Uitwerkingen derde deeltentamen Gameprogrammeren
Vrijdag 7 november 2014, 11.00-13.00 uur

---

1. Deze opgave bestaat uit een aantal deelvragen. Houd het antwoord kort: één of twee zinnen per onderdeel kan al genoeg zijn.

   (a) *(2 punten)* De volgende klasse is onderdeel van de library GameManagement:

   ```
   public class InputHelper
   {
       // ...
   }
   ```

   Wat betekent het woord **public** voor de klasse? Wat is het gevolg als we dat woord weg zouden laten?

   Antwoord: het woord **public** betekent dat de klasse toegankelijk is buiten zijn eigen assembly. Met andere woorden: andere applicaties kunnen deze klasse ook gebruiken. Als we het woord **public** weg zouden laten, dan is de klasse standaard *intern*, oftewel: hij kan alleen in zijn eigen assembly (GameManagement) gebruikt worden.

   (b) *(2 punten)* Eén van de volgende drie declaraties-met-toekenningen is correct. Welke is dat, en waarom zijn de andere twee niet correct?

   ```
   List<int> a = new IList<int>(); // versie 1
   IList<int> b = new List<int>(); // versie 2
   IList<int> c = new IList<int>(); // versie 3
   ```

   Antwoord: In versie 1 en 3 wordt een object van IList gecreëerd, maar dat kan niet want IList is een interface en geen klasse. Versie 2 is correct, want klasse List implementeert de interface IList, en mag dus die rol vervullen.

   (c) *(4 punten)* Beschouw de volgende klassen:

   ```
   abstract class GameObject {
       protected Vector2 position;
       protected Vector2 velocity;
       public bool visible;
   }

   class SpriteGameObject : GameObject {
       protected Texture2D sprite;
   }
   ```

   Kruis aan welke van de volgende instructies uitgevoerd mogen worden in een methode van een andere, ongerelateerde klasse:

   - [X] `GameObject obj;`
   - [ ] `GameObject obj2 = new GameObject();`
   - [ ] `SpriteGameObject obj3 = new SpriteGameObject() as GameObject;`
   - [X] `GameObject[] lijst;`
   - [X] `GameObject[] lijst2 = new GameObject[10];`
   - [ ] `(new GameObject()).visible = true;`
   - [ ] `(new GameObject()).position = Vector2.Zero;`
   - [ ] `(new SpriteGameObject()).position = Vector2.Zero;`

   (d) *(2 punten)* Wat is een *partial class*? Wanneer is het nuttig om een partial class te gebruiken?

   Antwoord: Een partial class is een klasse waarvan de definitie verspreid is over meerdere files. Dit is nuttig om te gebruiken als een klasse heel groot is, maar het niet logisch is om de code te splitsen in verschillende klassen.

(e) *(4 punten)* Kruis aan welke van de volgende stellingen waar zijn:

- ☐ Elke **if**-opdracht kan worden omgeschreven naar een **switch** opdracht en vice versa.
- X De alpha-testfase is over het algemeen intern (door developers gedaan), de beta-testfase is extern (grote groep spelers).
- ☐ Excepties gebruik je vooral om het programma te stoppen in het geval dat in de code een bug zit die geintroduceerd is door de programmeur.
- ☐ Een abstracte klasse is een klasse die geen membervariabelen, en implementaties van properties en methoden bevat.
- X Een klasse mag meerdere interfaces implementeren, maar mag niet erven van meerdere abstracte klassen.
- X Objecten van het type **string** zijn immutable.
- ☐ Botsingen tussen game objecten worden afgehandeld in de Draw methode, want we weten pas of objecten botsen nadat ze getekend zijn.
- X Om een struct object te creëren moet je altijd gebruik maken van de **new** operator.

(f) *(2 punten)* Definieer een property Getal die de waarde van een private **int**-variabele getal uit dezelfde klasse opvraagt/wijzigt, maar daarbij garandeert dat de nieuwe waarde niet groter dan 100 wordt.

Antwoord:

```
int Getal
{
    get { return getal;}
    set { if (value<=100) getal=value; }
}
```

(g) *(2 punten)* Een **string** is in feite gewoon een array van waarden van het type **char**. Geef twee redenen waarom het toch handig is dat er een klasse **string** bestaat.

Antwoord: String heeft een aantal extra methoden en properties die we kunnen gebruiken zoals SubString of IndexOf. Ook heeft string een dubbele quote-notatie om de string op te bouwen.

(h) *(2 punten)* Er zijn twee manieren om een twee-dimensionale array te declareren:

```
int [,] eerste;
int [][] tweede;
```

Wat is het verschil? In welke situatie is het handig om de tweede te gebruiken?

Antwoord: eerste bevat alle elementen in het array-object, tweede is eigenlijk een array van verwijzingen naar arrays. Dat laatste is handig als de rijen van de array niet allemaal even lang zijn.

2. In de klasse **string** zitten onder andere de volgende methoden:

```
static bool IsNullOrWhiteSpace(string)
static int Compare(string, string)
```

(a) *(5 punten)* De naam van IsNullOrWhiteSpace spreekt voor zichzelf. Onder 'whitespace' verstaan we spaties, tab-tekens en newline-tekens. Schrijf deze methode, *zonder* gebruik te maken van de bestaande IsNullOrEmpty en IsNullOrWhiteSpace methoden.

Antwoord:

```
static bool IsNullOrWhiteSpace(string s)
{
    if (s==null) return true;
    for (int t=0; t<s.Length; t++)
        if (s[t]!=' ' && s[t]!='\t' && s[t]=='\n')
            return false;
    return true;
}
```

(b) *(5 punten)* De methode Compare in **string** levert 0 op als de twee parameters precies gelijk zijn. Hij levert een negatief getal op (bijvoorbeeld -1, maar iets anders mag ook) als de eerste parameter kleiner is dan de tweede, en een positief getal (bijvoorbeeld 1) als die groter is. Met kleiner en groter wordt hier de woordenboek-ordening bedoeld: de eerste letter waar de strings verschillen bepaalt de ordening (volgens de Unicodes van die letters). Is de ene string een beginstuk van de andere, dan is de kortste de kleinste. Spaties en leestekens tellen gewoon mee, die hoeven dus niet speciaal behandeld te worden.

Voorbeelden:

`string.Compare("aap", "noot")`	geeft een negatief getal, want 'a'<'n'
`string.Compare("noot", "nieten")`	geeft een positief getal, want 'o'>'i'
`string.Compare("niet", "nietmachine")`	geeft een negatief getal vanwege de lengte
`string.Compare("noot", "noot")`	geeft 0, want precies gelijk
`string.Compare("noot", "NOOT")`	geeft een positief getal, want 'n'>'N'

De methode neemt aan dat de parameters niet **null** zijn (en controleert dat ook niet). Schrijf deze methode, *zonder* gebruik te maken van de bestaande Compare en CompareTo methoden.

Antwoord:

```
static int Compare(string s, string t)
{
    int m = s.Length;
    int n = t.Length;
    for (int i=0; i<Math.Min(m,n); i++)
    {
        char c = s[i];
        char d = t[i];
        if (c!=d) return c−d;
    }
    return m−n;
}
```

3. (a) *(2 punten)* Indien de ReadLevel methode een exceptie werpt, dan crasht het programma. Geef aan hoe we de GameWorld constructor moeten aanpassen zodat dit niet meer gebeurt.

Antwoord: we moeten de aanroep van de ReadLevel methode in een **try**-blok plaatsen en met het **catch**-gedeelte de exceptie afvangen, als volgt:

```
try
{
    ReadLevel("Content/level.txt");
}
catch(IOException e)
{
    // doe iets met de foutmelding
}
```

(b) *(5 punten)* Werk de header en body van de ReadLevel methode uit. Gebruik de **switch** opdracht om de verschillende soorten objecten af te handelen.

Antwoord:

```
public void ReadLevel(string path)
{
    StreamReader fileLezer = new StreamReader(path);
    string regel = fileLezer.ReadLine();
    while (regel != null)
    {
        string[] obj = regel.Split(' ');
        Vector2 position = new Vector2(float.Parse(obj[1]), float.Parse(obj[2]));
        switch(obj[0])
        {
            case "asteroid": gameObjects.Add(new Asteroid(asteroid, position)); break;
            case "planet1": gameObjects.Add(new SpriteGameObject(planet1, position)); break;
            case "planet2": gameObjects.Add(new SpriteGameObject(planet2, position)); break;
            default: throw new IOException("Unknown object type: " + obj[0]);
        }
        regel = fileLezer.ReadLine();
    }
    fileLezer.Close();
}
```

(c) *(3 punten)* Werk de Update methode van Asteroid verder uit, zodat de positie wordt aangepast en wrapping plaatsvindt. Let hierbij op dat de wrapping netjes geïmplementeerd is, oftewel: asteroïden mogen niet verplaatst worden als ze nog niet helemaal het scherm uitgevlogen zijn, en asteroïden mogen niet al deels in het scherm staan als ze weer tevoorschijn komen.

Antwoord:

```
public override void Update(GameTime gameTime)
{
    this.velocity *= 1.00001f;
    base.Update(gameTime);
    if (this.position.X + this.sprite.Width < 0)
        this.position.X = Asteroids.Screen.X;
    else if (this.position.X > Asteroids.Screen.X)
        this.position.X = -this.sprite.Width;
    if (this.position.Y + this.sprite.Height < 0)
        this.position.Y = Asteroids.Screen.Y;
    else if (this.position.Y > Asteroids.Screen.Y)
        this.position.Y = -this.sprite.Height;
}
```