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 - Persoon

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 alleen toegankelijk in de klasse.
4. 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!
5. Zorg ervoor dat je in de default constructor de constructor oproept met de 2 String-parameters. Geef als waarden mee “onbekend”,”onbekend”. Test uit!
6. 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!
   - 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 het hoofdprogramma.

7. 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.
8. 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. Maak hierbij gebruik van een constante MAX_LENGTE. Test uit!
9. 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!

Oefening 2 - Bankrekening

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. In de constructor met de defaultwaarden roep je de andere constructor op en geef je de juiste waarden mee.
   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.
   Probeer zoals altijd dubbele code te vermijden.
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(double bedrag)* om een bedrag te storten op de bankrekening.
6. Voorzie een methode *neemOp(double bedrag)* 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). Geef de lijst van verrichtingen door aan de methode d.m.v. een array.
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. Voorzie een methode *valideerRekening()* die een boolean waarde retourneert op basis van de geldigheid (enkel *true* indien rekeningnummer en naam ingevuld zijn). Pas alle methoden met verrichtingen aan zodat er geen acties uitgevoerd kunnen worden op een ongeldige rekening en er in dat geval een melding getoond wordt.
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:

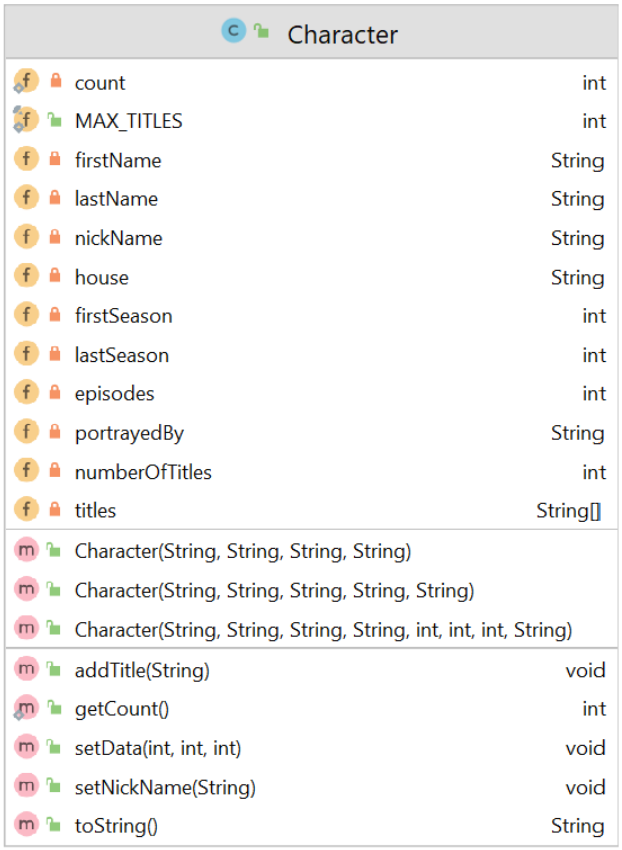
```
Bankrekening geopend met saldo 100,00 euro
Saldo op spaarrekening 000-01574125-77 op naam van Jef Klak bedraagt 100,00 euro
> Na opname van 50,00 euro
Saldo op spaarrekening 000-01574125-77 op naam van Jef Klak bedraagt 50,00 euro
> Na storting van 20,00 euro
Saldo op spaarrekening 000-01574125-77 op naam van Jef Klak bedraagt 70,00 euro
> Rente wordt bijgeschreven voor 0,84 euro
Saldo op spaarrekening 000-01574125-77 op naam van Jef Klak bedraagt 70,84 euro
```

12. Maak een klasse BankrekeningApp met een main-methode waarin je een bankrekening-object aanmaakt met beide constructors.
    Probeer alle verschillende verrichtingen uit en ga na of alle gegevens kloppen.
    Controleer of verrichtingen op ongeldige rekeningnummers correct afgehandeld worden.
    Schrijf vervolgens al het geld over van rekening 1 naar rekening 2.

Oefening 3 - Game of Thrones

Je maakt in deze oefening een programma om karakters of personages uit de populaire serie *Game of Thrones* op te slaan.

Voorzie eigenschappen om voornaam, achternaam, bijnaam en het huis (familie) van het personage bij te houden (Strings). Daarnaast hou je bij in welk seizoen het karakter voor het eerst en voor het laatst te zien is en in hoeveel episodes het personage in totaal voor komt. De naam van de acteur of actrice die het personage speelt hou je ook in een String bij en je voorziet een array van Strings om de extra titels van het karakter in op te slaan.



De grootte van deze array wordt bepaald door de constante klassevariabele *MAX_TITLES*. Standaard krijgt deze de waarde 3.

Hou bij hoeveel objecten van de *Character* klasse er aangemaakt worden door een klasse-eigenschap *count* (int) te gebruiken.

Deze waarde (count) moet ook opgevraagd kunnen worden aan de hand van de klassemethode *getCount()*.

Een *Character* object kan op 3 manieren aangemaakt worden:

- *firstName, lastName, house, portrayedBy*
- *firstName, lastName, nickName, house, portrayedBy*
- *firstName, lastName, nickName, house, firstSeason, lastSeason, episodes, portrayedBy*

Met de methode setData() kunnen waarden voor *firstSeason, lastSeason* en *episodes* in één keer ingesteld worden.

De methode *addTitle()* voegt de meegegeven *title* (argument) toe aan de array met titels, maar enkel indien deze nog niet volledig gevuld is. Gebruik hiervoor de eigenschap *numberOfTitles*.

De *toString()* methode toont de gegevens van een karakter als volgt:

```
Tyrion "The Imp" Lannister of house Lannister
*** "Hand of the King"
*** "Warden of the West"
*** "Lord of Caterly Rock"
Played by: Peter Dinklage in season 1-8 (67 episodes)
```

Zorg ervoor dat de *toString()* enkel de gegevens opneemt die een waarde hebben in het object.

Voer tenslotte de main methode in *GameOfThrones.java* uit.

De output zou er als volgt moeten uitzien:

```
Petyr "Littlefinger" Baelish of house Baelish
Played by: Aidan Gillen


Tyrion "The Imp" Lannister of house Lannister
*** "Hand of the King"
*** "Warden of the West"
*** "Lord of Caterly Rock"
Played by: Peter Dinklage in season 1-8 (67 episodes)


Catelyn Stark of house Tully
Played by: Michelle Fairley in season 1-3 (26 episodes)


Samwell "Sam" Tarly of house Tarly
*** "Grand Maester"
Played by: John Bradley in season 1-8 (48 episodes)


Margaery Tyrell of house Tyrell
*** "Queen Consort"
Played by: Nathalie Dormer in season 2-6 (26 episodes)


5 characters registered
```

Oefening 4 - Input helper klasse

Om de kracht van klassemethoden aan te tonen bouw je in deze oefening een eenvoudige klasse met de naam *Input*. Deze zal je helpen om input aan een gebruiker te vragen op een snelle manier.

Zoals je in de vorige oefeningen al gemerkt hebt, zijn er heel wat kleine details waar je op moet letten wanneer je een Scanner gaat gebruiken om de gebruiker om input te vragen. Daarom zou het handig zijn als we deze functionaliteit één keer kunnen schrijven en deze daarna makkelijk kunnen herbruiken.

Voeg een klassevariabele scanner toe van type Scanner. Maak een klassemethode *getScanner()* die het scanner object initialiseert indien dat nog niet gebeurd is (*null*) en daarna het object teruggeeft.

Maak daarna in de klasse *Input* een klassemethode *readLine()* die het scanner object opvraagt met de *getScanner()* methode, input in de vorm van een String opvraagt en daarna de input retourneert. De Scanner moet **niet** gesloten worden maar blijft bestaan in de klasse.

Als je dit correct hebt gedaan, kan je een klasse InputApp maken met een main methode en daarin de methode als volgt gebruiken:

```
public static void main(String[] args) {
        String naam = Input.readLine();
}
```

Voeg een extra methode *readLine(String msg)* toe waaraan je een bericht kan meegeven dat getoond wordt voor dat de gebruiker input moet geven. Herbruik steeds code waar mogelijk.

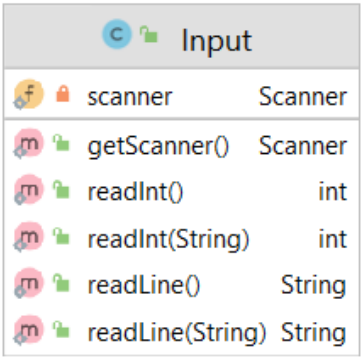
Voeg daarna op analoge wijze de klassemethoden readInt() en readInt(String msg) toe. Denk er aan om steeds de *getScanner()* methode te gebruiken wanneer je het scanner object nodig hebt. De laatste twee methoden kan je dan normaal gezien op onderstaande wijze gebruiken:

```
public static void main(String[] args) {
        String naam = Input.readLine("Geef je naam in.");
        int leeftijd = Input.readInt("Geef je leeftijd in.");
        System.out.println("Welkom, " + naam);
        System.out.println("Wauw, al " + leeftijd + "?");

}
```

Output:

```
Geef je naam in.
Sam
Geef je leeftijd in.
34
Welkom, Sam
Wauw, al 34?
```



In de verdere oefeningen van dit vak kan je nu steeds teruggrijpen naar deze klasse als je input van de gebruiker nodig hebt. Doordat je geen instanties moet maken, is deze klasse zeer makkelijk in gebruik.

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”.

Extra oefening 1 - Tijdstip

1. Maak een klasse Tijdstip met het volgende kenmerk: seconden (= int).
   Bv. 83705 stelt het tijdstip 23:15:05 voor (23 * 3600 + 15 * 60 + 5 = 83705)
   3850 stelt het tijdstip 01:04:10 voor (1 * 3600 + 4 * 60 + 10 = 3850)
   Dit kenmerk mag van buiten de klasse niet toegankelijk zijn.
2. Belangrijke opmerking: ten allen tijde moet gezorgd worden dat er geen foutieve waarde in de eigenschap voorkomt. Tijdstippen lopen van 00:00:00 (0 sec) tot 23:59:59 (86399 sec). Indien bvb. een waarde wordt gegeven 91326 sec (25:22:06) dan moet dit herleid worden tot 4926 sec (01:22:06).
   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
   - met als parameter een ander tijdstip-object: alle waarden worden overgenomen.
4. Voorzie de methoden bepaalUren(), bepaalMinuten(), bepaalSeconden() om de waarden van het tijdstip op te vragen. Bvb. 4926 sec stelt het tijdstip 01:22:06 voor. bepaalUren() geeft 1 terug, bepaa/lMinuten() geeft 22 terug en bepaalSeconden() geeft 6 terug.
5. Voorzie de methoden wijzigUren(), wijzigMinuten(), wijzigSeconden() om nieuwe waarden toe te kennen aan het tijdstip. Bv. tijdstip 20:02:13 als je hierop wijzigUren(5) toepast, is dit het tijdstip 05:02:13.
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).

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.