Derde deeltentamen Gameprogrammeren
Vrijdag 7 november 2014, 11.00-13.00 uur

Naam:	
Studentnummer:	

- Het tentamen bestaat uit 3 opgaven. Opgaven 1 levert 20 punten op, opgave 2 levert 10 punten op, en opgave 3 levert 10 punten op. Je cijfer is het totaal aantal punten gedeeld door 4. Als je een deel van een opgave niet weet, probeer dan toch zo veel mogelijk op te schrijven!
- Het is niet toegestaan om boeken, aantekeningen of ander materiaal te gebruiken, met uitzondering van de lijst met standaardklassen, -methoden, en -properties. Deze lijst na afloop graag weer inleveren.
- Je mag eventuele klassenbijlagen van de toets loshalen om niet steeds heen-en-weer te hoeven bladeren.

Veel succes!

---

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?

   (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
   ```

(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:

- ☐ `GameObject obj;`
- ☐ `GameObject obj2 = new GameObject();`
- ☐ `SpriteGameObject obj3 = new SpriteGameObject() as GameObject;`
- ☐ `GameObject[] lijst;`
- ☐ `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?

(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.
- ☐ 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.
- ☐ Een klasse mag meerdere interfaces implementeren, maar mag niet erven van meerdere abstracte klassen.
- ☐ 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.
- ☐ 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.

(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.

(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?

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.

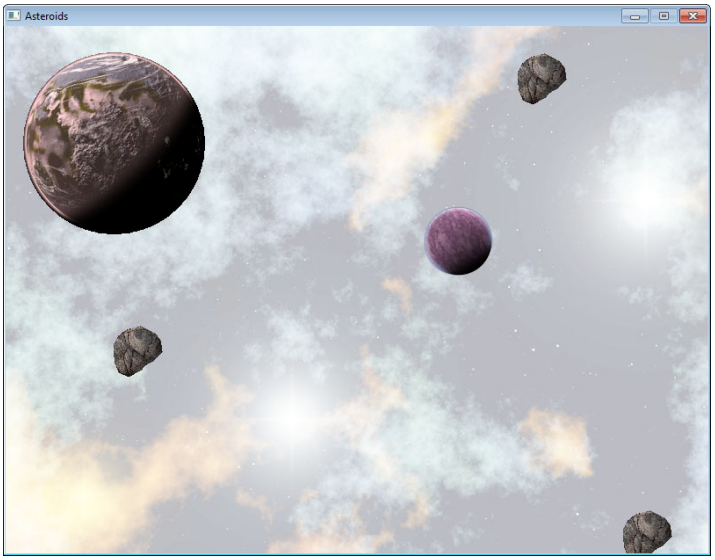
(b) *(5 punten)* De methode Compare in **string** levert 0 op als de twee parameters precies gelijk zijn. Hij levert een negatief getal op (bijvoorbeeld -1, maar iets anders mag ook) als de eerste parameter kleiner is dan de tweede, en een positief getal (bijvoorbeeld 1) als die groter is. Met kleiner 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.

3. In deze opgave gaan we een eenvoudige gamewereld bestaande uit planeten en rondvliegende asteroïden bouwen. Zo kan de gamewereld er bijvoorbeeld uitzien:



De hoeveelheid en initiële positie van de planeten en asteroïden worden geladen uit een tekstbestand. Dit is een voorbeeld van de inhoud van zo'n tekstbestand:

```
asteroid 120 340
asteroid 700 550
asteroid 580 30
planet1 460 200
planet2 20 30
```

Voor elke asteroïde of planeet wordt de x en y positie in het tekstbestand bewaard. Ook wordt aangegeven of het gaat om een asteroïde ('asteroid') een planeet type 1 ('planet1') of een planeet van type 2 ('planet2'). Het lezen van het level uit het tekstbestand gebeurt in de GameWorld klasse door de methode ReadLevel aan te roepen. Deze methode leest het tekstbestand uit en creëert de verschillende game objecten (planeten en asteroïden). Als de methode een onbekend type game object tegenkomt (oftewel iets anders dan 'asteroid', 'planet1' of 'planet2') tijdens het lezen van de file, dan werpt de methode een IOException.

(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.

(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.

In het begin krijgen alle asteroïden een willekeurige snelheid en richting. In de Update methode van Asteroid wordt deze snelheid vermenigvuldigd. Hierdoor gaan de asteroïden steeds sneller vliegen. Als we gewoon de positie van de asteroïde zouden aanpassen aan de hand van zijn snelheid, dan zouden alle asteroïden het scherm uitvliegen, waardoor de game niet zo interessant is. Wat we graag willen is dat als de asteroïde uit het scherm vliegt, hij aan de tegenovergestelde kant weer tevoorschijn komt. Dit effect wordt ook wel *wrapping* genoemd.

(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.

Appendix: klassen

Asteroids

```
1  public class Asteroids : Game
2  {
3      private GraphicsDeviceManager graphics; private SpriteBatch spriteBatch;
4      public static GameWorld gameWorld; public static Random random;
5      private static Vector2 screen;
6
7      static void Main() {
8          Asteroids game = new Asteroids();
9          game.Run();
10     }
11
12     public Asteroids() {
13         graphics = new GraphicsDeviceManager(this); screen = new Vector2(800, 600);
14         graphics.PreferredBackBufferHeight = (int)screen.Y;
15         graphics.PreferredBackBufferWidth = (int)screen.X;
16         Content.RootDirectory = "Content";
17     }
18     protected override void LoadContent() {
19         spriteBatch = new SpriteBatch(GraphicsDevice);
20         random = new Random();
21         gameWorld = new GameWorld(Content);
22     }
23
24     protected override void Update(GameTime gameTime) {
25         gameWorld.Update(gameTime);
26     }
27
28     protected override void Draw(GameTime gameTime) {
29         spriteBatch.Begin();
30         gameWorld.Draw(gameTime, spriteBatch);
31         spriteBatch.End();
32     }
33
34     public static Vector2 Screen {
35         get { return screen; }
36     }
37
38 }
```

SpriteGameObject

```
1  public class SpriteGameObject
2  {
3      protected Texture2D sprite;
4      protected Vector2 position, velocity;
5
6      public SpriteGameObject(Texture2D spr, Vector2 pos) {
7          this.sprite = spr; this.position = pos;
8      }
9      public virtual void Update(GameTime gameTime) {
10         this.position += this.velocity;
11     }
12     public virtual void Draw(GameTime gameTime, SpriteBatch spriteBatch) {
13         spriteBatch.Draw(this.sprite, this.position, Color.White);
14     }
15 }
```

GameWorld

```
public class GameWorld
{
    private Texture2D planet1, planet2, asteroid;
    private List<SpriteGameObject> gameObjects;

    public GameWorld(ContentManager Content) {
        asteroid = Content.Load<Texture2D>("asteroid");
        planet1 = Content.Load<Texture2D>("planet1");
        planet2 = Content.Load<Texture2D>("planet2");
        gameObjects = new List<SpriteGameObject>();
        gameObjects.Add(new SpriteGameObject(Content.Load<Texture2D>("background"), Vector2.Zero));
        ReadLevel("Content/level.txt");

        // TO DO: constructor uitbreiden/aanpassen (a)
    }

    // TO DO: ReadLevel methode (b)

    public void Update(GameTime gameTime) {
        foreach (SpriteGameObject obj in gameObjects)
            obj.Update(gameTime);
    }

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch) {
        foreach (SpriteGameObject obj in gameObjects)
            obj.Draw(gameTime, spriteBatch);
    }
}
```

Asteroid

```
class Asteroid : SpriteGameObject
{
    public Asteroid(Texture2D geladenSprite, Vector2 pos)
        : base(geladenSprite, pos) {
        this.velocity.X = (float)Asteroids.random.NextDouble() - 0.5f;
        this.velocity.Y = (float)Asteroids.random.NextDouble() - 0.5f;
        this.velocity *= (30.0f * (float)Asteroids.random.NextDouble());
    }

    public override void Update(GameTime gameTime) {
        this.velocity *= 1.00001f;
        // TO DO: positie berekenen (c)
    }
}
```

GAME OVER