Uitwerkingen eerste deeltentamen Gameprogrammeren
Vrijdag 27 september 2013, 8.30-10.30 uur

---

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 wordt er verstaan onder de *syntax* van een (programmeer)taal-constructie? En wat is de *semantiek* van een taal-constructie?

   Antwoord: De syntax van een taalconstructie is de grammaticale opbouw. De semantiek is de betekenis ervan.

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

   - X Elk C# programma moet één klasse hebben met een statische methode die de naam `Main` draagt.
   - ☐ Een klasse is een groepje variabelen. *Nee, een object is een groepje variabelen. Een object heeft als type een klasse.*
   - ☐ Als het woord **new** gebruikt wordt om een object te creëren, dan weten we dat het object als type een klasse heeft. *Nee, het type kan ook een struct zijn.*
   - ☐ Het aanroepen van een methode met als resultaatwaarde **void** is een expressie. *Nee, een instructie, want* **void** *betekent 'geen resultaatwaarde'.*
   - X Het Halting probleem is onoplosbaar.
   - ☐ Scripttalen zijn nooit declaratief. *Nee, het feit dat iets een scripttaal is zegt niets over het soort taal, alleen over de manier waarop de taal gebruikt wordt.*

   (c) *(2 punten)* De vertaling van een C#-programma naar machinecode gebeurt in twee stappen. Eerst wordt het programma vertaald naar een intermediate language, en die wordt vervolgens weer vertaald naar machinecode. Noem twee voordelen van het vertalen in twee stappen.

   Antwoord: (1) Het vertalen van een *intermediate language* naar platformspecifieke machinecode is veel eenvoudiger dan het vertalen van een hoog-niveau programmeertaal naar diezelfde machinecode. (2) Een tweede voordeel is dat we op deze manier ook verschillende hoog-niveau talen kunnen vertalen naar de intermediate language zonder dat we na hoeven te denken over het platform dat we gaan gebruiken.

   (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=0; x<5; x++)
	  this.iets();
	for (y=0; y>5; y++)
	  this.iets();
		5 keer: 5 keer voor elke x, en daarna 0 keer, want de tweede loop stopt onmiddellijk (y is nooit groter dan 5

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

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

4	**while** (**false**)
	  this.iets();
	if (**true**)
	  this.iets();
		1 keer (de **while** body wordt niet uitgevoerd (want de conditie is **false**), de **if** body wordt wel uitgevoerd)

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)

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

Antwoord:

int i, j;	D	position.Zero;	X	Color.White	E	**true**=**false**	X
(**float**)	X	'\\'	C	Game	T	**for**(;**false**;);	O
Vector2 position;	D	**new** Random()	E	1 + 2 == 3	E	i = Color.Black.R;	O
new Vector2(1, 2) * 3	E	i<5!=j>5	E	SpriteBatch	T	1E1	C
while(**true**) **while**(**false**);	O	/*hallo	X	(**bool**)**true**	E	i = j++;	O

3. (a) *(3 punten)* Schrijf een methode IsPrimeNumber die, gegeven een geheel getal, aangeeft of het een priemgetal is. Een getal is een priemgetal als dat getal alleen maar deelbaar is door zichzelf en door één. Je mag er vanuit gaan dat altijd een positief getal meegegeven wordt aan de methode. **Antwoord:**

```
public bool IsPrimeNumber(int n)
{
    for (int i = 2; i < n; i++)
        if (n % i == 0)
            return false;
    return true;
}
```

(b) *(5 punten)* Elk positief geheel getal kan uitgedrukt worden als een vermenigvuldiging van priemfactoren. Bijvoorbeeld:

$$102 = 2 \times 3 \times 17$$

$$712 = 2 \times 2 \times 2 \times 89$$

De priemfactoren kun je vinden door te beginnen met het delen door twee. Als dat lukt, dan is 2 de eerste priemfactor. Als het niet lukt, hoog de deler op en kijk of het daarmee kan, enzovoorts:

$$712/2 = 356 \qquad 356/2 = 178 \qquad 178/2 = 89 \qquad 89/89 = 1$$

Het getal 102 heeft 3 priemfactoren (2, 3, en 17), en het getal 712 heeft er vier (2, 2, 2, en 89). Schrijf een methode CountPrimeFactors die het aantal priemfactoren uitrekent van een geheel getal.

Antwoord:

```
public int CountPrimeFactors(int n)
{
    int factors = 0;
    for (int d = 2; d <= n; d++)
        while (n % d == 0)
        {
            n = n / d;
            factors++;
        }
    return factors;
}
```

Het is in dit geval niet eens nodig om te kijken of de deler wel een priemgetal is. Dat wordt namelijk al automatisch gegarandeerd doordat we de kleinere delers al eerder hebben bekeken.

(c) *(2 punten)* Schrijf een stukje programma dat, door gebruik te maken van de CountPrimeFactors methode, uitrekent wat het kleinste positieve gehele getal is dat vijf priemfactoren heeft. Dit getal moet vervolgens op het scherm afgedrukt worden.

Antwoord:

```
int i = 1;
while (CountPrimeFactors(i) != 5)
    i++;
Console.WriteLine("Eerste getal met 5 priemfactoren is " + i);
```

4. (a) *(2 punten)* Wat is de reden dat sommige membervariabelen in de AnnoyedElephants klasse **static** zijn?

Antwoord: Door deze variabelen static te maken heb je geen object nodig om ze te kunnen gebruiken. De game wereld zit in een static variabele zodat alle game objecten erbij kunnen, net zoals de willekeurige getallengenerator (random) en de schermdimensies. Bovendien heb je (in de context van deze game) maar één game wereld, één willekeurige getallengenerator, en één set schermdimensies.

(b) *(4 punten)* Het tekenen van de bananen gebeurt in de methode DrawBananas uit de GameWorld klasse. Werk deze methode uit (header en body). Let erop dat de bananen altijd netjes helemaal rechts in het scherm getekend worden, ook als de speler minder dan drie bananen verzameld heeft.

Antwoord:

```
public void DrawBananas(SpriteBatch spriteBatch)
{
    for (int i = 0; i < points; i++)
        spriteBatch.Draw(banana, new Vector2(AnnoyedElephants.Screen.X − (i % 3 + 1) ∗ banana.Width,
                                             i/3 ∗ banana.Height), Color.White);
}
```

(c) *(2 punten)* Er mist nog een property Points in de GameWorld klasse. Werk deze property uit. Zorg ervoor dat het aantal verkregen punten nooit kleiner kan zijn dan nul.

Antwoord:

```
public int Points
{
    get { return points; }
    set { if (value >= 0) points = value; }
}
```

(d) *(2 punten)* Als een olifant botst met een aap, dan wordt de aap op een willekeurige plek onderaan het scherm geplaatst. Echter, de x-positie van de aap mag nooit kleiner zijn dan 100, omdat de aap dan op de positie van de olifant zou kunnen komen. Het plaatsen van de aap gebeurt in de Reset methode. Werk de body van deze methode uit.

Antwoord:

```
public void Reset()
{
    position = new Vector2(AnnoyedElephants.Random.Next(100, (int)AnnoyedElephants.Screen.X − sprite.Width),
                           AnnoyedElephants.Screen.Y − sprite.Height);
}
```