Eerste deeltentamen Gameprogrammeren
Vrijdag 26 september 2014, 8.30-10.30 uur

Naam:	
Studentnummer:	

- 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!
- 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 vragen. Houd het antwoord kort: één of twee zinnen per onderdeel kan al genoeg zijn.

   (a) *(2 punten)* Wat is een statische membervariabele? Geef een voorbeeld in de context van games waarin een statische membervariabele nuttig is.

   (b) *(3 punten)* Kruis aan welke van de volgende stellingen waar zijn:

   - ☐ Elke klasse moet een statische methode hebben die de naam `Main` draagt.
   - ☐ Een object is een groepje variabelen.
   - ☐ Objecten met een klasse als type worden als verwijzing doorgegeven, objecten met een struct als type worden als waarde doorgegeven.
   - ☐ Een methode met een lege body heeft altijd als resultaatwaarde **void**.
   - ☐ Een opdracht is een stukje programmacode met een waarde.
   - ☐ Elke imperatieve taal is ook procedureel.

(c) *(2 punten)* Als we de waarde van een integer variabele `i` willen toekennen aan een double variabele `d`, dan doen we dat met de toekenning `d = i`. Andersom (toekennen van double waarde `d` aan `i`) ziet er iets anders uit: `i = (int)d;`. In beide gevallen vindt een typeconversie plaatst. Wat is een ander woord voor zo'n typeconversie? En waarom moet er in het tweede geval (**int**) voor de rechterzijde van de toekenning geschreven worden?

(d) *(3 punten)* Hieronder staan zes programma-fragmenten. Je mag er vanuit gaan dat alle variabelen die gebruikt worden vantevoren gedeclareerd en van het type **int** zijn. Geef in elk van de gevallen aan *hoe vaak* de methode `iets` wordt aangeroepen. Licht het antwoord kort toe.

1	**for** (x=1; x<=6; x+=2)
	  this.iets();
	for (y=0; y>5; y++)
	  this.iets();	

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

3	**for** (x=0; x<10; x++)
	  if (x % 2 == 0)
	  {
	  this.iets();
	  }	

4	**while** (**false**);
	  if (**true**)
	  this.iets();	

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

6	**for** (x=1; x<18; x+=x)
	  for (y=0; y<x; y++)
	  this.iets();	

2. *($\frac{1}{2}$ punt per goed antwoord)* Hieronder staat 20 fragmenten uit een programma. Schrijf in de tabel hieronder achter elk programmafragment één daarbij passende letter, als volgt:

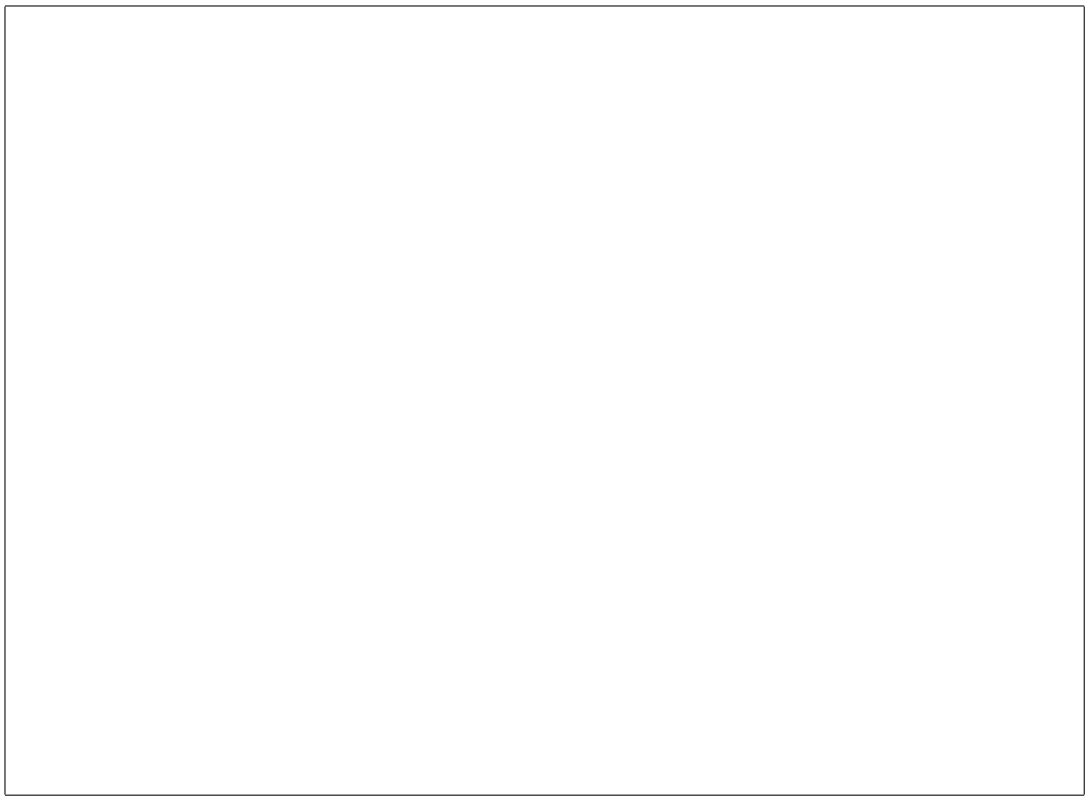
   - **T** als het programmafragment een **type** is
   - **E** als het programmafragment een **expressie** (maar geen constante) is
   - **O** als het programmafragment een **opdracht** is
   - **D** als het programmafragment een **declaratie** is
   - **C** als het programmafragment een **constante** (en dus ook een expressie) is
   - **X** als het programmafragment geen van bovenstaande dingen is

int i, j;		position.Zero;		Color.White		**true**=**false**	
(**float**)		'\\'		Game		**for**(;**false**;);	
Vector2 position;		**new** Random()		1 + 2 == 3		i = Color.Black.R;	
new Vector2(1, 2) * 3		i<5!=j>5		SpriteBatch		1E1	
while(**true**) **while**(**false**);		/*hallo		(**bool**)**true**		i = j++;	

Opgave 3 en 4 vragen 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. Je hoeft niet aan te geven welke **using***-opdrachten nodig zijn om de klassen te kunnen gebruiken.*

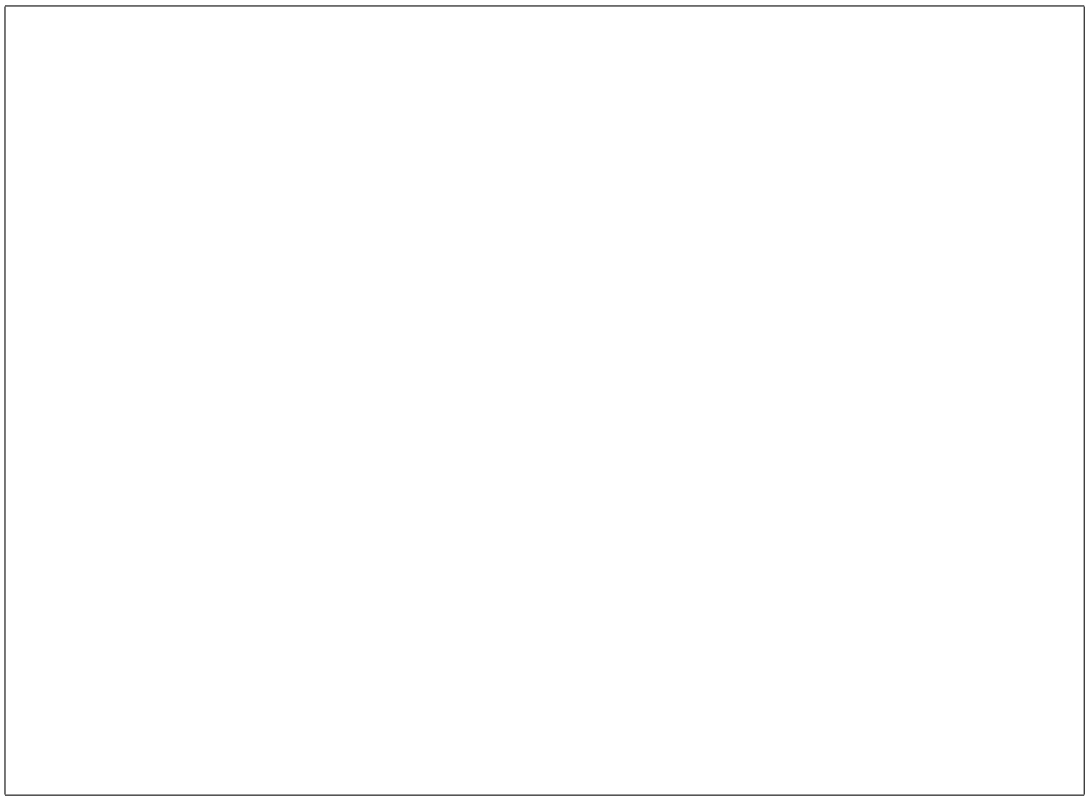
3. (a) *(5 punten)* Schrijf een methode cijfer met twee getallen als parameter. Als de eerste parameter 0 is, geeft de methode het *laatste cijfer* van de tweede parameter terug, als de eerste parameter 1 is, geeft de methode het *voorlaatste cijfer* van de tweede parameter terug, als de eerste parameter 2 is, geeft de methode het *op twee na laatste cijfer* van de tweede parameter terug, enzovoorts.

   Bijvoorbeeld: cijfer(0,456) geeft 6 terug, cijfer(2,98765) geeft 7 terug, en cijfer(4,1234) geeft 0 terug. Je mag zonder controle aannemen dat beide parameters niet negatief zijn.

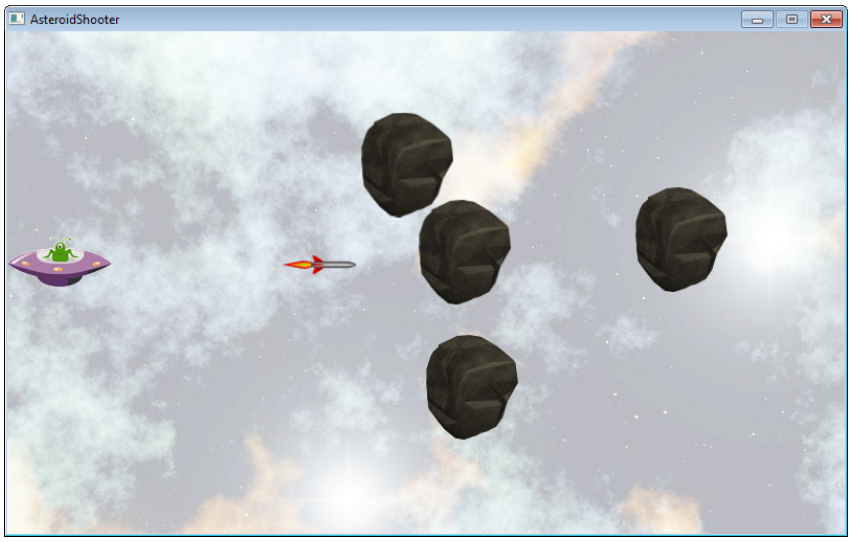
   

   (b) *(5 punten)* Bankrekeningnummers (met uitzondering van de vroegere gironummers) bestaan uit 9 cijfers. Maar niet elk getal van 9 cijfers is een geldig rekeningnummer. Om te controleren of er geen tikfouten zijn gemaakt bij het invoeren van rekeningnummers, doet online banking software de volgende controle: Het eerste cijfer wordt vermenigvuldigd met 9, het tweede cijfer wordt vermenigvuldigd met 8, het derde cijfer met 7, enzovoort, en het laatste cijfer met 1. Alle uitkomsten worden opgeteld. Als het totaal deelbaar is door 11, is het een geldig rekeningnummer. Bijvoorbeeld: voor de controle van 839801149 wordt uitgerekend: $8 \times 9 + 3 \times 8 + 9 \times 7 + 8 \times 6 + 0 \times 5 + 1 \times 4 + 1 \times 3 + 4 \times 2 + 9 \times 1 = 231$, en dat is inderdaad deelbaar door 11.

   Schrijf een methode controleer met een getal als parameter, dat teruggeeft of dat getal een geldig bankrekeningnummer is. Vermijd daarbij om 9 maal vrijwel dezelfde expressie op te schrijven; gebruik in plaats daarvan een C#-opdracht om de regelmaat uit te buiten. Je mag hierbij de methode cijfer uit de vorige deelvraag hergebruiken.



4. In het spel "Asteroid Shooter" is het de bedoeling om asteroïden kapot te schieten. Je bestuurt een ufo die altijd links in het scherm getekend wordt. De ufo volgt de muispointer, maar hij kan alleen verticaal bewegen. De asteroïden verschijnen rechts in het scherm en vliegen naar links. Door te klikken schiet de ufo een raket af. Als de raket botst met een van de asteroïden, dan verdwijnt de asteroïde van het scherm en de raket wordt ge-reset zodat hij weer afgeschoten kan worden. Er zijn maximaal vier asteroïden tegelijk zichtbaar. In de appendix van deze toets staan de belangrijkste klassen die gebruikt worden in deze game. Dit is een screenshot van het spel:



Naast de klassen in de appendix is een klasse InputHelper beschikbaar die onder andere de volgende methoden en properties heeft:

- **public** Vector2 MousePosition (property)
- **public bool** MouseLeftButtonPressed() (methode)

(a) *(1 punt)* Wat is het type van het object waar **this** naar wijst in regel 15 in de klasse AsteroidShooter?

(b) *(1 punt)* De ufo beweegt van boven naar beneden mee met de muispointer, maar blijft altijd aan de linkerkant van het scherm (zie ook de screenshot), en beweegt nooit uit het scherm. Schrijf de body van de HandleInput methode van de Ufo klasse op.

(c) *(2 punten)* De Missile klasse heeft ook een HandleInput methode. Als de speler op de linkermuisknop klikt, dan moet de raket een positieve x-snelheid krijgen van 400, maar dat mag alleen als de raket niet al aan het vliegen is. Werk de body van deze methode uit.

(d) *(3 punten)* De Update methode van Missile moet ervoor zorgen dat de positie van de raket wordt bijgewerkt. Werk de body van deze methode uit. Zorg dat alle mogelijke gevallen afgehandeld worden, zoals wat er gebeurt als de raket uit het scherm vliegt. Als de raket nog niet is afgeschoten, dan moet hij achter de ufo getekend worden, zodat hij niet zichtbaar is. Let op: we handelen in deze methode geen botsingen met asteroïden af.

(e) *(3 punten)* In de Update methode van Asteroid moet een aantal dingen gebeuren. Ten eerste moet de positie van de asteroïde aan de hand van de snelheid worden aangepast. Als de asteroïde links uit het scherm gevlogen is, dan moet hij een nieuwe (willekeurige) positie krijgen rechts buiten het scherm. Om ervoor te zorgen dat asteroïden niet altijd op hetzelfde moment het scherm in en uit vliegen moet deze nieuwe positie niet altijd onmiddellijk toegekend worden, maar na een willekeurig aantal iteraties van de game loop. Als de asteroïde botst met een raket, dan moeten de asteroïde en de raket ge-reset worden. Werk de body van deze methode uit.

Appendix: klassen

AsteroidShooter

```
1  class AsteroidShooter : Game {
2      SpriteBatch spriteBatch;
3      InputHelper inputHelper;
4      static Random random;
5      static GameWorld gameWorld;
6      static Vector2 screen;
7
8      static void Main() {
9          AsteroidShooter game = new AsteroidShooter();
10         game.Run();
11     }
12
13     public AsteroidShooter() {
14         Content.RootDirectory = "Content";
15         GraphicsDeviceManager graphics = new GraphicsDeviceManager(this);
16         random = new Random();
17         inputHelper = new InputHelper();
18     }
19
20     protected override void LoadContent() {
21         spriteBatch = new SpriteBatch(GraphicsDevice);
22         screen = new Vector2(GraphicsDevice.Viewport.Width, GraphicsDevice.Viewport.Height);
23         gameWorld = new GameWorld(Content);
24     }
25
26     protected override void Update(GameTime gameTime) {
27         inputHelper.Update();
28         gameWorld.HandleInput(inputHelper);
29         gameWorld.Update(gameTime);
30     }
31
32     protected override void Draw(GameTime gameTime) {
33         GraphicsDevice.Clear(Color.White);
34         spriteBatch.Begin();
35         gameWorld.Draw(gameTime, spriteBatch);
36         spriteBatch.End();
37     }
38
39     public static Vector2 Screen {
40         get { return screen; }
41     }
42
43     public static Random Random {
44         get { return random; }
45     }
46
47     public static GameWorld GameWorld {
48         get { return gameWorld; }
49     }
50 }
```

GameWorld

```
1  class GameWorld
2  {
3      Texture2D background;
4      Ufo ufo;
5      Asteroid ast1, ast2, ast3, ast4;
6      Missile missile;
7
8      public GameWorld(ContentManager Content)
9      {
10         background = Content.Load<Texture2D>("background");
11         ufo = new Ufo(Content);
12         ast1 = new Asteroid(Content);
13         ast2 = new Asteroid(Content);
14         ast3 = new Asteroid(Content);
15         ast4 = new Asteroid(Content);
16         missile = new Missile(Content);
17     }
18
19     public void HandleInput(InputHelper inputHelper)
20     {
21         ufo.HandleInput(inputHelper);
22         missile.HandleInput(inputHelper);
23     }
24
25     public void Update(GameTime gameTime)
26     {
27         ast1.Update(gameTime);
28         ast2.Update(gameTime);
29         ast3.Update(gameTime);
30         ast4.Update(gameTime);
31         missile.Update(gameTime);
32     }
33
34     public void Draw(GameTime gameTime, SpriteBatch spriteBatch)
35     {
36         spriteBatch.Draw(background, Vector2.Zero, Color.White);
37         ast1.Draw(gameTime, spriteBatch);
38         ast2.Draw(gameTime, spriteBatch);
39         ast3.Draw(gameTime, spriteBatch);
40         ast4.Draw(gameTime, spriteBatch);
41         missile.Draw(gameTime, spriteBatch);
42         ufo.Draw(gameTime, spriteBatch);
43     }
44
45     public Ufo Ufo
46     {
47         get { return ufo; }
48     }
49
50     public Missile Missile
51     {
52         get { return missile; }
53     }
54 }
```

Asteroid

```
class Asteroid
{
    Texture2D sprite;
    Vector2 position, velocity;

    public Asteroid(ContentManager Content) {
        sprite = Content.Load<Texture2D>("rock");
        Reset();
    }

    public void Update(GameTime gameTime) {
        // TODO (e)
    }

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch) {
        spriteBatch.Draw(sprite, position, Color.White);
    }

    public void Reset() {
        position = new Vector2(-sprite.Height, 0);
        velocity = new Vector2(-200, 0);
    }

    public bool CollidesWith(Missile m) {
        // code weggelaten
    }
}
```

Ufo

```
class Ufo
{
    Texture2D sprite;
    Vector2 position;

    public Ufo(ContentManager Content) {
        sprite = Content.Load<Texture2D>("ufo");
        position = Vector2.Zero;
    }

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

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch) {
        spriteBatch.Draw(sprite, position, Color.White);
    }

    public Vector2 Position
    {
        get { return position; }
    }
}
```

Missile

```
1  class Missile
2  {
3      Texture2D sprite;
4      Vector2 position, velocity;
5
6      public Missile(ContentManager Content) {
7          sprite = Content.Load<Texture2D>("missile");
8          Reset();
9      }
10
11     public void HandleInput(InputHelper inputHelper) {
12         // TODO (c)
13     }
14
15     public void Update(GameTime gameTime) {
16         // TODO (d)
17     }
18
19     public void Draw(GameTime gameTime, SpriteBatch spriteBatch) {
20         spriteBatch.Draw(sprite, position, Color.White);
21     }
22
23     public void Reset() {
24         velocity = Vector2.Zero;
25     }
26 }
```

EINDE TOETS