# Opdracht: Bankrekeningen beheren

## Beschrijving:

We gaan een klein programma schrijven om bankrekeningen te beheren. Je moet in staat zijn om nieuwe rekeningen aan te maken, geld te storten en op te nemen, en het saldo van een rekening te controleren.

Instructies:

Maak een klasse genaamd BankAccount met de volgende attributen:

* accountNumber (een String die het rekeningnummer vertegenwoordigt)
* balance (een double die het saldo van de rekening voorstelt)

Voeg de volgende methoden toe aan de BankAccount-klasse:

* getAccountNumber: geeft het rekeningnummer terug. //printstatement met rekeningnummer.
* ~~getBalance: geeft het saldo van de rekening terug. //printstatement met hoeveelheid.~~
* ~~deposit: neemt een bedrag als parameter en stort dat op de rekening. //geef een bedrag in en dan +=deposit.~~
* ~~withdraw: neemt een bedrag als parameter en neemt dat bedrag op van de rekening. Zorg ervoor dat je controleert of er voldoende saldo is voordat je een opname uitvoert. // geef een bedrag en dan -=withdraw. En een if-statement zodat hij niet onder de 0 euro komt.~~
* ~~toString: geeft een representatie van het bankrekeningobject als een string terug. 🡪 moet ik nog uitzoeken. Misschien een dictionary gebruiken.~~

Maak een Bank-klasse waarin je een lijst van BankAccount-objecten kunt bijhouden. Voeg de volgende methoden toe aan de Bank-klasse:

* ~~createAccount: neemt een rekeningnummer en een beginsaldo als parameters en maakt een nieuw BankAccount-object aan.~~ Voeg dit object toe aan de lijst van rekeningen. //aanmaken van een rekeningnummer aan de hand van rekeningnummer string en beginsaldo int. 🡪 lijst moet ik nog even uitzoeken. Eerst maar 2 account aanmaken en daarna kijken voor de lijst. Ik dnek met objecten in een lijst dan.
* ~~getAccount: neemt een rekeningnummer als parameter en geeft het bijbehorende BankAccount-object terug. // nummer string en balance printen. Dus string en double.~~
* ~~deposit: neemt een rekeningnummer en een bedrag als parameters en stort dat bedrag op de bijbehorende rekening.~~
* ~~withdraw: neemt een rekeningnummer en een bedrag als parameters en neemt dat bedrag op van de bijbehorende rekening. Zorg ervoor dat je controleert of er voldoende saldo is voordat je een opname uitvoert.~~

Schrijf een Main-klasse om je programma te testen. Maak een Bank-object aan en voer een aantal operaties uit, zoals het aanmaken van rekeningen, het storten en opnemen van geld en het controleren van saldi.

Hieronder is de main klasse, deze mag je niet aanpassen (tenzij je extra features wilt testen 😉)

public class Main {

public static void main(String[] args) {

Bank bank = new Bank();

// Nieuwe rekeningen aanmaken

bank.createAccount("NL01ABCD1234567890", 1000.0);

bank.createAccount("NL02EFGH1234567890", 500.0);

// Geld storten

bank.deposit("NL01ABCD1234567890", 250.0);

// Geld opnemen

bank.withdraw("NL02EFGH1234567890", 200.0);

// Saldo controleren

System.out.println(bank.getAccount("NL01ABCD1234567890").getBalance());

}

}

Vergeet niet de error-handeling, je mag bijvoorbeeld geen negatief geld overmaken, of in de min komen te staan na een transactie!

Als je hiermee klaar bent denk dan aan extra functies, bijvoorbeeld:

* Overboekingen tussen rekeningen: Breid de functionaliteit uit zodat gebruikers geld kunnen overboeken tussen verschillende bankrekeningen. Voeg een methode toe aan de Bank-klasse om een overboeking uit te voeren, waarbij je het bedrag en de bron- en doelrekeningen specificeert.
* Valutawissel: Implementeer een functie waarmee je valutawissel kunt simuleren. Je kunt een methode toevoegen aan de Bank-klasse waarmee je het saldo van een rekening in een andere valuta kunt weergeven. Je kunt bijvoorbeeld een wisselkoersattribuut gebruiken om de conversie uit te voeren.
* Transactiegeschiedenis: Implementeer een functionaliteit waarmee je de transactiegeschiedenis van een bankrekening kunt bijhouden. Je kunt bijvoorbeeld een lijst of een logboek van transacties maken waarin elke storting of opname wordt geregistreerd. Voeg vervolgens een methode toe aan de BankAccount-klasse om de transactiegeschiedenis op te vragen en te tonen.
* Wees creatief, en bedenk zelf handige functies die je mist