# Code Test: Banking

## About This Test

The purpose of this code test is to show us your skills in:

- Object Oriented Programming
- TDD
- Clean code
- Git
- Problem solving

Please keep these aspects in mind as you develop your solution. Also, your chosen implementation doesn't necessarily have to be the best you can think of, but one that you can implement in the allocated time.

## Instructions

- Create a project which models the situation below.
- The language used should be **Ruby** (unless previously agreed otherwise).
- All code should be tested following **TDD**.
- A **README** file must be attached explaining your implementation.
- Create a .zip with your **code** and **git** directory.
- Answer our email with the **zip** of the project.

## Background

The software you write in this test will be used for banks. Banks have accounts. Accounts hold money. Transfers can be made between accounts. Banks store the history of transfers.

There can be two types of transfers:

- **Intra-bank** transfers, between accounts of the same bank. They don't have commissions, they don't have limits and they always succeed.
- **Inter-bank** transfers, between accounts of different banks. They have 5€ commissions, they have a limit of 1000€ per transfer. And they have a 30% chance of failure.

## Part 1

Create the models and their relationships to accurately reflect banks, accounts and transfers. Make sure that new types of accounts and transfers can be added to the bank with minimal effort.

## Part 2

After modeling the domain(part 1), create a file called **show_me_the_money.rb*** in charge of reproducing the next situation:

*Jim has an account on the bank A and Emma has an account on the bank B. Jim owes Emma 20000€. Emma is already a bit angry, because she did not get the money although Jim told her that he already sent it.*

*Help Jim send his money by developing a **transfer agent**. This agent assures that everybody gets their money. When the agent receives an order to transfer money from account A to account B, he issues transfers considering commissions, transfer limits and possibility of transfer failures.*

The execution of the script will print the balance of every account before and after the transfers and the history of the transfer of every bank.

## Questions:

Please also supply your answer to the following questions:

- How would you improve your solution?
- How would you adapt your solution if transfers are not instantaneous?