Oefeningen Hoofdstuk 3

Maak in IntelliJ een module aan met naam “H3”. Per oefening maak je een aparte package. Voor oefening 1 geef je deze als naam “be.pxl.h3.oef1”.

Oefening1

We maken gebruik van de **klasse Persoon en PersoonApp uit oefening1 van H2**. Kopieer de klasse naar deze package van H3.

1. Vervang de eigenschap geboortejaar in een geboortedatum van type LocalDate. Voorzie getter en setter en pas code aan waar nodig. Pas de methode getLeeftijd() aan zodat die het jaartal van de geboortedatum aftrekt van het huidige jaartal.

2. Pas nu in de klasse Persoon volgende methoden aan:
   - De methode toString(): de naam en de voornaam van de persoon moet in hoofdletters gegeven worden. Maak op een zinvolle manier gebruik van een StringBuilder.
   - De methode berekenBmi() geeft de BMI afgerond op 1 decimaal terug.
   - De methode voegVoornamenToe : maak op een zinvolle manier gebruik van de klasse StringBuilder

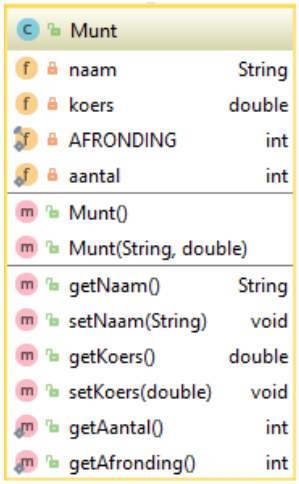
3. Voorzie in de klasse Persoon volgende methoden:
   - Een methode geefNaamAfgekort(). Deze methode geeft de eerste letter van de voornaam in hoofdletter gevolgd door een punt, gevolgd door de achternaam waarbij de 1ste letter in hoofdletter is en de andere letters in kleine letters zijn.
     Bvb. voor Pipi Langkous geeft de methode de string P.Langkous terug
   - Een methode encrypteerNaam() met als parameter een geheel getal. Deze methode geeft de geëncrypteerde **afgekorte** naam terug. Het geheel getal is >=1 en <=20.
     Encryptie verloopt als volgt: bij elk teken wordt het getal bijgeteld.
     Bvb. “Pipi Langkous” wordt dan “Q/Mbohlpvt” als het getal gelijk is aan 1.

4. Test je Persoon klasse met PersoonTest.java die je in de startbestanden kan vinden.

5. Maak in de klasse PersoonApp een persoon aan.
   - Voor het gewicht genereer je een willekeurig geheel getal ≥ 40 en ≤100.
   - Voor de lengte genereer je een willekeurig getal met 2 decimalen >1,57 m en < 2,10 m
   - Druk de afgekorte naam en de geëncrypteerde naam af. De sleutel om de naam te encrypteren is een willekeurig geheel getal >=1 en <=20.

Oefening2

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



   Houd rekening met het volgende:

   - De default-munt is de euro met een koers van 1.
   - De afronding is standaard ingesteld op 3, d.w.z. dat de (nieuwe) koers afgerond wordt op 3 decimalen, alsook getoond wordt met 3 decimalen in de klasse MuntApp. Als het getal van de afronding (in de klasse Munt) aangepast zou worden, moet nergens anders een aanpassing gebeuren, en wordt alles afgerond en getoond rekening houdend met de nieuwe afronding.
   - De methode getNaam() geeft de naam in hoofdletters terug.
   - De methode getAantal() geeft het aantal aangemaakte munten terug.

2. Maak een klasse MuntApp waarin je minstens 4 verschillende munten aanmaakt en maximaal 10 die je allemaal stockeert in een ArrayList. Doorloop nadien de lijst en druk van elke munt de naam en de koers af.
   Vervolgens maak je een overzicht van alle munten met hun koers t.o.v. de eerste munt die in de lijst zit. Zorg ervoor dat de eerste munt in de lijst geen euro is.

Voorbeeld van de output:

```
0,985 BRITSE POND
1,287 DOLLAR
86,950 RUSSICHE ROEBEL
1,000 EURO
Overzicht koersen tov BRITSE POND: 1 BRITSE POND =
1,306 DOLLAR
88,256 RUSSICHE ROEBEL
1,015 EURO
```

Oefening 3

Implementeer de klasse Gondelbaan als volgt:

- Een gondelbaan heeft volgende eigenschappen:
  - naam
  - land
  - hoogteDalstation (in meter)
  - hoogteBergstation (in meter)
  - lengte (in kilometer)
  - snelheid (in meter per seconde (m/s))
  - passagiersPerGondel
  - aantalGondels
- Voorzie alle setters en getters.
  Opmerking: Voor hoogteDalstation en hoogteBergstation voorzie je enkel getters. Je voorziet de methode **setHoogte**() met 2 parameters. De hoogste waarde gebruik je voor hoogteBergstation, de laagste waarde gebruik je voor hoogteDalstation.
  Voorzie een methode **getHoogteVerschil**() die het hoogteverschil tussen bergstation en dalstation teruggeeft.
- Voorzie volgende constructoren:
  - Eén constructor met 4 parameters: naam, land, lengte en snelheid.
  - Eén constructor met 2 parameters: naam en land. De waarde voor lengte wordt in dit geval 2 km en de waarde voor snelheid 6 m/s.
  - Voorzie GEEN default constructor.
  - Vermijd dubbele code!

Enkele belangrijke opmerkingen:

- Voor **land** zijn enkel de volgende waarden geldig: "Frankrijk", "Oostenrijk", "Zwitserland" en "Italië". Land wordt "Onbekend" indien een ongeldige waarde wordt gegeven.
- Voor **snelheid** is een waarde tussen 3 m/s en 8 m/s toegelaten (beide grenzen inclusief). Indien een gegeven waarde te laag is, wordt de minimumwaarde (3 m/s) gebruikt. Indien de waarde te hoog is, wordt de maximumwaarde (8 m/s) gebruikt.
- Bij de **naam** van een gondelbaan wordt ieder woord in de naam (woorden worden gescheiden door spaties) getoond met de eerste letter in hoofdletter en de rest in kleine letters. Bijv. "AiGUille dU MIDi" wordt "Aiguille Du Midi".

- Voorzie de methode **getDuur**() die de tijd geeft in minuten om van dalstation naar bergstation (of andersom) te gaan.

- Voorzie een methode **getVervoersCapaciteit()** die als resultaat het aantal passagiers geeft dat op een uur tijd de afstand van dal- tot bergstation (of andersom) kan overbruggen. Deze methode moet een int teruggeven. Het getal dat je met onderstaande formule bekomt, moet naar onder afgerond worden.
  Formule: vervoerscapaciteit = (60 * T * C) / D
  D = duur = ritduur uitgedrukt in minuten
  C = capaciteit van een gondel (aantal passagiers per gondel)
  T = aantal gondels
- Voorzie de **toString**() methode die de naam van de gondelbaan en het hoogteverschil teruggeeft. Bijvoorbeeld: 3 Vallées Express [1472m]

Test je Gondelbaan klasse met GondelbaanTest.java die je kan vinden in de startbestanden.

Maak tenslotte een klasse GondelbaanApp met een main-methode waarin je zelf een gondelbaan aanmaakt en enkele methoden uitprobeert.

Oefening 4

Maak een programma ScoutsKalenderApp dat door een scoutsvereniging gebruikt kan worden om de activiteiten van een bepaalde maand te plannen en af te drukken.

Om te starten maak je een klasse Activiteit, met als eigenschappen de datum en de naam van de activiteit. Voorzie een constructor met twee parameters en een get-methode voor beide eigenschappen. Daarnaast maak je ook een toString() methode die de gegevens teruggeeft.

De ScoutsKalenderApp werkt als volgt:

a. Na het opstarten wordt éénmalig het maandnummer en het jaar ingevoerd van de maand die gepland wordt.
b. Vervolgens wordt telkens een dagnummer ingevoerd en de activiteit voor die dag. De invoer stopt als dagnummer '0' wordt ingevoerd of als er 10 activiteiten zijn toegevoegd.
c. Alle gegevens worden bijgehouden door het programma. Maak hiervoor gebruik van een ArrayList.
d. Aan het einde van een programma wordt een lijst afgedrukt van alle data met bijhorende activiteit. Gebruik hiervoor een foreach-lus.

Voorbeeld van de werking:

```
Geef het jaar in
2022
Geef maandnummer in
3
Geef een dag in
5
Geef de activiteit in
Boswandeling
Geef een dag in
12
Geef de activiteit in
Wafelverkoop
Geef een dag in
19
Geef de activiteit in
Carnaval
Geef een dag in
26
Geef de activiteit in
Dropping
Geef een dag in
0
*** Kalender voor MARCH 2022 ***
2022-03-05     Boswandeling
2022-03-12     Wafelverkoop
2022-03-19     Carnaval
2022-03-26     Dropping
```

Extra oefeningen Hoofdstuk 3

Gebruik de IntelliJ module met de naam “H3”.

Per oefening maak je een aparte package. Voor oefening 1 geef je deze als naam “be.pxl.h4.exoef1”.

Extraoefening1

Schrijf een programma om $e \cdot \sqrt{x^2 + y^3}$ te berekenen.

x is een geheel getal dat via het toetsenbord wordt ingevoerd.

y is een willekeurig gegenereerd positief reëel getal met 3 decimalen dat ten hoogste 6000 is.

Geef het resultaat op het beeldscherm afgerond tot op 3 cijfers na de komma.

Extraoefening 2

1. Maak een klasse Gsm.
2. Een gsm heeft volgende eigenschappen: gsmNummer (String bv. 0474343434: de voorloopnul moet ook opgenomen worden), bouwjaar, merk, naamEigenaar, voornaamEigenaar, krediet (int), meterBatterij (int) (dit stelt de lading van de batterij voor uitgedrukt in minuten).
3. Het gsmnummer moet bestaan uit 10 cijfers waarvan het eerste een 0 is. Indien hier niet aan voldaan is, wordt aan het gsmnummer de waarde “ongeldig” toegekend. Programmeer deze code 1 keer en roep ze op wanneer nodig!
4. Zorg ervoor dat een GSM-object kan gemaakt worden op volgende 3 manieren.
   -) door het meegeven van het merk en gsmnummer
   -) door het meegeven van het merk, gsmnummer en meterbatterij
   -) door geen enkel argument mee te geven (gsmnummer krijgt de waarde “onbekend”)
5. De gegevens van een GSM-object kunnen afgedrukt worden via de methode toonGSM(). Deze methode geeft de volgende output:

   Het gsmnummer …… van ……..(naam en voornaam eigenaar) heeft de volgende kenmerken :
   Bouwjaar : ….
   Merk : …..
   Krediet : …..
   Meterbatterij : ….

   Voorzie in de afdruk dat voor een object aangemaakt via de default constructor (en waarvan de velden nog niet gewijzigd zijn) een eigen boodschap komt :

   Dit object is een leeg object en heeft geen eigen waarden voor zijn eigenschappen.

Opmerking: bij het afdrukken van de naam en voornaam van de eigenaar wordt de 1ste letter van de naam in hoofdletter weergegeven en de rest in kleine letters, analoog voor de voornaam.

6. Voorzie voor alle eigenschappen methoden om nieuwe waarden toe te kennen aan de eigenschappen en de waarden van de eigenschappen op te vragen.
   Opmerking: wanneer het gsmnummer van een gsm-object ongeldig of onbekend is, moet eerst het gsmnummer gewijzigd worden. Zorg ervoor dat je een mededeling afdrukt, zodra de gebruiker eerst een ander kenmerk van dit gsm-object probeert te wijzigen.
7. Maak volgende methodes
   - toonBeltijd() → de beltijd = krediet x 60 (seconden)
   - tik() zonder parameters. Hierdoor neemt de meterbatterij met 2 eenheden af. Zorg ervoor dat je batterij niet negatief kan gaan.
   - laadOp() zonder parameters. Hierdoor wordt de meter van de batterij onmiddellijk op 200 gezet.
8. Maak een methode getAantalGsm() welke je het aantal tot nu gemaakte GSM-objecten teruggeeft.
9. Maak als hoofdprogramma een klasse GsmApp die een main bevat.
   - Maak een array met 3 gsm-objecten aan. Test bovenstaande methoden uit.
   - Doorloop deze array en druk de gegevens van elk gsm-object in de array af.
   - Druk het aantal gsm-objecten af. Maak hierbij gebruik van de methode getAantalGsm().