Tweede deeltentamen **Gameprogrammeren**
Vrijdag 15 oktober 2010, 11.00-13.00 uur

- **Let op: Lever het tentamen in op twee bladen! Opgaven 1 en 2 moeten uitgewerkt worden op het eerste blad, opgaven 3 en 4 op het tweede blad.** Schrijf op elk ingeleverd blad je naam en je studentnummer.
- De opgaven en de lijst met standaardfuncties mag je houden (behalve als je heel vroeg vertrekt).
- Het tentamen bestaat uit 4 opgaven. Elke opgave 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!

Veel succes!

---

1. *(2 punten per deelvraag)* Deze opgave bestaat uit een aantal tekstvragen. Houd het antwoord kort: een of twee zinnen per onderdeel kan al genoeg zijn.

   (a) Wat is het verschil tussen `this` en `base`? Geef een voorbeeld van een situatie waarin het gebruik van `base` noodzakelijk is.

   (b) 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.

   (c) Wat betekent het dat een object *immutable* is? Geef een voorbeeld van een type object dat immutable is.

   (d) Wat is het verschil tussen een klasse en een object?

   (e) Geef een voorbeeld van hoe een ééndimensionale array in games gebruikt kan worden. Geef ook een voorbeeld van hoe een multidimensionale array in games gebruikt kan worden.

2. (a) *(5 punten)* Gegeven de volgende methodeheader:

   ```
   static bool IsWhiteSpace(string)
   ```

   Deze methode kijkt of een string bestaat uit alleen maar 'whitespace'. Onder 'whitespace' verstaan we spaties, tab-tekens en newline-tekens.

   Schrijf deze methode, *zonder* gebruik te maken van de bestaande `IsNullOrEmpty` en `IsNullOrWhiteSpace` methoden in de string-klasse.

   (b) *(5 punten)* In de klasse `String` zit onder andere de methode

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

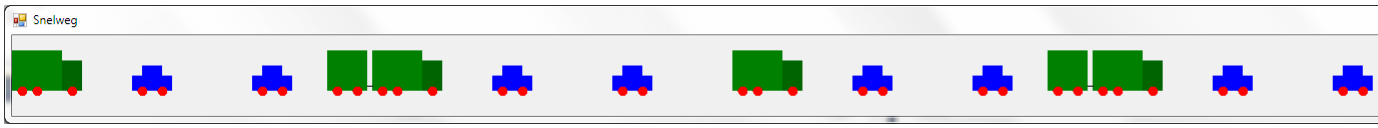
   De methode `Compare` levert 0 op als de twee parameters precies gelijk zijn. Hij levert een negatief getal op (bijvoorbeeld -1, maar iets anders mag ook) als de eerste parameter kleiner is dan de tweede, en een positief getal (bijvoorbeeld 1) als die groter is. Met kleiner en groter wordt hier de woordenboek-ordening 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'`

   Schrijf deze methode, *zonder* gebruik te maken van de bestaande `Compare` en `CompareTo` methoden.

3. Bekijk het onderstaande programma. Het moet een file auto's op de snelweg tekenen, zoals in de screendump hieronder. Elke derde auto is een vrachtwagen, en elke tweede vrachtwagen is een combinatie met aanhanger.




```
public class Snelweg : Game
{   public SpriteBatch spriteBatch;
    public Texture2D wiel, auto, vrachtwagen, verbinding, aanhanger;
    public Snelweg()
    {   graphics = new GraphicsDeviceManager(this);
        graphics.PreferredBackBufferWidth = 1800;
        graphics.PreferredBackBufferHeight = 80;
        Content.RootDirectory = "Content";
        // TODO: ontbrekend deel van de constructor
    }
    protected override void LoadContent()
    {
        spriteBatch = new SpriteBatch(this.GraphicsDevice);
        wiel = Content.Load<Texture2D>("wiel");
        auto = Content.Load<Texture2D>("auto");
        vrachtwagen = Content.Load<Texture2D>("vrachtwagen");
        verbinding = Content.Load<Texture2D>("verbinding");
        aanhanger = Content.Load<Texture2D>("aanhanger");
    }
    protected override void Draw(GameTime gameTime)
    {   GraphicsDevice.Clear(Color.Grey);
        spriteBatch.Begin();
        for (int t = 0; t < rijbaan.Length; t++)
            rijbaan[t].Draw(this, t*120, 60);
        spriteBatch.End();
    }
    static void Main()
    {
        Snelweg spel = new Snelweg();
        spel.Run();
    }
}
class MotorVoertuig
{
    public void Draw(Game g, int x, int y)
    {
    }
}
class PersonenAuto : MotorVoertuig
{   public void Draw(Game g, int x, int y)
    {   g.spriteBatch.Draw(g.auto, new Vector2(x, y - 30), Color.Blue);
        g.spriteBatch.Draw(g.wiel, new Vector2(x + 5, y - 10), Color.Red);
        g.spriteBatch.Draw(g.wiel, new Vector2(x + 25, y - 10), Color.Red);
    }
}
class Vrachtwagen : MotorVoertuig
{   public void Draw(Game g, int x, int y)
    {   g.spriteBatch.Draw(g.vrachtwagen, new Vector2(x, y - 45), Color.Green);
        g.spriteBatch.Draw(g.wiel, new Vector2(x + 5, y - 10), Color.Red);
        g.spriteBatch.Draw(g.wiel, new Vector2(x + 20, y - 10), Color.Red);
        g.spriteBatch.Draw(g.wiel, new Vector2(x + 55, y - 10), Color.Red);
    }
}
```

```
class Combinatie : Vrachtwagen
{   public void Draw(Game g, int x, int y)
    {   // de vrachtwagen
        g.spriteBatch.Draw(g.vrachtwagen, new Vector2(x, y - 45), Color.Green);
        g.spriteBatch.Draw(g.wiel, new Vector2(x + 5, y - 10), Color.Red);
        g.spriteBatch.Draw(g.wiel, new Vector2(x + 20, y - 10), Color.Red);
        g.spriteBatch.Draw(g.wiel, new Vector2(x + 55, y - 10), Color.Red);
        // de aanhanger
        g.spriteBatch.Draw(g.verbinding, new Vector2(x - 5, y - 10), Color.Black);
        g.spriteBatch.Draw(g.aanhanger, new Vector2(x - 45, y - 45), Color.Green);
        g.spriteBatch.Draw(g.wiel, new Vector2(x - 40, y - 10), Color.Red);
        g.spriteBatch.Draw(g.wiel, new Vector2(x - 20, y - 10), Color.Red);
    }
}
```

(a) *(1 punt)* Er ontbreekt nog een declaratie. Schrijf deze declaratie, en geef aan waar die moet staan.

(b) *(2 punten)* Schrijf het ontbrekende deel van de constructor van `Snelweg`.

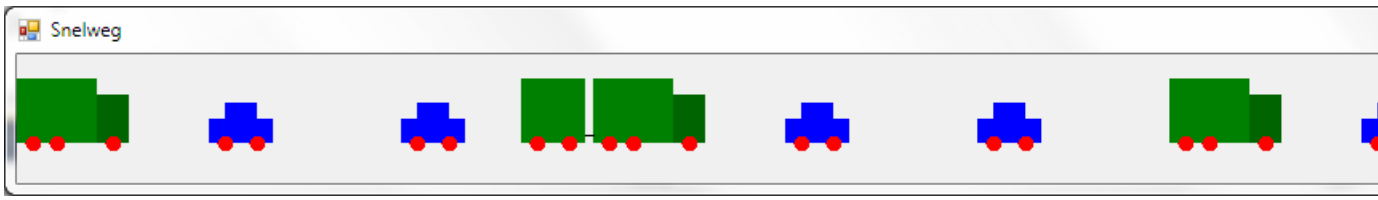
(c) *(2 punten)* In het gegeven programma zit nog een fout, waardoor er helemaal niets zichtbaar wordt. Hoe komt dat, en hoe kan de fout worden verbeterd?

(d) *(1 punt)* De programmeur heeft een flink stuk code met copy&paste gedupliceerd. Waarom is dat geen goed idee?

(e) *(2 punten)* Hoe had het dupliceren van de code het beste vermeden kunnen worden?

(f) *(2 punten)* We willen de object-georiënteerde opzet van het programma nog verder doorvoeren, zo dat ook de concepten 'wiel' en 'aanhanger' met klassen worden gemodelleerd. Hoe kan dat netjes worden aangepakt? Zorg ervoor dat code-duplicatie zo veel mogelijk vermeden kan worden, en dat er nooit door een programmeerfout een losse aanhanger op de weg terecht kan komen.

Je hoeft dit niet helemaal uit te programmeren; geef alleen aan welke klassen er komen, hoe hun subklasse-relatie is, en welke declaraties er in (bestaande en/of nieuwe) klassen komen te staan.



op de achterkant staat nog een opgave!

4. Beschouw de volgende klasse:

```
class GameObject
{
    protected Texture2D sprite;
    protected Vector2 positie, snelheid;

    public GameObject(Texture2D geladenSprite)
    {
        this.sprite = geladenSprite;
        this.positie = Vector2.Zero;
        this.snelheid = Vector2.Zero;
    }

    public virtual void Update()
    {
        this.positie += this.snelheid;
    }

    public virtual void Draw(SpriteBatch spriteBatch)
    {
        spriteBatch.Draw(this.sprite, this.positie, Color.White);
    }
}
```

In de volgende deelvragen gaan we een aantal methoden en properties toevoegen aan deze klasse. Schrijf alleen de opdrachten op die toegevoegd moeten worden, je hoeft niet steeds de hele klasse over te schrijven.

(a) *(3 punten)* Breid de `GameObject`-klasse uit met een property `Zichtbaar`. Deze property moeten we kunnen lezen en schrijven met een waarheidswaarde, en geeft aan of het game object getekend moet worden. Voeg membervariabelen toe indien nodig. Herschrijf ook de Draw-actie zodat rekening gehouden wordt met deze property.

(b) *(3 punten)* Schrijf een klasse `Diamant` die een subklasse is van de `GameObject`-klasse. Een speler kan met diamanten punten verdienen. Voeg een property `Punten` toe aan de `Diamant`-klasse waarmee de waarde van de diamant opgevraagd en veranderd kan worden. Let op dat de waarde van de diamant nooit kleiner dan 0 mag zijn. Voeg membervariabelen toe indien nodig. Standaard is een diamant 10 punten waard.

(c) *(2 punten)* In de `Diamant`-klasse willen we dat de Draw-actie ander gedrag vertoont. Namelijk als een diamant 50 punten of meer waard is, dan moet de sprite niet met een witte kleur (`Color.White`) getekend worden, maar met een rode kleur (`Color.Red`). Breid de `Diamant`-klasse uit zodat dit bewerkstelligd wordt.

(d) *(2 punten)* Zou je de membervariabelen van de `GameObject`-klasse ook als *private* in plaats van protected kunnen declareren? Zo nee: waarom niet? Zo ja: is dat een goed idee?

EINDE TENTAMEN