Uitwerkingen eerste deeltentamen Gameprogrammeren
Vrijdag 28 september 2012, 11.00-13.00 uur

---

1. Deze opgave bestaat uit een aantal vragen. Houd het antwoord kort: één of twee zinnen per onderdeel kan al genoeg zijn.

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

   - ☐ Elke klasse moet tenminste één methode hebben die de naam Main draagt. *Nee, er moet één klasse in het programma zitten die een Main methode heeft.*
   - X Er bestaan methoden en properties die geen objecten bewerken.
   - ☐ Als het woord **new** gebruikt wordt om een object te creëren, dan weten we dat dat object als type een klasse heeft. *Nee, het type kan ook een struct zijn.*
   - X Het woord **this** mag niet gebruikt worden in een **static** methode.
   - X Elke **while** opdracht kan omgeschreven worden naar een **for** opdracht en vice versa.
   - ☐ Scripttalen zijn altijd imperatief. *Nee, het feit dat iets een scripttaal is zegt niets over het soort taal, alleen over de manier waarop de taal gebruikt wordt.*

   (b) *(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.

   (c) *(3 punten)* Beschouw de volgende methode:

```
public int deelsom(int n)
{
    int resultaat = 0;
    for (int i = n; i >= 0; i -= 2)
        resultaat += i;
    return resultaat;
}
```

   Geef aan wat de uitkomst is van de volgende expressies:

   Antwoord:

   deelsom(8): 8+6+4+2=20

   deelsom(5): 5+3+1=9

   deelsom(1): 1

deelsom(−1): 0

Schrijf de **for** loop om naar een **while** loop.
Antwoord:

```
int i = n;
while (i >= 0)
{
    resultaat += i;
    i -= 2;
}
```

(d) *(2 punten)* Wat betekent het als het woord **void** voor de naam van een methode in diens methodeheader wordt geschreven? Is de aanroep van die methode dan een expressie of een opdracht? Of allebei? Of geen van beiden?

Antwoord: Het woord **void** voor een methodenaam betekent dat de methode geen resultaatwaarde oplevert. De methode-aanroep is een opdracht, maar geen expressie, want een expressie is een stukje programma met een waarde.

2. *($-\frac{1}{2}$ punt per fout)* Hieronder staat 16 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**)	X	'\\'	C	**decimal**	T	**for**(;**false**;);	O
int i;	D	Texture2D.Height;	X	Color.White	E	**true**==**false**	E
Texture2D t;	D	**new** SpriteBatch(GraphicsDevice)	E	1 + 2 = 3	X	1E1	C
3==i<5	E of X	Vector2.Zero / 2	E	SpriteBatch	T	/*1 */2 //3	C

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. Als de som van de alle cijfers tot de macht 3 van een getal hetzelfde is als dat getal, dan noemen we het getal ook wel een *Armstronggetal*. Bijvoorbeeld, 153 is een Armstronggetal, want:

$$153 = (1 \times 1 \times 1) + (5 \times 5 \times 5) + (3 \times 3 \times 3)$$

   (a) *(5 punten)* Schrijf een statische methode isArmstrongNumber die van een gegeven getal bepaalt of het een Armstronggetal is. Je mag er hierbij vanuit gaan dat altijd een positief getal meegegeven wordt aan de methode.

   Antwoord:

```
public static bool isArmstrongNumber(int n)
{
    int number = n;
    int sum = 0, r = 0;
    while (n > 0)
    {
        r = n % 10;
        n = n / 10;
        sum += r * r * r;
    }
    return sum == number;
}
```

   (b) *(3 punten)* Schrijf een statische methode isPrimeNumber die uitrekent of een gegeven getal een priemgetal is. Een priemgetal is een getal dat alleen deelbaar door één en door zichzelf is. Ook hier mag je er vanuit gaan dat altijd een positief getal meegegeven wordt aan de methode.

   Antwoord:

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

   (c) *(2 punten)* Schrijf een serie opdrachten die alle getallen tussen 1 en 500 (eenmalig) op het scherm afdrukt die een Armstronggetal of een priemgetal zijn, of beide. Gebruik de methoden uit de vorige deelopdrachten. Je mag er vanuit gaan dat deze opdrachten in een console applicatie uitgevoerd worden.

   Antwoord:

```
for (int i = 1; i < 500; i++)
{
    if (isArmstrongNumber(i) || isPrimeNumber(i))
        Console.WriteLine(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);
}
```