Uitwerkingen tweede deeltentamen Gameprogrammeren
Vrijdag 19 oktober 2012, 11.00-13.00 uur

---

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.

   Antwoord:

   ```
   public double Totaal
   {
       get
       { double res = 0.0;
          for (int t=0; t<getallen.Length; t++)
               res += getallen[t];
          return res;
       }
   }
   ```

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

   - X Een klasse mag meerdere interfaces implementeren.
   - X 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();
		11 keer (5 keer voor elke x, en dan nog eens 6 keer voor elke y (inclusief y == 5))

2	**for** (x=0; x<5; x++) **for** (y=0; y<x; y++) **this**.iets();	10 keer (0+1+2+3+4 keer)

3	**for** (x=0; x<5; x++)
	  while (x % 2 == 0)
	  {
	  this.iets();
	  x++;
	  }
		3 keer (voor x is 0, 2 en 4)

4	x=0;
	while (x<0) ;
	  this.iets();
	  x=x+1;
		1 keer (de **while** opdracht heeft een lege body, daarna is iets aan de beurt)

5	**for** (x=0; x<x; x++) **this**.iets();	0 keer (de voorwaarde is meteen **false**)

6	**for** (x=20; x>=0; x/=3)
	  for (y=1; y<=x; y++)
	  this.iets();
		28 keer (20+6+2 keer)

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

Antwoord: **base** en **this** in een methode wijzen allebei naar de huidige instantie waar de methode op wordt aangeroepen, maar **base** wijst naar het deel van de instantie dat is gedefinieerd in de superklasse.

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

- ☐ A erft van B.
- X 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.
- X 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:

- X `this.method1();`
- ☐ `this.method2();`
- ☐ `this.method3();`
- X `base.method1();`
- ☐ `base.method2();`
- ☐ `base.method3();`
- ☐ `int x1 = base.var1;`
- X `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:

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

Antwoord: de volgende instructie mist nog (initialisatie van grid):

```
foodgrid = new SpriteGameObject[cols, rows];
```

Deze instructie hoort tussen regelnummers 13 en 14.

(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).

Antwoord:

```
string[] assetSplit = assetname.Split('@');
if (assetSplit.Length > 1)
{
    string sheetNrData = assetSplit[assetSplit.Length - 1];
    string[] colrow = sheetNrData.Split('x');
    this.sheet_columns = int.Parse(colrow[0]);
    if (colrow.Length == 2)
        this.sheet_rows = int.Parse(colrow[1]);
}
```

(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).

Antwoord:

```
if (inputHelper.MouseLeftButtonPressed())
{
    int xpos = (int)inputHelper.MousePosition.X / grid_width;
    int ypos = (int)inputHelper.MousePosition.Y / grid_height;
    if (xpos > 0)
    {
        SpriteGameObject tmp = foodgrid[xpos, ypos];
        foodgrid[xpos, ypos] = foodgrid[xpos - 1, ypos];
        foodgrid[xpos - 1, ypos] = tmp;
    }
}
```

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).

Antwoord:

```
for (int x = 0; x < cols; x++)
    for (int y = 0; y < rows; y++)
    {
        if (y > 0 && foodgrid[x, y].SheetIndex == foodgrid[x, y − 1].SheetIndex)
        {
            toreplace[x, y] = true;
            toreplace[x, y − 1] = true;
        }
        if (x > 0 && foodgrid[x, y].SheetIndex == foodgrid[x − 1, y].SheetIndex)
        {
            toreplace[x, y] = true;
            toreplace[x − 1, y] = true;
        }
    }
```

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

Antwoord:

```
for (int x = 0; x < cols; x++)
    for (int y = 0; y < rows; y++)
        if (toreplace[x,y])
            foodgrid[x, y] = new SpriteGameObject("food@7x2", Content, FoodFight.Random.Next(14));
```