Uitwerking Tentamen Modelleren en Programmeren - versie 1
Woensdag 1 februari 2017, 17–20 uur

- Schrijf op elk ingeleverd blad je naam. Schrijf op het eerste blad ook je studentnummer en het aantal ingeleverde bladen.
- De lijst met standaardfuncties na afloop graag weer inleveren. De antwoorden komen binnenkort op de website.
- Opgave 1 t/m 10 zijn meerkeuzevragen, die meetellen voor $10 \times 3 = 30$ punten.
  Opgave 11 en 12 zijn tekstvragen, die meetellen voor $2 \times 15 = 30$ punten.
  Opgave 13 en 14 zijn programmeervragen, die meetellen voor $2 \times 20 = 40$ punten.

Meerkeuzevragen: de letter van het goede antwoord volstaat.
Belangrijk: dit is versie **1** van het tentamen, vermeld dat boven je antwoorden.

1. Een *klasse* is *niet*:

   (a) het type van een object
   (b) een groepje methoden
   (c) • een groepje variabelen
   (d) een groepje declaraties

   Toelichting op het antwoord: een groepje variabelen heet een *object*, en dat is wat anders.

2. Iemand schrijft een methode om te kunnen bepalen of een array van characters een bepaalde waarde bevat:

```
bool Bevat(char[] a, char x)
{
    for (int t=0; t < a.Length; t++)
        if (a[t] == x)
             return true;
        else return false;
}
```

   Wat is er fout in deze methode?

   (a) In de test moeten aanhalingstekens staan om de `x`, dus `a[t]=='x'`
   (b) De array `a` is nog niet met `new` aangemaakt en heeft dus geen lengte
   (c) • Het keyword `else` moet hier niet staan
   (d) De hele `if`/`else`-opdracht moet worden vervangen door `return a[t]==x;`

   Toelichting op het antwoord: (a) is niet waar: dan zou je alleen naar de specifiek de 24e letter van het alfabet zoeken, en blijft de parameter `x` helemaal ongebruikt. (b) is niet waar: het aanmaken van de array is de verantwoordelijkheid van de aanroeper van de methode. (c) is goed: door deze else vliegt de methode er bij de eerste karakter al uit, terwijl ook de overige karakters getest moeten worden. Dus `return false` moet pas helemaal aan het eind van de methode staan, en dat is wat er in feite staat als je de `else` weg denkt. (d) deze opdracht is equivalent aan de gebruikte if-opdracht, en weliswaar stylistisch mooier, maar dan zit de bug er nog steeds in.

3. Iemand schrijft een methode om een getal `x` tot een niet-negatieve macht `e` te verheffen:

```
int Macht(int x, int e)
{
    int res = 1;
    for (int t=1; t<=e; t++)
        x *= res;
    return res;
}
```

Welk ongewenst effect heeft deze methode?

(a) de herhaling gaat één stap te lang door

(b) het werkt niet als `e` gelijk is aan 0

(c) het werkt niet als `x` gelijk is aan 1

(d) • de uitkomst is altijd 1

Toelichting op het antwoord: (a) is fout: weliswaar staat er `<=`, en dat is gevaarlijk, maar omdat de teller bij 1 begint gaat het toch goed. (d) is het goede antwoord: nadat `res` de waarde 1 heeft gekregen, verandert hij nooit meer. Aan het eind is hij dus nog steeds 1, en dat was natuurlijk niet de bedoeling. De bug in het programma is de opdracht `x *= res;`. Dit had moeten zijn: `res *= x;`. Doordat het antwoord altijd 1 is, werkt de methode juist *wel* als `e==0` of `x==1`.

4. Een `try-catch` opdracht kan worden gebruikt om

   (a) het optreden van exceptions te voorkomen
   (b) programmeerfouten op te vangen
   (c) • een foutsituatie netjes af te handelen
   (d) in een methode ongeldige waarden van een parameter te detecteren

5. Aanroep van de methode `Start` van een `Thread`-object heeft tot gevolg dat

   (a) de methode `Run` wordt aangeroepen
   (b) de methode `Run` steeds opnieuw wordt aangeroepen
   (c) • de methode die bij de constructor van `Thread` werd meegegeven wordt aangeroepen
   (d) de constructormethode van `Thread` wordt aangeroepen

6. Welk van de volgende fragmenten kan gebruikt worden om de grootste waarde `m` van een array `a` te bepalen?

   (a) • `m=a[0]; for (int t=1; t<a.Length; t++) if (a[t]>m) m=a[t];`
   (b) `m=a[0]; for (int t=1; t<a.Length; t++) if (a[t]>m) a[t]=m;`
   (c) `m=a[0]; for (int t=1; t<a.Length; t++) if (m>a[t]) m=a[t];`
   (d) `m=a[0]; for (int t=1; t<a.Length; t++) if (m>a[t]) a[t]=m;`

7. `Length` is

   (a) een member-variabele van de klasse `String`
   (b) een methode van de klasse `String`
   (c) • een property van de klasse `String`
   (d) een parameter van de klasse `String`

8. Het is handiger om een `List` te gebruiken in plaats van een array als je

   (a) van tevoren precies weet hoeveel elementen er zijn
   (b) • later nog elementen wilt kunnen tussenvoegen
   (c) de elementen later wilt kunnen sorteren
   (d) met een `foreach`-opdracht de elementen wilt langsgaan

   Toelichting op het antwoord: (a) is juist een reden om arrays te gebruiken; (c) en (d) zijn zowel voor arrays als string mogelijk en dus geen voordeel van lists.

9. Een verschil tussen een *list* en een *collection* is

   (a) • Een collection heeft geen indexer-property
   (b) In een collection kun je het aantal elementen niet bepalen
   (c) Een collection kun je niet met `foreach` doorlopen
   (d) In een collection zitten geen dubbele elementen

   Toelichting op het antwoord: De indexer-property indexeert met een *volgnummer*, en dat is specifiek voor List. Collection heeft wel een Count. Collection is een subinterface van Enumerable, en mag je dus gebruiken in foreach. Geen dubbele elementen zitten in een Set.

10. Is het mogelijk om in een klasse onder andere een member-variabele te declareren met diezelfde klasse als type?

    (a) Nee, een object van dit type zou oneindig veel geheugen vragen

    (b) Nee, ophalen van de waarde van deze member zou oneindig veel tijd vragen

    (c) Ja, maar alleen als je zorgt dat deze variabele niet verwijst naar het object waar het deel van is

    (d) • Ja, dit kan altijd

    Tekstvragen: Geef een korte beschrijving.
    Dit kan in één zin per deelvraag, die op elkaar mogen aansluiten.

11. In sommige libraries worden naast klassen ook *interfaces* gedefinieerd.

    (a) Wat staat er in de body van zo'n interface?

    (b) Welke relatie kan er bestaan tussen zo'n interface en klassen die daarna gedefinieerd worden, en hoe schrijf je dat op?

    (c) Welke verplichting brengt het met zich mee als een klasse K zo'n relatie met een interface I heeft?

    (d) Wat is er mogelijk geworden als een klasse K zo'n relatie met een interface I heeft?

    (e) Waarom is dat handig bij de ontwikkeling van een programma?

    Antwoord:

    (a) In een interface staan alleen maar methode-*headers*, dus zonder body.

    (b) In een klasse-header kun je met `class K : I` aangeven dat de klasse K interface I implementeert.

    (c) Dat wil zeggen dat de klasse alle methodes gedefinieerd moeten worden die in de interface staan gespecificeerd.

    (d) Daarna is een object van K acceptabel als rechterkant van een toekenning aan een variabele van type I.

    (e) Je kunt in een laat stadium nog kiezen voor een bepaalde implementatie van je interface. *of:* Je kunt in een recursieve datastructuur objecten van verschillende klassen door elkaar gebruiken.

12. De frequentieverdeling van de letters van het alfabet in een tekst is tamelijk specifiek voor de taal waarin de tekst is geschreven.

    (a) Hoe kun je zo'n frequentieverdeling in een programma bepalen?

    (b) Hoe kun je de mate van overeenkomst tussen twee frequentieverdelingen bepalen? (de beschrijving mag kort zijn, maar wel zo precies dat een programmeur dit verder kan uitwerken zonder nog over het probleem na te denken).

    (c) Hoe kun je het inrichten dat een programma zelf talen kan 'leren'?

    (d) Hoe zou je dit soort AI karakteriseren?

    Antwoord:

    (a) Declareer een array van tellertjes; ga met een for-opdracht alle letters van de tekst langs en hoog steeds de bijbehorende teller op.

    (b) Bereken het totaal van de absolute waarde van de verschillen van de relatieve frequentie van overeenkomstige letters.

(c) Geef het programma voorbeeld-teksten van elke taal, en laat het de frequentieverdeling van de onbekende tekst vergelijken met die van elke taal.

(d) Dit is subsymbolische AI: met statistische technieken wordt intelligentie gesimuleerd.

Programmeervragen: Geef het gevraagde stukje code.
Het hoeft geen compleet programma te zijn, en je hoeft ook de `using`-regels niet te geven.

13. Bekijk de volgende `Main` methode:

```
static void Main()
{   Console.WriteLine( Prog.Druiven(6) );
}
```

Als dit programma wordt uitgevoerd, verschijnt de volgende tekst op de console:

```
o-o-o-o-o-o
.o-o-o-o-o.
..o-o-o-o..
...o-o-o...
....o-o....
.....o.....
```

De parameter van de methode `Druiven` bepaalt het aantal regels tekst. In dit voorbeeld is dat 6, maar als er een groter getal wordt gebruikt verschijnen er meer regels, die ook langer zijn: steeds zo, dat er een driehoek van `o`-tekens ontstaat, met streepjes ertussen, en stippen eromheen.

Als de parameter 1 is, verschijnt er alleen een enkel `o`-teken.
Als de parameter 0 is of kleiner, dan verschijnt er helemaal niets.

(a) Schrijf een methode `Kopieer` die een string als resultaat oplevert die uit een bepaald aantal kopieën van een bepaalde string bestaat. Zowel het aantal als de te kopiëren string zijn een parameter van deze methode.
Antwoord:

```
string Kopieer(int n, string x)
{   string res = "";
    for (int t=0; t<n; t++)
        res += x;
    return res;
}
```

(b) Schrijf deze methode `Druiven`.
Antwoord:

```
static string Druiven(int n)
{   string res="", marge, bolstreep;
    int t;
    for (t=0; t<n; t++)
    {    marge     = Prog.Kopieer(t,     "." );
         bolstreep = Prog.Kopieer(n-1-t, "o-") + "o";
         res      += marge + bolstreep + marge + "\n";
    }
    return res;
}
```

14. In de library `System.Drawing` zitten klassen (of eigenlijk structs, maar dat verschil is hier niet belangrijk) `Point` en `Rectangle`. Een object van de klasse `Rectangle` beschrijft de plaats en grootte van een rechthoek. We gaan een klasse `Blok` schrijven die ook een rechthoek beschrijft, maar dan net een beetje anders dan `Rectangle`. Je mag bij het schrijven ervan niet de bestaande `Rectangle`-klasse gebruiken, maar wel de klasse `Point`.

Een object van de klasse `Blok` legt de eigenschappen van een rechthoek vast door de *linker bovenhoek*, en de *breedte* en de *hoogte* op te slaan. De klasse heeft de volgende methoden:

- een constructormethode waaraan de gewenste linker bovenhoek, breedte, en hoogte als parameter worden meegegeven
- een constructormethode waaraan twee tegenoverliggende hoekpunten van de gewenste rechthoek worden meegegeven
- een methode om de oppervlakte van de rechthoek te berekenen
- een methode om de rechter onderhoek van de rechthoek te bepalen
- een methode om de gegevens van de rechthoek in een string weer te geven
- een methode om de string van de vorige methode weer om te zetten in een rechthoek (naar keuze een extra constructormethode, of de andere gangbare manier om dit te doen)
- een methode die bepaalt of de rechthoek het punt, dat als parameter wordt meegegeven, bevat

Antwoord:

```
class Blok
{
    Point linksboven;
    int breedte, hoogte;

    Blok(Point p, int b, int h)
    {
        linksboven = p;
        breedte = b;
        hoogte = h;
    }
    Blok(Point p, PointFq)
    {
        linksboven = new Point( Math.Min(p.X, q.X), Math.Min(p.Y, q.Y) );
        breedte = Math.Abs(p.X - q.X);
        hoogte  = Math.Abs(p.Y - q.Y);
    }
    float Oppervlakte()
    {
        return breedte*hoogte;
    }
    Point Rechtsonder()
    {
        return new Point(linksboven.X+breedte, linksboven.Y+hoogte);
    }
    override string ToString()
    {
        return $"{linksboven.X} {linksboven.Y} {breedte} {hoogte}";
    }
    Blok(string s)
    {
        string[] velden = s.Split();
        linksboven = new Point(int.Parse(velden[0]), int.Parse(velden[1]));
        breedte = int.Parse(velden[2]);
        hoogte  = int.Parse(velden[3]);
    }
    bool Bevat(Point p)
    {
        return p.X >= linksboven.X && p.X <= linksboven.X+breedte &&
               p.Y >= linksboven.Y && p.Y <= linksboven.Y+hoogte  ;
    }
}
```