
		4	`x=0;`
			`while (x<0) ;`
			`  this.iets();`
			`  x=x+1;`

5	`for (x=0; x<x; x++)` `  this.iets();`		

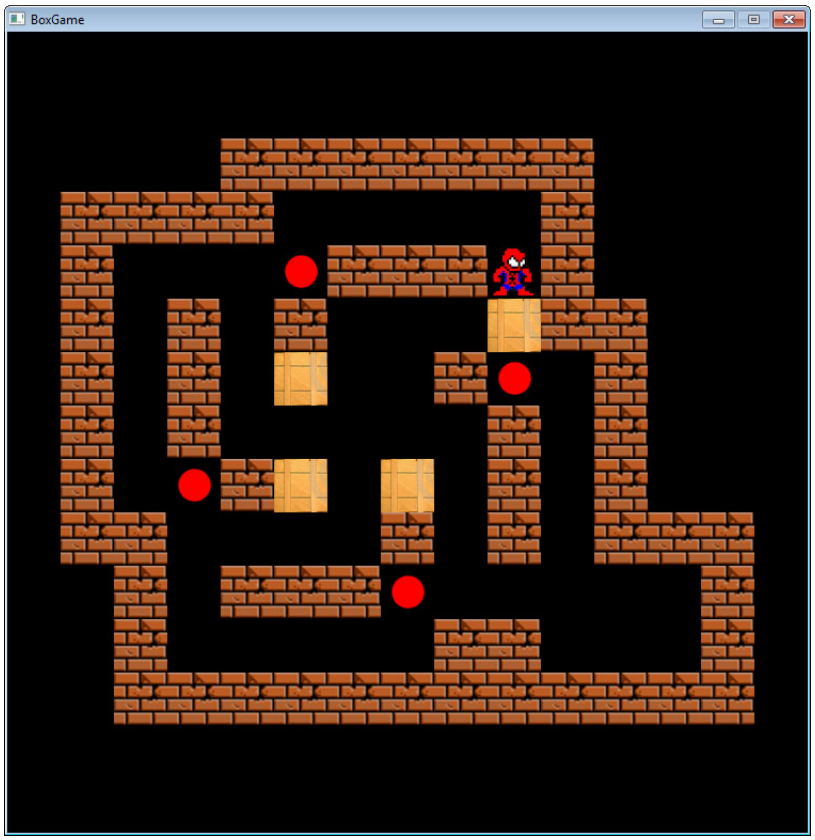
(e) *(3 punten)* Gegeven de volgende klassen:

```
abstract class A
{
    public abstract void methode1();

    public void methode2()
    {
        return;
    }
}
class B : A
{
    public override void methode1()
    {
        return;
    }

    public void methode3(A a)
    {
        a.methode1();
    }
}
```

Geef voor elk van de volgende opdrachten aan of ze mogen:

```
A obj;
obj = new A();
obj = new B();
obj.methode1();
obj.methode2();
obj.methode3(object);
B anderObject = (B)(new A());
A nogEenObject = (A)obj;
obj.methode3(nogEenObject);
A[] lijst;
lijst = new A[10];
lijst[0] = new A();
lijst[1] = new B();
List<A> nogEenLijst = new List<A>();
```

2. In deze opgave gaan we een eenvoudig 'box moving' puzzelspel maken. Het doel van het spel is om een aantal dozen te verschuiven zodanig dat ze op gegeven posities terechtkomen. Als speler kun je in een level bewegen met behulp van de vier pijltjestoetsen. Als je tegen een doos aanloopt, dan wordt deze één positie verschoven in dezelfde richting, behalve als aan de andere kant een muur staat. Als alle dozen op een doelpositie staan, dan is het level af. Een voorbeeld van een level is gegeven als volgt:



In dit level zijn vier dozen geplaatst die verschoven moeten worden naar de doelposities aangegeven met een stip. Om een voorbeeld te geven van hoe het spel reageert op spelersinvoer, laten we kijken wat er gebeurt als een speler de pijltjestoets omlaag indrukt. Onder de speler staat een doos, dus die wordt één positie naar beneden verschoven, oftewel, de doos komt op een doelpositie terecht. De speler komt terecht op de oude positie van de doos. Als de speler nu nog een keer op de pijltjestoets omlaag drukt, dan gebeurt er niks, want onder de doos staat nu een muur. Omdat aan de linkerkant van de doos nu ook een muur staan, kun je deze doos zelfs helemaal niet meer bewegen!

In ons spel worden levels geladen uit tekstbestanden. Het tekstbestand dat dit level omschrijft ziet er als volgt uit:

```
................
................
....WWWWWWWW....
.WWWW.....W....
.W...GWWWPW....
.W.W.W...BWW...
.W.W.B..WG.W...
.W.W.....W.W...
.W.GWB.B.W.W...
.WW....W.W.WWW.
..W.WWWG.....W.
..W.....WW...W.
..WWWWWWWWWWW.
................
................
```

Hierbij staan de verschillende karakters voor verschillende soorten tiles in het spel:

- een 'W' (wall) staat voor een muurtile
- een 'G' (goal) staat voor ee doeltile
- een 'B' (box) staat voor een doostile
- een 'P' (player) staat voor de startpositie van de speler
- een '.' staat voor een lege tile

Een deel van de game is al uitgewerkt, zie de appendix. We gebruiken een klasse `Tile` om de verschillende tiles te representeren in het level. Deze klasse wordt voor het gemak ook gebruikt om de speler en de dozen voor te stellen. In de `Level` klasse laden we de tiles uit een bestand, en handelen we de verdere gameplay af. De klasse `BoxGame` is de klasse die erft van de XNA `Game` klasse, en die maakt een `Level` instantie, en roept de juiste methoden aan in de `Update` en `Draw` methoden.

(a) *(2 punten)* De methode `laadTiles` in de `Level` klasse roept een methode `laadTile` aan die een tile bouwt aan de hand van het gelezen karakter. Deze methode zorgt er ook voor dat de lijst met dozen gevuld wordt en dat het speler object gemaakt wordt. Werk deze methode uit met behulp van de `switch` opdracht.

(b) *(2 punten)* Om makkelijk uit te kunnen vinden of een doos zich bevindt op een bepaalde (x, y)-positie willen we ook een methode `VindDoos` maken in de `Level` klasse. Deze methode krijgt als parameter twee `int`s (de x- en de y-positie) mee en die loopt door de lijst van dozen om te zien of een van die dozen op de gevraagde positie staat. Zo ja, dan wordt die doos als resultaat opgeleverd. Zo nee, dan levert de methode de waarde `null` op. Implementeer deze methode.
Let op: het gaat hier niet om de pixel-coördinaten, maar om de locatie in het tweedimensionale tileveld!

(c) *(4 punten)* De belangrijkste methode is de methode `VerplaatsSpeler` in de `Level` klasse. Gegeven een toets die de speler heeft ingedrukt, verplaatst deze methode de speler naar de nieuwe positie, indien mogelijk. Als er een doos op de gewenste positie van de speler staat, dan moet de doos verplaatst worden. Echter, dat mag alleen als de nieuwe positie van de doos géén muur of een andere doos is. Werk deze method uit. Maak hierbij gebruik van de methode `VindDoos` die we in het vorige onderdeel hebben uitgewerkt.
Hint: begin met het bepalen van de doelposities van de speler en een eventuele doos. Kijk daarna of je de speler en de doos (als die er staat) mag verplaatsen!

(d) *(1 punt)* Een level is afgemaakt als alle dozen op een doelpositie staan. Implementeer de methode `LevelAfgemaakt`. Deze methode geeft waarheidswaarde `true` als resultaat als het level is afgemaakt, en in alle andere gevallen `false`.

(e) *(1 punt)* Is het een goede ontwerpkeuze geweest om de `Tile` klasse ook te gebruiken voor de speler en de dozen? Zo ja, wat zijn de nadelen om het anders te doen? Zo nee, wat zou een betere keuze zijn geweest?

EINDE TOETS

Appendix: klassen

BoxGame

```
class BoxGame : Game
{
    static void Main()
    {
        BoxGame spel = new BoxGame();
        spel.Run();
    }

    SpriteBatch spriteBatch;
    Level huidigLevel;
    KeyboardState vorigeToetsenbordStatus;

    public BoxGame()
    {
        GraphicsDeviceManager graphics = new GraphicsDeviceManager(this);
        Content.RootDirectory = "Content";
        graphics.PreferredBackBufferWidth = 750;
        graphics.PreferredBackBufferHeight = 750;
    }

    protected override void LoadContent()
    {
        this.spriteBatch = new SpriteBatch(this.GraphicsDevice);
        huidigLevel = new Level(Content, "Content/level.txt");

    }
    protected override void Update(GameTime gameTime)
    {
        KeyboardState huidigeToetsenbordStatus = Keyboard.GetState();
        Keys[] keys = huidigeToetsenbordStatus.GetPressedKeys();
        foreach (Keys k in keys)
            if (vorigeToetsenbordStatus.IsKeyUp(k))
                huidigLevel.VerplaatsSpeler(k);
        vorigeToetsenbordStatus = huidigeToetsenbordStatus;
    }

    protected override void Draw(GameTime gameTime)
    {
        GraphicsDevice.Clear(Color.Black);
        spriteBatch.Begin();
        huidigLevel.Draw(gameTime, spriteBatch);
        spriteBatch.End();
    }

}
```

Level

```
class Level
{
    Tile[,] tiles;
    Texture2D doosSprite, muurSprite, spelerSprite, doelSprite, levelAfSprite;
    List<Tile> dozen = new List<Tile>();
    Tile speler;
    private ContentManager Content;

    public Level(ContentManager c, string path)
    {
        this.Content = c;
        this.doosSprite = Content.Load<Texture2D>("doos");
        this.muurSprite = Content.Load<Texture2D>("muur");
        this.spelerSprite = Content.Load<Texture2D>("verhuizer");
        this.doelSprite = Content.Load<Texture2D>("doel");
        this.levelAfSprite = Content.Load<Texture2D>("levelAf");
        this.laadTiles(path);
    }

    private void laadTiles(string pad)
    {
        List<string> tekstregels = new List<string>();
        StreamReader fileLezer = new StreamReader(pad);
        string regel = fileLezer.ReadLine();
        int breedte = regel.Length;
        while (regel != null)
        {
            tekstregels.Add(regel);
            if (regel.Length != breedte)
                throw new Exception("De lengte van de regels is verschillend.");
            regel = fileLezer.ReadLine();
        }
        this.tiles = new Tile[breedte, tekstregels.Count];
        for (int y = 0; y < tekstregels.Count; ++y)
            for (int x = 0; x < breedte; ++x)
                this.tiles[x, y] = laadTile(tekstregels[y][x], x, y);
    }

    private Tile laadTile(char tileType, int x, int y)
    {
        // TODO
    }

    public void VerplaatsSpeler(Keys key)
    {
        // TODO
    }

    public bool LevelAfgemaakt()
    {
        // TODO
    }

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch)
    {
        // code weggelaten
    }
}
```

Tile

```
enum TileType
{
    Achtergrond, Muur, Doel, Doos, Speler
}

class Tile
{
    private Texture2D sprite;
    private TileType type;
    int x, y;

    public const int Breedte = 50;
    public const int Hoogte = 50;

    public Tile(Texture2D sprite, TileType type, int x, int y)
    {
        this.sprite = sprite;
        this.type = type;
        this.x = x;
        this.y = y;
    }

    public TileType TileType
    {
        get { return this.type; }
    }

    public int X
    {
        get { return x; }
        set { if (value >= 0) x = value; }
    }

    public int Y
    {
        get { return y; }
        set { if (value >= 0) y = value; }
    }

    public virtual void Draw(SpriteBatch spriteBatch)
    {
        if (sprite != null)
        {
            Vector2 positie = new Vector2(x * Tile.Breedte, y * Tile.Hoogte);
            spriteBatch.Draw(this.sprite, positie, Color.White);
        }
    }
}
```