Tentamen Object-oriëntatie (NWI-MOL098)

Vrijdag 13 april 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 *parameter* en een *lokale variabele*? Geef van beide een voorbeeld.

Solution: Een parameter is een variabele die hoort bij een methode waarvan de initiële waarde bij de aanroep (altijd) wordt opgegeven. Een lokale variabele is een variabele die binnen een methode gedeclareerd en (mogelijk) geïnitialiseerd wordt. Voorbeeld parameter

```
void f (int i)
```

Voorbeeld locale variabele

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

3 ptn **1.b)** Bekijk de volgende declaratie:

```
    private static final String WORLD_FILE_NAME = "worldwith10eggs";
```

Beschrijf de betekenis van **alle** zes onderdelen (de '=' en de ';' mag je achterwege laten) in deze declaratie kort en krachtig.
Solution:

`private`: zichtbaarheid enkel binnen de klasse zelf.

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

`final`: de waarde van deze variable kan veranderd worden; een constante.

`String`: het type van de constante dat aangeeft wat voor soort waarde in de constante is opgeslagen, hier een tekst.

`WORLD_FILE_NAME`: de naam van de van de constante.

`"worldwith10eggs"`: de waarde van de constante, een tekst.

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 methode:

```
private int calculateSomething( int n ) {
    int s = 0;
    int i = 1;
    while ( i <= n ) {
        s = s + 2 * i - 1;
        i++;
    }
    return s;
}
```

Wat is het resultaat van de aanroep `calculateSomething`(3)? En van `calculateSomething` (4)? Gebruik eventueel een tracing table. En wat is het resultaat van `calculateSomething` (n), uitgedrukt in n?
Solution: `calculateSomething`$(3) = 9$, `calculateSomething`$(4) = 16$, `calculateSomething` $(n) = n^2$

2 Complexiteit van algoritmen **totaal 8 punten**

We beschrijven eerst twee verschillende problemen.

Probleem 1: verkiezingsstrijd

Onderstaand plaatje geeft een sociaal netwerk weer: als twee personen in dit netwerk met elkaar verbonden zijn kennen ze elkaar en hebben ze ook regelmatig contact.

Persoon P (P maakt zelf geen deel uit van het netwerk) doet mee aan de verkiezingen en wil proberen de stemmen van de mensen uit het netwerk te krijgen. Daarvoor huurt hij aan

aantal personen uit het netwerk in die hun kennissenkring ervan moeten overtuigen om op P te stemmen. P wil zo min mogelijk geld hieraan uitgeven door zo min mogelijk mensen aan te stellen. Hij is dus op zoek naar een groep G van personen uit het netwerk zodat (1) ieder ander persoon direct door iemand uit G benaderbaar is en (2) het aantal mensen in groep G minimaal is. P wil voor de bepaling van deze groep een programma schrijven.

Probleem 2: riolering aanleggen

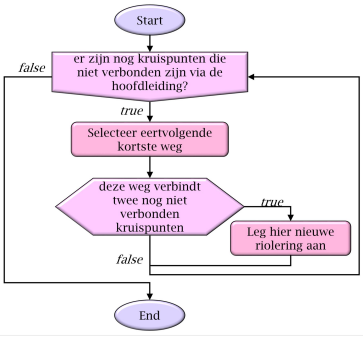
In een woonwijk wordt de capaciteit van de riolering vergroot door een deel van de bestaande riolering te vernieuwen. Om de kosten zo laag mogelijk te houden wil men de totale lengte van de nieuw te leggen riolering minimaliseren. De riolering wordt aangelegd onder de bestaande wegen en wel zodanig dat alle knooppunten (kruisingen) van wegen ondergronds met elkaar verbonden worden. Op deze kruisingen worden de rioleringsbuizen die niet vervangen worden gekoppeld aan de het vernieuwde hoofdstelsel. Ook voor dit probleem wil men een programma schrijven dat aangeeft welke wegen hiervoor in aanmerking komen.

Vragen

Beantwoord nu de volgende vragen.

4 ptn **2.a)** Geef voor één van beide problemen een oplossing in de vorm van een algoritme gespecificeerd door middel van een 'hoog niveau' flowchart (dat wil zeggen, een flowchart zonder veel details).

Solution: We geven een oplossing voor probleem 2.



4 ptn **2.b)** Wat is het fundamentele verschil tussen de beide problemen wanneer we de complexiteit van mogelijke oplossingen bekijken? Gebruik in je antwoord termen als Niet-Polynomiaal of Exponentieel.

Solution: Het fundamentele verschil tussen beide problemen is dat voor probleem 1 geen efficiënte oplossingsmethode bestaat (of eigenlijk niet bekend is) en dus enkel *brute-force* in exponentiële tijd kan worden opgelost. Voor probleem 2 bestaan wel algoritmes die dit probleem polynomiale tijd oplossen.

3 Loops: Fibonaccigetallen **totaal 8 punten**

De rij van Fibonacci begint met 1 dan weer 1 en vervolgens is elk volgend element van de rij steeds de som van de twee voorgaande elementen. De eerste 10 elementen van deze rij zijn:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

Dus voor het n^{de} fibonaccigetal f_n, met $n \geq 2$, geldt dat $f_n = f_{n-1} + f_{n-2}$.

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

```
public void fibonacciNumbers ( int size )
```

die allereerst een lijst waarin de eerste `size` fibonaccigetallen zijn opgeslagen aanmaakt en deze vervolgens afdrukt (met `System.out.print`).

Hierbij kun je gebruik maken van de volgende declaratie

```
List<Integer> fibNumList = new ArrayList<>( Arrays.asList( 1, 1 ) );
```

waarmee een lijst van grootte 2 genaamd `fibNumList` wordt aangemaakt die de eerste twee getallen uit de Fibonaccireeks, hier dus 1 en 1, bevat. Je hoeft deze declaratie niet precies te begrijpen, maar wel kunnen gebruiken.

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 fibonacciNumbers ( int size ) {
    List<Integer> fibNumList = new ArrayList<>(Arrays.asList( 1, 1 ));
    for( int i = 2; i < size; i++ ) {
        fibNumList.add(fibNumList.get(i-1) + fibNumList.get(i-2));
    }
    for (Integer fn: fibNumList) {
        System.out.print( fn + " ");
    }
    System.out.println();
}
```

4 Cryptografie: Bitcoins **totaal 15 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)** 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.d)** Wat is een cryptografische hashfunctie? Noem de twee belangrijkste eigenschappen van zo'n hashfunctie.

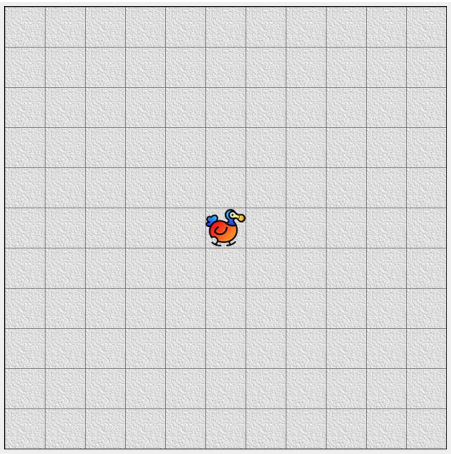
Solution: Een cryptografische hashfunctie is een algorime dat een willekeurige tekst omzet in een bitreeks van een vooraf vastgestelde grootte, bijvoorbeeld 256 bits. Het resultaat wordt ook wel de hashcode genoemd. De 2 belangrijkste eigenschappen zijn: Het is onmogelijk om van een gegeven hashcode 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.e)** Leg uit hoe hashfuncties gebruikt worden om Bitcoin-transacties vast te leggen.

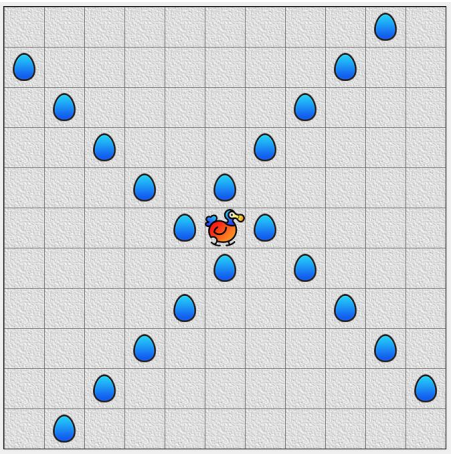
Solution: Voor het accepteren van een Bitcoin-transactie wordt gezocht naar een willekeurige tekst die samengevoegd met de transactie een hashcode oplevert die begint met een vooraf vastgestelde hoeveelheid nullen, bijvoorbeeld 60 bits. Vanwege het onvoorspelbare gedrag van de hashfunctie kan bij dit 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 molen **totaal 15 punten**

Dodo staat in het midden van de wereld (zie onderstaand linkerplaatje). Haar doel is om het eierenpatroon zoals getoond wordt in het rechterplaatje, te produceren (in het midden van dit plaatje bevindt zich overigens geen ei).



(a) Beginsituatie



(b) Eindsituatie

Figuur 1: Dodo maakt een diamant

5.a) Geef, in Java, een algoritme dat het rechterplaatje oplevert. Dit algoritme dient generiek te zijn, dat wil zeggen, de lengte van de te produceren wieken van de molen dient als parameter te worden meegegeven. Je hoeft niet te controleren of de molen met de opgegeven lengte van de wieken ook inderdaad past binnen Dodo's wereld. Verder mag je aannemen dat de wereld een vierkant is met een oneven grootte waarbij Dodo precies in het midden staat. Je mag **geen** aannames maken over richting waarin Dodo kijkt. Gebruik in je programma enkel de methodes `move`, `turnLeft`, `turnRight` en `layEgg` (dus bijvoorbeeld geen `getX`, `getY` of `setLocation`).

Hint: Definieer eerst een hulpmethode `void diagonal( int length )` die een diagonaal met de opgegeven lengte produceert. Maak hiermee een aparte methode voor tekenen van een wiek waarin je niet alleen de diagonaal tekent maar ook weer terugkeert naar je oorspronkelijke positie. Misschien dat je voor dit laatste iets hebt aan onderstaande methode (je mag deze gewoon gebruiken in je eigen code):

```
public void jump( int distance ) {
    int nrStepsTaken = 0;                 // set counter to 0
    while ( nrStepsTaken < distance ) { // check if more steps must be taken
        move();                           // take a step
        nrStepsTaken++;                   // increment the counter
    }
}
```

Solution:

```
public void diagonal( int length ) {
    int nrStepsTaken = 0;
    while ( nrStepsTaken < length ){
        move();
        layEgg();
        turnRight();
        move();
        turnLeft();
        nrStepsTaken++;
    }
}

public void sail( int length ){
    diagonal( length );
    turnLeft();
    jump( length );
    turnLeft();
    jump( length );
    turnLeft();
}

public void windMill( int length ){
    int nrOffSails = 0;
    while ( nrOffSails < 4 ){
        sail( length );
        nrOffSails++;
    }
}
```

– EINDE TENTAMEN –