Tentamen Object-oriëntatie (NWI-MOL098)

Vrijdag 6 juli 2018, 12:30-15:30

Dit is een gesloten boek tentamen. In totaal kun je voor de 5 vragen in dit tentamen 60 punten halen.

1 Enkele kleine opgaven **totaal 14 punten**

We beginnen met enkele korte vragen.

3 ptn **1.a)** Wat is het verschil tussen een instantievariabele en een lokale variabele? Geef van beide een voorbeeld.

Solution: Een instantievariabele geeft een eigenschap van een object weer. Een lokale variable dient enkel voor (tijdelijke) opslag van een waarde binnen een methode. Voorbeeld instantievariabele:

```
Class C {
    private int myProperty;
}
```

Voorbeeld lokale variabele:

```
void f () {
   int n = 3;
}
```

3 ptn **1.b)** Bekijk de volgende *methode-header* (ook wel *signatuur* genoemd):

```
    public static Egg findClosestEgg( Egg egg, List<Egg> eggs )
```

Benoem **alle** onderdelen in deze header kort en krachtig. Probeer niet te bedenken wat de methode mogelijk doet; geef enkel aan wat de betekenis van iedere component is (gebruik termen als *type*, *parameter*, etc.).

Solution:

`private`: zichtbaarheid enkel binnen de klasse zelf.

`static`: behoort tot de klasse, niet tot een object.

`Egg`: het resultaattype, de methode zal een `Egg`-object opleveren.

`findClosestEgg`: de naam van de methode

`Egg`: het type van de eerste parameter: een `Egg`

`egg`: de naam van de eerste parameter

`List<Egg>`: het type van de tweede parameter: een lijst van `Egg`-elementen

`balls`: de naam van de tweede parameter

3 ptn **1.c)** Wat is het doel van een constructor? In wat voor opzicht(en) onderscheidt een constructor zich van andere methodes? Noem 2 verschillen.
Solution: Een constructor heeft als doel om de instantievariabelen van een nieuw object een initiële waarde te geven. Een constructor kan niet expliciet worden aangeroepen, heeft geen resultaattype en heeft dezelfde naam als de klasse.

5 ptn **1.d)** Bekijk de volgende 2 methodes:

```
public int surprise1( int n ) {
    int i = 0;
    int p = 1;
    while( i < n ) {
        p = p * n;
        i++;
    }
    return p;
}

public int surprise2( int m ) {
    int i = 1;
    int s = 0;
    while( i <= m ) {
        s = s + surprise1(i);
        i++;
    }
    return s;
}
```

Wat is het resultaat van de aanroep `surprise1`(2)? En van `surprise1`(4)? Gebruik eventueel een tracing table. En wat is het resultaat van `surprise1`(n), uitgedrukt in n? Tot slot, wat is het resultaat van `surprise2`(n), uitgedrukt in n?
Solution: `surprise1`$(2) = 4$, `surprise1`$(4) = 256$, `surprise1`$(n) = n^n$, `surprise2`$(n) = \sum_{m=1}^{n} m^m$

2 Complexiteit van algoritmen **totaal 8 punten**

Een graaf bestaat uit een aantal *knopen* die verbonden zijn voor *kanten*. Tijdens het college zijn verschillende problemen aan de orde gekomen die we door middel van een graaf konden modelleren en vervolgens op konden lossen door gebruik te maken van een bepaald graafalgoritme. Het kleuren van grafen is een bekend voorbeeld van een graafprobleem. Hierbij ken je aan iedere knoop een kleur toe en wel zodanig dat twee verbonden knopen verschillende kleuren hebben. Het doel is om het aantal benodigde kleuren te minimaliseren.

4 ptn **2.a)** Wat kun je zeggen over de complexiteit van mogelijke graaf kleuralgoritmen? Gebruik in je antwoord termen als Polynomiaal/Niet-Polynomiaal of Lineair/Kwadratisch/Exponentieel/....

4 ptn **2.b)** Stel je wil het volgende examenroosteringsprobleem oplossen: We gaan uit van N verschillende cursussen en M verschillende studenten die zich hebben ingeschreven voor 1 of meerdere tentamens van deze cursussen. Voor elk tentamen is een tijdblok van 2 uur beschikbaar en iedere student kan per tijdblok natuurlijk maar aan 1 tentamen deelnemen. We willen zo min mogelijk tijdblokken reserveren (oftewel zo veel mogelijk tentamens parallel roosteren). Modelleer dit probleem als een graaf kleurprobleem. Dus wat representeer je als knopen in de graaf en wat stellen de kanten voor (dit in termen van cursussen en studenten)? En wat is de relatie tussen het kleuren van knopen en het oorspronkelijke probleem?

3 Loops en lijsten: teksten versleutelen **totaal 8 punten**

Caesar-substitutie is een eenvoudig versleutelingsalgoritme waarbij elke letter van de oorspronkelijke tekst wordt vervangen door een letter die n posities verderop in het alfabet staat. Bij deze methode is n dus de sleutel. Zo is het resultaat van de versleuteling van de tekst `"HALLO"` met sleutel 4 de tekst "LEPPS". We kunnen dit algoritme wat verbeteren door niet een sleutel van 1 getal te nemen, maar een die uit meerdere getallen bestaat. Laten we even aannemen dat dat we een sleutel hebben van grootte M en dat s_j met $0 \leq j < M$, het j^{de} getal van de sleutel aangeeft. Het idee van de versleuteling is nu als volgt. De eerste letter van de boodschap wordt s_0 posities opgeschoven, de tweede s_1, etc. Als we op die manier M letters hebben versleuteld, dan gebruiken we de sleutel weer opnieuw voor de volgende M letters net zolang totdat we hele boodschap omgezet is. Voorbeeld: Stel we hebben de tekst `"DITISEENGEHEIMEBOODSCHAP"` en sleutel die uit het volgende rijtje getallen bestaat: [2, 17, 9]. Versleuteling hiervan levert de tekst `"FZCKJNGEPGYNKDNDFXFJLJRY"` op, immers 'D' + 2 = 'F', 'I' + 17 = 'Z', 'T' + 9 = 'C',... etc.

We nemen even aan dat onze tekst niet uit letters, maar uit getallen (tussen 0 en 25) bestaat; dat rekent wat makkelijker.

8 ptn **3.a)** Definieer een methode

```
public void encrypt ( List<Integer> text, List<Integer> key )
```

die de meegegeven lijst van integers met als naam `text` omzet in een versleutelde tekst waarbij de opgegeven parameter `key` als sleutel gebruikt wordt. Je hoeft het versleutelde bericht niet op te slaan maar mag de tekens meteen (als getal) op de console weergeven.

Enkele `List`-methodes die mogelijk van pas kunnen komen zijn:

```
int size ()
Object get(int index)     // levert het element op positie index op
boolean add (Object o)    // voegt object o aan het einde van de lijst toe
boolean remove (Object o) // verwijdert object o uit de lijst
boolean isEmtpy ()
```

Solution:

```
public void encrypt( List<Integer> text, List<Integer> key ) {
    int i = 0;
    for ( int nextChar : text ){
        int eChar = ( nextChar + key.get( i % key.size() ) ) % 26;
        System.out.print( eChar );
        i++;
    }
}
```

4 Cryptografie: Bitcoins **totaal 18 punten**

In de cryptografie wordt gebruik gemaakt van twee soorten versleuteling: *Symmetrische* en *asymmetrische*.

3 ptn **4.a)** Leg uit wat verschil is tussen beide methoden.

Solution: By symmetrische cryptografie wordt gebruik gemaakt van één sleutel die door beide partijen wordt gedeeld en die zowel voor het vercijferen van berichten als voor het ontcijferen ervan wordt gebruikt. Bij asymmetrische cryptografie wordt gebruikgemaakt van twee aparte sleutels: één sleutel voor het vercijferen en de tweede sleutel voor het ontcijferen.

3 ptn **4.b)** Stel Alice wil Bob een geheim bericht sturen gebruikmakende van asymmetrische versleuteling. Beschrijf het *protocol* dat Alice en Bob hiervoor kunnen gebruiken (leg uit wie wat nodig heeft en welke stappen worden uitgevoerd zodat Bob aan het einde het bericht heeft ontvangen).

Solution: Bob heeft twee sleutels: een publieke en een geheime. De publieke sleutel wordt door Alice gebruikt om de boodschap te vercijferen. Deze vercijferde boodschap wordt naar Bob gestuurd die met zijn eigen geheime sleutel het oorspronkelijke bericht reconstrueert

3 ptn **4.c)** Leg uit hoe een kwaadwillende tussenpersoon Eve dit protocol kan verstoren. Eve is hierbij in staat om boodschappen te onderscheppen en eventueel te vervangen door andere boodschappen.

3 ptn **4.d)** Hoe kun je asymmetrische versleuteling gebruiken voor *digitale handtekeningen*?

Solution: Voor het plaatsen van een digitale handtekening beschikt Alice wederom over een publieke en geheime sleutel. Dit keer wordt echter de geheime sleutel gebruikt om het bericht te vercijferen. Met behulp van de publieke sleutel kan nu het bericht worden ontcijfert. Alice kan nu niet ontkennen het bericht van haar afkomstig was: zij was de enige die in staat was om dit vercijferde bericht te maken. Hiervoor is immers haar geheime sleutel noodzakelijk.

Bij de *Bitcoin* worden naast digitale handtekeningen ook *cryptografische hashfuncties* gebruikt.

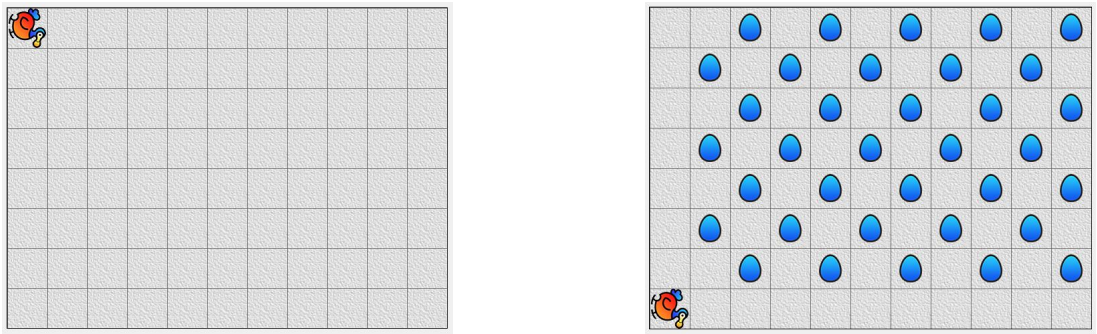
3 ptn **4.e)** Wat is een cryptografische hashfunctie? Noem de twee belangrijkste eigenschappen van zo'n hashfunctie.

Solution: Een cryptografische hashfunctie is een algoritme dat een willekeurige tekst omzet in een bitreeks van een vooraf vastgestelde grootte, bijvoorbeeld 256 bits. Het resultaat wordt ook wel de hashcode genoemd. De 2 belangrijkste eigenschappen zijn: Het is onmogelijk om van een gegeven hashcode te bepalen wat de oorspronkelijke tekst was. En verder is het onmogelijk twee verschillende teksten te vinden/maken die dezelfde hashcode hebben

3 ptn **4.f)** Leg uit hoe hashfuncties gebruikt worden om Bitcoin-transacties vast te leggen.
Solution: Voor het accepteren van een Bitcoin-transactie wordt gezocht naar een willekeurige tekst die samengevoegd met de transactie een hashcode oplevert die begint met een vooraf vastgestelde hoeveelheid nullen, bijvoorbeeld 60. Vanwege het onvoorspelbare gedrag van de hashfunctie kan bij dit zoeken enkel gebruik worden gemaakt van een brute-force strategie: men doorloopt systematisch alle mogelijke teksten totdat men er een gevonden heeft die de gewenste hashcode tot gevolg heeft (het zogenaamde minen). De hoeveelheid werk die hiermee gemoeid is, is dermate groot dat het voor een bedrieger onmogelijk is om zelf een reeks van dit soort geaccepteerde Bitcoin-transacties te genereren. Hij zou hiervoor moeten concurreren met de massa die over aanzienlijk meer rekenkracht beschikt. Op den duur zal hierdoor het bedrog aan het licht komen.

5 Algoritmen: Dodo tekent een dambord **totaal 15 punten**

Dodo staat in één van de 4 hoeken van de wereld. Haar doel is om het eierenpatroon zoals getoond wordt in het rechterplaatje, te produceren.



(a) Beginsituatie (b) Eindsituatie

Figuur 1: Dodo maakt een diamant

15 ptn **5.a)** Geef, in Java, een algoritme dat het rechterplaatje oplevert. Dit algoritme dient generiek te zijn, dat wil zeggen, het moet werken voor alle mogelijke dimensies van de wereld. Je mag **geen** aannames maken over de hoek waarin Dodo aan het begin staat; je mag eventueel wel aannames maken over richting waarin ze kijkt (bijvoorbeeld dat de wereldgrenzen zich achter en rechts van haar bevinden, zoals in bovenstaand plaatje het geval is). In je programma mag je enkel de methodes `borderAhead`, `move`, `turnLeft`, `turnRight` en `layEgg` gebruiken (dus bijvoorbeeld geen `getX`, `getY` of `setLocation`).

Hint: Definieer eerst een hulpmethode `makeRowOfCheckersBoard( boolean nextFree )` die een rij van het dambord aanmaakt. De boolean parameter `nextFree` geeft aan of deze rij al dan niet moet beginnen met een lege cel. Misschien dat je voor je algoritme iets aan onderstaande methode hebt (je mag deze gewoon gebruiken in je eigen code):

```
private void walkToBorder() {
    while ( ! borderAhead() ) {
        move();
    }
}
```

Solution:

```
public void makeCheckersBoard() {
    boolean nextFree = true;
    while (! borderAhead() ){
        turnLeft();
        makeRowOfCheckersBoard( nextFree );
        nextFree = ! nextFree;
        turnRight();
        move();
        turnRight();
        walkToBorder();
        turnLeft();
    }
}

private void makeRowOfCheckersBoard( boolean nextFree ){
    while ( ! borderAhead() ) {
        move();
        if ( nextFree ) {
            nextFree = false;
        } else {
            layEgg();
            nextFree = true;
        }
    }
}
```