Uitwerking Tweede deeltentamen Imperatief programmeren - versie 1
Vrijdag 21 oktober 2016, 13.00-15.00 uur

1. De situatie die ontstaat door `class A : B { C D; }` kan beschreven worden door

   (a) `B` is een `A`

   (b) • `A` heeft een `C`

   (c) `C` is een `A`, en dus ook een `B`

   (d) `A` erft een `C` van `B`

   Toelichting op het antwoord: C is een membervariabele in de klasse A, dus elk A-object 'heeft een' C. (a) is fout, het is andersom: A is een B. (c) is pure onzin. (d) is fout, want A erft weliswaar alles van B, maar de C declareert hij toch echt zelf.

2. In een property-definitie heeft het keyword `value` een bijzondere betekenis. Dit is:

   (a) De waarde die de property heeft in de `get`-minimethode

   (b) De waarde die de property krijgt in de `get`-minimethode

   (c) De waarde die de property heeft in de `set`-minimethode

   (d) • De waarde die de property krijgt in de `set`-minimethode

   Toelichting op het antwoord: Bij een toekenning aan een property, zoals `a.Prop=5;` wordt de `set`-minimethode aangeroepen, waarbij `value` dan de waarde voorstelt die aan de property wordt toegekend (hier: 5), dus de waarde die de property *krijgt*.

3. Gegeven een klasse `K`, en de declaratie `K p = null;`

   (a) kun je wel properties van `p` gebruiken, maar er geen member-variabelen van veranderen

   (b) • kun je wel `p` meegeven als parameter van een methode, maar er geen methoden van aanroepen

   (c) kun je wel waarden aan `p` toekennen, maar hem niet met andere variabelen van type `K` vergelijken

   (d) kun je wel waarden aan `p` toekennen, maar hem niet meegeven als parameter van een methode

   Toelichting op het antwoord: De `null` waarde kun je wel manipuleren (toekennen, meegeven als parameter, vergelijken met een andere object-verwijzing), zolang je maar niets met de *inhoud* van het object doet (members of properties opvragen, methoden aanroepen). (b) is de enige combinatie die hiermee klopte.

4. Een *array bound exception* treedt op als

   (a) de index niet numeriek is

   (b) de array als geheel nog niet is geïnitialiseerd

   (c) • de index te groot is

   (d) de array op een bepaalde plek nog niet is geïnitialiseerd

   Toelichting op het antwoord: (a) komt de compiler niet door, dus dan is er helemaal geen sprake van exceptions. (b) geeft een null-pointer exception, en (d) geeft helemaal geen exception.

5. Na de declaratie `int[,,]a = new int[2,3,4];` heeft de array `a`

(a) ruimte voor 3 getallen

(b) ruimte voor 9 getallen

(c) • ruimte voor 24 getallen

(d) ruimte voor een aantal getallen dat verschilt voor elk van de 3 dimensies

Toelichting op het antwoord: Deze driedimensionale tabel heeft ruimte voor $2 \times 3 \times 4 = 24$ getallen. (a) zou het geval zijn bij een initialisatie zoals `int[]a={2,3,4}`.

6. Iemand schrijft een methode om te kunnen testen of een bepaalde array van characters een bepaald character bevat:

```
bool Bevat(char[] a, char x)
{
    for (int t=0; t < a.Length; t++)
        if (a[t] == 'x')
             return true;
        return false;
}
```

Wat is er fout in deze methode?

(a) • In de test moeten geen aanhalingstekens staan om de `x`

(b) De array `a` is nog niet met `new` aangemaakt en heeft dus geen lengte

(c) Er moeten accolades rond de body van de `for`-opdracht

(d) Er moet nog `else` staan voor `return false;`

Toelichting op het antwoord: Met deze aanhalingstekens worst altijd op specifiek de letter iks gezocht, en wordt de parameter helemaal genegeerd. (b) is fout, want het aanmaken van de array moet de aanroeper doen, niet de methode. (c) is fout, want de body van de for-opdracht bestaat uit één opdracht, dus accolades zijn niet nodig. (d) is fout: dan zou hij er altijd meteen de eerste ronde uitvliegen, en komen de overige letters van de string niet aan de beurt.

7. De parameter die je meegeeft aan de constructor van `Thread`

(a) • is een methode die door `Start` wordt aangeroepen

(b) is een methode die door `Start` steeds opnieuw wordt aangeroepen

(c) is een object waarvan `Start` wordt aangeroepen

(d) is een object waarvan `Start` steeds opnieuw wordt aangeroepen

Toelichting op het antwoord: Je geeft een methode mee: dit is een soort event-handler. 'Steeds opnieuw' moet je zelf doen in de methode die je meegeeft.

8. Je kunt met `s[i]` het character op positie `i` in een string `s` opvragen omdat

(a) dan automatisch eerst de methode `ToCharArray` zal worden aangeroepen

(b) • in de klasse `String` de indexer-property is gedefinieerd

(c) een string immutable is

(d) dan automatisch de methode `IndexOf` zal worden aangeroepen

9. 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`.

10. De klasse `object`

(a) erft de methodes van alle andere klasses

(b) • is de enige klasse zonder superklasse

(c) is automatisch een subklasse van alle andere klasses

(d) is het type van een object dat nog de waarde `null` heeft

Toelichting op het antwoord: (a) is fout, het is andersom: alle klasses erven (direct of indirect) van `object`. (c) is fout, het is andersom: `object` is automatisch de *superklasse* van alle klasses zonder expliciete superklasse. (d) is onzin: het type hangt nooit af van de huidige waarde.

Tekstvragen: geef kort en duidelijk antwoord in woorden of (bij 12 en 15) een stukje code.

11. De klasse `Control` bevat een methode `Invalidate`.

Welk effect heeft het aanroepen van deze methode?

In welke situatie is het zinvol om deze methode aan te roepen?

Antwoord: Invalidate forceert een `Paint`-event voor de control waarmee je het aanroept. Dat doe je typisch als je de variabelen hebt veranderd die de toestand van het programma voorstellen, en deze toestand moet in beeld gebracht worden.

12. Door het uitvoeren van de opdracht

```
x = int.Parse(s);
```

kan er een exception optreden. In die situatie willen we `x` de waarde 0 geven.

(a) Schrijf een stukje code, waarin deze opdracht is opgenomen, waarmee de exception opgevangen wordt.
Antwoord:

```
try
{    x = int.Parse(s);
}
catch (Exception e)
{    x = 0;
}
```

(b) Schrijf een stukje code, waarin deze opdracht is opgenomen, waardoor de exception helemaal niet meer kan optreden.

Antwoord:

```
int n = s.Length; bool ok = true;
for (t=0; t<n; t++)
    if (s[t]<'0' || s[t]>'9')
        ok = false;
if (ok && n>0)
     x = int.Parse(s);
else x = 0;
```

ook goed (maar niet behandeld op college) is:

```
if (! int.TryParse(s,x))
    x = 0;
```

13. Een `class` kan het type van een object zijn. Maar ook een `struct` kan het type van een object zijn. Wat is het verschil?

    In welke situatie kun je beter voor een `struct` kiezen, en in welke situatie voor een `class`?

    Antwoord: Een variabele met een `class` als type bevat een *verwijzing* naar het object, een variabele met een `struct` als type bevat direct het object (zie sectie 4.3 van het dictaat). Men gebruikt `struct` voor kleine objecten, die uit niet meer dan een paar variabelen bestaan. Een `class` gebruik je voor grotere objecten, zodat die niet bij elke toekenning helemaal geopieerd hoeven te worden.

14. Soms staat er in de header van een methode `virtual` of `override`. In welke situatie kun je deze keywords gebruiken?

    Wat verandert er in de semantiek als je deze keywords weglaat?

    Antwoord: Als in een subklasse S een methode M van de superklasse K een nieuwe invulling moet krijgen, dan gebruik je in de subklasse `override`, en in de superklasse `virtual`. Na de declaratie `K a = new S();` wordt dan bij de aanroep `a.M()` de methode van S aangeroepen.

    Zonder de keywords wordt in deze situatie de methode van K aangeroepen.

15. *Het is in deze opgave niet toegestaan om de methode* `ToLower` *uit de klasse* `String` *te gebruiken.*

    Schrijf een statische methode `HoofdKlein` die een hoofdletter in de range A–Z omzet in de overeenkomstige kleine letter a–z. Andere waarden van de parameter worden ongewijzigd teruggegeven.

    Antwoord:

```
static char HoofdKlein(char c)
{    if (c>='A' && c<='Z')
         c = (char)(c+'a'-'A');
     return c;
}
```

Programmeervragen: hier schrijf je een stukje programma

16. In de klasse `String` zit een methode `Replace`. Deze methode levert een nieuwe string op, waarin elk voorkomen van het character dat als eerste parameter wordt meegegeven, is vervangen door het character dat als tweede parameter wordt meegegeven. Bijvoorbeeld:

```
"Utrecht".Replace('t','x')    geeft "Uxrechx"
"A+2+#@?".Replace('+','9')    geeft "A929#@?"
```

Ook is er een methode `EndsWith`, die oplevert of een string eindigt met de string die als parameter wordt meegegeven. Bijvoorbeeld:

```
"Utrecht".EndsWith("recht")    geeft true
```

Stel dat je de auteur van de klasse `String` bent. Veel andere members van die klasse zijn al geschreven (die mag je dus gebruiken), maar nog niet de `Substring` en `IndexOf` methoden (die mag je dus niet gebruiken).

(a) Schrijf de methode `Replace`.

Antwoord:

```
string Replace(char oud, char nieuw)
{
    string res = "";
    for (int t=0; t<this.Length; t++)
    {   if (this[t]==oud)
            res += nieuw;
        else res += this[t];
    }
    return res;
}
```

(b) Schrijf de methode `EndsWith`.

Antwoord:

```
bool EndsWith(string s)
{
    int sl = s.Length;
    int tl = this.Length;
    if (sl>tl)
        return false;
    for (int n=1; n<=sl; n++)
        if (this[tl-n] != s[sl-n])
            return false;
    return true;
}
```

17. Onderstaand console-programma vraagt aan de gebruiker steeds opnieuw om een woord, totdat de gebruiker alleen maar op Enter drukt.

```
class Program
{
    static void Main()
    {
        // hier komt code A
        Console.Write("Geef een woord: ");
        string s = Console.ReadLine(); // eerste regel
        while (s != "")
        {
            // hier komt code B
            Console.Write("Geef een woord: ");
            s = Console.ReadLine(); // volgende regel
        }
        // hier komt code C
        Console.ReadKey();
    }
}
```

Geef de ontbrekende code op de plaatsen A, B en C zodat het programma aan het eind een tabelletje afdrukt waarin voor elk getal vanaf 1 met een rij sterretjes wordt aangegeven hoeveel woorden van die lengte er zijn ingetikt.

Je mag ervan uitgaan dat alle woorden minder dan 100 letters hebben. Het tabelletje moet stoppen na de lengte van het langste woord.

Bij het runnen zou het er bijvoorbeeld als volgt uit kunnen zien:

```
Geef een woord: koe
Geef een woord: paard
Geef een woord: cavia
Geef een woord: kip
Geef een woord: zebra
Geef een woord: olifant
Geef een woord: vogel
Geef een woord: hond
Geef een woord:
1:
2:
3: **
4: *
5: ****
6:
7: *
```

Antwoord:

```
// hier komt code A
int[] turf = new int[100];
int max = 0;

// hier komt code B
turf[s.Length]++;
if (s.Length > max)
    max = s.Length;

// hier komt code C
for (int t=1; t<=max; t++)
{
    Console.Write(t + ": ");
    for (int x = 0; x < turf[t]; x++)
        Console.Write('*');
    Console.WriteLine();
}
```