Oefeningen Hoofdstuk 2

Maak in IntelliJ een nieuwe module aan met naam “H2”.

Per oefening maak je in deze module een aparte package. Voor oefening 1 geef je deze als naam “be.pxl.h2.oef1”.

Oefening1

1. Maak een klasse Persoon.
2. Maak als hoofdprogramma een klasse PersoonApp die een main-methode bevat. Hierin maak je een object van de klasse Persoon.
3. Een persoon heeft volgende eigenschappen: voornaam, naam, lengte (in m), gewicht (in kg, voorzie decimalen), geboortejaar. Deze variabelen zijn van overal toegankelijk.
   Ken aan je persoon-object in het hoofdprogramma waarden toe. Druk de naam van je persoon af.
4. Zorg dat de eigenschappen van een Persoon niet meer toegankelijk zijn van buiten de klasse. In de klasse PersoonApp kan je voorgaande instructies die nu problemen opleveren in commentaar zetten.
   Voorzie volgende methoden:
   - set-methoden om alle eigenschappen een waarde te geven.
   - een methode toString() om alle gegevens in de vorm van een String terug te geven.
     De lay-out moet identiek zijn met onderstaande lay-out.

     ```
     Deze persoon is Michael Mystery
     gewicht          : 59,00
     lengte           : 1,85
     geboortejaar     : 1984
     ```

     Let op de uitlijning! Gebruik de Formatter.
   - een methode berekenBmi() om de Body Mass Index van een persoon te berekenen en terug te geven. De BMI wordt als volgt berekend: het gewicht (in kg) gedeeld door het kwadraat van de lengte (in m).

- een methode geefOmschrijving() die een gepaste omschrijving in de vorm van een String teruggeeft:
  - BMI lager dan 18 ➔ ondergewicht
  - BMI vanaf 18 tot 25 ➔ normaal
  - BMI vanaf 25 tot 30 ➔ overgewicht
  - BMI vanaf 30 tot 40 ➔ obesitas
  - BMI vanaf 40 ➔ morbide obesitas
- een methode voegVoornamenToe() om in 1 keer meerdere voornamen aan een persoon te kunnen toevoegen;
  Alle voornamen worden toegevoegd aan de String voornaam met telkens een spatie tussen.
- Test alle methoden uit in de hoofdprogramma.

5. Voeg aan de klasse Persoon de nodige methoden toe om alle eigenschappen van je persoon op te kunnen vragen (getters).
   Voorzie een methode getLeeftijd() om de leeftijd op te kunnen vragen.
   Test deze methoden uit in je hoofdprogramma.
6. Voor de methode setLengte() bouw je een controle in: als er een lengte opgegeven wordt van meer dan 2,40 m, wordt de lengte ingesteld op 2,40. Test uit!
7. Een persoon kan groeien. Voorzie hiervoor 2 methoden groei(): 1 zonder parameters (de persoon groeit dan gewoon 1 cm), en 1 met als parameter het aantal cm dat de persoon groeit. Om rekening te houden met hetgeen in opdracht6 staat, roep je best de methode setLengte() op. Test uit!
8. Voeg een constructor toe met 2 parameters voor naam en voornaam.
   Voeg een constructor toe zonder parameters.
   Voeg een constructor toe die een persoon maakt op basis van een bestaande persoon. Test uit!
9. Zorg ervoor dat je in de default constructor de constructor oproept met de 2 String-parameters. Geef als waarden mee “onbekend”,”onbekend”. Test uit!

Oefening2

1. Maak een klasse Tijdstip met de volgende kenmerken: uren, minuten en seconden (= int). De kenmerken mogen van buiten de klasse niet toegankelijk zijn.
2. Belangrijke opmerking: ten allen tijde moet gezorgd worden dat er geen foutieve waarden in de eigenschappen kunnen opgenomen worden. Indien het argument voor uren > 23 dan wordt dit herrekend (bv. 25 wordt op 1 gezet, 24 wordt op 0 gezet). Indien minuten 125 is dan wordt dit afgehandeld als 2 uur en 5 minuten, 60 minuten worden omgezet naar 1uur en 0 minuten (idem voor seconden). Programmeer deze code slechts 1 keer en roep ze op wanneer nodig!
3. Voorzie de nodige constructors om een tijdstip op volgende manieren te creëren:
   - met 3 parameters (bv. uren : 20, minuten : 50, seconden : 10).
   - met 1 parameter, nl een aantal seconden. Aan de hand van dit getal wordt een tijdstip berekend. Bv. 3672 wordt 1u, 1m en 12 s.
   - met als parameter een ander tijdstip-object: alle waarden worden overgenomen.
4. Voorzie de methoden setUren(), setMinuten(), setSeconden() om nieuwe waarden toe te kennen aan de eigenschappen.
5. Voorzie de methoden getUren(), getMinuten(), getSeconden() om de waarde van de eigenschappen op te vragen.
6. Voorzie de methode voegUrenToe() met als parameter het aantal toe te voegen uren. Doe hetzelfde voor voegMinutenToe() en voegSecondenToe().
   Maak vervolgens een variant op de methode voegUrenToe() waarbij je geen argument meegeeft. Het tijdstip wordt dan met 1 uur opgehoogd. Om zo weinig mogelijk programmatiewerk te hebben roep je vanuit deze methode de methode voegUrenToe() op en je geeft als argument “1” mee.
7. Voorzie een methode toStringTijd() met als parameter een boolean om aan te duiden of het tijdstip volgens de Engelse notatie moet weergeven worden.
   Als de waarde ‘false’ is, geef je het tijdstip als een String terug met de volgende lay-out:

bv. voor de voormiddag:	8:03 u
bv. voor de namiddag:	20:05 u
middernacht wordt afgedrukt als	0:00 u
’s middags wordt afgedrukt als	12:00 u

   Als de waarde ‘true’ is, geef je het tijdstip als een String terug met de volgende lay-out:

bv. voor de voormiddag:	08:30 AM
bv. voor de namiddag:	08:30 PM
middernacht wordt	12:00 AM (midnight)
’s middags wordt	12:00 PM (noon)

8. Voorzie een methode toStringTechnisch() waarbij het tijdstip wordt teruggeven in de vorm van een String 15:05:23, 05:16:03 (je moet zelf voor het juiste aantal nullen in de uitvoer zorgen).
9. Maak een klasse TijdstipApp met een main-methode waarin je een aantal tijdstip-objecten aanmaakt en alle methoden uittest.
   Maak eveneens een array met daarin minstens 4 tijdstip-objecten. Doorloop de array en druk elk tijdstip af.

Oefening3

1. Maak een klasse Bankrekening met de volgende kenmerken: een rekeningnummer (String), de naam van de eigenaar, een saldo en een rentepercentage. De kenmerken mogen van buiten de klasse niet toegankelijk zijn.
2. Een bankrekening-object kan op 2 manieren aangemaakt worden : ofwel worden voor de 4 kenmerken waarden meegegeven ofwel worden defaultwaarden genomen. Als defaultwaarden gelden voor rekeningnummer “geen”, voor naam “onbekend”, voor percentage “1,2” en voor saldo de waarde 0.
   Een bankrekening kan nooit geopend worden met een negatief saldo. Indien dit toch gebeurt moet het saldo op 0 gezet worden.
   Idem voor het rentepercentage.
   In de constructor met de defaultwaarden roep je de andere constructor op en geef je de juiste waarden mee.
3. Voorzie een methode setNaam() om de naam te wijzigen en een methode setRekeningNummer() om het rekeningnummer te wijzigen.
4. Voorzie een methode getSaldo() om het saldo op te vragen.
5. Voorzie een methode stort() om een bedrag te storten op de bankrekening.
6. Voorzie een methode neemOp() om een bedrag van de bankrekening op te nemen. Let op: je mag niet meer opnemen als het bedrag op deze rekening. Indien je dit toch probeert, kan je enkel het saldo opnemen en verschijnt een melding. Bv. het saldo is 50 en je wil 70 euro afhalen: er verschijnt een melding “u mag enkel 50 euro opnemen” en er gaat 50 euro van de rekening. Als het saldo 0 is verschijnt simpelweg “u kan geen geld opnemen”.
7. Voorzie een methode doeVerrichting() om een aantal verrichtingen in 1 keer te kunnen doorvoeren. De positieve bedragen worden op de rekening gestort (roep de methode stort() op), de negatieve bedragen worden van de rekening afgehaald (roep de methode neemOp() op).
8. Voorzie een methode schrijfRenteBij() om de rente te berekenen (= saldo * percentage) en aan het saldo toe te voegen.
9. Belangrijke opmerking: de methodes stort(), neemOp(), doeVerrichting() en schrijfRenteBij() kunnen slechts toegepast worden als je een rekeningnummer hebt en de naam van de eigenaar is ingevuld. Indien er een storting of opname gebeurt op een “geen”-rekeningnummer-object, dan geef je een foutenboodschap “sorry, geen rekeningnummer” en gebeurt er verder niets voor deze rekening. Indien er een storting of opname gebeurt op een rekeningnummer waar de naam van de eigenaar op “onbekend” staan, moet eerst de naam van de eigenaar gevraagd worden via het toetsenbord en ingevuld worden vooraleer de verrichting kan doorgaan. Vermijd het schrijven van dubbele code.

10. Voorzie een methode print() die alle gegevens van een bankrekening als volgt afdrukt:

```
Saldo op spaarrekening 000-01574125-77 op naam van Jef Klak bedraagt 70,00 euro
```

11. Na iedere actie op de bankrekening wordt de toestand getoond als volgt:

```
na opname van 50,00 euro
Saldo op spaarrekening 000-01574125-77 op naam van Jef Klak bedraagt 50,00 euro
na storting van 20,00 euro
Saldo op spaarrekening 000-01574125-77 op naam van Jef Klak bedraagt 70,00 euro
rente wordt bijgeschreven voor 0,60 euro
Saldo op spaarrekening 000-2536987-33 op naam van Jacqueline Klak bedraagt 30,60 euro
```

12. Maak een klasse BankrekeningApp met een main-methode waarin je een bankrekening-object aanmaakt met de default constructor en tracht geld te storten.
    Zorg er vervolgens voor dat je wel verrichtingen kan doen op deze rekening.
    Test alle methoden uit.
    Maak vervolgens een bankrekening-object met je de constructor met 4 parameters.
    Schrijf vervolgens al het geld over van rekening2 naar rekening1.

Oefening4

Maak een klasse Bewerkingen om een aantal wiskundige basisbewerkingen te kunnen uitvoeren.

Maak hierin volgende methoden:

1. trekAf() met 2 parameters van het type double. Het tweede getal wordt afgetrokken van het eerste. Er gebeurt een afdruk zoals volgend voorbeeld: 21,45 – 0,03 = 21,42.
   Van elk getal worden 2 decimalen getoond.
   Opmerking: als de 2de parameter negatief is, moet er een plusteken afgedrukt worden.
2. trekAf() met 2 parameters van het type int. De werking is dezelfde als bij voorgaande methode, behalve dat geen decimalen gedrukt worden.
3. telOp() met 2 parameters van het type double. De 2 getallen worden opgeteld en er gebeurt een afdruk zoals volgend voorbeeld: 2,23 + 1,70 = 3,93.
   Van elk getal worden 2 decimalen getoond.
   Opmerking: als de 2de parameter negatief is, moet er een minteken afgedrukt worden.
4. telOp() met als parameter een array van het type double. Alle getallen worden opgeteld en er gebeurt een afdruk zoals volgend voorbeeld: 2,02 + 3,01 + 0,10 = 5,13
   Van elk getal worden 2 decimalen getoond.
   Opmerking: afhankelijk van het teken van de parameters, moet er voor de bewerking een plusteken of een minteken afgedrukt worden.
5. deel() met 2 parameters van het type int. Het eerste getal wordt gedeeld door het tweede en er gebeurt een afdruk zoals volgende voorbeeld: 7 / 3 = 2,33
   Enkel van de uitkomst worden 2 decimalen getoond.
6. faculteit() met 1 parameter van het type int. Bereken de faculteit en zorg voor een afdruk zoals volgend voorbeeld: de faculteit van 5 is 120
   ter info: de faculteit van 5 is 1 x 2 x 3 x 4 x 5 = 120
   Opmerkingen:
   De faculteit kan niet berekend worden van een negatief getal. Zorg ervoor dat er in de console een foutmelding wordt afgedrukt als het argument negatief is en dat er via het toetsenbord aan de gebruiker gevraagd wordt om een nieuwe waarde in te geven.

Maak vervolgens een klasse BewerkingenApp met een main-methode waarin je een object maakt van de klasse Bewerkingen en alle methoden uittest.
Test ook uit vanaf wanneer je problemen krijgt als je in de methode faculteit alleen gebruik maakt van int / long. Pas de methode faculteit aan zodat de gebruiker een foutmelding krijgt en er via het toetsenbord aan de gebruiker gevraagd wordt om een nieuwe waarde in te geven.

Oefening5

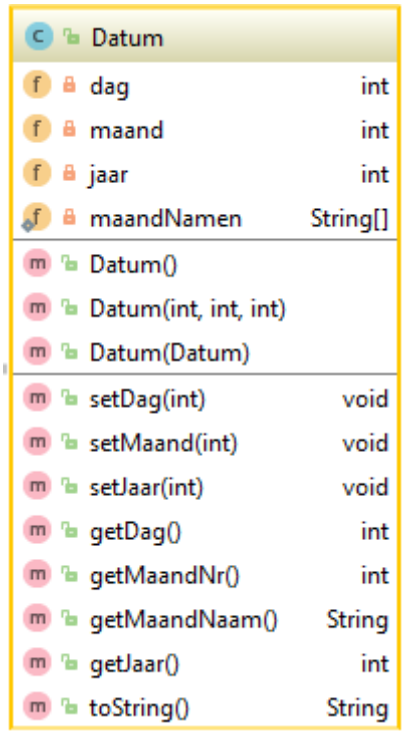
1. Maak een klasse Auto met de volgende kenmerken: merk, model, bouwjaar, kleur maximumSnelheid. Voorzie ook de klasse-variabele MAXIMUM (maximum toegelaten snelheid) die de waarde 180 bevat en zorg ervoor dat de maximumSnelheid van elke wagen nooit dit maximum overschrijdt (als daar op welke manier dan ook een poging toe ondernomen wordt, wordt de maximumSnelheid ingesteld op dit MAXIMUM). De kenmerken mogen van buiten de klasse niet toegankelijk zijn.
2. Voorzie de nodige constructors om een auto op volgende manieren te creëren:
   - met een waarde voor alle velden.
   - zonder parameters: de default-waarden voor een auto-object zijn: VW, Polo, 2018, grijs, 160.
   - met als parameter een ander auto-object: alle waarden worden overgenomen.

   Schrijf zo weinig mogelijk code en roep op een zinvolle manier een constructor op vanuit een andere constructor.
3. Voorzie getters en setters om de waarde te kunnen opvragen en wijzigen.
4. Voorzie een methode print() die alle gegevens van een auto-object afdrukt.
5. Doe het nodige om het aantal auto-objecten te kunnen opvragen.
6. Maak een klasse AutoApp met een main-methode waarin je een aantal auto-objecten aanmaakt en alle methoden uittest.

   Maak eveneens een array met daarin minstens 3 auto-objecten. Doorloop de array en druk de gegevens van elke auto af. Druk ook af hoeveel procent van de auto's niet de hoogst toegelaten snelheid (MAXIMUM) kan behalen. Druk dit percentage afgerond op 1 decimaal af.

Oefening6

1. Maak een klasse Datum volgens onderstaand UML-schema:



   Hou rekening met het volgende:

   - De default-datum is 1/1/2018.
   - De waarde voor 'maand' mag nooit meer zijn dan het aantal maanden in de maandNamen-array en nooit kleiner dan 1. Als er een waarde wordt ingegeven kleiner dan 1, wordt de waarde 1. Als er een waarde wordt ingegeven groter dan 12, wordt de waarde 12.
   - De methode toString() geeft de datum als een String terug als volgt: 5 april 2018.
   - Roep zoveel mogelijk bestaande code op en vermijd om 2 keer hetzelfde te programmeren!

2. Maak eveneens een klasse ScoutsKalenderApp die de scoutsvereniging zal gebruiken om de activiteiten van een bepaalde maand te plannen en af te drukken (zie volgende pagina voor een concreet voorbeeld). Deze applicatie maakt gebruik van de klasse Datum en werkt als volgt:
   a. Na het opstarten wordt éénmalig het maandnr en het jaar ingevoerd van de maand die gepland wordt.
   b. Vervolgens wordt telkens een dagnr ingevoerd en de activiteit voor die dag. De invoer stopt als dagnr '0' wordt ingevoerd.

c. Alle gegevens worden 'onthouden' door het programma (tip: maak telkens een datum-object dat je stockeert in een array. De activiteit wordt eveneens bijgehouden in een array. Je mag veronderstellen dat er nooit meer dan 10 activiteiten gepland worden in 1 maand).

d. Aan het einde van een programma wordt een lijst afgedrukt van alle data met bijhorende activiteit. Zie hieronder voor een voorbeeld van output.

Voorbeeld van de werking:

```
geef een maandnr in
1
geef een jaar in
2018
geef een dag in
9
geef de activiteit in
winterboswandeling
geef een dag in
17
geef de activiteit in
wafelverkoop
geef een dag in
24
geef de activiteit in
viering 25-jarig bestaan
geef een dag in
0
**** Kalender voor januari 2018 ****
9 januari 2018            winterboswandeling
17 januari 2018           wafelverkoop
24 januari 2018           viering 25-jarig bestaan
```

Extra oefeningen Hoofdstuk 2

Gebruik de module met de naam “H2”.

Per oefening maak je een aparte package. Voor oefening 1 geef je deze als naam “be.pxl.h2.exoef1”.

Extraoefening1

Maak een klasse DatumBewerking om een aantal bewerkingen op datums te kunnen uitvoeren. Voor deze oefening kan je eventueel gebruik maken van oefening 6.

Maak hierin volgende methoden:

1. printDatum() met 3 parameters van het type int (dag, maand, jaar). Deze methode drukt een datum af zoals volgend voorbeeld: 4 maart 1992 (neem geen nieuwe lijn).
2. bepaalSchrikkelJaar() met als parameter een jaartal.
   Een jaar is een schrikkeljaar als het deelbaar is door 4; maar als het jaartal deelbaar is door 100 is het geen schrikkeljaar, tenzij het deelbaar is door 400 (Bv.: het jaar 2000 is een schrikkeljaar, want deelbaar door 400; het jaar 1900 is geen schrikkeljaar, want deelbaar door 100 en niet door 400).
   De methode retourneert een boolean met waarde true als het een schrikkeljaar is en false als het geen schrikkeljaar is.
3. berekenDagVanHetJaar() met 3 parameters van het type int (dag, maand, jaar). Deze methode retourneert de hoeveelste dag van het jaar de meegegeven datum is (denk aan schrikkeljaren!!).
4. rangschikData() met 6 parameters van het type int (dag, maand, jaar, dag, maand, jaar). Deze methode beschouwt de eerste 3 parameters als 1 datum en de volgende 3 als een tweede datum. Maak gebruik van voorgaande methoden om het volgende te doen: bepaal de chronologische volgorde van de data en zorg voor een afdruk zoals volgend voorbeeld:
   29 februari 2018 ligt voor 1 maart 2018
   Tip: maak een methode druk() die op haar beurt gebruik maakt van de methode printDatum().

Maak vervolgens een klasse DatumBewerkingApp met een main-methode waarin je een object maakt van de klasse DatumBewerkingen en alle methoden uittest.

Test zeker meerdere datums!

Bv. 4/3/1992 is dag 64, 4/3/1990 is dag 63, 29/2/1992 is dag 60, 4/1/1992 is dag 4

Extraoefening2

Maak een klasse Tekening met volgende methoden:

1. tekenDriehoek() met als parameters 1 int (de grootte van de driehoek) en 1 char. Volgende schermafdruk wordt getekend bij parameters 9 en '§' :

```
        §
       §§
      §§§
     §§§§
    §§§§§
   §§§§§§
  §§§§§§§
 §§§§§§§§
§§§§§§§§§
```

Ter info: de hoogte en breedte van de driehoeken zijn gelijk

2. tekenRechthoekVol() met als parameters de breedte (aantal tekens per lijn) en de hoogte (het aantal lijnen). Maak een output zoals volgend voorbeeld (breedte 6 / hoogte 4):

```
* * * * * *
* * * * * *
* * * * * *
* * * * * *
```

3. tekenRechthoekLeeg() met ongeveer dezelfde werking als de vorige methode, maar met een output zoals in volgend voorbeeld:

```
* * * * * *
*         *
*         *
* * * * * *
```

4. Maak vervolgens een klasse TekeningApp met een main-methode waarin je een object maakt van de klasse Tekening en alle methoden uittest.