Tweede deeltentamen Gameprogrammeren
Vrijdag 18 oktober 2013, 11.00-13.00 uur

Naam:	
Studentnummer:	

- Het tentamen bestaat uit 3 opgaven. Opgaven 1 levert 10 punten op, opgave 2 levert 6 punten op, en opgave 3 levert 24 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.

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)* Gegeven de volgende methode:

```
// Neem aan dat x groter dan of gelijk aan y
int f(int x, int y)
{
    if (y == 0 || y == x)
        return 1;
    return f(x−1,y) + f(x−1, y−1);
}
```

   Wat is het resultaat van de aanroep f(f(4,2), 1)?

   (b) *(2 punten)* Stel dat de interface Demo is gedeclareerd, en een variabele test met die interface als type:

```
interface Demo
{
    public int Aantal();
}
Demo test;
```

   Het is niet mogelijk om deze variabele te initialiseren met

```
test = new Demo();
```

   i. Hoe kan deze variabele toch zinvol worden geïnitialiseerd (anders dan met **null**)?

ii. Waarom kan het handig zijn om test te declareren met Demo als type, in plaats van met het type van de expressie waarmee hij wordt geïnitialiseerd?

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

```
A obj = new A();
```

Deze instructie bestaat uit een aantal onderdelen. Schrijf hieronder achter elk onderdeel wat het onderdeel precies doet:

declaratie:

new operator:

constructor:

toekenning:

(d) *(2 punten)* Beschouw de volgende instructie:

```
IList<GameObject> gameObjects = new List<GameObject>();
```

Kruis aan welke van de volgende stellingen waar zijn:

- ☐ Het aantal elementen in de lijst gameObjects mag niet gewijzigd worden, want het geheugen is reeds gereserveerd.
- ☐ gameObjects mag objecten van het type GameObject bevatten, maar ook objecten die een subklasse van GameObject als type hebben.
- ☐ gameObjects mag objecten van elk type bevatten waarvoor geldt dat GameObject de klasse is die aan de top van de overervingshiërarchie van dat type staat.
- ☐ Deze instructie compileert niet, want gameObjects is van het type IList<GameObject> terwijl we een object toekennen van het type List<GameObject>.

2. *(6 punten)* Gegeven zijn de volgende klassedefinities:

```
class One
{
    int x;
    public One()
    {
        x = 0;
    }
    public void SetX(int a)
    {
        x = a;
    }
}
```

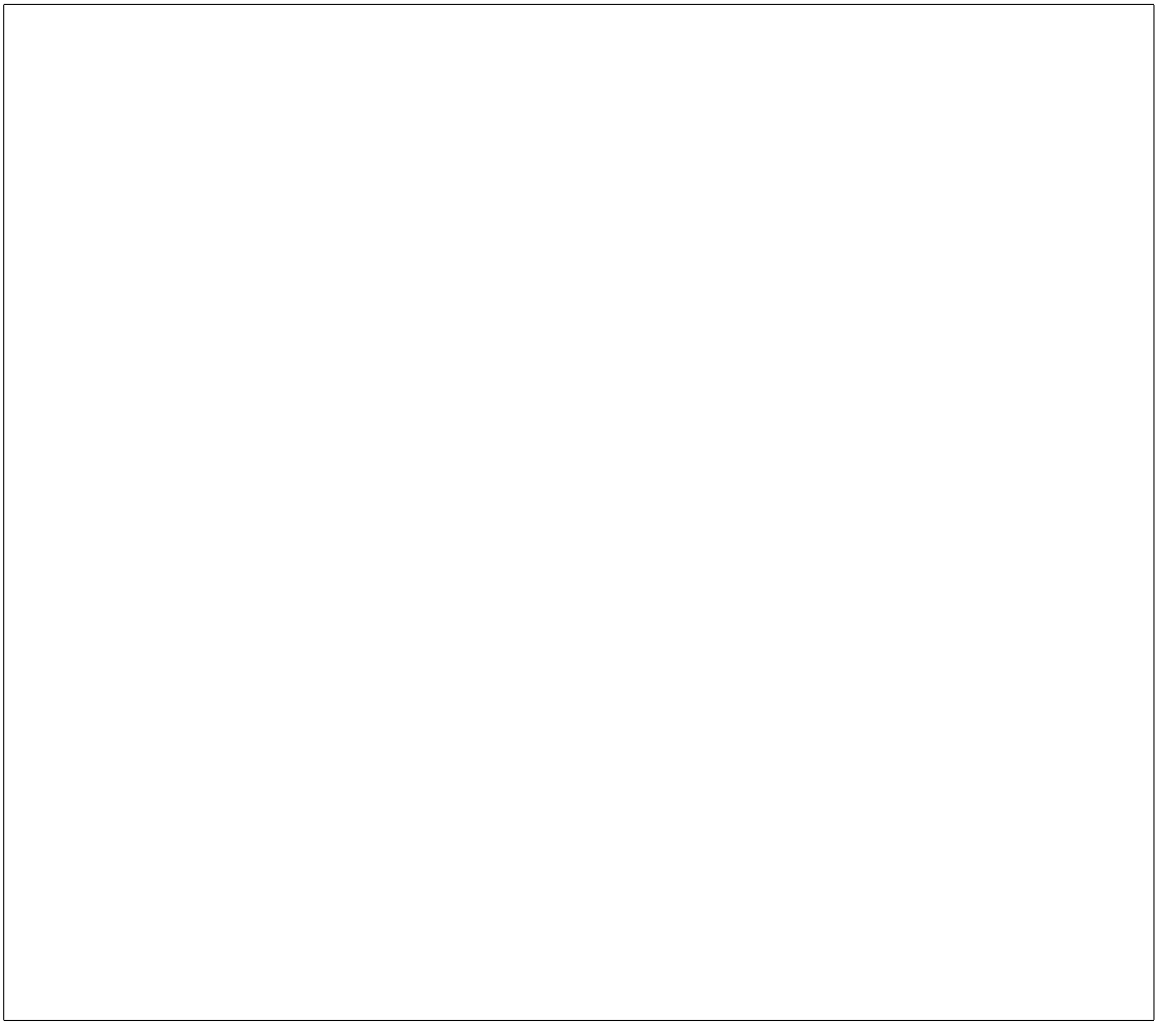
```
class Two
{
    int x;
    One o;

    public Two(One b, int c)
    {
        o = b;
        x = c+1;
    }
    public One GetO()
    {
        return o;
    }
}
```

```
class Three : One
{
    Two p, q;
    public Three()
    {
        p = new Two(new One(), 1);
        p.GetO().SetX(7);
        q = new Two(p.GetO(), 2);
        q.GetO().SetX(8);
        p = new Two(this, 3);
        p.GetO().SetX(9);
    }
}
```

Teken hoe het geheugen eruit ziet na het uitvoeren van de volgende instructie:

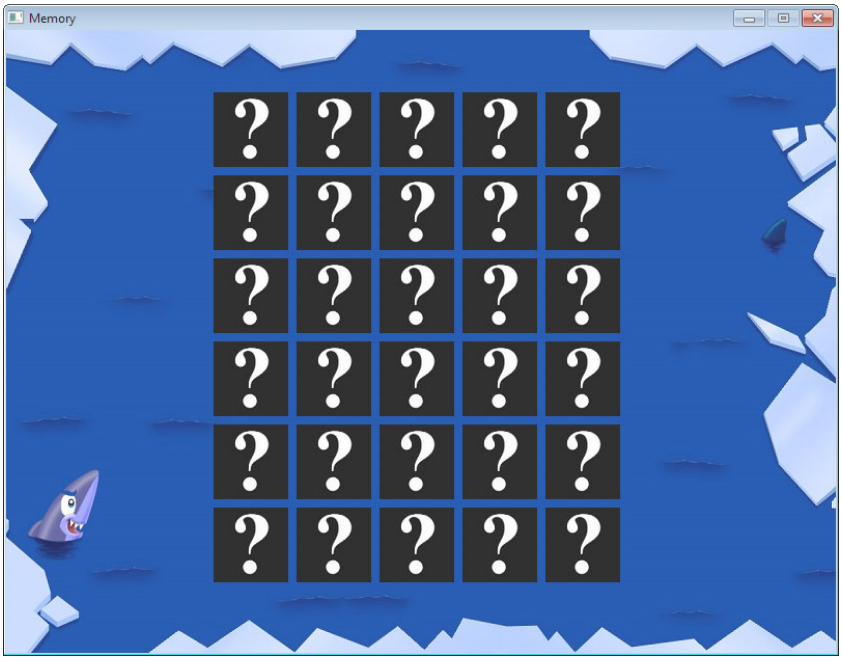
```
Three t = new Three();
```

Maak in je tekening onderscheid tussen de namen en de waarden van variabelen: de naam hoort naast het blok te staan, en de waarde in het blok. Verwijzingen naar objecten beginnen met een duidelijke stip in het blok van de referentievariabele, en de pijl wijst naar de rand van het object.



Opgave 3 vraagt een stukje programma. Kleine schrijffoutjes (hoofdletters, puntkomma's enz.) worden niet streng afgerekend, maar de elementen die de structuur van het programma bepalen (haakjes, accolades, aanhalingstekens enz.) zijn wel belangrijk. Schrijf die dus duidelijk en op de goede plaats op! Het is toegestaan (maar niet nodig) om C#-constructies die (nog) niet zijn behandeld toch te gebruiken.

3. In deze opgave gaan we de klassieke game "Memory" uitwerken. Het spel bestaat uit een grid van kaarten dat bij aanvang van het spel gevuld wordt met willekeurige kaarten. In de bijlage van deze toets vind je een aantal klassen die al deels ingevuld zijn. Kaarten hebben een voor- en een achterkant. Als het spel begint liggen alle kaarten met de achterkant naar boven. Dit is een screenshot van het spel:



Op elke kaart staat een plaatje, en de bedoeling van het spel is om telkens twee kaarten om te draaien door erop te klikken. Als de twee kaarten hetzelfde zijn, dan worden ze verwijderd. Als de kaarten een verschillend plaatje hebben, dan worden ze weer omgedraaid.

(a) *(1 punt)* In de GameWorld klasse mist nog een declaratie. Welke declaratie is dit?

(b) *(3 punten)* In de HandleInput methode van de MemoryCard klasse moet gekeken worden of de speler op de memory kaart geklikt heeft. Is dit het geval, dan moet de toestand van de kaart veranderd worden naar 'open' (door middel van de variabele Open). Werk deze methode uit (zie volgende pagina voor het antwoordvak).

(c) *(3 punten)* In de GameWorld klasse staan de variabelen die de omvang en plaatsing van het kaarten grid bepalen. In dit voorbeeld bestaat de grid uit 6 rijen en 5 kolommen. Elk element in de grid is 80 pixels breed, en 80 pixels hoog. Je mag aannemen dat sprites van de kaarten zelf (voor- en achterkant) diezelfde dimensies hebben. De hoogte en breedte is vastgelegd in de variable cellSize in de GameWorld klasse. De grid moet in het midden van het scherm getekend worden.

Schrijf een methode AddToGrid in de GameWorld klasse to die drie parameters meekrijgt: een kaart, een x-index, en een y-index. Deze methode voegt de memory kaart toe aan het grid op de x-index en y-index die als parameter zijn meegegeven. Daarnaast moet bij het toevoegen van de MemoryCard objecten aan de grid de juiste positie berekend worden.

(d) *(8 punten)* In de constructor van de GameWorld klasse moet het grid gevuld worden met willekeurige memory kaarten. Schrijf de instructies die dit bewerkstelligen. Let op: van elke soort kaart moet er een even aantal kaarten in het grid zitten, want anders kun je niet alle kaarten bij elkaar zoeken. Daarnaast moeten de kaarten in een willekeurige volgorde in het grid zitten. Je mag er vanuit gaan dat de grid bestaat uit een even aantal elementen. *Hint: vul eerst het grid met de juiste kaarten, en 'schud' het grid daarna zodat de kaarten in willekeurige volgorde liggen.*

(e) *(5 punten)* In de Update methode van GameWorld moeten we, indien er twee kaarten open liggen kijken of de kaarten hetzelfde plaatje voorstellen. Is dit het geval, dan worden de kaarten onzichtbaar. Als de kaarten niet hetzelfde plaatje voorstellen, dan moeten ze weer omgedraaid worden. Schrijf de instructies op die dit bewerkstelligen.

(f) *(2 punten)* De SpriteSheet klasse heeft een lees-schrijf property SheetIndex. Werk deze property uit. Zorg dat de sheet index nooit buiten het bereik van de sprite kan vallen.

(g) *(2 punten)* In de huidige implementatie van deze game gaat nog iets mis. De speler ziet namelijk nooit allebei de kaarten waarop hij/zij heeft geklikt. Waarom niet? Leg kort uit hoe je dit zou kunnen oplossen, je hoeft het niet daadwerkelijk uit te programmeren.

Appendix: klassen

Memory

```
class Memory : Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;
    InputHelper inputHelper;

    static Random random;
    static GameWorld gameWorld;
    static Vector2 screen;

    static void Main()
    {
        Memory game = new Memory();
        game.Run();
    }

    public Memory()
    {
        // code weggelaten
    }

    protected override void LoadContent()
    {
        spriteBatch = new SpriteBatch(GraphicsDevice);
        screen = new Vector2(GraphicsDevice.Viewport.Width, GraphicsDevice.Viewport.Height);
        gameWorld = new GameWorld(Content);
    }
    protected override void Update(GameTime gameTime)
    {
        inputHelper.Update();
        gameWorld.HandleInput(inputHelper);
        gameWorld.Update(gameTime);
    }
    protected override void Draw(GameTime gameTime)
    {
        GraphicsDevice.Clear(Color.White);
        spriteBatch.Begin();
        gameWorld.Draw(gameTime, spriteBatch);
        spriteBatch.End();
    }
    public static Vector2 Screen
    {
        get { return screen; }
    }
    public static Random Random
    {
        get { return random; }
    }
    public static GameWorld GameWorld
    {
        get { return gameWorld; }
    }
}
```

GameWorld

```
class GameWorld
{
    Texture2D background;
    int rows = 6, cols = 5, cellsize = 80;
    // TODO: a) declaratie toevoegen

    public GameWorld(ContentManager Content)
    {
        background = Content.Load<Texture2D>("spr_background");
        grid = new MemoryCard[cols, rows];

        // TODO: d) grid vullen met kaarten
    }

    // TODO: c) voeg een methode AddToGrid toe

    public void HandleInput(InputHelper inputHelper)
    {
        foreach (MemoryCard m in grid)
            m.HandleInput(inputHelper);
    }

    public void Update(GameTime gameTime)
    {
        // TODO: e) handel twee open kaarten af
    }

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch)
    {
        spriteBatch.Draw(background, Vector2.Zero, Color.White);
        for (int x = 0; x < cols; x++)
            for (int y = 0; y < rows; y++)
                grid[x, y].Draw(gameTime, spriteBatch);
    }
}
```

SpriteSheet

```
class SpriteSheet
{
    protected Texture2D sprite;
    protected int sheetIndex, sheetColumns, sheetRows;

    public SpriteSheet(ContentManager Content, string assetname, int sheetIndex = 0)
    {
        // code weggelaten
    }

    public void Draw(SpriteBatch spriteBatch, Vector2 position, Vector2 origin)
    {
        // code weggelaten
    }

    public int Width
    {
        get { return sprite.Width / sheetColumns; }
    }

    public int Height
    {
        get { return sprite.Height / sheetRows; }
    }

    public int SheetIndex
    {
        // TODO: f) property uitwerken
    }
}
```

InputHelper

```
public class InputHelper
{
    MouseState currentMouseState, previousMouseState;
    KeyboardState currentKeyboardState, previousKeyboardState;

    public void Update()
    { /* code weggelaten */ }

    public Vector2 MousePosition
    {
        get { return new Vector2(currentMouseState.X, currentMouseState.Y); }
    }

    public bool MouseLeftButtonPressed()
    {
        return currentMouseState.LeftButton == ButtonState.Pressed
               && previousMouseState.LeftButton == ButtonState.Released;
    }
}
```

MemoryCard

```
class MemoryCard
{
    SpriteSheet cardFront, cardBack;
    public bool Open, Visible;
    public Vector2 Position;

    public MemoryCard(ContentManager Content, int sheetIndex)
    {
        cardFront = new SpriteSheet(Content, "spr_card_front@8", sheetIndex);
        cardBack = new SpriteSheet(Content, "spr_card_back");
        Open = false;
        Visible = true;
    }

    public void HandleInput(InputHelper inputHelper)
    {
        // TODO: b) klik afhandelen
    }

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch)
    {
        if (!Visible)
            return;
        if (Open)
            cardFront.Draw(spriteBatch, Position, Vector2.Zero);
        else
            cardBack.Draw(spriteBatch, Position, Vector2.Zero);
    }

    public int SheetIndex
    {
        get { return this.cardFront.SheetIndex; }
    }
}
```

EINDE TOETS