samen sterk voor werk

Java

JAVA PROGRAMMING FUNDAMENTALS

Takenbundel

Peoplesoftcode:
Wettelijk depot:

versie: 02/02/2016

INHOUD

1 Taken bij hoofdstuk 4: Variabelen en operatoren

1.1 Student scores

Een student behaalt op een examen volgende scores: 8, 6, 9 en 4. De maximum score voor ieder vak is 10. Maak een programma dat het gemiddelde berekent (als decimaal getal) en het behaalde percentage.

1.2 Omrekening seconden

Maak een programma dat een geheel aantal seconden, bijvoorbeeld 5924, omrekent in uren, minuten en seconden.
Het resultaat kan als volgt getoond worden : U:1 M:38 S:44

1.3 Snoepautomaat

Gegeven: een te betalen bedrag voor een snoep uit een snoepautomaat. De kostprijs van de verschillende snoepen varieert tussen € 0,30 en € 1,20. De klant kan enkel betalen met een muntstuk van € 2. Het programma dient het wisselgeld uit te rekenen: hoeveel muntstukken van € 1, € 0.50, € 0.20, € 0.10, € 0.05, € 0.02 en € 0.01 dienen er teruggegeven te worden? Steeds met zo weinig mogelijk munten!

Test het programma voor een aankoop van € 0.42 , voor een aankoop van € 1.02 , …

1.4 Bankrekeningnummer

Schrijf een programma om te testen of een bankrekeningnummer een geldig nummer is.
Je kan dit testen door het getal gevormd door de eerste 10 cijfers van het bankrekeningnummer te delen door 97. De rest van deze deling zou gelijk moeten zijn aan het getal gevormd door de 2 laatste cijfers van het bankrekeningnummer.

Gebruik volgende correcte rekeningnummers ter controle:

823445816730
237824199569
662431212859
737524091952

Gebruik volgende foutieve rekeningnummers ter controle:

111224444891
777553241844

2 Taken bij hoofdstuk 5: Arrays

2.1 Array van vijf integers

Maak een array van vijf integers. De waarden van de array worden opgevuld met willekeurige getallen tussen 1 en 100 (grenswaarden inbegrepen). Gebruik hiervoor de method `Math.random()`. Deze method geeft een *double* terug tussen 0 (inbegrepen) en 1 (niet inbegrepen).

Nadat de array gevuld is, worden volgende gegevens getoond:

- de vijf getallen van de array,
- de som van de vijf getallen,
- de gemiddelde waarde (als decimaal getal) van de vijf getallen.
  Tip: om het gemiddelde te berekenen gebruik je een property van de array om te weten hoeveel elementen er in de array aanwezig zijn.

3 Taken bij hoofdstuk 6: Programmaverloop

3.1 Array van vijf integers (met iteraties)

Herneem vorige opdracht maar realiseer nu de oplossing door gebruik te maken van iteraties.

3.2 Randomgenerator

Controle op de random-generator.
Laat de randomgenerator 10.000 willekeurige gehele getallen genereren tussen 1 en 100 (grenswaarden inbegrepen).
Toon op het scherm hoe dikwijls ieder getal is gegenereerd.

3.3 Randomgenerator2

Laat met de randomgenerator 100 getallen genereren tussen 1 en 1000 (grenswaarden inbegrepen) en stop ze in een array. Sorteer de getallen en voer ze uit van klein naar groot.

3.4 Lotto

Genereer lotto-getallen, d.w.z. 6 verschillende getallen tussen 1 en 42 (inbegrepen) + een reservegetal.
Toon ze op scherm in opklimmende volgorde.

3.5 Huisdieren

Maak een programma waarin je de gebruiker vraagt hoeveel huisdieren hij/zij heeft. Toon voor de aantallen 0 t/m 3 telkens een gepaste melding op het scherm. Voor een aantal groter dan 3, toon dan een standaardbericht op het scherm.

4 Taken bij hoofdstuk 7: OO, classes en objects

4.1 Een class Getal en een main-programma GetalMain.

a) Maak 2 classes als volgt:

- Maak een class Getal die enkel een public membervariabele x van het type int heeft (geen constructor).

- Maak een tweede class GetalMain met een main-method (een main-class). In deze method maak je een object (of ook wel instance genoemd) van de class Getal aan. Plaats de waarde van x op het scherm door rechtstreeks de membervariabele x aan te spreken.

b) Maak in class Getal de membervariabele x private. Wat stel je vast in je main-class en waarom ?

c) Voeg aan de class Getal de public method `print()` toe die de waarde van de membervariabele x op het scherm toont.

Wijzig de method `main()` zodat deze gebruik maakt van de nieuwe method om de waarde van de membervariabele x op het scherm te tonen.

d) Voeg nu een constructor aan de class Getal toe. Deze constructor heeft een parameter (een int a). De constructor kent de waarde van a toe aan de membervariabele x.

Herschrijf daarop de regel uit de `main()` die het object van Getal aanmaakt zodat deze nu de nieuwe constructor gebruikt. Merk op dat er nu geen default constructor meer wordt gemaakt of voorzien door de compiler.

e) Voeg aan Getal de method `absoluut()` toe. Deze method heeft als returnwaarde de absolute waarde van de membervariabele x (x zelf mag niet veranderen).

Tip : om de absolute waarde te berekenen kan je de functie `Math.abs(x)` gebruiken.

Wijzig de `main()` zodat deze gebruik maakt van deze method of anders gezegd: laat de `main()` bijvoorbeeld de absolute waarde van -45 op het beeldscherm tonen.

f) Voeg aan Getal een method `som(int a)` toe die de som van de membervariabele x en a teruggeeft. Test deze nieuwe method in de `main()`.

g) Voeg aan Getal een method `add(int a)` toe die de membervariabele x verhoogt met a. Wijzig de `main()` zodat die een object van de class Getal aanmaakt, de waarde van de membervariabele x op het scherm toont, deze waarde verhoogt met een int-getal d.m.v. de nieuwe method en tot slot deze nieuwe waarde van de membervariabele x op het scherm toont.

h) Nu gaan we de method `som(int a)` overloaden met nieuwe methods:
- `float som(float a)` en
- `double som(double a)`.
Test deze nieuwe methods in de main().

i) Maak een method `toDouble()` die als returnwaarde de waarde van membervariabele x als een double teruggeeft. Test deze nieuwe method in de main().

j) Maak een method `getX()` die de waarde van de membervariabele x teruggeeft en een method `setX(int a)` die de waarde van a toekent aan de membervariabele x.

Wijzig de constructor die dus gebruik zal gaan maken van deze set()-method.
Probeer deze methods in de main door eerst de setter en vervolgens de getter uit te voeren.

4.2 Student

Maak een class *Student* met volgende membervariabelen:

- naam (String)
- score (int)

Schrijf een constructor met enkel een parameter voor de naam.

Schrijf een tweede constructor met parameters voor de naam en de score.

Schrijf getters en setters voor de membervariabelen.

Schrijf een main-class *StudentMain* en maak hier twee studenten aan (gebruik beide constructors één keer). Toon de gegevens van de studenten op het scherm. Wijzig de score en/of naam met de setters en toon de gegevens opnieuw op het scherm.

4.3 Waarnemer

Maak een class Waarnemer. Deze class kan gegevens van temperaturen bijhouden. Na het registreren van een temperatuur (of meerdere temperaturen), kan de minimumtemperatuur, de maximumtemperatuur, het aantal waarnemingen en een gemiddelde temperatuur opgevraagd worden.
Denk zelf na over de membervariabelen en hun types en de methods die hiervoor nodig zijn.

Maak vervolgens een main-class aan, nl. WaarnemerMain. De bedoeling is om meerdere temperaturen in te geven en deze te laten registreren door de class Waarnemer. Bij ingave van temperatuur 999 stopt de invoer en worden volgende gegevens getoond:

- het aantal ingegeven temperaturen
- de hoogste temperatuur
- de laagste temperatuur
- de gemiddelde temperatuur

4.4 Kaart

Schrijf een class *Kaart*. Een kaart heeft:

- een **kleur** (harten, ruiten, klaveren of schoppen) en
- een **rang** (2, 3, 4, 5, 6, 7, 8, 9, 10, boer, vrouw, heer, aas)

Maak een constructor die een willekeurige kaart aanmaakt.

Maak een method `printKaart()` die de kleur en de rang van een kaart weergeeft.

Voeg aan de class Kaart een method `isHogerDan` toe die de kaart vergelijkt met een andere kaart: wanneer de kaart hoger in kleur is of bij gelijke kleur hoger in rang is dan de andere kaart, geeft deze

method de waarde `true` terug, anders `false`. Het argument van deze method is een andere kaart of m.a.w. object van de class Kaart.

De volgorde van de kleur en rang, die tussen de haakjes vermeld staat, is van klein naar groot opgesomd.

Schrijf tevens een main()-programma (*KaartMain*) waarin:

- een eerste kaart wordt aangemaakt
- de gegevens van deze kaart worden getoond op het scherm
- een tweede kaart wordt aangemaakt
- en ook hiervan de gegevens op het scherm worden getoond
- de eerste kaart vergeleken wordt met de tweede kaart en aangeeft of deze kaart hoger in rang is dan de tweede kaart.

5 Taken bij hoofdstuk 8: Inheritance

5.1 Voertuigen

5.1.1 De class Voertuig

Schrijf een class Voertuig met setter en getter methods voor:

- een membervariabele *polishouder* van het type String
- een membervariabele *kostprijs* van het type float
- een membervariabele *pk* van het type int
- een membervariabele *gemVerbruik* van het type float
- een membervariabele *nummerplaat* van het type String

De default waarde voor *polishouder* en *nummerplaat* is “onbepaald”. De default waarde voor *kostprijs* en *gemVerbruik* is 0.0; voor *pk* is dat 0.

Maak een constructor zonder argumenten.

Maak ook een constructor met argumenten voor polishouder, kostprijs, pk, gemiddeld verbruik en nummerplaat. Controleer de waardes: tekstvelden mogen niet leeg zijn en de numerieke velden moeten groter zijn dan 0.

Return alle gegevens in één String in een toString() method. Eerst de polishouder, dan de kostprijs, het aantal pk, het gemiddeld verbruik en de nummerplaat. Scheid de gegevens met een puntkomma.

Schrijf ook een method toon(), die uiteraard alle gegevens van het voertuig toont op het scherm. Voorzie hierbij de nodige opmaak.

5.1.2 De class Vrachtwagen

Maak een class Vrachtwagen, die is afgeleid van Voertuig en setter en getter methods heeft voor:

- een membervariabele *maxLading* van het type float

De default waarde voor *maxLading* is 10 000 kg en mag niet negatief zijn.

Voorzie ook een constructor waarbij je alle waardes via parameters meegeeft.

Override de toString() method en de method toon().

5.1.3 De class Personenwagen

Maak een class Personenwagen, die is afgeleid van Voertuig en setter en getter methods heeft voor:

- een membervariabele *aantalDeuren* van het type int
- een membervariabele *aantalPassagiers* van het type int

De default waarde voor *aantalDeuren* is 4 en voor *aantalPassagiers* is 5. Deze membervariabelen mogen niet negatief zijn.

Voorzie ook een constructor waarbij je alle waardes via parameters meegeeft.

Override de toString() method en de method toon().

5.1.4 De abstracte method getKyotoScore ()

Maak de class Voertuig abstract en voeg de volgende abstracte method toe :

```
public abstract double getKyotoScore();
```

Deze method returnt een Kyoto-score. Voor een personenwagen is de Kyoto-score gelijk aan het verbruik vermenigvuldigd met het aantal pk, gedeeld door het aantal passagiers.

Voor een vrachtwagen is de Kyoto-score gelijk aan het verbruik vermenigvuldigd met het aantal pk, gedeeld door de lading in ton.

Werk getKyotoScore() uit in de diverse classes.

5.1.5 Het hoofdprogramma

Schrijf een main()-programma met de naam TestProgramma. Maak hier een aantal Voertuigobjecten aan, zowel vrachtwagens als personenwagens. Gebruik de verschillende constructors, dus zowel zonder als met argumenten.

Toon de gegevens van de voertuigen. Gebruik hiervoor zowel de toString() als de method toon().

Toon eveneens de kyotoscore van alle voertuigen.

Maak vervolgens een array van voertuigen en itereer hierover. Een keer om de gegevens te tonen m.b.v. de toString() method, en vervolgens nog een keer om de gegevens te tonen m.b.v. de toon() method.

6 Taken bij hoofdstuk 9: Strings

6.1 Klinkers

Schrijf een(main-) programma waarbij je de gebruiker eerst een zin laat intikken. Toon vervolgens op het scherm hoeveel klinkers er in die zin staan.

6.2 Rekenaar

Schrijf een class die een input onder de vorm

17 + 38 * 2 – 22

aanvaardt, dit uiteenrafelt in integers en bewerkingstekens, de bewerkingen uitvoert en het resultaat bewaart. Het scheidingsteken is een spatie!
Voer alle bewerkingen uit van links naar rechts en niet volgens de juiste wiskundige rekenvolgorde (* / + -).

Test deze class m.b.v. een main()-programma. Gebruik hard gecodeerde expressies, maar laat ook de gebruiker een rekenkundige expressie ingeven. (Ga er van uit dat de gebruiker geen fouten maakt bij het ingeven).

6.3 Palindroom

Maak een class die in staat is om te onderzoeken of een ingevoerde tekst een palindroom is, hierbij al dan niet rekening houdend met hoofd- en kleine letters.
Voorbeelden van palindrooms: kok, lepel, parterretrap, snelmeetsysteemlens.

Maak een main waarin je een woord opvraagt en dit woord test.

Tip:
Gebruik een StringBuilder of StringBuffer. Beide classes hebben een method ***reverse()*** die de karakters van de string in omgekeerde volgorde plaatst. Zo kan je onmiddellijk vergelijken of een tekst en zijn omgekeerde gelijk zijn.

7 Taken bij hoofdstuk 10: Interfaces

7.1 VoertuigenI

Voor deze oefening werk je verder met de oplossing van oefening 'Voertuigen' van hoofdstuk 8 *Inheritance*. De oefening wordt uitgebreid met een interface Vervuiler :

```
package jpfhfdst10oef;
public interface Vervuiler {
    double berekenVervuiling();
}
```

- Implementeer deze interface in de classes Personenwagen en Vrachtwagen. Voor de personenwagen is de vervuiling gelijk aan de Kyotoscore maal 5, voor de vrachtwagen is dat de Kyotoscore maal 20.

- Maak ook een nieuwe class Stookketel. Deze class heeft een membervariabele van type float *CONorm* (met bijhorende getter en setter). Laat deze class de interface Vervuiler implementeren. De vervuiling is de CONorm maal 100.

- In het hoofdprogramma maak je vervolgens een array van type Vervuiler. Je plaatst hierin een aantal objecten die de interface Vervuiler implementeren. Je overloopt de array en van elk object toon je het resultaat van de method *berekenVervuiling()*.

- Maak twee interfaces : *Privaat* en *Milieu*. In de interface Privaat voorzie je een void-method *geefPrivateData()*, in de interface Milieu een void-method *geefMilieuData()*.

- Werk deze interfaces uit in Voertuig door in de method geefPrivateData () enkel de polishouder en de nummerplaat uit te voeren naar het scherm. Maak ook een method geefMilieuData() waarin je de properties pk, kostprijs en verbruik weergeeft.

- In het hoofdprogramma maak je vervolgens een array van objecten van type Privaat. Stop een aantal voertuigen (vrachtwagens en personenwagens) in deze array en laat in een lus alle private gegevens zien. Tracht om ook eens een array van objecten van type Milieu te maken met eveneens een aantal voertuigen erin en ga na welke method(s) nu beschikbaar zijn.

7.2 Voorwerpen

Maak een interface Voorwerp. Deze interface bevat een void-method gegevensTonen() en een double-method winstBerekenen().

Maak volgende classes die allen de interface Voorwerp implementeren :

- Een class boekenrek met properties eigenaar (String), hoogte (int), breedte (int), aankoopprijs (float) en een winstmarge (float).
  De winst is gelijk aan de aankoopprijs x 2.

- Een class boek met properties titel (String), auteur (String), eigenaar (String), aankoopprijs (float) en een genre (String).

- Een class leesboek, afgeleid van boek, met property onderwerp (String) en een winstmarge (float) die gelijk is aan de aankoopprijs x 1,5.

- Een class woordenboek, afgeleid van boek, met property taal (String) en een winstmarge gelijk aan de aankoopprijs x 1,75.

Alle voorwerpen, zowel een boekenrek als een boek, hebben als eigenaar “VDAB”.
Gegevens zoals een breedte, hoogte, enz. kunnen niet negatief zijn. Strings mogen niet null zijn.

Voorzie voor alle classes een toString()-method.

De methods van de interface dienen gedefinieerd te worden:

- gegevensTonen(): toont alle waarden van de membervariabelen + de berekende winst
- winstBerekenen(): returnt het resultaat van de berekening

In het hoofdprogramma maak je een array van het interfacetype Voorwerp. Hierin stop je volgende objecten: 2 boekenrekken, 2 leesboeken en 2 woordenboeken. Deze objecten maak je telkens door één keer gebruik te maken van een defaultconstructor en één keer aan de hand van een constructor met parameters.
Itereer over deze array en toon alle gegevens van de arrayelementen. Totaliseer de winst en toon ze.

8 Taken bij hoofdstuk 11: Packages

8.1 Voorwerpen (vervolg)

In deze oefening werk je de oplossing van oefening Voorwerpen (uit het vorig hoofdstuk) verder uit. Stop de classes als volgt in een package-structuur:

Package **be.vdab.util** bevat:

- de interface Voorwerp

Package **be.vdab.voorwerpen** bevat:

- de class Boekenrek
- de class Boek
- de class Leesboek
- de class Woordenboek

Het hoofdprogramma zit in de package **be.vdab**.

8.2 VoertuigenP

In deze oefening werk je de oplossing van oefening VoertuigenI (uit het vorig hoofdstuk) verder uit.

Stop de classes als volgt in een package-structuur:

Package **be.vdab.util** bevat:

- de interface Vervuiler
- de interface Privaat
- de interface Milieu

Package **be.vdab.voertuigen** bevat:

- de class Voertuig
- de class Vrachtwagen
- de class Personenwagen

Package **be.vdab.verwarming** bevat:

- de class Stookketel

Het hoofdprogramma zit in de package **be.vdab**.

9 Taken bij hoofdstuk 12: Exception handling

9.1 TestExceptions

Bestudeer het volgende main-programma en de exception class en noteer wat de output is :

```
package jpfhfdst12oef;
public class TestExceptions {
    public static void main(String[] args) {
        String test = "no";
      try {
            System.out.println("start try");
            doRisky(test);
            System.out.println("end try");
        }
        catch (ScaryException ex ) {
            System.out.println("scary exception");
        }
        finally {
            System.out.println("finally");
        }
        System.out.println("end of main");
    }

    static void doRisky(String test) throws ScaryException {
      .out.println("start risky");
            if ("yes".equals(test)) {
            throw new ScaryException();
        }
        System.out.println("end risky");
    }
}
```

De exception class:

```
package jpfhfdst12oef;
public class ScaryException extends java.lang.Exception {

    public ScaryException() { }
    public ScaryException(String msg) {
        super(msg);
    }
}
```

Noteer vervolgens wat de output is indien de vierde regel gewijzigd wordt in

```
        String test = "yes";
```

9.2 Controle ISBN13nummer

In deze oefening werk je verder met de classes Boek, Leesboek en Woordenboek, gemaakt in het vorige hoofdstuk.
De class Boek wordt uitgebreid met een private membervariabele *ISBN13* van type String. Bedoeling is om aan de class Boek een ISBNnummer toe te voegen. Het is het ISBNnummer bestaande uit 13 cijfers en heeft het formaat met of zonder streepjes (dus 13 cijfers achter elkaar, xxx-xxxxxxxxxx, xxx-

x-xxx-xxxxx-x of xxx-xx-xxx-xxxx-x). Er is geen vast formaat voor de plaats van de streepjes. Wat wel zeker is, is dat het 13 cijfers moet bevatten.

Dit ISBN-nummer dient gevalideerd te worden. Bedoeling is om het 13e cijfer te controleren als volgt:

- Voor de eerste 12 cijfers doe je het volgende: elk cijfer wordt beurtelings vermenigvuldigd met 1 of 3.
- Van deze producten maak je de som.
- Je berekent de modulo (rest van de deling) van de deling van deze som door 10.
- Je berekent het verschil van 10 – rest. Indien het resultaat gelijk is aan 10, wordt het resultaat 0 (resultaat moet altijd een waarde zijn gaande van 0 tot en met 9).
- Het laatste cijfer van het ISBNnumer moet gelijk zijn aan dit resultaat.

Voorbeeld: 978-0-306-40615-7

som = 9**x1** + 7**x3** + 8**x1** + 0**x3** + 3**x1** + 0**x3** + 6**x1** + 4**x3** + 0**x1** + 6**x3** + 1**x1** + 5**x3**
= 93
93 % 10 → rest = 3
10 – 3 = **7**

Voorzie verder een eigen exception class *ISBN13Exception* die gethrowd wordt wanneer er een foutief ISBNnummer wordt opgegeven.

10 Taken bij hoofdstuk 14: Collections

10.1 Land

Maak een class Land. Deze class bevat de membervariabelen *landCode*, *landNaam* en *hoofdstad*, alle drie van type String. Verder bevat ze nog de membervariabele *aantalInwoners* van type BigInteger en *oppervlakte* van type BigDecimal.
Voorzie enkel een constructor met parameters.
Voorzie ook getters en setters. De stringvariabelen moeten ingevuld worden, het aantal inwoners en de oppervlakte moeten groter dan 0 zijn. Er wordt een LandException gethrowed met een passende foutboodschap wanneer niet aan deze voorwaarden wordt voldaan. Verder bereken je de bevolkingsdichtheid van het land.

Maak vervolgens een main-class waarin je een ArrayList vult met minstens 10 landen. Geef de gegevens en de bevolkingsdichtheid weer van deze landen.

Bereken de gemiddelde bevolkingsdichtheid van alle landen en geef deze weer.
Geef daarna weer welk land een bevolkingsdichtheid heeft die het dichtst aanleunt bij de gemiddelde bevolkingsdichtheid.

De LandException plaats je in de package *be.vdab.util*, de class Land in *be.vdab.land* en het main-programma in *be.vdab*.

Publiceer alle berekende aantallen met 2 cijfers na het decimaal teken.
Tip: gebruik hiervoor de class DecimalFormat of gebruik de method *format()* van de class String.

10.2 VoertuigenC

Voor deze oefening werk je verder met de oplossing van oefening 'VoertuigenP' van hoofdstuk 11 *Packages*.

- De natural ordening van voertuigen in een stijgende volgorde van nummerplaat.
- Creëer in het main-programma enkele voertuigen van verschillende types (zowel personenwagens als vrachtwagens) en voeg die toe aan een Set.
- Doorloop de Set en toon de informatie van alle voertuigen.

10.3 Tienkamper

Maak een class Tienkamper met de properties *naam* (String) en *punten* (int). De naam mag niet *null* zijn en de punten mogen niet negatief zijn. Voeg de nodige getters en setters toe. Schrijf de toString(), de equals(), de hashCode() en de compareTo()-method en baseer deze op de *naam*.

Hoofdprogramma:

- Vul een collection (ArrayList) met een aantal atleten en toon ze.
- Maak een tweede collection (TreeSet) en vul deze met dezelfde objecten. Toon opnieuw de lijst.

10.4 Beginletter

Schrijf een main-programma waarin je bepaalt hoeveel woorden beginnen met een bepaalde letter:

- Stockeer hiervoor eerst een aantal woorden in een array van Strings,
- Bepaal dan door gebruik te maken van een HashMap hoeveel van deze woorden beginnen met dezelfde letter. Deze hashMap zal dus key-value-paren bevatten (K=beginletter, V=aantal woorden),

Toon vervolgens volgende resulaten:

- aantal woorden per letter
- de grootte van de HashMap
- is de hashMap leeg?
- alle keys
- alle values
- alle entries

10.5 Winkel

Bewaar voor deze oefening de classes in package **be.vdab.winkel** en het main-progr. in **be.vdab.**

Schrijf een **class Product** met 2 membervariabelen:

- een omschrijving
- een prijs

Schrijf een **class Catalogus:**

- met 1 membervariabele: een verzameling van producten (List van Product)
- een constructor om deze verzameling te vullen

Schrijf een **class Mandje** die een verzameling kan bevatten van de gekochte producten met hun aantallen (Gebruik hiervoor een **Map**, nl. een `HashMap <Product,Integer>`).
Gebruik de class Catalogus om het mandje te vullen, zodat er in het mandje enkel producten kunnen zitten die in de catalogus staan.
Verder heeft deze class ook volgende methods:

- method add (Product, int): voegt een Product toe aan het mandje
- method set (Product, int): wijzigt het aantal van het gekochte product
- method remove(Product): verwijdert één product uit het mandje
- method clear(): maakt het mandje leeg
- method getTotalePrijs(): geeft het totaal bedrag van de aankopen in het mandje

Denk zelf na welke methods er nog nodig zijn bij bovenstaande classes.

Bedoeling is om via een main()-programma een overzicht te tonen van de inhoud van het mandje. Vul daarom eerst dit mandje programmatorisch m.b.v. de methods add en set. Controleer ook de werking van de overige methods. Toon ten slotte het totaal te betalen bedrag van de aankopen.

11 Taken bij hoofdstuk 15:Streams–File I/O en Serialization

11.1 Gastenboek

De bedoeling van deze oefening is om een programma te schrijven voor het bijhouden van een gastenboek. Het gastenboek is een binair bestand *gastenboek.dat*. In het gastenboek worden entries weggeschreven, bestaande uit een tijdstip, een naam (van de schrijver) en zijn/haar boodschap. Het programma laat ook toe om de lijst met alle entries te lezen en weer te geven naar het scherm.

Plaats alle classes in de package **be.vdab.gastenboek**.

Maak eerst een **GastenboekEntry**-class. Deze bevat membervariabelen:

- datum,
- schrijver
- boodschap

Overweeg welke methods nodig zijn.

Vervolgens maak je een **Gastenboek**-class. Deze bevat één membervariabele, namelijk een ArrayList van GastenBoekEntries. Verder twee methods: één om een entry toe te voegen aan de ArrayList en één om alle entries op te vragen.

Maak een class **GastenboekManager** die in staat is om in het gastenboek te schrijven. Per bericht in het gastenboek wordt een timestamp, de opsteller en de boodschap geregistreerd (omwille van praktische redenen is de boodschap beperkt tot één regel). Deze class heeft ook een tweede method waarmee je het gastenboek kan inlezen.

Maak een **main-class** die de keuze laat tussen lezen of schrijven en die de verdere input/output via de console afhandelt:

- lezen: het gastenboek wordt ingelezen en de gastenboekentries worden weergegeven. De meest recente items staan bovenaan, dus de volgorde van weergave is omgekeerd t.o.v. de ingave.
- schrijven: een naam en boodschap wordt gevraagd zodat er een gastenboekentry kan worden toegevoegd aan het gastenboek (met de huidige datum en tijd)

12 Taken bij hoofdstuk 16: Multithreading

12.1 GemiddeldeRekenaar

Je maakt een class GemiddeldeRekenaarMain met een method `public static void main(String args[]);`

Je maakt in deze method een double array met 1 000 000 random getallen.

Je wil het gemiddelde kennen van deze getallen.

Je doet dit op volgende manier:

Eén thread berekent het gemiddelde van de eerste 500.000 getallen.

Een tweede thread berekent gelijktijdig het gemiddelde van de laatste 500.000 getallen.

De method main() vraagt deze gemiddeldes op en maakt het gemiddelde van deze gemiddeldes. Dit is het gemiddelde van de 1.000.000 getallen.

12.2 Pannenkoek (uitbreiding)

Je voegt aan de applicatie uit de theorie met de classes Kok en Stapel een class Klant toe. Deze class implementeert de interface Runnable.

Deze class stelt een klant voor die 50 iteraties uitvoert.
Bij iedere iteratie neemt de klant een pannenkoek van de stapel, op voorwaarde dat de stapel minstens één pannenkoek bevat.
Als volgende stap binnen de iteratie wacht de Klant 100 milliseconden.

Je maakt in de method main van de class Main naast de twee threads die een Kok voorstellen ook vier threads die een Klant voorstellen en je voert alle threads uit.

Je toont daarna hoeveel pannenkoeken er op de stapel liggen.

Gezien 2 koks elk 100 pannenkoeken bakten en 4 klanten elk 50 pannenkoeken bestelden, moet de stapel op het einde van de applicatie 0 pannenkoeken bevatten.

13 Voorbeeldoplossingen taken van hoofdstuk 4

13.1 Student scores

```
public class Studentscores {
  public static void main(String[] args) {
    int score1 = 8;
    int score2 = 6;
    int score3 = 9;
    int score4 = 4;

    int totaalScore = score1 + score2 + score3 + score4;
    float gemiddelde = totaalScore / 4F;
    float percentage = totaalScore / 40F * 100;

    System.out.println("Het gemiddelde is " + gemiddelde + " op 10");
    System.out.println("Het behaalde percentage is " + percentage + " %");
  }
}
```

13.2 Omrekening seconden

```
public class OmrekeningSeconden {
  public static void main(String[] args) {
    int totSec = 5924;
    int uren, minuten, seconden, rest;
    uren = totSec / 3600; //3600 sec in een uur
    rest = totSec % 3600; //rest bevat resterende sec

    minuten = rest / 60;
    seconden = rest % 60;

    System.out.println("U:" + uren + " M:" + minuten + " S:" + seconden);
  }
}
```

13.3 Snoepautomaat

```
public class Snoepautomaat {
  public static void main(String[] args) {
    int ingave = 2;
    double kost = 0.42;
    int terug =  ingave * 100 - (int)(kost*100);  //in centen uitgedrukt

    System.out.println("Kost van " + kost + " euro, en ingave van " +
ingave + " euro" );
    System.out.println("Automaat geeft " + terug +  " cent(en) terug");

    int munt100, munt50, munt20, munt10, munt5, munt2, munt1;

    munt100 = terug / 100;
    terug -=munt100 * 100;

    munt50 = terug / 50;
    terug -= munt50 * 50;

    munt20 = terug / 20;
    terug -= munt20 * 20;
```

```
    munt10 = terug / 10;
    terug -= munt10 * 10;

    munt5 = terug / 5;
    terug -= munt5 * 5;

    munt2 = terug / 2;
    terug -= munt2 * 2;

    munt1 = terug;

    System.out.println("Munten van 1 EUR   : " + munt100);
    System.out.println("Munten van 0,50 EUR: " + munt50);
    System.out.println("Munten van 0,20 EUR: " + munt20);
    System.out.println("Munten van 0,10 EUR: " + munt10);
    System.out.println("Munten van 0,05 EUR: " + munt5);
    System.out.println("Munten van 0,02 EUR: " + munt2);
    System.out.println("Munten van 0,01 EUR: " + munt1);
  }
}
```

13.4 Rekeningnummer

```
public class Bankrekeningnummer {
  public static void main(String[] args) {

//    long bankreknr = 823445816730L; //OK
      long bankreknr = 237824199569L; //OK
//    long bankreknr = 662431212859L; //OK
//    long bankreknr = 737524091952L; //OK

//    long bankreknr = 111224444891L; //niet OK
//    long bankreknr = 777553241844L; //niet OK

    long bankreknrEerste10 = bankreknr / 100;
    System.out.println(bankreknrEerste10);
    int  bankreknrLaatste2 = (int) (bankreknr % 100);
    System.out.println(bankreknrLaatste2);

    int rest = (int) (bankreknrEerste10 % 97);
    System.out.println("BankrekeningNr: " + bankreknr);
    System.out.println("rest van de deling door 97: " + rest);
    System.out.println("laatste 2 cijfers: " + bankreknrLaatste2 );
  }
}
```

14 Voorbeeldoplossingen taken van hoofdstuk 5

14.1 Array van vijf integers

```
public class ArrayVanVijfIntegers {
    public static void main(String[] args) {
        int[] getallen = new int[5];
        int som;
        float gemiddelde;
        getallen[0] = (int)(Math.random()*100) + 1;
        getallen[1] = (int)(Math.random()*100) + 1;
        getallen[2] = (int)(Math.random()*100) + 1;
        getallen[3] = (int)(Math.random()*100) + 1;
        getallen[4] = (int)(Math.random()*100) + 1;

        som = getallen[0] + getallen[1] + getallen[2] +
              getallen[3] + getallen[4];
        gemiddelde = (float)som / getallen.length;

        System.out.println(getallen[0]);
        System.out.println(getallen[1]);
        System.out.println(getallen[2]);
        System.out.println(getallen[3]);
        System.out.println(getallen[4]);
        System.out.println("Som = " + som);
        System.out.println("Gemiddelde = " + gemiddelde);
    }
}
```

15 Voorbeeldoplossingen taken van hoofdstuk 6

15.1 ArrayVanVijfIntegersMetIteraties

```
public class ArrayVanVijfIntegersMetIteraties {
    public static void main(String[] args) {
        int[] getallen = new int[5];
        int som = 0;
        float gemiddelde;

        for (int i=0; i<5; i++) {
          getallen[i] = (int)(Math.random()*100) + 1;
          som = som + getallen[i];
        }
        gemiddelde = (float) som/getallen.length;

        for (int i = 0; i < getallen.length; i++) {
          System.out.println(getallen[i]);
        }

        System.out.println("Som = " + som);
        System.out.println("Gemiddelde = " + gemiddelde);
    }
}
```

15.2 Randomgenerator

```
public class Randomgenerator {
    public static void main(String[] args) {
        int[] getallen = new int[100]; //automatische initialisatie op 0

        int randGetal;
        for(int i=0; i<10_000; i++) {
           randGetal = (int)(Math.random()*100 + 1 );
           getallen[randGetal-1] +=1;
        }

        for(int i=0; i<getallen.length; i++) {
            System.out.println("getal " + (i+1) + " : " + getallen[i]);
        }
    }
}
```

15.3 Randomgenerator2

```
public class Randomgenerator2 {
    public static void main(String[] args) {
        int[] getallen = new int[100];
        int tempGetal;

        for(int i = 0; i < getallen.length; i++) {
           getallen[i] = (int)(Math.random()*1000 + 1);
        }

        // SORTEREN van de 100 getallen
        for(int pos=0; pos < getallen.length - 1; pos++) {
            for(int vgl = pos+1; vgl < getallen.length; vgl++) {
                if (getallen[pos] > getallen[vgl]) {
                    tempGetal = getallen[pos];
```

```
                    getallen[pos] = getallen[vgl];
                    getallen[vgl] = tempGetal;
                }
            }
        }


        for (int i = 0; i<getallen.length; i++) {
            System.out.print(getallen[i] + "\t") ;
        }
    }
}
```

In plaats van de sorteerroutine zelf te schrijven, kan je gebruik maken van een bestaande method om een tabel te sorteren:
Voeg als eerste regel bovenaan volgende regel code toe:

```
import java.util.Arrays;
```

De zelfgeschreven sorteerroutine in de code vervang je door:

```
Arrays.sort(getallen);
```

Nu maak je handig gebruik van een bestaande sorteermethod om de tabel te sorteren, nl. de method sort() van de class Arrays.

15.4 Lotto

```
public class Lotto {
    public static void main(String[] args) {
        int[] lotto = new int[7];
        int randGetal;
        int aantalGetallen=0;
        int tempGetal;
        boolean dubbel;

        while (aantalGetallen < 7) {
            dubbel=false;
            randGetal = (int)(Math.random()*42 +1);

            for(int i=0; i<lotto.length; i++) {                    (1)
                if (lotto[i] == randGetal) {
                    dubbel = true;
                    break;
                }
            }

            if(!dubbel) {
                lotto[aantalGetallen]=randGetal;
                aantalGetallen++;
            }
        }
        // er zijn nu 7 verschillende getallen gevonden
        // de eerste 6 worden gesorteerd, het zevende is het reservegetal
        for (int i=0; i<lotto.length -2; i++) {
            for (int j=i+1; j<lotto.length -1; j++) {
                 if (lotto[j] < lotto[i]) {
                    tempGetal = lotto[i];
                    lotto[i] = lotto[j];
```

```
                    lotto[j] = tempGetal;
                }
            }
        }

        System.out.println("De winnende lotto getallen zijn: ");
        for(int i=0; i < lotto.length -1; i++) {
           System.out.print(lotto[i]+ "\t");
        }
        System.out.println("\nHet reservegetal is: " +
            lotto[lotto.length - 1] );
    }
}
```

(1) Je kan ook op een andere manier controleren op dubbels, nl. zonder gebruik van *break*. Vervang hiervoor de for-lus door onderstaande code:

```
            int i=0;
            while ( (i< lotto.length) && (!dubbel) ) {
                if (lotto[i] == randGetal) {
                  dubbel=true ;
                }
                i++;
            }
```

Hieronder volgt een tweede versie van de oplossing waarbij *continue* met een *label* gebruikt wordt, waardoor de boolean variabele *dubbel* niet langer nodig is:

```
public class LottoVersie2 {
    public static void main(String[] args) {
        int[] lotto = new int[7];
        int randGetal=0;
        int aantalGetallen = 0;
        int tempGetal;
        //boolean dubbel;

        //opvullen van de tabel, controleren op dubbels!!
        begin:
        while (aantalGetallen < 7) {
            randGetal = (int)(Math.random()*42  + 1 ) ;
            for(int i=0; i<lotto.length; i++) {
                if (lotto[i] == randGetal) {
                    continue begin;
                }
            }
            lotto[aantalGetallen]=randGetal;
            aantalGetallen++;
        }

        // er zijn nu 7 verschillende getallen gevonden
        // de eerste 6 worden gesorteerd, het zevende is het reservegetal
        for(int i = 0; i<lotto.length -2; i++) {
            for(int j = i+1; j<lotto.length -1; j++) {
                if (lotto[j] < lotto[i]) {
                    tempGetal = lotto[i];
                    lotto[i] = lotto[j];
                    lotto[j] = tempGetal;
                }
            }
        }
```

```
        System.out.println("De winnende lotto getallen zijn: ");
        for(int i = 0; i < lotto.length -1; i++) {
           System.out.print(lotto[i]+ "\t");
        }
        System.out.println("\nHet reservegetal is: " +
            lotto[lotto.length - 1] );
     }
}
```

15.5 Huisdieren

```
public class Huisdieren {
   public static void main(String[] args) {
      int huisdieren;
      Scanner sc = new Scanner(System.in);

      System.out.println("\nHoeveel huisdieren heb je?");
      huisdieren = sc.nextInt();

      switch(huisdieren) {
          case 0:
             System.out.println("Niet echt verzot op dieren?");
             break;
          case 1:
             System.out.println("Een eenzaam beestje?");
             break;
          case 2:
             System.out.println("Twee beestjes maken ruzie.");
             break;
          case 3:
             System.out.println("Drie beestjes, leuk!");
             break;
          default:
             System.out.println("Zoveel dieren, lijkt wel klein Bokrijk!");
             break;
      }
   }
}
```

16 Voorbeeldoplossingen taken van hoofdstuk 7

16.1 Een class Getal en een main-programma GetalMain.

16.1.1 De class Getal

```
public class Getal {
    //public int x; // (a)
    private int x;  // (b)

    public Getal(int a) {
        //x = a;  //(d)
        setX(a);  //(j)
    }

    public void print() { // (c)
        System.out.println("x = " + x);
    }

    public int absoluut() { // (e)
        return Math.abs(x);
    }

    public int som(int a) { // (f)
        return x + a;
    }

    public void add(int a) { // (g)
        x = x + a;
    }

    public float som(float f) { // (h)
        return x + f;
    }

    public double som(double d) { // (h)
        return x + d;
    }

    public double toDouble(){  // (i)
        return (double) x;
    }

    public int getX() { // (j)
        return x;
    }

    public void setX(int a) { // (i)
        x = a;
    }
}
```

16.1.2 De class GetalMain

```
public class GetalMain {
    public static void main(String[] args) {
        //Getal getalA = new Getal();    (a)
        //System.out.println(getalA.x); (a)
```

```
        // (b) System.out.println(getalA.x);
        // kan niet meer omdat x private is en niet meer public

        Getal getalA = new Getal(-45); // (d)

        getalA.print(); // (c)

        int absWaarde = getalA.absoluut();  // (e)
        System.out.println("absolute waarde van x = " + absWaarde);

        int som = getalA.som(7);  // (f)
        System.out.println("som van x en a " + som );

        getalA.add(56);  // (g)
        getalA.print();

        float f = getalA.som(123.67F);  // (h)
        System.out.println("som van x en float " + f);

        double d = getalA.som(1234567.98765);  // (h)
        System.out.println("som van x en double " + d);

        double xd = getalA.toDouble(); // (i)
        System.out.println("toDouble van x " + xd);

        getalA.setX(99); // (j)
        getalA.print();
        int xx = getalA.getX();
        System.out.println("getX geeft " + xx);
    }
}
```

16.2 Student

16.2.1 De class Student

```
package jpfhfdst07oef;

public class Student {
    private String naam;
    private int score;

    public Student(String naam) {
        this.naam = naam;
    }

    public Student(String naam, int score) {
        this(naam); //oproep vorige constructor ipv this.naam = naam;
        this.score = score;
    }

    public void setNaam(String naam) {
        this.naam = naam;
    }

    public void setScore(int score) {
        this.score = score;
    }

    public String getNaam() {
        return naam;
```

```
    }

    public int getScore() {
        return score;
    }
}
```

16.2.2 De class StudentMain

```
package jpfhfdst07oef;

public class StudentMain {
    public static void main(String[] args) {
        Student student1 = new Student("Piet");
        Student student2 = new Student("Jef",14);
        String naam;
        int score;

        naam = student1.getNaam();
        score = student1.getScore();
        System.out.println("Student " + naam + " met score: " + score);

        naam = student2.getNaam();
        score = student2.getScore();
        System.out.println("Student " + naam + " met score: " + score);

        //Piet een score geven en terug geg tonen
        student1.setScore(9);
        naam = student1.getNaam();
        score = student1.getScore();
        System.out.println("Student " + naam + " met score: " + score);

        //Jef zijn naam wijzigen naar Jeff en score wijzigen van 14 naar 16
        student2.setNaam("Jeff");
        student2.setScore(16);
        naam = student2.getNaam();
        score = student2.getScore();
        System.out.println("Student " + naam + " met score: " + score);
    }
}
```

16.3 Waarnemer

16.3.1 De class Waarnemer

```
package jpfhfdst07oef;

public class Waarnemer {
    private int maxTemp;
    private int minTemp;
    private int aantalWaarnemingen;
    private double somTemp;

    public Waarnemer() {
        maxTemp = Integer.MIN_VALUE;
        minTemp = Integer.MAX_VALUE;
    }

    public int getMaxTemp() {
        if (aantalWaarnemingen > 0)
            return maxTemp;
```

```
        else
            return 0;
    }

    public int getMinTemp() {
        if (aantalWaarnemingen > 0)
            return minTemp;
        else
            return 0;
    }

    public int getAantalWaarnemingen() {
        return aantalWaarnemingen;
    }

    public double getGemTemp() {
        if (aantalWaarnemingen > 0)
            return somTemp/aantalWaarnemingen;
        else
            return 0.0;
    }

    public void registreer(int temp) {
        somTemp += temp;
        aantalWaarnemingen++;
        if (temp > maxTemp)  maxTemp = temp;
        if (temp < minTemp)  minTemp = temp;
    }
}
```

16.3.2 De class WaarnemerMain

```
package jpfhfdst07oef;

import java.util.*;

public class WaarnemerMain {
    public static void main(String[] args) {
       Waarnemer thermometer = new Waarnemer();

       Scanner sc = new Scanner(System.in) ;
       System.out.println("Geef een temperatuur in (stop = 999):");
       int temperatuur = sc.nextInt();

       while (temperatuur != 999) {
            thermometer.registreer(temperatuur);
            System.out.println("Geef een temperatuur in:");
            temperatuur = sc.nextInt();
       }

       System.out.println("Het aantal waarnemingen is: " +
            thermometer.getAantalWaarnemingen());
       System.out.println("De hoogste temperatuur is: " +
            thermometer.getMaxTemp());
       System.out.println("De laagste temperatuur is: " +
            thermometer.getMinTemp());
       System.out.println("De gemiddelde temperatuur is: " +
            thermometer.getGemTemp());
    }
}
```

16.4 Kaart

16.4.1 De class Kaart

```
package jpfhfdst07oef;

public class Kaart {

  private static String kleuren[] = {"harten", "ruiten", "klaveren",
      "schoppen" } ;
  private static String rangen[]  = {"2", "3", "4", "5", "6", "7", "8",
      "9", "10", "boer", "vrouw", "heer", "aas" } ;
  //static --> gelijk voor alle objecten!!
  private int kleur, rang ;

  public Kaart() {
    kleur = (int) (Math.random() * 4  ); //index van de array's
    rang  = (int) (Math.random() * 13 );
  }

  public String getKleur() {
    return kleuren[kleur];
  }

  public String getRang() {
    return rangen[rang];
  }

  public void printKaart() {
    System.out.println("kleur = " + getKleur() +
                       " en rang = " + getRang() );
  }

  public boolean isHogerDan(Kaart k2) {
    return (kleur > k2.kleur) ||
           ((kleur==k2.kleur) && (rang > k2.rang));
  }
}
```

16.4.2 De class KaartMain

```
package jpfhfdst07oef;

public class KaartMain {
  public static void main(String[] args) {
    Kaart kaart1 = new Kaart() ;
    kaart1.printKaart() ;

    Kaart kaart2 = new Kaart() ;
    kaart2.printKaart() ;

    if ( kaart1.isHogerDan(kaart2) )
      System.out.println("kaart1 is hoger dan kaart2" ) ;
    else
      System.out.println("kaart2 is hoger dan kaart1" ) ;
  }
}
```

17 Voorbeeldoplossingen taken van hoofdstuk 8

17.1 Voertuigen

17.1.1 De class Voertuig + getKyotoScore()

```
package jpfhfdst08oef;

public abstract class Voertuig {
    private String polishouder = "onbepaald";
    private float kostprijs;
    private int pk;
    private float gemVerbruik;
    private String nummerplaat = "onbepaald";

    public Voertuig() {
    }

    public Voertuig(String polishouder, float kostprijs,
            int pk, float gemVerbruik, String nummerplaat) {
        setPolishouder(polishouder);
        setKostprijs(kostprijs);
        setPk(pk);
        setGemVerbruik(gemVerbruik);
        setNummerplaat(nummerplaat);
    }

    public String getPolishouder() {
        return polishouder;
    }
    public final void setPolishouder(String polishouder) {
        if (polishouder != null && !polishouder.isEmpty())
            this.polishouder = polishouder;
    }

    public float getKostprijs() {
        return kostprijs;
    }
    public final void setKostprijs(float kostprijs) {
        if (kostprijs > 0.0F)
            this.kostprijs = kostprijs;
    }

    public int getPk() {
        return pk;
    }
    public final void setPk(int pk) {
        if (pk > 0)
            this.pk = pk;
    }

    public float getGemVerbruik() {
        return gemVerbruik;
    }
    public final void setGemVerbruik(float gemVerbruik) {
        if (gemVerbruik > 0.0F)
            this.gemVerbruik = gemVerbruik;
    }
```

```
    public String getNummerplaat() {
        return nummerplaat;
    }
    public final void setNummerplaat(String nummerplaat) {
        if (nummerplaat != null && !nummerplaat.isEmpty())
            this.nummerplaat = nummerplaat;
    }

    @Override
    public String toString() {
          return polishouder + " ; " + kostprijs + " ; " +
                 pk + " ; " + gemVerbruik + " ; " + nummerplaat;
    }

    public void toon() {
        System.out.println("polishouder: " + polishouder);
        System.out.println("kostprijs: " + kostprijs);
        System.out.println("pk: " + pk);
        System.out.println("gemVerbruik: " + gemVerbruik);
        System.out.println("nummerplaat: " + nummerplaat);
    }

    public abstract double getKyotoScore();
}
```

17.1.2 De class Vrachtwagen + getKyotoScore()

```
package jpfhfdst08oef;

public class Vrachtwagen extends Voertuig {

    private float maxLading = 10000.0F;

    public Vrachtwagen() {
    }

    public Vrachtwagen(String polishouder, float kostprijs, int pk,
            float gemVerbruik, String nummerplaat, float maxLading) {
        super(polishouder,kostprijs,pk,gemVerbruik,nummerplaat);
        setMaxLading(maxLading);
    }

    public float getMaxLading() {
        return maxLading;
    }
    public final void setMaxLading(float maxLading) {
        if (maxLading > 0.0F)
            this.maxLading = maxLading;
    }

    @Override
    public String toString() {
        return super.toString() + " ; " + maxLading;
    }

    @Override
    public void toon() {
        System.out.println("\nVRACHTWAGEN");
        super.toon();
        System.out.println("max. lading: " + maxLading);
    }
```

```
    @Override
    public double getKyotoScore() {
        return (getGemVerbruik() * getPk() / (maxLading / 1000.0F) );
        //lading omzetten van kg naar ton
    }
}
```

17.1.3 De class Personenwagen + getKyotoScore()

```
package jpfhfdst08oef;

public class Personenwagen extends Voertuig {
    private int aantalDeuren = 4;
    private int aantalPassagiers = 5;

    public Personenwagen() {
    }

    public Personenwagen(String polishouder, float kostprijs,
            int pk, float gemVerbruik, String nummerplaat,
            int deuren, int passagiers) {
        super(polishouder,kostprijs,pk,gemVerbruik,nummerplaat);
        setAantalDeuren(deuren);
        setAantalPassagiers(passagiers);
    }

    public int getAantalDeuren() {
        return aantalDeuren;
    }
    public final void setAantalDeuren(int aantalDeuren) {
        if (aantalDeuren > 0)
            this.aantalDeuren = aantalDeuren;
    }

    public int getAantalPassagiers() {
        return aantalPassagiers;
    }
    public final void setAantalPassagiers(int aantalPassagiers) {
        if (aantalPassagiers > 0)
            this.aantalPassagiers = aantalPassagiers;
    }

    @Override
    public String toString() {
        return super.toString() + " ; " +
               aantalDeuren + " ; " + aantalPassagiers ;
    }

    @Override
    public void toon() {
        System.out.println("\nPERSONENWAGEN");
        super.toon();
        System.out.println("deuren: " + aantalDeuren) ;
        System.out.println("passagiers: " + aantalPassagiers);
    }

    @Override
    public double getKyotoScore() {
        return getGemVerbruik() * getPk() / aantalPassagiers;
    }
```

```
}
```

17.1.4 Het hoofdprogramma

```
package jpfhfdst08oef;

public class TestProgramma {

    public static void main(String[] args) {
        Personenwagen opel1 = new Personenwagen();
        opel1.toon();
        System.out.println(opel1);

        Personenwagen opel2 = new Personenwagen("Jan Klaasen",
                14599.0F, 105, 6.8F, "1-KLM-099", 5, 5);
        opel2.toon();
        System.out.println(opel2);

        opel2.setKostprijs(-15000);
        opel2.setAantalDeuren(-7);
        opel2.setAantalPassagiers(0);
        System.out.println(opel2); //opel2 is niet gewijzigd

        Vrachtwagen volvo1 = new Vrachtwagen();
        volvo1.toon();
        System.out.println(volvo1);

        Vrachtwagen volvo2 = new Vrachtwagen("Michel Dewolf",
                214599.0F, 440, 33.1F, "1-PRD-441", 6000.0F);
        volvo2.toon();
        System.out.println(volvo2);

       System.out.println();
       System.out.println("Kyotoscore personenwagen 1: " + opel1.getKyotoScore());
       System.out.println("Kyotoscore personenwagen 2: " + opel2.getKyotoScore());
       System.out.println("Kyotoscore vrachtwagen 1: " + volvo1.getKyotoScore());
       System.out.println("Kyotoscore vrachtwagen 2: " + volvo2.getKyotoScore());

        //polymorfisme
        Voertuig voertuigen[] = new Voertuig[4];
        voertuigen[0] = opel1;
        voertuigen[1] = opel2;
        voertuigen[2] = volvo1;
        voertuigen[3] = volvo2;

        System.out.println("\n--- toString()---");
        for (Voertuig voertuig : voertuigen) {
            System.out.println(voertuig);
        }

        System.out.println("\n--- method toon() ---");
        for (Voertuig voertuig : voertuigen) {
            voertuig.toon();
        }
    }
}
```

18 Voorbeeldoplossingen taken van hoofdstuk 9

18.1 Klinkers

```
package jpfhfdst09oef;
import java.util.*;

public class Klinkers {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        char c;
        int aantal = 0;

        System.out.println("Geef een zin in:");
        String zin = sc.nextLine().toLowerCase() ;

        for (int i=0; i < zin.length(); i++) {
            c = zin.charAt(i);
//          if (c=='a' || c=='e' || c=='i' || c=='o' || c=='u') {
//             aantal++;
//          }
            //andere manier
            if ("aeiou".indexOf(c)> -1)
               aantal++;
        }
        System.out.println("Aantal klinkers: " + aantal);
    }
}
```

18.2 Palindroom

18.2.1 De class PalindroomTester

```
package jpfhfdst09oef;

public class PalindroomTester {
    private final StringBuilder orgTekst, revTekst;

    public PalindroomTester(String tekst) {
        orgTekst = new StringBuilder(tekst);
        revTekst = new StringBuilder(tekst);
        revTekst.reverse();
    }

    public boolean isPalindroom(boolean hoofdlettergevoelig) {
        if (!hoofdlettergevoelig) {
            return orgTekst.toString().equals(revTekst.toString() );
            //toString() is nodig: orgTekst en revTekst zijn StringBuilder
        }
        else {
            String ot = orgTekst.toString().toLowerCase();
            String rt = revTekst.toString().toLowerCase();
            return ot.equals(rt);
        }
    }
```

```
    public String getOrgTekst() {
        return orgTekst.toString();
    }
    public String getRevTekst() {
        return revTekst.toString();
    }
}
```

18.2.2 De main class

```
package jpfhfdst09oef;

import java.util.*;

public class PalindroomMain {
  public static void main(String[] args) {
     String woord;
     PalindroomTester palin;
     Scanner sc = new Scanner(System.in);

     System.out.println("Geef het te onderzoeken woord: ");
     woord = sc.next();
     palin = new PalindroomTester(woord);

     System.out.println("Het originele  woord is: " +
       palin.getOrgTekst());
     System.out.println("Het omgekeerde woord is: " +
       palin.getRevTekst());
     if (palin.isPalindroom(false))
        System.out.println("Het woord is een zuiver palindroom.");

     else {
      if (palin.isPalindroom(true))
       System.out.println("Het woord is,los van de hoofdletters, een palindroom");
      else
       System.out.println("Het woord is helemaal geen palindroom!");
       }
    }
}
```

18.3 Rekenaar

18.3.1 De class Rekenaar

```
package jpfhfdst09oef;

public class Rekenaar {
    private String tekst;
    private int totaal;

    public Rekenaar(String input) {
        tekst = input;
        totaal = berekenen(input);
    }

    public void setTekst(String input) {
        tekst = input;
        totaal = berekenen(input);
    }
//Het is belangrijk dat de status van het object in zijn geheel blijft kloppen.
//Wanneer de string-expressie wijzigt, wijzigt ook het resultaat van de berekening,
```

```
//daarom wordt meteen na het setten van de string tekst het totaal berekend.

    public String getTekst() {
        return tekst;
    }

    public int getTotaal() {
        return totaal;
    }

    private int berekenen(String expressie) {
        String[] delen = expressie.split(" ");
        //in geval van een lege expressie bevat de array delen slechts 1 element,
        //nl. een lege string: ""

        int getal;
        char bewerkingsTeken;

        // eerste cijfer in totaal zetten om te sommeren
        if (!delen[0].equals(""))
            totaal=Integer.parseInt(delen[0]);
        else
            totaal=0;

        int i=1;
        while (i<delen.length) {
            bewerkingsTeken = delen[i].charAt(0);
            i++;
            if (i<delen.length) {
                getal = Integer.parseInt(delen[i]);
                voerUit(bewerkingsTeken, getal);
                i++;
            }
        }
        return totaal;
    }

    private void voerUit(char bewTeken, int getal) {

        switch (bewTeken) {
          case '+':
            totaal += getal; break;
          case '-':
            totaal -= getal; break;
          case '*':
            totaal *= getal; break;
          case '/':
            if (getal != 0) {
                totaal /= getal;}
            break;
          default:
            break;
        }
    }

    @Override
    public String toString() {
        return tekst + ";" + totaal;
    }
}
```

18.3.2 De main-class:

```
package jpfhfdst09oef;
import java.util.*; // tbv Scanner

public class RekenaarMain {
    public static void main(String[] args) {
        Rekenaar rek = new Rekenaar("17 + 38 * 2 - 22");
        System.out.println("Totaal: " + rek.getTotaal());
        System.out.println(rek); //toString() wordt hier getoond

        //lege expressie
        rek.setTekst("");
        System.out.println("Totaal: " + rek.getTotaal());
        System.out.println(rek);

        //andere expressie
        rek.setTekst("7 - 5 * 10 / 0 ");
        System.out.println("Totaal: " + rek.getTotaal());
        System.out.println(rek);

        //expressie laten ingeven door gebruiker
        Scanner sc = new Scanner(System.in);
        System.out.println("Geef zelf een te berekenen expressie: ");
        String expressie = sc.nextLine();
        rek.setTekst(expressie);
        System.out.println("Totaal: " + rek.getTotaal());
        System.out.println(rek);
    }
}
```

19 Voorbeeldoplossingen taken van hoofdstuk 10

19.1 VoertuigenI

19.1.1 De class Personenwagen

```
package jpfhfdst10oef;
public class Personenwagen extends Voertuig implements Vervuiler {
…    //alles wat er reeds geschreven was

    @Override
    public double berekenVervuiling() {
        return getKyotoScore() * 5.0F;
    }
}
```

19.1.2 De class Vrachtwagen

```
package jpfhfdst10oef;

public class Vrachtwagen extends Voertuig implements Vervuiler {
…    //alles wat er reeds geschreven was

    @Override
    public double berekenVervuiling() {
        return getKyotoScore() * 20.0F;
    }
}
```

19.1.3 De class Stookketel

```
package jpfhfdst10oef;
public class Stookketel implements Vervuiler {
    private float CONorm;

    public Stookketel() {
    }

    public Stookketel(float CONorm) {
        setCONorm(CONorm);
    }

    public float getCONorm() {
        return this.CONorm;
    }

    public final void setCONorm(float CONorm) {
        if (CONorm > 0.0)
            this.CONorm = CONorm;     }

    @Override
    public double berekenVervuiling() {
        return getCONorm() * 100;
    }
}
```

19.1.4 Het hoofdprogramma uitgebreid met een array van Vervuiler

```
package jpfhfdst10oef;
import java.text.DecimalFormat;

public class TestProgramma {

    public static void main(String[] args) {
      Personenwagen opel1 = new Personenwagen();
      Personenwagen opel2 = new Personenwagen("Jan Klaasen",
                14599.0F, 105, 6.8F, "1-KLM-099", 5, 5);

      Vrachtwagen volvo1 = new Vrachtwagen();
      Vrachtwagen volvo2 = new Vrachtwagen("Michel Dewolf",
                214599.0F, 440, 33.1F, "1-PRD-441", 6000.0F);

      DecimalFormat fmt = new DecimalFormat("#0.00");

      System.out.println("\n*** Kyotoscore ***");
       System.out.println("Kyotoscore : " + fmt.format(opel1.getKyotoScore()));
       System.out.println("Kyotoscore : " + fmt.format(opel2.getKyotoScore()));
       System.out.println("Kyotoscore : " + fmt.format(volvo1.getKyotoScore()));
       System.out.println("Kyotoscore : " + fmt.format(volvo2.getKyotoScore()));

      System.out.println("\n*** Array van vervuilers ***");
      Vervuiler[] vervuilers = new Vervuiler[5];
      vervuilers[0] = opel1;
      vervuilers[1] = opel2;
      vervuilers[2] = volvo1;
      vervuilers[3] = volvo2;
      vervuilers[4] = new Stookketel(7.5F);
      for (Vervuiler vervuiler:vervuilers) {
       System.out.println("vuil: "+ fmt.format(vervuiler.berekenVervuiling()) );
        }

    }
}
```

19.1.5 Interface Milieu en Privaat

```
package jpfhfdst10oef;
public interface Milieu {
    public void geefMilieuData();
}

package jpfhfdst10oef;
public interface Privaat {
    public void geefPrivateData();
}
```

19.1.6 De class Voertuig

```
public abstract class Voertuig implements Privaat, Milieu {
…   //alles wat er reeds geschreven was

    @Override
    public void geefPrivateData() {
       System.out.println("--- Private data van voertuig ---");
       System.out.println("Polishouder :" + getPolishouder());
       System.out.println("Nummerplaat :" + getNummerplaat());
```

```
    }


    @Override
    public void geefMilieuData() {
       System.out.println("--- Milieu data van voertuig ---");
       System.out.println("PK :" + getPk());
       System.out.println("Kostprijs :" + getKostprijs());
       System.out.println("Gem. verbruik :" + getGemVerbruik());
    }
}
```

19.1.7 Het hoofdprogramma uitgebreid met een array van Privaat en Milieu

Volgende code kan toegevoegd worden:

```
        System.out.println("\n*** Array van private geg van auto's ***");
        Privaat[] voertuigen = new Privaat[4];
        voertuigen[0] = opel1;
        voertuigen[1] = opel2;
        voertuigen[2] = volvo1;
        voertuigen[3] = volvo2;
        for (Privaat auto:voertuigen) {
           auto.geefPrivateData();
        }

        System.out.println("\n*** Array van milieu geg van auto's ***");
        Milieu[] voertuigen2 = new Milieu[4];
        voertuigen2[0] = opel1;
        voertuigen2[1] = opel2;
        voertuigen2[2] = volvo1;
        voertuigen2[3] = volvo2;
        for (Milieu auto:voertuigen2) {
           auto.geefMilieuData();
        }
```

19.2 Voorwerpen

19.2.1 De interface Voorwerp

```
package jpfvoorwerpen;
public interface Voorwerp {
    public final static String EIGENAAR = "VDAB";

    void gegevensTonen();
    float winstBerekenen();
}
```

19.2.2 De class Boekenrek

```
package jpfvoorwerpen;
public class Boekenrek implements Voorwerp {
    private int hoogte = 0;
    private int breedte = 0;
    private float aankoopPrijs = 0;
    private static final float WINSTMARGE = 2F;

    public Boekenrek() {
      this (175, 100, 40.0F) ;  // eenheid in cm
    }
```

```
    public Boekenrek(int hoogte, int breedte, float aankoopPrijs) {
        setHoogte(hoogte);
        setBreedte(breedte);
        setAankoopPrijs(aankoopPrijs);
    }

    public int getHoogte() {
        return hoogte ;
    }
    public final void setHoogte(int hoogte) {
        if (hoogte > 0)
            this.hoogte = hoogte;
    }

    public int getBreedte() {
        return breedte ;
    }
    public final void setBreedte(int breedte) {
        if (breedte > 0)
            this.breedte = breedte ;
    }

    public float getAankoopPrijs() {
        return aankoopPrijs ;
    }
    public final void setAankoopPrijs(float aankoopPrijs) {
        if (aankoopPrijs > 0.0)
            this.aankoopPrijs = aankoopPrijs ;
    }

    @Override
    public void gegevensTonen() {
      System.out.println("GEGEVENS VAN DE BOEKENREK ") ;
      System.out.println("Het boekenrek is " + hoogte + " cm hoog en "
                                             + breedte  + " cm breed.");
      System.out.println("Het is eigendom van : " + EIGENAAR);
      System.out.println("Aankoopprijs        : " + aankoopPrijs);
      System.out.println("Winst               : " + winstBerekenen());
    }

    @Override
    public float winstBerekenen() {
        return aankoopPrijs * WINSTMARGE ;
    }

    @Override
    public String toString() {
        return (hoogte + " ; " + breedte + " ; " + EIGENAAR + " ; " +
                aankoopPrijs + " ; " + WINSTMARGE);
    }
}
```

19.2.3 De class Boek

```
package jpfvoorwerpen;

public abstract class Boek implements Voorwerp {
    private String titel;
    private String auteur;
    private float aankoopPrijs;
```

```
    private String genre;

    public Boek(String titel, String auteur, float aankoopPrijs, String genre) {
        setTitel(titel);
        setAuteur(auteur);
        setAankoopPrijs(aankoopPrijs);
        setGenre(genre);
    }

    public String getTitel() {
        return titel;
    }
    public final void setTitel(String titel) {
        if (titel != null)
            this.titel = titel;
    }

    public String getAuteur() {
        return auteur;
    }
    public final void setAuteur(String auteur) {
        if (auteur != null)
            this.auteur = auteur;
    }

    public float getAankoopPrijs() {
        return aankoopPrijs;
    }
    public final void setAankoopPrijs(float aankoopPrijs) {
        if (aankoopPrijs > 0.0)
            this.aankoopPrijs = aankoopPrijs;
    }

    public String getGenre() {
        return genre;
    }
    public final void setGenre(String genre) {
        if (genre != null)
            this.genre = genre;
    }

    @Override
    public void gegevensTonen() {
      System.out.println("GEGEVENS VAN EEN BOEK ") ;
      System.out.println("Titel                : " + titel) ;
      System.out.println("Auteur               : " + auteur) ;
      System.out.println("Het is eigendom van  : " + EIGENAAR);
      System.out.println("Aankoopprijs         : " + aankoopPrijs);
      System.out.println("Genre                : " + genre);
    }

    @Override
    public String toString() {
      return (titel + " ; " + auteur + " ; " + EIGENAAR + " ; " +
              aankoopPrijs + " ; " + genre);
    }
}
```

19.2.4 De class Leesboek

```
package jpfvoorwerpen;

public class Leesboek extends Boek {
    private String onderwerp;
    private static final float WINSTMARGE = 1.5F;

    public Leesboek() {
      this("Leesboek Java ", "O Reilly", 10.5F, "genre studie",
           "onderw Informatica" );
    }

    public Leesboek(String titel, String auteur, float aankoopPrijs,
                    String genre, String onderwerp ) {
      super(titel, auteur, aankoopPrijs, genre);
      setOnderwerp(onderwerp) ;
    }

    public String getOnderwerp() {
      return onderwerp;
    }
    public final void setOnderwerp(String onderwerp) {
        if (onderwerp != null)
            this.onderwerp = onderwerp;
    }

    public float getWINSTMARGE() {
        return WINSTMARGE;
    }

    @Override
    public void gegevensTonen() {
      super.gegevensTonen();
      System.out.println("Onderwerp           : " + onderwerp);
      System.out.println("Winst               : " + winstBerekenen() );
    }

    @Override
    public float winstBerekenen() {
      return super.getAankoopPrijs() * WINSTMARGE;
    }

    @Override
    public String toString() {
      return (super.toString() + " ; " + onderwerp + " ; " + WINSTMARGE);
    }
}
```

19.2.5 De class Woordenboek

```
package jpfvoorwerpen;

public class Woordenboek extends Boek {
    private String taal;
    private static final float WINSTMARGE = 1.75F;

    public Woordenboek() {
      this("Woordenboek Nederlands", "Van Dale", 25.8F,
           "verklarend woordenboek","taal Nederlands" );
    }
```

```
    public Woordenboek(String titel, String auteur, float aankoopPrijs,
                       String genre, String taal ) {
      super(titel, auteur, aankoopPrijs, genre);
      setTaal(taal) ;
    }

    public String getTaal() {
      return taal;
    }
    public final void setTaal(String taal) {
      if (taal != null)
        this.taal = taal;
    }

    public float getWINSTMARGE() {
        return WINSTMARGE;
    }

    @Override
    public void gegevensTonen() {
      super.gegevensTonen();
      System.out.println("Taal                : " + taal );
      System.out.println("Winst               : " + winstBerekenen() );
    }

    @Override
    public float winstBerekenen() {
      return super.getAankoopPrijs() * WINSTMARGE;
    }

    @Override
    public String toString() {
          return (super.toString() + " ; " + taal + " ; " + WINSTMARGE );
    }
}
```

19.2.6 Het hoofdprogramma

```
package jpfvoorwerpen;
public class Hoofdprogramma {

    public static void main(String[] args) {
      Voorwerp[] voorwerpen = {
        new Boekenrek(),
        new Boekenrek(75, 90, 28.5F),

        new Leesboek(),
        new Leesboek("Leesboek Java 7", "Oracle", 20.4F,
                     "genre Informatica", "onderw programmeren" ),

        new Woordenboek(),
        new Woordenboek("Woordenboek Engels", "Van Dale", 30.10F,
                        "genre vertaal woordenboek",
                        "taal Engels-Nederlands")
      } ;

      float totaleWinst = 0;

      for (Voorwerp eenVoorwerp: voorwerpen) {
        eenVoorwerp.gegevensTonen();
```

```
            System.out.println();
            totaleWinst += eenVoorwerp.winstBerekenen();
        }
        System.out.println("DE TOTALE WINST BEDRAAGT " + totaleWinst);
    }
}
```

20 Voorbeeldoplossingen taken van hoofdstuk 11

20.1 Voorwerpen (vervolg)

20.1.1 De interface Voorwerp

```
package be.vdab.util;
public interface Voorwerp {
    public final static String EIGENAAR = "VDAB";

    void gegevensTonen();
    float winstBerekenen();
}
```

20.1.2 De class Boekenrek

```
package be.vdab.voorwerpen;
import be.vdab.util.Voorwerp;
public class Boekenrek implements Voorwerp {
    …
}
```

20.1.3 De class Boek

```
package be.vdab.voorwerpen;
import be.vdab.util.Voorwerp;
public abstract class Boek implements Voorwerp {
    …
}
```

20.1.4 De class Leesboek

```
package be.vdab.voorwerpen;
public class Leesboek extends Boek {
    …
}
```

20.1.5 De class Woordenboek

```
package be.vdab.voorwerpen;
public class Woordenboek extends Boek {
    …
}
```

20.1.6 Het hoofdprogramma

```
package be.vdab;

import be.vdab.util.Voorwerp;
import be.vdab.voorwerpen.Boekenrek;
import be.vdab.voorwerpen.Leesboek;
import be.vdab.voorwerpen.Woordenboek;

public class Hoofdprogramma {
    …
}
```

20.2 VoertuigenP

20.2.1 De interface Vervuiler

```
package be.vdab.util;
public interface Vervuiler {
    double berekenVervuiling();
}
```

20.2.2 De interface Privaat

```
package be.vdab.util;
public interface Privaat {
    public void geefPrivateData();
}
```

20.2.3 De interface Milieu

```
package be.vdab.util;
public interface Milieu {
    public void geefMilieuData();
}
```

20.2.4 De class Voertuig

```
package be.vdab.voertuigen;
import be.vdab.util.Milieu;
import be.vdab.util.Privaat;
public abstract class Voertuig implements Privaat, Milieu {
    …
}
```

20.2.5 De class Vrachtwagen

```
package be.vdab.voertuigen;
import be.vdab.util.Vervuiler;
public class Vrachtwagen extends Voertuig implements Vervuiler {
    …
}
```

20.2.6 De class Personenwagen

```
package be.vdab.voertuigen;
import be.vdab.util.Vervuiler;
public class Personenwagen extends Voertuig implements Vervuiler {
    …
}
```

20.2.7 De class Stookketel

```
package be.vdab.verwarming;
import be.vdab.util.Vervuiler;
public class Stookketel implements Vervuiler {
    …
}
```

20.2.8 Het hoofdprogramma

```
package be.vdab;
```

```
import be.vdab.util.Milieu;
import be.vdab.util.Privaat;
import be.vdab.util.Vervuiler;
import be.vdab.verwarming.Stookketel;
import be.vdab.voertuigen.Personenwagen;
import be.vdab.voertuigen.Vrachtwagen;
import java.text.DecimalFormat;

public class TestProgramma {
    …
}
```

21 Voorbeeldoplossingen taken van hoofdstuk 12

21.1 Oplossing TestExceptions

Resultaat met `String test = "no";`

```
start try
start risky
end risky
end try
finally
end of main
```

Resultaat me t `String test = "yes";`

```
start try
start risky
scary exception
finally
end of main
```

21.2 Oplossing Controle ISBN13nummer

21.2.1 De ISBN13Exception class

```
package be.vdab.util;

public class ISBN13Exception extends Exception {
    public ISBN13Exception() {}
    public ISBN13Exception(String omschrijving) {
        super(omschrijving);
    }
}
```

21.2.2 De class Boek

Een private membervariabele ISBN13 met type String is toegevoegd. De constructor is hiervoor aangepast en er is een getter en setter voorzien. De setter bevat de controle op het laatste digit van het ISBN-nr. De methods gegevensTonen() en toString() bevatten nu ook het ISBN13 nummer. Overige methods blijven ongewijzigd.

```
package be.vdab.voorwerpen;

import be.vdab.util.ISBN13Exception;
import be.vdab.util.Voorwerp;

public abstract class Boek implements Voorwerp {
    private String ISBN13;
    private String titel;
    private String auteur;
    private float aankoopPrijs;
    private String genre;

    public Boek(String ISBN13, String titel, String auteur, float
aankoopPrijs, String genre) throws ISBN13Exception {
```

```
        setISBN13(ISBN13);
        setTitel(titel);
        setAuteur(auteur);
        setAankoopPrijs(aankoopPrijs);
        setGenre(genre);
    }

    public String getISBN13() {
        return ISBN13;
    }
    public final void setISBN13(String ISBN13) throws ISBN13Exception {
        if (checkISBN13(ISBN13)) {
            this.ISBN13 = ISBN13;
        }
        else {
            throw new ISBN13Exception("laatste cijfer is ongeldig");
        }
    }

    …

    @Override
    public void gegevensTonen() {
      System.out.println("GEGEVENS VAN EEN BOEK ") ;
      System.out.println("ISBN13                : " + ISBN13) ;
      System.out.println("Titel                 : " + titel) ;
      System.out.println("Auteur                : " + auteur) ;
      System.out.println("Het is eigendom van   : " + EIGENAAR);
      System.out.println("Aankoopprijs          : " + aankoopPrijs);
      System.out.println("Genre                 : " + genre);
    }

    @Override
    public String toString() {
      return (ISBN13 + " ; " + titel + " ; " + auteur + " ; " + EIGENAAR +
         " ; " + aankoopPrijs + " ; " + genre);
    }

    private boolean checkISBN13(String isbn) {
        if (isbn == null || isbn.isEmpty()) {
            return false;
        }

        //verwijder de streepjes
        isbn = isbn.replaceAll( "-", "" );

        //controle op lengte van 13 tekens
        if (isbn.length() != 13) {
            return false;
        }

        try {
            int tot = 0;
            int factor;
            for (int i = 0; i < 12; i++ ) {
                int digit = Integer.parseInt(isbn.substring( i, i + 1 ));
                //bepalen of er vermenigvuldigd moet worden met 1 of 3
                factor = (i % 2 == 0) ? 1 : 3;
                tot += digit * factor;
            }
```

```
            int checksom = 10 - (tot % 10);
            if (checksom == 10) {
                checksom = 0;
            }

            return checksom == Integer.parseInt(isbn.substring(12));
        }
        catch ( NumberFormatException ex ) {
            return false;
        }
    }
```

21.2.3 De class Leesboek

Een import statement voor de exception is vereist en verder dienen enkel de constructors aangepast te worden:

```
    public Leesboek() throws ISBN13Exception {
      this("978-05-960-0920-5", "Leesboek Java", "O Reilly", 10.5F,
           "genre studie", "onderw Informatica");
    }

    public Leesboek(String ISBN13, String titel, String auteur,
      float aankoopPrijs, String genre, String onderwerp)
      throws ISBN13Exception {
        super(ISBN13, titel, auteur, aankoopPrijs, genre);
        setOnderwerp(onderwerp) ;
    }
```

21.2.4 De class Woordenboek

Een import statement voor de exception is vereist en verder dienen enkel de constructors aangepast te worden:

```
    public Woordenboek() throws ISBN13Exception {
      this("978-90-664-8384-2", "Woordenboek Engels", "Van Dale", 25.8F,
           "verklarend woordenboek","taal Nederlands");
    }

    public Woordenboek(String ISBN13, String titel, String auteur,
      float aankoopPrijs, String genre, String taal)
      throws ISBN13Exception {
        super(ISBN13, titel, auteur, aankoopPrijs, genre);
        setTaal(taal) ;
    }
```

21.2.5 Het hoofdprogramma

```
package be.vdab;

import be.vdab.util.ISBN13Exception;
import be.vdab.util.Voorwerp;
import be.vdab.voorwerpen.Leesboek;
import be.vdab.voorwerpen.Woordenboek;

public class Hoofdprogramma {

    public static void main(String[] args) {
        Voorwerp[] voorwerpen = new Voorwerp [3];
```

```
        try {
            voorwerpen[0]= new Leesboek();
        }
        catch (ISBN13Exception ex) {
            System.out.println(ex.getMessage());
        }

        try {
            voorwerpen[1]= new Leesboek("978-14-302-4764-7",
            "Studieboek Java 7", "Oracle", 20.4F,
            "genre Informatica", "onderw programmeren" );
        }
        catch (ISBN13Exception ex) {
            System.out.println(ex.getMessage());
        }

        try {
            voorwerpen[2] = new Woordenboek("978-90-664-8143-9",
             "Woordenboek Engels", "Van Dale", 30.10F,
             "genre vertaal woordenboek", "taal Engels-Nederlands");
            }
        catch (ISBN13Exception ex) {
            System.out.println(ex.getMessage());
        }

        float totaleWinst = 0;
        for (Voorwerp eenVoorwerp: voorwerpen) {
            if (eenVoorwerp != null) {
                eenVoorwerp.gegevensTonen();
                System.out.println();
                totaleWinst += eenVoorwerp.winstBerekenen();
            }
        }
        System.out.println("DE TOTALE WINST BEDRAAGT " + totaleWinst);
    }
}
```

Het laatste woordenboek veroorzaakt de fout (het laatste cijfer van het ISBNnr moet immers een 5 zijn.

22 Voorbeeldoplossingen taken van hoofdstuk 14

22.1 Land

22.1.1 De class LandException

```
package be.vdab.util;

public class LandException extends Exception {
    public LandException() {}
    public LandException(String omschrijving) {
        super(omschrijving);
    }
}
```

22.1.2 De class Land

```
package be.vdab.land;

import be.vdab.util.LandException;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.math.RoundingMode;

public class Land {
    private String landCode, landNaam, hoofdstad;
    private BigInteger aantInwoners;
    private BigDecimal oppervlakte;

    public Land(String code, String naam, String hoofdstad,
                BigInteger inwoners, BigDecimal opp) {
        try {
            setLandCode(code);
            setLandNaam(naam);
            setHoofdstad(hoofdstad);
            setAantalInwoners(inwoners);
            setOppervlakte(opp);
        }
        catch (LandException ex) {
            System.out.println(ex.getMessage());
        }
    }

    public String getLandCode() {
        return landCode;
    }
    public final void setLandCode(String landCode) throws LandException {
        if ((landCode==null) || landCode.isEmpty()) {
            throw new LandException("Landcode verplicht in te vullen");
        }
        else {
            this.landCode = landCode;
        }
    }

    public String getLandNaam() {
        return landNaam;
    }
    public final void setLandNaam(String landNaam) throws LandException {
```

```
        if ((landNaam==null) || landNaam.isEmpty()) {
            throw new LandException("Landnaam verplicht in te vullen");
        }
        else {
            this.landNaam = landNaam;
        }
    }

    public String getHoofdstad() {
        return hoofdstad;
    }
    public final void setHoofdstad(String hoofdstad) throws LandException {
        if ((hoofdstad==null) || hoofdstad.isEmpty()) {
            throw new LandException("Hoofdstad verplicht in te vullen");
        }
        else {
            this.hoofdstad = hoofdstad;
        }
    }

    public BigInteger getAantInwoners() {
        return aantInwoners;
    }
    public final void setAantalInwoners(BigInteger inwoners) throws
        LandException {
        if (inwoners.compareTo(BigInteger.ZERO)<=0) {
            throw new LandException("Aantal inwoners moet groter dan 0
            zijn");
        }
        else {
            aantInwoners = inwoners;
        }
    }

    public BigDecimal getOppervlakte() {
        return oppervlakte;
    }
    public final void setOppervlakte(BigDecimal opp) throws LandException {
        if (opp.compareTo(BigDecimal.ZERO)<=0) {
            throw new LandException("Oppervlakte moet groter dan 0 zijn");
        }
        else {
            oppervlakte = opp;
        }
    }

    public BigDecimal bevolkingsDichtheid() {
        BigDecimal bevolkingsDichtHeid;
        bevolkingsDichtHeid = (new BigDecimal(aantInwoners)
                              ).divide(oppervlakte, 2, RoundingMode.HALF_UP);
        return bevolkingsDichtHeid;
    }

    @Override
    public String toString() {
        return landCode + " ; "+ landNaam + " ; " + hoofdstad + " ; " +
               aantInwoners + " ; " + oppervlakte + " ; " +
               String.format("%5.2f",this.bevolkingsDichtheid());
    }
}
```

22.1.3 De main-class

```
package be.vdab;
import be.vdab.land.Land;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.math.RoundingMode;
import java.text.DecimalFormat;
import java.util.List;
import java.util.ArrayList;

public class LandMain {
  public static void main(String[] args) {
      Land hetLand=null;
      BigInteger totInwoners = BigInteger.ZERO;
      BigDecimal totOppervlakte = BigDecimal.ZERO;
      BigDecimal gemBevolkingsdichtheid;// = BigDecimal.ZERO;
      BigDecimal absVerschil = BigDecimal.valueOf(Double.MAX_VALUE);

      List<Land> landen = new ArrayList<>();
      landen.add(new Land("Ag", "Argentinië", "Buenos Aires",
BigInteger.valueOf(38500000L), BigDecimal.valueOf(2825647.56)) );
      landen.add(new Land("Bg", "Bulgarije", "Sofia",
BigInteger.valueOf(7800000L), BigDecimal.valueOf(111002.42)) );
      landen.add(new Land("Ey", "Egypte", "Cairo",
BigInteger.valueOf(72000000L), BigDecimal.valueOf(997739.83)) );
      landen.add(new Land("In", "India", "New Delhi",
BigInteger.valueOf(1060000000L), BigDecimal.valueOf(3336251.12)) );
      landen.add(new Land("It", "Italië", "Rome",
BigInteger.valueOf(57840000L), BigDecimal.valueOf(301268.14)) );
      landen.add(new Land("L", "Luxemburg", "Luxemburg",
BigInteger.valueOf(462690L), BigDecimal.valueOf(2586.27)) );
      landen.add(new Land("N", "Noorwegen", "Oslo",
BigInteger.valueOf(4600000L), BigDecimal.valueOf(386958.22)) );
      landen.add(new Land("B", "België", "Brussel",
BigInteger.valueOf(10400000L), BigDecimal.valueOf(30528.56)) );
      landen.add(new Land("Si", "Singapore", "Singapore",
BigInteger.valueOf(4200000L), BigDecimal.valueOf(640.94)) );
      landen.add(new Land("Us", "Verenigde Staten", "Washington DC",
BigInteger.valueOf(293000000L), BigDecimal.valueOf(9165487.63)) );

      for (Land eenLand : landen) {
          if(eenLand != null) {
            totOppervlakte = totOppervlakte.add( eenLand.getOppervlakte());
            totInwoners = totInwoners.add(eenLand.getAantInwoners());
            System.out.println(eenLand);
          }
      }
      gemBevolkingsdichtheid = (new BigDecimal(totInwoners)
                  ).divide(totOppervlakte, 2, RoundingMode.HALF_UP) ;

      for (Land eenLand : landen) {
          if ( ((eenLand.bevolkingsDichtheid()
                  .subtract(gemBevolkingsdichtheid))
                .abs())
                .compareTo(absVerschil) < 0 ) {
                hetLand = eenLand;
                absVerschil = (eenLand.bevolkingsDichtheid()
                              .subtract(gemBevolkingsdichtheid)).abs();
          }
      }
```

```
        DecimalFormat fmt = new DecimalFormat("#0.00");
        System.out.println("\nDe gemiddelde bevolkingsdichtheid is " +
            fmt.format(gemBevolkingsdichtheid));
        System.out.println("Het land dat het dichtst aanleunt bij dit
            gemiddelde is " + hetLand.getLandNaam() +
            " met een bevolkingsdichtheid van " +
            fmt.format(hetLand.bevolkingsDichtheid() ));
    }
}
```

22.2 VoertuigenC

22.2.1 De class Voertuig

Hier wordt weergegeven: de header van de class Voertuig met implements van de interface Comparable<Voertuig>, de equals method, de hashCode() method en de compareTo() method.

```
package be.vdab.voertuigen;
import be.vdab.util.Milieu;
import be.vdab.util.Privaat;
import java.util.Objects;

public abstract class Voertuig implements Privaat, Milieu,
Comparable<Voertuig> {

…

    @Override
    public boolean equals(Object o) {
        if (o == null) {
            return false;
        }
        if (!(o instanceof Voertuig)) {
            return false;
        }
        Voertuig v = (Voertuig) o;
        return nummerplaat.equals(v.getNummerplaat() );
    }

    @Override
    public int hashCode() {
        int hash = 5;
        hash = 43 * hash + Objects.hashCode(this.nummerplaat);
        return hash;
    }

    @Override
    public int compareTo(Voertuig v) {
        if (v == null)
            throw new NullPointerException();
        else
            if ( this.equals(v) )
                return 0;
            else
                return nummerplaat.compareTo(v.getNummerplaat()) ;
    }

…

}
```

De equals() method is gegenereerd door NetBeans (laten baseren op de nummerplaat). De class Objects is afgeleid van Object en beschikt over statische methods die toegepast kunnen worden op objecten, zoals het bepalen van de hashcode van een string.

22.2.2 Het Hoofdprogramma

Uitbreidingen aan het vorige TestProgramma zijn de volgende:

```
import java.util.Set;
import java.util.TreeSet;
```

In de main kan je bijvoorbeeld volgende lijnen toevoegen:

```
        Set<Voertuig> setVoertuigen = new TreeSet<>();
        setVoertuigen.add(opel2);
        setVoertuigen.add(new Personenwagen("Piet Peeters",
                18321.0F, 110, 7.1F, "1-OPQ-099", 5, 5));
        setVoertuigen.add(volvo2);
        setVoertuigen.add(new Vrachtwagen("Jan Vos",
                254612.0F, 450, 33.1F, "1-AZE-123", 6200.0F));

        System.out.println("\n*** TreeSet van voertuigen ***");
        for (Voertuig eenVoertuig : setVoertuigen) {
            System.out.println(eenVoertuig);
```

22.3 Tienkamper

22.3.1 De class Tienkamper

```
package be.vdab.tienkamper;
import java.util.Comparator;

public class Tienkamper implements Comparable<Tienkamper> {
    private String naam="";
    private int punten;

    public Tienkamper(String naam, int punten) {
        setNaam(naam);
        setPunten(punten);
    }

    public String getNaam() {
        return naam;
    }
    public final void setNaam(String naam) {
        if (naam != null) {
            this.naam = naam;
        }
    }
    public int getPunten() {
        return punten;
    }
    public final void setPunten(int punten) {
        if (punten>=0) {
            this.punten = punten;
        }
    }
```

```
    @Override
    public String toString() {
        return naam + " - " + punten;
    }

    @Override
    public boolean equals(Object object) {
        if (object == null)
            return false;
        if (! (object instanceof Tienkamper)) {
          return false;
        }
        Tienkamper andere = (Tienkamper) object ;
        return naam.equals(andere.naam);
    }

    @Override
    public int compareTo(Tienkamper tienkamper) {
        return naam.compareTo(tienkamper.naam);
    }

    @Override
    public int hashCode() {
        return naam.hashCode();
    }
}
```

22.3.2 Het hoofdprogramma

```
package be.vdab.tienkamper;
import java.util.List;
import java.util.ArrayList;
import java.util.Set;
import java.util.TreeSet;

public class TienkamperMain {

  public static void main(String[] args) {

    Tienkamper Jurgen = new Tienkamper("Jurgen Hingsen",8832);
    Tienkamper Roman = new Tienkamper("Roman Sebrle",8891);
    Tienkamper Daley = new Tienkamper("Daley Thompson",8847);
    Tienkamper Dan = new Tienkamper("Dan O'Brien",8891);

    List<Tienkamper> lijst = new ArrayList<>();
    lijst.add(Jurgen);
    lijst.add(Roman);
    lijst.add(Daley);
    lijst.add(Dan);

    System.out.println("Alle tienkampers uit de arraylist (=volgorde van
toevoegen):");
    for (Tienkamper eenTienkamper:lijst)
       System.out.println(eenTienkamper);
    System.out.println();

    Set<Tienkamper> set = new TreeSet<>();
    set.add(Jurgen);
    set.add(Roman);
    set.add(Daley);
```

```
    set.add(Dan);

    System.out.println("Alle tienkampers uit de treeset (=gesorteerd op
naam):");
    for (Tienkamper eenTienkamper:set)
       System.out.println(eenTienkamper);
    System.out.println();
  }
}
```

22.4 Beginletter

```
package be.vdab;
import java.util.*;

public class BeginLetter {
  private static final String woorden[] = { "een", "twee", "drie",
      "vier", "vijf", "zes", "zeven", "acht", "negen", "tien" };

  public static void main( String args[] ) {
    Map <Character,Integer> hashMapAantal = new HashMap<>();

    //1e manier: lezen met key en begin-value putten of value verhogen
    //en terug putten
    for (String woord : woorden) {
      if (hashMapAantal.get(woord.charAt(0)) == null) {
         hashMapAantal.put(woord.charAt(0), 1);
      }
      else {
         hashMapAantal.put(woord.charAt(0),hashMapAantal.get(woord.charAt(0))+1) ;
      }
    }

    System.out.println(
      "\nAantal woorden die beginnen met elke letter:    " );
    System.out.println(hashMapAantal); //toString() HashMap wordt gebruikt
    System.out.println("size: " + hashMapAantal.size());
    System.out.println("isEmpty: " + hashMapAantal.isEmpty());

    System.out.println("------ Set van de keys ------");
    for (char c : hashMapAantal.keySet()) {
          System.out.println(c);
    }

    System.out.println("------ Set van de values ------");
    for (int i : hashMapAantal.values()) {
        System.out.println(i);
    }

    System.out.println("------ Set van de map.entries ------");
    for (Map.Entry<Character,Integer> entry : hashMapAantal.entrySet()) {
        System.out.print(entry); //toString() Map.Entry wordt gebruikt
        System.out.println(" of anders: "  + entry.getKey() + " : " +
         entry.getValue() );
    }
  }
}
```

Er is nog een tweede manier om het aantal letters te tellen in de hashMap:
vervang de for-lus van de 1e manier als volgt:

```
    // 2e manier: met contains eerst nagaan of key aanwezig is
    for (String woord : woorden) {
       if (hashMapAantal.containsKey(woord.charAt(0)) == false) {
         hashMapAantal.put(woord.charAt(0) , 1);
       }
       else {
         hashMapAantal.put(woord.charAt(0) ,hashMapAantal.get(woord.charAt(0))+1);
       }
    }
```

22.5 Winkel

22.5.1 De class Product

```
package be.vdab.winkel;

public class Product {
    private String omschrijving="";
    private double prijs;

    public Product() {
    }

    public Product(String oms, double prijs ) {
       setOmschrijving(oms);
       setPrijs(prijs);
    }

    public String getOmschrijving() {
        return omschrijving;
    }

    public final void setOmschrijving(String omschrijving) {
        if (omschrijving != null && !omschrijving.isEmpty()) {
            this.omschrijving = omschrijving;
        }
    }

    public double getPrijs() {
        return prijs;
    }

    public final void setPrijs(double prijs) {
        if (prijs >=0 ){
            this.prijs = prijs;
        }
    }

    @Override
    public boolean equals(Object o) {
        if (o == null) {
            return false;
        }
        if (!(o instanceof Product)) {
            return false;
        }
        Product p = (Product) o;
        return (omschrijving.equals(p.omschrijving) && prijs == p.prijs);
    }

    @Override
```

```
    public int hashCode() {
        return toString().hashCode();
    }

    @Override
    public String toString() {
        return String.format ("%s ; %.2f ",  omschrijving, prijs);
    }
}
```

22.5.2 De class Catalogus

```
package be.vdab.winkel;

import java.util.ArrayList;
import java.util.List;

public class Catalogus {
    private final List <Product> lijstCatalogus;

     // Opmerking: List: er kan 2x hetzelfde product in de catalogus zitten
     // --> wat wil je ermee kunnen doen, wat is er beschreven

    public Catalogus() {
        lijstCatalogus = new ArrayList<> ();
        lijstCatalogus.add(new Product ("1kg appelen", 1.34 ) );
        lijstCatalogus.add(new Product ("1kg peren", 1.56) );
        lijstCatalogus.add(new Product ("netje citroenen", 0.64 ) ) ;
        lijstCatalogus.add(new Product ("bakje aardbeien", 2.85 ) ) ;
        lijstCatalogus.add(new Product ("1kg mandareinen", 1.60 ) ) ;
        lijstCatalogus.add(new Product ("0.5kg noten", 2.35 ) ) ;
        lijstCatalogus.add(new Product ("0.5kg kastanjes", 3.15 ) ) ;
        lijstCatalogus.add(new Product ("zakje rozijnen", 1.90 ) ) ;
    }

    public List<Product> getLijstCatalogus() {
        return lijstCatalogus;
    }

    @Override
    public String toString() {
        return lijstCatalogus.toString();
    }
}
```

22.5.3 De class Mandje

```
package be.vdab.winkel;

import java.util.HashMap;
import java.util.Map;

public class Mandje {
    private final Map<Product,Integer> mandje = new HashMap <>();

    public Mandje() {
    }

    public Map<Product,Integer> getMandje() {
        return mandje;
    }
```

```
    public void add(Product product, int aantal) {
        if (! mandje.containsKey(product) )
            mandje.put(product, aantal);
        else
            set(product, aantal);
    }

    public void set(Product product, int aantal) {
        int oudeAantal=mandje.get(product);
        mandje.put(product,oudeAantal + aantal);
    }

    public void remove(Product product) {
        mandje.remove(product);
    }

    public void clear() {
        mandje.clear();
    }

    public double getTotalePrijs() {
        double som = 0;
        for (Product p: mandje.keySet() ) {
            som = som + (p.getPrijs() * mandje.get(p) )  ;
        }
        return som;
    }

    @Override
    public String toString() {
        return mandje.toString();
    }
}
```

22.5.4 Het main-programma

```
package be.vdab;

import be.vdab.winkel.Catalogus;
import be.vdab.winkel.Mandje;
import be.vdab.winkel.Product;
import java.util.Map.Entry;

public class WinkelProgramma {
    public static void main(String[] args) {
        Catalogus catalogus = new Catalogus(); //wordt meteen gevuld door
        de constructor
        Mandje mandje = new Mandje ();

        //Catalogus overlopen + mandje programmatorisch vullen
        int i = 1;
        for (Product eenProduct : catalogus.getLijstCatalogus()) {
            if (i%2==0)
                mandje.add(eenProduct,i);
            i++;
        }

        System.out.println("u kocht:");
        for (Entry eenAankoop : mandje.getMandje().entrySet()) {
```

```
            System.out.println( (eenAankoop.getKey()).toString() + ";
            aantal stuks: " + eenAankoop.getValue() );
        }

        System.out.println("\nu kocht voor een totaal van: " +
          mandje.getTotalePrijs() );
    }
}
```

23 Voorbeeldoplossingen taken van hoofdstuk 15

23.1 Gastenboek

23.1.1 De class GastenboekEntry

```
package be.vdab.gastenboek;

import java.io.Serializable;
import java.util.Date;

public class GastenboekEntry implements Serializable {
    public static final long serialVersionUID = 1L;
    private final Date datum = new Date();
    private String schrijver = "";
    private String boodschap = "";

    public GastenboekEntry(String naam, String boodschap) {
        if (naam!=null && !(naam.isEmpty()) ) {
            schrijver = naam;
        }
        if (boodschap!=null && !(boodschap.isEmpty()) ) {
            this.boodschap = boodschap;
        }
    }

    public String getDatum() {
        return datum.toString();
    }

    public String getSchrijver() {
        return schrijver;
    }

    public String getBoodschap() {
        return boodschap;
    }
}
```

23.1.2 De class Gastenboek

```
package be.vdab.gastenboek;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class Gastenboek implements Serializable {
    public static final long serialVersionUID = 1L;
    List<GastenboekEntry> gastenboek = new ArrayList<>();

    public Gastenboek() {
    }

    public void toevoegen(GastenboekEntry gastenboekEntry) {
        gastenboek.add(gastenboekEntry);
    }
    public List<GastenboekEntry> getGastenboek() {
        return gastenboek;
```

```
    }
}
```

23.1.3 De class GastenboekManager

```
package be.vdab.gastenboek;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class GastenboekManager {
  private Gastenboek gastenboek = new Gastenboek() ;

  public GastenboekManager() {
    leesGastenboek(); //initieel reeds het geschreven gastenboek inlezen
  }

  public Gastenboek getGastenboek() {
      return gastenboek;
  }

  private void leesGastenboek() {
    try (FileInputStream fis = new
              FileInputStream("D:/JPF/gastenboek.dat");
         ObjectInputStream ois = new ObjectInputStream(fis);){
         gastenboek = (Gastenboek) ois.readObject();
    }
    catch (IOException e) {
        System.out.println("Kan het gastenboek niet vinden, er wordt
gestart met een nieuw gastenboek!");
        gastenboek = new Gastenboek(); //starten met een leeg gastenboek
    }
    catch (ClassNotFoundException e) {
        System.out.println("Kan de klasse niet vinden : ");
    }
  }

  public void schrijfEntry(GastenboekEntry entry) {
      gastenboek.toevoegen(entry);
  }

  public void schrijfGastenboek() {
      try (FileOutputStream fos = new
                FileOutputStream("D:/JPF/gastenboek.dat");
           ObjectOutputStream oos = new ObjectOutputStream(fos); ){
           oos.writeObject(gastenboek);
      }
      catch (IOException e) {
         System.out.println("Gastenboek kan niet weggeschreven worden!");
      }
  }
}
```

23.1.4 De main-class

```
package be.vdab.gastenboek;

import java.util.List;
```

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        GastenboekManager manager = new GastenboekManager();
        String keuze, naam, boodschap;

        try (Scanner sc = new Scanner(System.in)) {
            sc.useDelimiter("\n"); //om er voor te zorgen dat een naam en
boodschap uit meerdere woorden kan bestaan

            System.out.println("Wat wil je doen? \nTonen, Schrijven of
Eindigen\nGeef T, S of E\n");
            keuze = sc.next();

            while (!keuze.equalsIgnoreCase("E")) {
                switch (keuze) {
                    case "S":
                    case "s":
                        System.out.println("Geef uw naam: ");
                        naam = sc.next();
                        System.out.println("Geef uw boodschap: ");
                        boodschap = sc.next();
                        manager.schrijfEntry(
                                new GastenboekEntry(naam, boodschap));
                        break;

                    case "T":
                    case "t":
                        List<GastenboekEntry> alleEntries =
                            manager.getGastenboek().getGastenboek();
                        System.out.println("Aantal entries: " +
                            alleEntries.size() );

                        for (int i = alleEntries.size()-1; i>=0; i--) {
                          System.out.println(
                           alleEntries.get(i).getDatum() + "\t" +
                           alleEntries.get(i).getSchrijver() + "\t" +
                           alleEntries.get(i).getBoodschap());
                        }
                        break;

                    default:
                        System.out.println("Foute keuze");
                }

                System.out.println("\nWat wil je doen? \nTonen, Schrijven
of Eindigen\nGeef T, S of E\n");
                keuze = sc.next();
            }
        }
        //op het einde het gastenboek volledig schrijven naar de file
        manager.schrijfGastenboek();
    }
}
```

24 Voorbeeldoplossingen taken van hoofdstuk 16

24.1 GemiddeldeRekenaar

24.1.1 De class GemiddeldeRekenaar

```
package be.vdab.jpfhfdst16oef;

public class GemiddeldeRekenaar implements Runnable {
  private final double[] getallen;
  private final int vanafIndex;
  private final int totIndex;
  private double gemiddelde;

  public GemiddeldeRekenaar(double[] getallen, int vanafIndex,
                            int totIndex) {
    this.getallen = getallen;
    this.vanafIndex = vanafIndex;
    this.totIndex = totIndex;
  }

  @Override
  public void run() {
    double totaal = 0;
    for (int i = vanafIndex; i != totIndex; i++) {
      totaal += getallen[i];
    }
    gemiddelde = totaal / (totIndex - vanafIndex);
  }

  public double getGemiddelde() {
    return gemiddelde;
  }
}
```

24.1.2 De class GemiddeldeRekenaarMain

```
package be.vdab.jpfhfdst16oef;

public class GemiddeldeRekenaarMain {
  public static void main(String[] args) {
    double getallen[] = new double[1000000];

    for (int i = 0; i != getallen.length; i++) {
      getallen[i] = Math.random();
    }

    GemiddeldeRekenaar gemiddeldeRekenaar1 =
      new GemiddeldeRekenaar(getallen, 0, getallen.length / 2);

    GemiddeldeRekenaar gemiddeldeRekenaar2 =
      new GemiddeldeRekenaar(getallen, getallen.length / 2,
                             getallen.length);

    Thread thread1 = new Thread(gemiddeldeRekenaar1);
    Thread thread2 = new Thread(gemiddeldeRekenaar2);
    thread1.start();
    thread2.start();
```

```
    try {
      thread1.join();
      thread2.join();
    } catch (InterruptedException ex) {
      System.err.println(ex);
    }

    System.out.println(
      (gemiddeldeRekenaar1.getGemiddelde() +
       gemiddeldeRekenaar2.getGemiddelde()) / 2);
  }
}
```

24.2 Pannenkoek (uitbreiding)

24.2.1 De gewijzigde class Stapel

```
package be.vdab.jpfhfdst16oef;

public class Stapel {
  private int aantalPannenkoeken;

  public void voegPannenkoekToe() {
    System.out.println("Nog een pannenkoek gebakken");
    synchronized (this) {
        ++aantalPannenkoeken;
    }
  }

  public boolean neemPannenkoekWeg() {
    System.out.println("Nog een pannenkoek gegeten hmm...");
    synchronized (this) {
        if (aantalPannenkoeken > 0) {
          --aantalPannenkoeken;
          return true;
        }
        return false;
    }
  }

  public int getAantalPannenkoeken() {
    return aantalPannenkoeken;
  }
}
```

24.2.2 De nieuwe class Klant

```
package be.vdab.jpfhfdst16oef;

public class Klant implements Runnable {
  private final Stapel stapel;

  public Klant(Stapel stapel) {
    this.stapel = stapel;
  }

  @Override
  public void run() {
    int aantalPannenKoekenGegeten = 0;

    while (aantalPannenKoekenGegeten != 50) {
```

```
      if (stapel.neemPannenkoekWeg()) {
        aantalPannenKoekenGegeten++;
      }
      try {
        Thread.sleep(100);
      } catch (InterruptedException ex) {
        System.err.println(ex);
      }
    }
  }
}
```

24.2.3 De gewijzigde main-class

```
package be.vdab.jpfhfdst16oef;

import java.util.ArrayList;
import java.util.List;

public class PannenkoekenMain {
  public static void main(String[] args) {

    Stapel stapel = new Stapel();
    List<Thread> threads = new ArrayList<>();

    for (int i = 0; i != 2; i++) {
      threads.add(new Thread(new Kok(stapel)));
    }
    for (int i = 0; i != 4; i++) {
      threads.add(new Thread(new Klant(stapel)));
    }
    for (Thread thread : threads) {
      thread.start();
    }
    try {
      for (Thread thread : threads) {
        thread.join();
      }
    } catch (InterruptedException ex) {
      System.err.println(ex);
    }
    System.out.println(stapel.getAantalPannenkoeken());
  }
}
```

COLOFON

Domeinexpertisemanager	Jean Smits
Moduleverantwoordelijke	Brigitte Loenders
Auteurs	Hans Desmet - Brigitte Loenders
Versie	02/02/2016
Codes	Peoplesoftcode: Wettelijk depot:

Omschrijving module-inhoud

Abstract	Doelgroep	Opleiding Enterprise Java Ontwikkelaar
	Aanpak	Begeleide zelfstudie
	Doelstelling	De fundamentals van de programmeertaal Java leren.
Trefwoorden		JPF
Bronnen/meer info		