Uitwerking Tweede deeltentamen Imperatief programmeren - versie 1
Vrijdag 17 oktober 2014, 8.30-10.30 uur

1. 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

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

   (a) kun je wel properties van `p` gebruiken, maar er geen methodes van aanroepen
   (b) • kun je wel `p` met andere variabelen van type `K` vergelijken, maar er geen member-variabelen van gebruiken
   (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

3. Je kunt een array-variabele een waarde geven die wordt aangeduid door een opsomming van de elementen tussen accolades, maar alleen

   (a) • als dit direct bij de declaratie gebeurt
   (b) als de array precies zo lang is als het aantal opgesomde elementen
   (c) als de array minstens zo lang is als het aantal opgesomde elementen
   (d) als de array één-dimensionaal is

4. Het keyword `value` kan worden gebruikt binnen de definitie van een property

   (a) in de `set`-minimethode, en heeft dan de waarde van de parameter van de property
   (b) • in de `set`-minimethode, en heeft dan de waarde van de rechterkant van een toekenning aan de property
   (c) in de `get`-minimethode, en heeft dan de waarde van het resultaat van de property
   (d) in de `get`-minimethode, en heeft dan de waarde van de `private` variabele die door de property wordt afgeschermd

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. Een animatie die door het starten van een `Thread`-object is begonnen kan weer worden gestopt door

   (a) aanroep van de methode `Stop`
   (b) aanroep van de methode `Sleep`

(c) toekenning van de waarde `null` aan het `Thread`-object

(d) • de door `Start` gestarte methode te laten eindigen

7. 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;`

8. Na de declaratie `char c = (char) 0;` bevat de variabele `c`

   (a) het cijfer 0

   (b) een spatie

   (c) het symbool @

   (d) • een symbool dat niet getoond kan worden

9. `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`

10. Het aantal variabelen waaruit een array `a` bestaat

    (a) wordt vastgelegd bij de declaratie van de array

    (b) • wordt vastgelegd als het array-object wordt gecreëerd

    (c) groeit door een toekenning `a[t]=...` voor een index `t` die nog niet eerder werd gebruikt

    (d) groeit door de opdracht `a++;`

Tekstvragen: geef kort en duidelijk antwoord in woorden.

11. Met de definitie van klassen kun je allerlei typen objecten definiëren die in een programma een rol spelen. Hoe kun je daarbij aangeven

    (a) dat een bepaald type object zich ook kan gedragen als object van een ander type?

    (b) dat een bepaald type object een ander type object als onderdeel heeft?

    (Beschrijf beide situaties in woorden, en geef van allebei ook een voorbeeld, naar keuze uit de standaardlibraries of uit een denkbeeldig programma).

    Antwoord: Een object dat een subklasse van klasse A als type heeft, kan zich ook gedragen als een object van type A. Voorbeeld: elk Button-object kan zich ook gedragen als Control-object.

    Als in de klasse B een variabele van type C staat gedeclareerd, heeft elk object van type B een object van type C als onderdeel. Voorbeeld: elk Form-object heeft een Color-object 'achtergrondkleur' als onderdeel.

12. Hoe kun je het voor elkaar krijgen dat in één array objecten van verschillende typen worden opgeslagen?

    Antwoord: Maak al deze typen een subklasse van klasse K, en declareer de array als array-van-K-objecten.

13. Wat is de rol van een *virtual* methode in de situatie uit het vorige onderdeel?

    Antwoord: Je kunt een virtuele methode definiëren in de klasse K, en hem dan *overriden* in de subklassen, zodat je hem kunt aanroepen voor alle elementen van de array en er dan voor elk type iets anders gebeurt.

14. Wanneer gebruik je het keyword `base` ? Het gebruik ervan stimuleert een goede programmeergewoonte. Welke is dat? Waarom is dat een goede gewoonte vanuit software engineering perspectief?

    Antwoord: Door `base` te gebruiken in plaats van `this` kun je vanuit een overriden methode de oorspronkelijke methode uit de superklasse aanroepen. Daarmee voorkom je dat de code van de oorspronkelijke methode gecopy-paste moet worden. Dat is wenselijk, want een eventuele bugfix of uitbreiding van die code hoeft dan maar één keer gedaan te worden.

15. Welke bijzondere rol heeft de klasse `object` ? Wat heeft dat voor gevolg voor de methodes ervan?

    Antwoord: De klasse `object` is het begin van de subklasse-hiërarchie. Zijn methodes zijn dus toepasbaar op elk type object, en kunnen in alle klassen worden overriden.

Programmeervragen: hier schrijf je een stukje programma

16. In de klasse `String` zitten onder andere de volgende methodes:

```
string Insert   (int n, string s)
int    CompareTo(string s)
```

    Deze methode `Insert` geeft een nieuwe string waarbij op de positie aangeduid door de eerste parameter, de string in de tweede parameter wordt ingevoegd. Als de positie nul of negatief is wordt de string vooraan ingevoegd, als de positie te groot is wordt de string achteraan toegevoegd. Voorbeelden:
    `"hallo".Insert( 2, "XYZ")` geeft `"haXYZllo"`
    `"hallo".Insert( 0, "XYZ")` geeft `"XYZhallo"`
    `"hallo".Insert(-5, "XYZ")` geeft `"XYZhallo"`
    `"hallo".Insert( 7, "XYZ")` geeft `"halloXYZ"`

    Stel dat je de auteur van de klasse `String` bent. De meeste methoden en properties zijn al beschikbaar, maar de methode `Insert` nog niet. Schrijf de methode `Insert` in de klasse `String`.

    Antwoord:

```
public string Insert(int n, string s)
{   int t;
    string res = "";
    for (t=0; t<n && t<this.Length; t++)
        res += this[t];
    res += s;
    for ( ; t<this.Length; t++)
        res += this[t];
    return res;
}
```

17. De string-methode `CompareTo` levert 0 op als de string precies gelijk is aan de parameter. Hij levert een negatief getal op (bijvoorbeeld -1, maar iets anders mag ook) als de string kleiner is dan de parameter, en een positief getal (bijvoorbeeld 1) als die groter is. Met kleiner en groter wordt hier de woordenboek-ordening bedoeld: de eerste letter waar de strings verschillen bepaalt de ordening (volgens de Unicodes van die letters). Is de ene string een beginstuk van de andere, dan is de kortste de kleinste. Spaties en leestekens tellen gewoon mee, die hoeven dus niet speciaal behandeld te worden.

    Voorbeelden:

`"aap".CompareTo("noot")`	geeft een negatief getal, want `'a'<'n'`
`"noot".CompareTo("nieten")`	geeft een positief getal, want `'o'>'i'`
`"niet".CompareTo("nietmachine")`	geeft een negatief getal vanwege de lengte
`"noot".CompareTo("noot")`	geeft 0, want precies gelijk
`"noot".CompareTo("NOOT")`	geeft een positief getal, want `'n'>'N'`

    Stel dat je de auteur van de klasse `String` bent. De meeste methoden en properties zijn al beschikbaar, maar de methoden `Compare` en `CompareTo` nog niet. Schrijf de methode `CompareTo` in de klasse `String`.

    Antwoord:

```
int CompareTo(string t)
{
    int m = this.Length;
    int n = t.Length;
    for (int i=0; i<Math.Min(m,n); i++)
    {
        char c = this[i];
        char d = t[i];
        if (c!=d) return c-d;
    }
    return m-n;
}
```

18. Schrijf een statische methode `VaakstVoorkomendeLengte` met als parameter een array van strings. De methode moet opleveren welke lengte van de strings het vaakste voorkomt.

    Bijvoorbeeld, als de array de strings `"hier", "zijn", "een", "paar", "woorden"` bevat, dan is het resultaat 4, omdat de 4-letterwoorden in de meerderheid zijn (het zijn er drie, terwijl er maar één 3-letterwoord, en één 6-letterwoord is).

    Je mag aannemen dat alle strings minder dan 50 tekens bevatten. Als er meerdere stringlengtes zijn die precies even vaak voorkomen, dan kies je daarvan de kortste.

    Je moet er wel op rekenen dat de parameter een erg lange array kan zijn met misschien wel miljoenen strings, en dat het ongewenst is om zo'n lange array vaker dan nodig is helemaal te doorlopen.

    Antwoord:

```
int VaakstVoorkomendeLengte(string [] a)
{
   int turf[] = new int[50];
   for (int t=0; t<a.Length; t++)
       turf[ a[t].Length ]++;
   int res=0;
   for (int t=0; t<turf.Length; t++)
       if (turf[t] > turf[res])
           res = t;
   return res;
}
```