EECS 3311 – W16 - Lab 2 – Bank

Student name: Jhan Perera

Prism Login: cse13187

I completed and submitted the whole lab: YES

Comments:

Chart View: BANK

```
class
        BANK

General
        cluster: model
                create: make

Ancestors
        ANY

Queries
        comment (s: STRING_8): BOOLEAN
        count: INTEGER_32
        customer_exists (a_name: STRING_8): BOOLEAN
        customer_id (a_name: STRING_8): INTEGER_32
        customer_with_name (a_name: STRING_8): CUSTOMER
        customers: LIST [CUSTOMER]
        customers_string: STRING_8
        customers_unchanged_other_than (a_name: STRING_8; old_customers: [like customers] LIST
[CUSTOMER]): BOOLEAN
        flag: BOOLEAN
        one: VALUE
        out: STRING_8
        partition (lo, hi: INTEGER_32): INTEGER_32
        sum_of_all_balances: VALUE
        total: VALUE
        unique_customers: BOOLEAN
        zero: VALUE

Commands
        deposit (a_name: STRING_8; a_value: VALUE)
        new (name1: STRING_8)
        quick_sort (a_first, a_last: INTEGER_32)
        swap (i, j: INTEGER_32)
```

```
        transfer (name1, name2: STRING_8; a_value: VALUE)
        withdraw (a_name: STRING_8; a_value: VALUE)

Constraints
        value constrsaints
        consistent count
        consistent total
        customer names unique
        customers are sorted
```

Contract View: BANK

```
class interface
        BANK

create
        make

        deposit (a_name: STRING_8; a_value: VALUE)
                        -- Deposit an amount of 'a_value' into account owned by 'a_name'.
                require
                        customer_exists: customer_exists (a_name)
                        positive_amount: (a_value > zero)
                ensure
                        deposit_num_customers_unchanged: count = old count
                        total_increased: total = old total.add (a_value)
                        deposit_customer_balance_increased: customers [customer_id
(a_name)].balance = old customers [customer_id (a_name)].balance + a_value
                        deposit_other_customers_unchanged: customers_unchanged_other_than
(a_name, old customers.deep_twin)
                        total_balance_increased: sum_of_all_balances = old sum_of_all_balances +
a_value

        withdraw (a_name: STRING_8; a_value: VALUE)
                        -- Withdraw an amount of 'a_value' from account owned by 'a_name'.
                require
                        customer_exists: customer_exists (a_name)
                        positive_amount: (a_value > zero)
                        sufficient_balance: not (customer_with_name (a_name).balance.is_less
(a_value))
                ensure
                        withdraw_num_customers_unchanged: count = old count
                        total_decreased: total = old total.minus (a_value)
                        withdraw_customer_balance_decreased: customers [customer_id
(a_name)].balance = old customers [customer_id (a_name)].balance - a_value
                        withdraw_other_customers_unchanged: customers_unchanged_other_than
(a_name, Current.customers.deep_twin)
                        total_balance_decreased: sum_of_all_balances = old sum_of_all_balances -
a_value

feature -- Command using multiple accounts

        transfer (name1: STRING_8; name2: STRING_8; a_value: VALUE)
                        -- Transfer an amount of 'a_value' from
                        -- account `name1' to account `name2'
                require
                        distinct_accounts: not (name1.is_case_insensitive_equal_general (name2))
                        customer1_exists: customer_exists (name1)
                        customer2_exists: customer_exists (name2)
                        sufficient_balance: not (customer_with_name (name1).balance.is_less
(a_value))
                ensure
                        same_total: total = old total
                        same_count: count = old count
                        total_balance_unchanged: sum_of_all_balances = old sum_of_all_balances
```

```
                        customer1_balance_decreased: customers [customer_id (name1)].balance =
old customers [customer_id (name1)].balance - a_value
                        customer2_balance_increased: customers [customer_id (name2)].balance =
old customers [customer_id (name2)].balance + a_value
                        other_customers_unchanged: customers_unchanged_other_than (name1,
Current.customers.deep_twin) and then customers_unchanged_other_than (name2,
Current.customers.deep_twin)



invariant
        value_constrsaints: zero = zero.zero and one = one.one
        consistent_count: count = customers.count
        consistent_total: total = sum_of_all_balances
        customer_names_unique: unique_customers
        customers_are_sorted: ∀ i ϵ customers and  ∀ j ϵ customers ( i.name <= j.name)

end -- class BANK
```

BON diagram: BANK CLUSTER