Tutorial 2: Simple Bank

Ken Robinson

25th May 2000

Contents

1 SimpleBank0 2
2 SimpleBank 5
The solution to the simple bank specification is given as a document developed within the BToolkit. The English requirements will be distributed through the formal text.

1 SimpleBank0

Although not specified in the tutorial the machine will be given a parameter, \( \text{maxaccount} \), that will be used to set the size of the \( \text{ACCOUNT} \) set.

**MACHINE**  \( \text{SimpleBank0 ( maxaccount )} \)

Since \( \text{maxaccount} \) is to be the size of a deferred set, it must be a non-zero natural number.

**CONSTRAINTS**  \( \text{maxaccount} \in \mathbb{N}_1 \)

- **Requirement:** the bank customers are represented by accounts. Account identifiers are allocated from a pool (set) of identifiers maintained by the bank.

**SETS**  \( \text{ACCOUNT} \)

**PROPERTIES**  \( \text{card ( ACCOUNT )} = \text{maxaccount} \)

**VARIABLES**

- \( \text{accounts} \) contains the set of account numbers allocated to customers
- \( \text{accounts} \),
- and \( \text{balance} \) will contain the balance in each account.
- \( \text{balance} \)

**INVARIANT**

- \( \text{accounts} \subseteq \text{ACCOUNT} \land \\
  \text{Requirement: the bank needs to maintain a balance for all accounts.} \\
  \text{Thus we model balance as a total function with domain } \text{accounts}. \text{ We will model money amounts using } \mathbb{N}, \text{ cents.}  \\
  \text{balance} \in \text{accounts} \rightarrow \mathbb{N} \\
**INITIALISATION**

- \( \text{accounts} , \text{balance} := \{} , \{} \)

**OPERATIONS**

- **NewAccount**

  - **Requirement:** an operation by which a customer obtains an account identifier.
  
  Account identifiers are allocated from the set \( \text{ACCOUNT} \). Since this is a finite set we can eventually run out of identifiers. The precondition ensures that there are still identifiers remaining.
account ← NewAccount ≡
PRE accounts ≠ ACCOUNT
THEN
ANY acc
WHERE acc ∈ ACCOUNT − accounts
THEN account := acc ∥ accounts := accounts ∪ \{ acc \} ∥ balance ( acc ) := 0
END
END ;

Deposit
Requirement: an operation to add an amount to an account balance. The account id must be valid.

Deposit ( account , amount ) ≡ PRE account ∈ accounts ∧ amount ∈ N THEN balance ( account ) := balance ( account ) + amount END ;

WithDraw
Requirement: an operation to withdraw an amount from an account. Customers cannot withdraw more than the balance in their account.

WithDraw ( account , amount ) ≡ PRE account ∈ accounts ∧ amount ∈ N ∧ amount ≤ balance ( account ) THEN balance ( account ) := balance ( account ) − amount END ;

Balance
Requirement: an enquiry operation for a customer to obtain the balance in their account.

bal ← Balance ( account ) ≡ PRE account ∈ accounts THEN bal := balance ( account ) END ;

holdings ← Holdings ≡ BEGIN holdings := ∑ account . ( account ∈ accounts | balance ( account ) ) END
2 SimpleBank

The machine SimpleBank provides a robust interface to NewAccount1 and WithDraw, as well as providing an enquiry operation ValidAccount that can be used to check whether an account id is valid.

MACHINE SimpleBank (maxaccount)  
CONSTRAINTS maxaccount \in \mathbb{N}_1  
SEES Bool_TYPE  
INCLUDES SimpleBank0 (maxaccount)  
PROMOTES Deposit, Balance, Holdings

OPERATIONS

ok, account \leftarrow NewAccount1 \triangleq

\text{IF } \text{accounts} \neq \text{ACCOUNT} \\
\text{THEN }

\text{account} \leftarrow \text{NewAccount} \parallel 
\text{ok} := \text{TRUE}

\text{ELSE }

\text{account} \in \text{ACCOUNT} \parallel 
\text{ok} := \text{FALSE}

\text{END} ;

Cross-references

ACCOUNT SimpleBank0 \quad \text{SETS}
NewAccount SimpleBank0 \quad \text{OPERATIONS}
accounts SimpleBank0 \quad \text{VARIABLES}

ok \leftarrow \text{WithDraw1 (account, amount)} \triangleq

\text{PRE } \text{account} \in \text{ACCOUNT} \land \text{amount} \in \mathbb{N}
\text{THEN }

\text{IF } \text{account} \in \text{accounts} \land 
\text{amount} \leq \text{balance (account)}
\text{THEN } \text{WithDraw (account, amount)} \parallel \text{ok} := \text{TRUE}
\text{ELSE } \text{ok} := \text{FALSE}
\text{END}

\text{END} ;

Cross-references

ACCOUNT SimpleBank0 \quad \text{SETS}
WithDraw SimpleBank0 \quad \text{OPERATIONS}
accounts SimpleBank0 \quad \text{VARIABLES}
balance SimpleBank0 \quad \text{VARIABLES}

ok \leftarrow \text{ValidAccount (account)} \triangleq
PRE \( account \in \text{ACCOUNT} \)

THEN

\[ ok := \text{bool} \left( account \in \text{accounts} \right) \]

END

Cross-references

<table>
<thead>
<tr>
<th>ACCOUNT</th>
<th>SimpleBank0</th>
<th>SETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>accounts</td>
<td>SimpleBank0</td>
<td>VARIABLES</td>
</tr>
</tbody>
</table>

Cross-references for SimpleBank

<table>
<thead>
<tr>
<th>ACCOUNT</th>
<th>SimpleBank0</th>
<th>SETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance</td>
<td>SimpleBank0</td>
<td>OPERATIONS</td>
</tr>
<tr>
<td>Bool_TYPE</td>
<td>SimpleBank0</td>
<td>MACHINE</td>
</tr>
<tr>
<td>Deposit</td>
<td>SimpleBank0</td>
<td>OPERATIONS</td>
</tr>
<tr>
<td>Holdings</td>
<td>SimpleBank0</td>
<td>OPERATIONS</td>
</tr>
<tr>
<td>NewAccount</td>
<td>SimpleBank0</td>
<td>OPERATIONS</td>
</tr>
<tr>
<td>SimpleBank0</td>
<td>SimpleBank0</td>
<td>MACHINE</td>
</tr>
<tr>
<td>WithDraw</td>
<td>SimpleBank0</td>
<td>OPERATIONS</td>
</tr>
<tr>
<td>accounts</td>
<td>SimpleBank0</td>
<td>VARIABLES</td>
</tr>
<tr>
<td>balance</td>
<td>SimpleBank0</td>
<td>VARIABLES</td>
</tr>
</tbody>
</table>
Index

ACCOUNT  2
        2, 5, 6
Balance  ??
        5
Bool_TYPE ??
        5
Deposit  ??
        5
Holdings ??
        5
NewAccount ??
        5
SimpleBank  5
        5
SimpleBank0  2
        2, 5
WithDraw  ??
        5
accounts  2
        2, 5, 6
balance  2
        2, 5