Aanvullende toets Gameprogrammeren (INFOB1GP)
Woensdag 24 december 2014, 13.30 - 15.30 uur

Naam:	
Studentnummer:	

- Het tentamen bestaat uit 4 opgaven. Opgaven 1 levert 8 punten op, opgave 2 levert 12 punten op, en opgaven 3 en 4 leveren elk 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 het verschil tussen een abstracte klasse en een interface? Geef een voorbeeld van wat niet mogelijk is met een interface en wel met een abstracte klasse.

   (b) *(2 punten)* Wat is de betekenis van de operator `%`? Geef een expressie die voor positieve getallen `x` en `y` dezelfde waarde heeft als `x%y`, zonder daarbij de operator `%` te gebruiken.

(c) *(2 punten)* Wat betekent het woord **virtual** als het in de definitie van een methode gebruikt wordt? Mag **virtual** ook gebruikt worden voor properties?

(d) *(2 punten)* Gegeven zijn de volgende twee declaraties van variabelen:

```
1 string s;
2 double waarde;
```

Iemand schrijft de volgende opdracht om de wortel van de waarde te berekenen en aan de gebruiker te tonen:

```
1 s = "De wortel is " + Math.Sqrt(waarde);
```

Maar als waarde negatief is wordt het programma afgebroken met een foutmelding.
In plaats daarvan willen we liever dat de tekst "onmogelijk" in de string bewaard wordt. Je kunt dit op twee manieren voor elkaar krijgen:

1. Vooraf controleren of de foutsituatie zich gaat voordoen
2. De wortel gewoon maar uitrekenen en de foutsituatie opvangen

Geef voor beide aanpakken aan hoe de opdracht er dan uit komt te zien.

1.

2.

2. *(12 punten)* Geven is de volgende membervariabele in een klasse:

```
1  int[] array = { 3, 1, 5, 3, 8, 2, −1, 1, 1 };
```

Hieronder staan zes methoden uit diezelfde klasse. Geef in elk van de gevallen aan wat de return-waarde van de methode-aanroep is. Licht het antwoord kort toe.

1	`int Methode1() {`
	`  int result = 0;`
	`  for (int i = 0; i < array.Length; ++i)`
	`  result += array[i];`
	`  return result;`
	`}`	

2	`int Methode2() {`
	`  int result = 0, i = 0;`
	`  while (i < array.Length) {`
	`  result += array[i];`
	`  i += array[i];`
	`  }`
	`  return result;`
	`}`	

3	`int Methode3() {`
	`  int result = 0;`
	`  for (int i = 0; i < array.Length; ++i)`
	`  if (i % 2 == 0)`
	`  result += array[i];`
	`  return result;`
	`}`	

4	`int Methode4() {`
	`  int result = 0, i = 0;`
	`  while (i <= array[i]) {`
	`  result += array[i];`
	`  ++i;`
	`  }`
	`  return result;`
	`}`	

5	`int Methode5() {`
	`  int result = 0;`
	`  for (int i = 0; i < array.Length − 1; ++i)`
	`  if (array[i] > array[i + 1])`
	`  result += array[i];`
	`  return result;`
	`}`	

6	`int Methode6() {`
	`  int result = −1000;`
	`  for (int i = 0; i < array.Length; ++i)`
	`  if (array[i] > result)`
	`  result = array[i];`
	`  return result;`
	`}`	

3. Gegeven zijn de volgende twee klassedefinities:

```
1  class BaseClass {
2      private int var1;
3      protected int var2;
4      public int var3;
5      int var4;
6  }
7
8  class DerivedClass : BaseClass {
9      public int var5;
10
11     public void Method1() {
12         int i = 0;
13     }
14 }
```

(a) *(2.5 punten)* Kruis aan welke van de volgende opdrachten mogen worden uitgevoerd in de body van Method1 in de klasse DerivedClass, na de declaratie van variabele i:

- ☐ i = var1;
- ☐ i = var2;
- ☐ i = var3;
- ☐ i = var4;
- ☐ i = var5;

(b) *(7.5 punten)* De volgende opdrachten staan in een methode van een andere (ongerelateerde) klasse:

```
1  BaseClass obj = new BaseClass();
2  DerivedClass obj2 = new DerivedClass();
3  BaseClass obj3 = obj2;
4  int i = 0;
```

Kruis aan welke van de volgende opdrachten mogen worden uitgevoerd in de body van die methode:

- ☐ i = obj.var1;
- ☐ i = obj.var2;
- ☐ i = obj.var3;
- ☐ i = obj.var4;
- ☐ i = obj2.var1;
- ☐ i = obj2.var2;
- ☐ i = obj2.var3;
- ☐ i = obj2.var4;
- ☐ i = obj2.var5;
- ☐ i = obj3.var1;
- ☐ i = obj3.var2;
- ☐ i = obj3.var3;
- ☐ i = obj3.var4;
- ☐ i = obj3.var5;
- ☐ obj3.Method1();

4. Gegeven is de volgende klasse:

```
1  class TetrisBlok
2  {
3      private bool[,] configuratie;
4      private const int size = 4;
5
6      public TetrisBlok() {
7      }
8  }
```

Deze klasse gebruiken we om een blok voor te stellen in de game Tetris. We gaan er in deze klasse vanuit dat elk tetrisblok in een configuratie van size $*$ size deelblokjes beschreven wordt, waarbij **true** staat voor bezet en **false** staat voor vrij. In de tweedimensionale array staat de eerste dimensie voor de x-positie en de tweede dimensie voor de y-positie van elke deelblokje in de configuratie.

(a) *(2 punten)* Er mist nog iets in deze klasse. Wat is dat? Schrijf de missende opdracht(en) op en geef aan waar ze toegevoegd moeten worden.

(b) *(4 punten)* We willen dat andere objecten kunnen opvragen of een bepaald deelblokje in de configuratie bezet is. Schrijf een methode IsBezet die berekent of een gegeven positie binnen het tetrisblok bezet is. Indien de indices gegeven als parameter buiten het bereik van de array vallen, dan dient de methode een IndexOutOfRangeException object te werpen.

(c) *(4 punten)* Schrijf een methode genaamd DraaiRechtsom. Deze methode draait het tetrisblok 90 graden naar rechts.

GAME OVER