Tweede deeltentamen Gameprogrammeren
Vrijdag 19 oktober 2012, 11.00-13.00 uur

Naam:	
Studentnummer:	

- Het tentamen bestaat uit 3 opgaven. Opgaven 1 en 2 leveren allebei 10 punten op. Opgave 3 levert 20 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!
- De lijst met standaardfuncties 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 is een klasse Data met daarin een membervariabele getallen, en methoden die deze variabele een zinvolle waarde geeft:

   ```
   class Data
   {
       private double [] getallen;

       // hier staan de bestaande methoden
       // hier komt een nog te schrijven property
   }
   ```

   Schrijf een read-only property Totaal waarmee het totaal van de opgeslagen waarden bepaald kan worden.

   (b) *(2 punten)* Kruis aan welke van de volgende stellingen waar zijn:

   - ☐ Een klasse mag meerdere interfaces implementeren.
   - ☐ Een klasse mag een subklasse zijn van slechts één klasse.
   - ☐ Als het woord **virtual** voor een methode staat, dan is de methode niet geïmplementeerd in die klasse.
   - ☐ **foreach** mag alleen in combinatie met Collection objecten gebruikt worden (dus bijvoorbeeld niet om door arrays te lopen).

(c) *(3 punten)* Hieronder staan vijf 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=0; x<5; x++)
	  this.iets();
	for (y=0; y<=5; y++)
	  this.iets();	

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

3	**for** (x=0; x<5; x++)
	  while (x % 2 == 0)
	  {
	  this.iets();
	  x++;
	  }	

4	x=0;
	while (x<0) ;
	  this.iets();
	  x=x+1;	

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

6	**for** (x=20; x>=0; x/=3)
	  for (y=1; y<=x; y++)
	  this.iets();	

(d) *(1 punt)* Wat is het verschil tussen **base** en **this**?

(e) *(2 punten)* Klasse B is een subklasse van A. Kruis aan welke van de volgende stellingen waar zijn:

- ☐ A erft van B.
- ☐ Elk object van het type B is ook van het type A.
- ☐ Elk object van het type A heeft een membervariabele van het type B.
- ☐ A is de parent klasse van B.

2. Gegeven zijn de volgende klassedefinities:

```
class A {
    public int var1;
    protected int var2;
    private void method1() { }
    public virtual void method2() {}
    public void method3() { }
}

class B : A {
    protected int var3;
    public override void method2() {}
}

class C : B {
    public int var4;
    public override void method2() {}
}
```

(a) *(5 punten)* Kruis aan welke van de volgende opdrachten *niet* mogen worden uitgevoerd in de body van method2 in de klasse C:

- ☐ **this**.method1();
- ☐ **this**.method2();
- ☐ **this**.method3();
- ☐ **base**.method1();
- ☐ **base**.method2();
- ☐ **base**.method3();
- ☐ **int** x1 = **base**.var1;
- ☐ **int** y1 = **base**.var4;
- ☐ **int** x2 = **this**.var1;
- ☐ **int** y2 = **this**.var4;

(b) *(5 punten)* De volgende opdrachten staan in een methode iets van een andere (ongerelateerde) klasse:

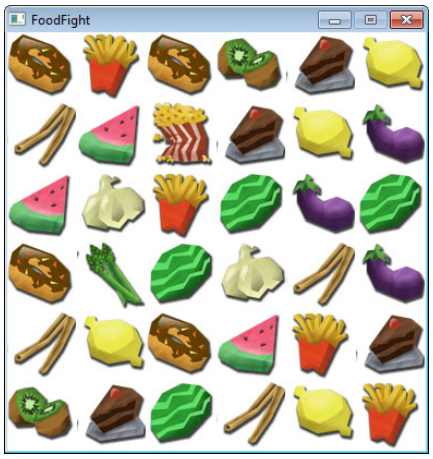
```
A obj1 = new C();
C obj2 = new C();
B obj3 = obj2 as B;
```

Kruis aan welke van de volgende opdrachten *niet* mogen worden uitgevoerd in de body van de methode iets:

- ☐ obj1.method1();
- ☐ **int** x = obj1.var1;
- ☐ **int** y = obj1.var2;
- ☐ obj2.method1();
- ☐ obj2.method2();
- ☐ **int** z = obj2.var4;
- ☐ obj3.method1();
- ☐ obj3.var1;
- ☐ obj3.var3;
- ☐ **int** z2 = obj3.var4;

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 game "Food Fight" uitwerken. Het spel bestaat uit een grid van game objecten (verschillende soorten voedsel) dat bij het opstarten willekeurig gevuld wordt. In de bijlage van deze toets vind je een aantal klassen die al deels ingevuld zijn. Dit is een voorbeeld van hoe het spel er vlak na het opstarten uit kan zien:



Het doel van het spel is ervoor zorgen dat dezelfde soorten voedsel naast elkaar of boven elkaar komen te staan. Als dat gebeurt, dan worden die objecten van het speelveld gehaald, en worden er op de lege plekken nieuwe objecten geplaatst. De enige manier om objecten te verplaatsen is door ze te verwisselen met hun *linkerbuurman*. Bijvoorbeeld: je zou kunnen klikken op het stukje taart in rij één, kolom vijf. Die wordt dan verwisseld met zijn linkerbuurman (de kiwi), en nu bevinden zich twee stukjes taart boven elkaar. Die worden daarna beide verwijderd, en weer vervangen door twee nieuwe (willekeurige) objecten.

Het kan ook gebeuren dat door het verwisselen van twee objecten een rij of kolom ontstaat van meer dan twee gelijke objecten. Bijvoorbeeld: als de meloen en de aubergine in rij drie, kolom vijf en zes, verwisseld worden, dan hebben we drie aubergines boven elkaar, en twee meloenen naast elkaar. Al die objecten moeten dan verwijderd worden en vervangen door nieuwe objecten. In de volgende screenshot zijn een paar voorbeelden opgenomen van combinaties die in de game kunnen ontstaan en waar de game mee om moet kunnen gaan:

(a) *(2 punten)* In de constructor van de GameWorld klasse mist nog een instructie. Welke instructie is dit? Tussen welke regelnummers hoort die instructie te staan?

De sprite met alle voedselobjecten erin is een *sprite sheet*. Een SpriteGameObject instantie kan hiermee omgaan, want daarin wordt bijgehouden hoeveel kolommen en rijen de sheet heeft, en wat de huidige sheet index is. Het aantal kolommen en rijen in een sheet is verwerkt in de filenaam. Bijvoorbeeld: de file "food@7x2" is een sheet met zeven kolommen en twee rijen. De file "background" is een enkele sprite (oftewel: een sheet met één rij en één kolom). Tenslotte: de file "onoffbutton@2" is een sprite sheet met één rij en twee kolommen (oftewel een *sprite strip*).

(b) *(4 punten)* De code die het aantal rijen en kolommen uit de filenaam haalt hoort in de constructor van SpriteGameObject te staan. Schrijf de missende instructies op. Let op: SpriteGameObject moet met *alle* soorten sprites om kunnen gaan (enkel, strip, of sheet).

(c) *(4 punten)* In de HandleInput methode van GameWorld wordt het klikken op een van de objecten in de grid afgehandeld. Als dat gebeurt, dan moet het object met zijn linkerbuurman verwisseld worden. Werk deze methode uit. Let op: we kijken nu enkel naar het verwisselen van twee objecten. Het vervangen van dezelfde objecten die naast elkaar staan door nieuwe objecten gebeurt in de Update methode (zie de laatste twee onderdelen van deze vraag).

In de Update methode van GameWorld zorgen we ervoor dat groepen van dezelfde objecten verwijderd en weer vervangen worden door nieuwe (willekeurige) voedsel objecten. Dit gebeurt in twee stappen. In de eerste stap wordt het hele grid doorgelopen, en wordt voor elk object bewaard of het object vervangen moet worden of niet. Deze informatie wordt bewaard in de lokale variabele toreplace (zie de GameWorld klasse in de bijlage).

(d) *(6 punten)* Schrijf de instructies uit die door het grid lopen en voor elk object in de variabele toreplace markeren of het object vervangen moet worden of niet. Let op dat je rekening houdt met alle mogelijkheden (voor een paar voorbeelden, zie de vorige screenshot).

(e) *(4 punten)* De laatste stap is het vervangen van alle gemarkeerde objecten door nieuwe (willekeurige) objecten. Schrijf de instructies op die dit bewerkstelligen.

Appendix: klassen

FoodFight

```
1  class FoodFight : Game
2  {
3      GraphicsDeviceManager graphics;
4      SpriteBatch spriteBatch;
5      InputHelper inputHelper;
6      GameWorld gameWorld;
7      static Random random;
8
9      static void Main()
10     {
11         FoodFight game = new FoodFight();
12         game.Run();
13     }
14
15     public FoodFight()
16     {
17         Content.RootDirectory = "Content";
18         IsMouseVisible = true;
19         graphics = new GraphicsDeviceManager(this);
20         graphics.PreferredBackBufferWidth = 384;
21         graphics.PreferredBackBufferHeight = 384;
22         random = new Random();
23         inputHelper = new InputHelper();
24     }
25
26     protected override void LoadContent()
27     {
28         spriteBatch = new SpriteBatch(GraphicsDevice);
29         gameWorld = new GameWorld(Content);
30     }
31
32     protected override void Update(GameTime gameTime)
33     {
34         inputHelper.Update();
35         gameWorld.HandleInput(inputHelper);
36         gameWorld.Update(gameTime);
37     }
38
39     protected override void Draw(GameTime gameTime)
40     {
41         GraphicsDevice.Clear(Color.White);
42         spriteBatch.Begin();
43         gameWorld.Draw(gameTime, spriteBatch);
44         spriteBatch.End();
45     }
46
47     public static Random Random
48     {
49         get { return random; }
50     }
51 }
```

GameWorld

```
1  class GameWorld
2  {
3      SpriteGameObject[,] foodgrid;
4      int grid_width, grid_height;
5      int rows, cols;
6      ContentManager Content;
7
8      public GameWorld(ContentManager Content)
9      {
10         this.Content = Content;
11         grid_width = 64;
12         grid_height = 64;
13         rows = 6; cols = 6;
14         for (int x = 0; x < cols; x++)
15             for (int y = 0; y < rows; y++)
16                 foodgrid[x, y] = new SpriteGameObject("food@7x2", Content, FoodFight.Random.Next(14));
17     }
18
19     public void HandleInput(InputHelper inputHelper)
20     {
21         // TODO (c)
22     }
23
24     public void Update(GameTime gameTime)
25     {
26         bool[,] toreplace = new bool[cols, rows];
27
28         // TODO: markeer te vervangen objecten (d)
29
30         // TODO: vervang alle gemarkeerde objecten door nieuwe (willekeurige) objecten (e)
31     }
32
33     public void Draw(GameTime gameTime, SpriteBatch spriteBatch)
34     {
35         for (int x = 0; x < cols; x++)
36             for (int y = 0; y < rows; y++)
37                 foodgrid[x, y].Draw(gameTime, spriteBatch, new Vector2(x * grid_width, y * grid_height));
38     }
39 }
```

SpriteGameObject

```
1  public class SpriteGameObject
2  {
3      protected Texture2D sprite;
4      protected int sheet_index, sheet_columns, sheet_rows;
5
6      public SpriteGameObject(string assetname, ContentManager Content, int sheetIndex = 0)
7      {
8          sprite = Content.Load<Texture2D>(assetname);
9          this.sheet_index = sheetIndex;
10         this.sheet_columns = 1;
11         this.sheet_rows = 1;
12
13         // TODO: haal de sheet dimensies uit de filenaam (b)
14     }
15
16     public void Draw(GameTime gameTime, SpriteBatch spriteBatch, Vector2 position)
17     {
18         int column_index = sheet_index % sheet_columns;
19         int row_index = sheet_index / sheet_columns % sheet_rows;
20         int width = sprite.Width / sheet_columns;
21         int height = sprite.Height / sheet_rows;
22         Rectangle spritePart = new Rectangle(column_index * width, row_index * height, width, height);
23         spriteBatch.Draw(sprite, position, spritePart, Color.White);
24     }
25
26     public int SheetIndex
27     {
28         get { return this.sheet_index; }
29     }
30 }
```

InputHelper

```
1  public class InputHelper
2  {
3      MouseState currentMouseState, previousMouseState;
4      KeyboardState currentKeyboardState, previousKeyboardState;
5
6      public void Update()
7      { // code weggelaten }
8
9      public Vector2 MousePosition
10     {
11         get { return new Vector2(currentMouseState.X, currentMouseState.Y); }
12     }
13
14     public bool MouseLeftButtonPressed()
15     {
16         return currentMouseState.LeftButton == ButtonState.Pressed
17                 && previousMouseState.LeftButton == ButtonState.Released;
18     }
19 }
```

EINDE TOETS