Java Advanced – Week 4: Lambda expressies

GitHub: https://github.com/PXLJavaAdvanced2017/week4-lambdaexpressions

Opdracht 1

De klasse NumberMachine bevat een lijst van integers die verwerkt moeten worden. De klasse heeft ook een functie *processNumbers*, die de lijst van integers zal gaan inspecteren en afhankelijk hiervan een selectie van getallen zal samenzetten in een String waarde, die daarna als return waarde wordt gegeven. De manier waarop de integers verwerkt worden, wordt bepaald door een object dat de interface NumberFilter implementeert.

Pas de functie wat aan zodat de getallen in de String gescheiden worden door een streepje, bv. 1-2-3. (**Let op**: geen streepje voor of achteraan!)

Schrijf de functionele interface NumberFilter met als enige functie ***boolean check(int number)***.

Maak een klasse ***NumberSelector*** met als member variabele een NumberMachine object. Dit object wordt aan de constructor meegegeven en daar ook in de member variabele opgeslagen. Definieer in deze klasse de functie ***showEvenNumbers().*** Zorg dat deze functie de functie *processNumbers* van het NumberMachine object oproept en de NumberFilter op die manier implementeert zodat enkel even getallen afgeprint zullen worden. Doe dit door gebruik te maken van een anonieme geneste klasse.

Test nadien je code met de EvenNumbersTest.

Maak nu een tweede functie: ***showNumbersAbove()***, eveneens in de NumberSelector klasse. Deze functie krijgt een integer als parameter mee en zal opnieuw de *processNumbers* functie gebruiken, maar deze keer om enkel getallen uit te printen die boven het meegegeven getal liggen. Definieer de instantie van NumberFilter met behulp van een **lambda expressie**.

Test nadien je code met de NumbersAboveTest.

Maak tenslotte een derde methode ***printHexNumbers()***. Deze functie moet de integers printen in hexadecimale notatie, die steeds bestaat uit 4 karakters. (bv. 46 => 2E) Hier kan je geen gebruik maken maken van de NumberFilter, omdat deze een boolean moet returnen. Maak voor deze oefening een functie ***convertNumbers*** in NumberMachine, die een standaard functionele interface gebruikt die geschikt is voor deze functie. De functie geeft de hexadecimale waarden in een String als returnwaarde terug. De waarden zijn opnieuw gescheiden door een streepje -. Gebruik tenslotte de functie *convertNumbers* in combinatie met een lambda expressie om het gewenste resultaat te bekomen.

Test je code opnieuw met de HexNotationTest.

Opdracht 2

Cryptografie is het versleutelen van geheime boodschappen en gaat terug tot het oude Egypte. De geleerden en wetenschappers die zich er mee bezig houden, worden ook cryptografen genoemd. We hebben in het skelet voor deze oefening een functionele interface *Encryptie* voorzien, die als enige taak heeft om een bericht te versleutelen.

Maak een klasse **Bericht** met de member variabele *tekst*. Voorzie een setter functie voor deze variabele en zorg dat de variabele *tekst* ook gezet kan worden via de constructor. Maak tevens een functie **encrypt** in deze klasse, die een implementatie van de Encryptie interface als argument mee krijgt en daarmee de variabele *tekst* gaat versleutelen. De functie moet het resultaat teruggeven als return waarde.

Maak een klasse **Cryptograaf**. Maak hierin een static methode **encryptFirst** aan, die als argument de String meekrijgt die versleuteld moet worden en als return value het resultaat geeft. Zorg dat deze functie een *Bericht* aanmaakt en daarop de **encrypt** methode aanroept. Geef hierin een implementatie van Encryptie door, door middel van een **anonieme geneste klasse**.
Zorg dat de encryptie het bericht versleuteld door de gehele inhoud van de String om te draaien.

Voorbeeld: “Sam0!” wordt “!0maS”

Test het resultaat met de FirstEncryptionTest.

Verander je implementatie van de *encryptFirst()* methode, zodat deze nu gebruik maakt van een lambda expressie, in plaats van een anonieme geneste klasse. Vereenvoudig zo veel mogelijk.

Test het resultaat opnieuw met de FirstEncryptionTest, zodat je zeker bent dat het nog steeds werkt.

Creëer in *Cryptograaf* nu de static methode **encryptSimple**, die op dezelfde manier een Bericht zal aanmaken en versleutelen. Geef deze keer opnieuw een implementatie van Encryptie door middel van een **anonieme geneste klasse**.
Zorg dat de encryptie het bericht versleuteld door:

1. Het hele bericht in hoofdletters om te zetten
2. De karakterwaarde van elke letter naar een cijfer om te zetten **(zie verder > ‘Tip’)**
3. Alle andere tekens (cijfers, leestekens, speciale karakters, …) blijven gewoon staan
4. Het geheel aan elkaar te plakken door alle elementen te scheiden met een – teken.

Voorbeeld: “Sam1!” wordt “28-10-22-1!”

Tip: Je zal hiervoor de String moeten omzetten in een *char* array. Een *char* is een type dat één karakter voorstelt in Unicode. Zoek zelf naar een manier (bv. op Internet) om de integer waarde van zo’n *char* type vast te krijgen en te checken of de waarde een letter voorstelt of niet.

Controleer het resultaat me de SimpleEncryptionTest.

Maak een derde static methode in *Cryptograaf* aan, genaamd encryptLambda. Deze volgt het zelfde principe, maar deze keer geef je een implementatie van Encryptie door middel van een **lambda expressie**.
Zorg dat deze encryptie het bericht versleuteld door:

1. Het hele bericht om te zetten naar kleine letters
2. Elke letter in de String te 'verhogen' (= volgende letter in het alfabet) met de totale lengte van de originele string. (zie verder > **'Tip'**) Als dit algoritme verder dan de letter 'z' gaat, ga dan verder met 'a', 'b', …
3. Alle andere tekens (cijfers, leestekens, speciale karakters, …) blijven gewoon staan

Voorbeeld: "Sam2!" wordt "xfr2!" (telkens 3 letters verder)

Tip: Gebruik hiervoor opnieuw de omzetting van String naar *char* array. Met de *char* values kan je bewerkingen doen, waarna je de *char* array weer kan omzetten naar een String om deze terug te geven.

Test of je encryptie algoritme werkt door gebruik te maken van de LambdaEncryptionTest.

Opdracht 3

Maak een klasse **VideoGame**, met als member variabelen: *name* (String)*, price* (double), *rating* (double) en *genres* (ArrayList van Strings). De constructor moet deze variabelen ontvangen. Let wel op: de genres moeten aan de constructor meegegeven worden als een String array, en in de constructor omgezet worden naar een ArrayList. Voorzie verder get- en set-functies, waarbij de set functie van genres opnieuw een array verwacht. Zorg er ook voor dat de genre-namen in lowercase opgeslagen worden.

Creëer nu een **GameCollection** klasse, met een ArrayList van VideoGames als member variabelen. Voorzie een *addGame* functie die een VideoGame object als argument krijgt en dit object gaat toevoegen aan de collectie.

Voeg nu een functie ***selectGames*** toe aan de GameCollection klasse. Deze functie krijgt als argument een object dat één of andere filter implementeert. Zoek zelf in je cursus of op internet welke standaard functionele interface je hier best voor kan gebruiken. De functie *selectGames* zal door de lijst met games in de collectie lopen en alle VideoGames die voldoen aan de eisen van de meegegeven filter, toevoegen aan een nieuwe ArrayList van VideoGames. Deze ArrayList is het resultaat en wordt teruggegeven als return waarde door deze functie.

Maak tenslotte een klasse **GameBrowser**. Deze klasse heeft als member variabele een GameCollection object. Deze variabele wordt gezet in de constructor (meegegeven als argument).

Maak een functie ***showGamesForSearch*** in GameBrowser. Deze functie krijgt een String *search* als parameter en gaat daarmee de games uit de collectie filteren die deze String in hun naam bevatten. Dit doe je door een gepaste implementatie van het filter object te voorzien en deze mee te geven aan de *selectGames* functie van GameCollection. Doe dit in deze functie met behulp van een **anonieme geneste klasse**.
Geef nadien de resulterende ArrayList van VideoGames terug als return waarde.

Je kan de functie controleren met de GamesSearchTest.

Schrijf een functie ***showFreeGames*** in deze klasse. Roep daarvoor opnieuw de functie *selectGames* uit GameCollection aan en voorzie een gepaste implementatie van de filter, maar deze keer aan de hand van een **lambda expressie**. Geef nadien de resulterende ArrayList van VideoGames terug als return waarde.

Controleren kan je met de FreeGamesTest.

Voeg nadien een functie toe met de naam ***showGamesInGenre***. Deze heeft een String *genre* als parameter en gaat daarmee alle games in de collectie selecteren die dat genre bevatten. Verder werkt de functie analoog aan de functie *showFreeGames*. (werk opnieuw met een lambda expressie)

Check of alles naar behoren werkt met de GamesGenreTest.