Oorerwing / Inheritance

Vir die vorige weke se prakties moes jy klasse skryf wat inligting oor beide spaarrekenings en kredietrekenings bestuur.
Hierdie twee klasse het 'n aantal voorkomsveranderlikes en metodes in gemeen. Verder is daar ook metodes in die twee klasse met dieselfde naam (en parameters) maar waarvan die implementerings verskil. Daar is van oorerwing van een of meer superklasse gebruik gemaak om soveel moonlik kode te hergebruik.

For the practicals of the previous weeks you had to write classes to manage information of both savings accounts as well as credit accounts.
These two classes have a number of instance variables and methods in common. Furthermore, the two classes also have methods with the same name (and parameters) but which have different implementations. Inheritence of one or more super classes was used to ensure the re-use of as much code as possible.

Geldigheidstoetsing / Validity checking

'n Rekeningnommer moet presies 8 syfers lank wees. Geldigheidstoetsing word gedoen deur van Luhn se algoritme gebruik te maak. Die syfers van die rekeningnommer word vermenigvuldig met gewigte wat begin by 8 vir die eerste (mees beduidende) syfer, 7 vir die tweede syfer tot by 1 vir die laaste syfer. Nadat al die syfers met die gewigte vermenigvuldig is word hulle bymekaar getel en deur gedeel. Indien die res gelyk is aan 0, is die rekeningnommer geldig. 'n Nie-nul res beteken dat die rekeningnommer ongeldig is.
Die geldigheidstoetsing vir die geldige rekeningnommer 53456785 lyk dus so:
Stap 1: Die geweegde som van die syfers word bereken:

An account number must have exactly 8 digits. Validity testing is done using Luhn's algorithm. The digits of the account number are multiplied with weights, starting with a weight of 8 for the first (most significant) digit, 7 for the second digit and so on with a weight of 1 for the last digit. After the digits have been multiplied with the weights they are added together and divided by 11. If the remainder of the division is equal to 0 the account number is valid. A non-zero remainder means that the account number is invalid.
The validity test for the valid account number 53456785 is done as follows:
Step 1: The weighted sum of the digits is calculated:

$$\begin{aligned} s &= (5*8)+(3*7)+(4*6)+(5*5)+(6*4)+(7*3)+(8*2)+(5*1) \\ &= 40+21+24+25+24+21+16+5 \\ &= 176 \end{aligned}$$

Stap 2: Die modulus van die geweegde som met 11 word bereken:

Step 2: The modulus of the weighted sum with 11 is calculated:

$$176\%11 = 0$$

Aangesien die modulus gelyk is aan 0 volg dit dat die rekeningnommer 53456785 geldig is.

As the modulus is equal to 0 it follows that the account number 53456785 is valid.

Vir die ongeldige rekeningnommer 53456763 lyk die berekening soos volg:
Stap 1:

The calculations for the invalid account number 53456763 is done as follows:
Step 1:

$$\begin{aligned} s &= (5*8)+(3*7)+(4*6)+(5*5)+(6*4)+(7*3)+(6*2)+(3*1) \\ &= 40+21+24+25+24+21+12+3 \\ &= 170 \end{aligned}$$

Stap 2: Die modulus van die geweegde som met 11 word bereken:

Step 2: The modulus of the weighted sum with 11 is calculated:

$$170\%11 = 5$$

Aangesien die modulus ongelyk is aan 0 volg dit dat die rekeningnommer 53456763 ongeldig is.

As the modulus is not equal to 0 it follows that the account number 53456763 is invalid.

Opdrag / Assignment

- Skryf die "*unchecked*" uitsondering klas **InvalidAccountNumberException**.
- Die stel metode vir die rekeningnommer voorkoms veranderlike moet toets of die rekeningnommer wat ontvang is voldoen aan die vereistes hierbo genoem. Indien die rekeningnommer ongeldig is moet die stel metode 'n **InvalidAccountNumberException** gooi.
- Skryf die program **TestExceptions.java** wat twee objekte elk met geldige en ongeldige rekeningnommers skep. Elke objek moet in 'n aparte *try-catch* blok geskep word en in elke geval moet die uitsondering wat gegooi is, gevang en hanteer word deur 'n foutboodskap te vertoon.

- Write the unchecked exception class **InvalidAccountNumberException**.
- The set method for the account number instance variable must test the validity of the account number that it receives against the criteria given above. If the account number is invalid the set method must throw an **InvalidAccountNumberException**.
- Write the program **TestExceptions.java** that creates two objects each with valid and invalid account numbers. Each object must be created in a separate try-catch block and in each of the cases the exception that is thrown must be caught and handled by displaying an error message.