

Exercise 1: Abstract Data Types

Issued: **April 12, 2007**. Return for corrections: **April 26, 2007**.

We have the following requirements for the implementation of a `BANK_ACCOUNT` class:

1. Every `BANK_ACCOUNT` has an *owner* and a *balance*.
2. The balance is recorded in “Rappen” (as an `INTEGER`).
3. The owner is recorded with his/her name (as a `STRING`).
4. It should always be possible to retrieve the balance and owner for any given `BANK_ACCOUNT`.
5. It is possible to *deposit* money to and *withdraw* money from the `BANK_ACCOUNT`.
6. The balance on the `BANK_ACCOUNT` is adjusted accordingly.
7. The balance of any `BANK_ACCOUNT` should never become negative.

Here is a first version of an Abstract Data Type (with the abstract data types for `INTEGER` and `STRING` given with the standard operations) that tries to implement the requirements:

TYPES

`BANK_ACCOUNT`

FUNCTIONS

`new_account`: `STRING` \rightarrow `BANK_ACCOUNT`

`owner`: `BANK_ACCOUNT` \rightarrow `STRING`

`balance`: `BANK_ACCOUNT` \rightarrow `INTEGER`

`deposit`: `BANK_ACCOUNT` \times `INTEGER` \rightarrow `BANK_ACCOUNT`

`withdraw`: `BANK_ACCOUNT` \times `INTEGER` \rightarrow `BANK_ACCOUNT`

PRECONDITIONS (with $v \in \text{INTEGER}$, $a \in \text{BANK_ACCOUNT}$)

`withdraw` (a , v) **require** `balance` (a) $\geq v$ **and** $v \geq 0$

`deposit` (a , v) **require** $v \geq 0$

AXIOMS (with $o \in \text{STRING}$, $v \in \text{INTEGER}$, $a \in \text{BANK_ACCOUNT}$)

`balance` (`new_account` (o)) = 0

`owner` (`new_account` (o)) = o

`balance` (`deposit` (a , v)) = `balance` (a) + v

`balance` (`withdraw` (a , v)) = `balance` (a) - v

To Do:

1. Prove by structural induction of bank accounts that the value returned by “balance” is never negative.
2. The specification is not sufficiently complete; show why. Add axiom(s) to make it sufficiently complete, and prove that, with such an extension, it is sufficiently complete.
3. It should be possible to transfer money from one account to the other. Please model a “transfer” function that transfers money from one bank account to another by adding types, functions, preconditions, and axioms.