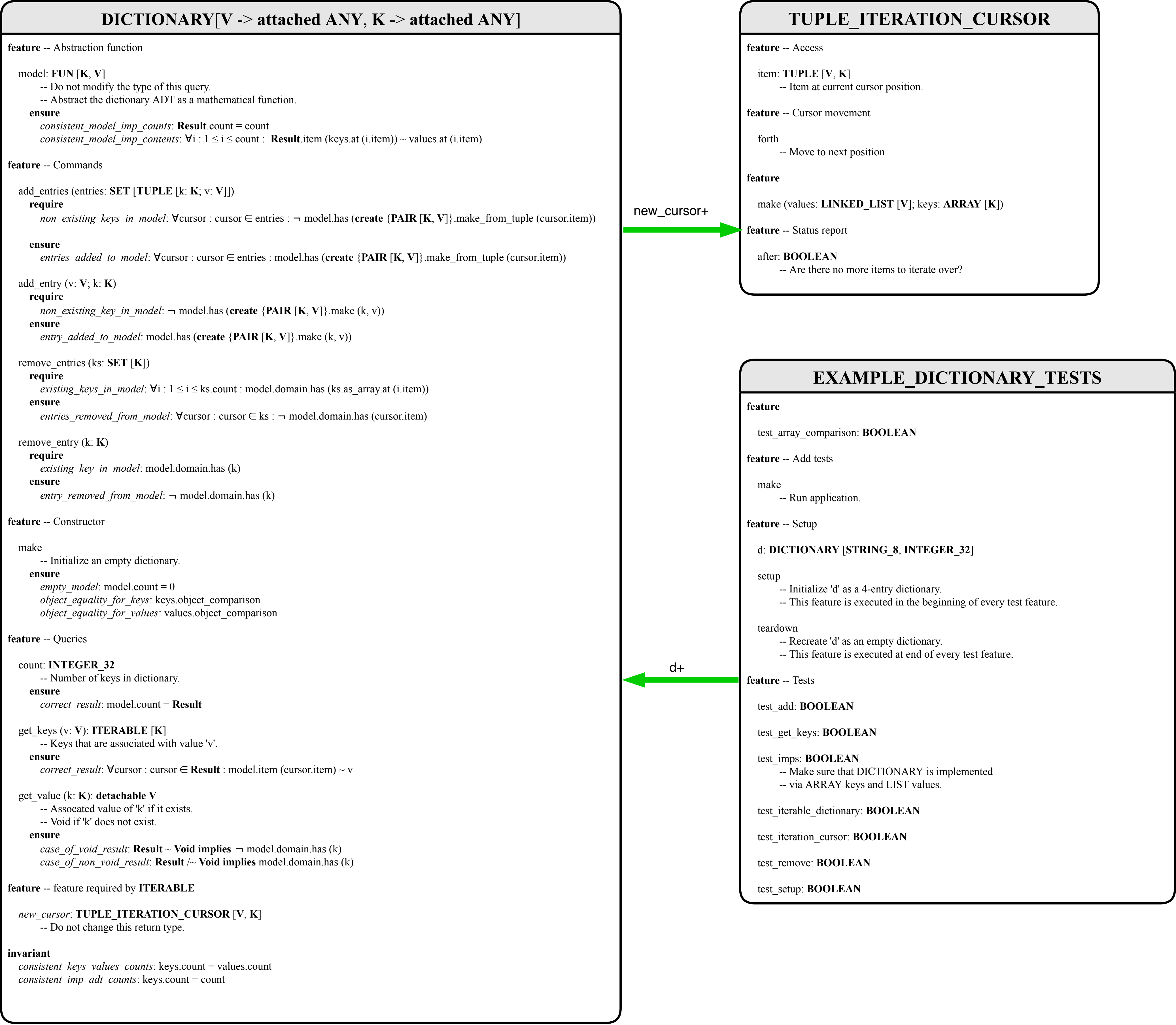

DICTIONARY[V -> attached ANY, K -> attached ANY]

feature -- Abstraction function

model: **FUN** [**K**, **V**]
-- Do not modify the type of this query.
-- Abstract the dictionary ADT as a mathematical function.
ensure
consistent_model_imp_counts: **Result**.count = count
consistent_model_imp_contents: ∀i : 1 ≤ i ≤ count : **Result**.item (keys.at (i.item)) ~ values.at (i.item)

feature -- Commands

add_entries (entries: **SET** [**TUPLE** [k: **K**; v: **V**]])
require
non_existing_keys_in_model: ∀cursor : cursor ∈ entries : ¬ model.has (**create** {**PAIR** [**K**, **V**]}.make_from_tuple (cursor.item))
ensure
entries_added_to_model: ∀cursor : cursor ∈ entries : model.has (**create** {**PAIR** [**K**, **V**]}.make_from_tuple (cursor.item))

add_entry (v: **V**; k: **K**)
require
non_existing_key_in_model: ¬ model.has (**create** {**PAIR** [**K**, **V**]}.make (k, v))
ensure
entry_added_to_model: model.has (**create** {**PAIR** [**K**, **V**]}.make (k, v))

remove_entries (ks: **SET** [**K**])
require
existing_keys_in_model: ∀i : 1 ≤ i ≤ ks.count : model.domain.has (ks.as_array.at (i.item))
ensure
entries_removed_from_model: ∀cursor : cursor ∈ ks : ¬ model.domain.has (cursor.item)

remove_entry (k: **K**)
require
existing_key_in_model: model.domain.has (k)
ensure
entry_removed_from_model: ¬ model.domain.has (k)

feature -- Constructor

make
-- Initialize an empty dictionary.
ensure
empty_model: model.count = 0
object_equality_for_keys: keys.object_comparison
object_equality_for_values: values.object_comparison

feature -- Queries

count: **INTEGER_32**
-- Number of keys in dictionary.
ensure
correct_result: model.count = **Result**

get_keys (v: **V**): **ITERABLE** [**K**]
-- Keys that are associated with value 'v'.
ensure
correct_result: ∀cursor : cursor ∈ **Result** : model.item (cursor.item) ~ v

get_value (k: **K**): **detachable V**
-- Assocated value of 'k' if it exists.
-- Void if 'k' does not exist.
ensure
case_of_void_result: **Result** ~ **Void implies** ¬ model.domain.has (k)
case_of_non_void_result: **Result** /~ **Void implies** model.domain.has (k)

feature -- feature required by **ITERABLE**

new_cursor: **TUPLE_ITERATION_CURSOR** [**V**, **K**]
-- Do not change this return type.

invariant
consistent_keys_values_counts: keys.count = values.count
consistent_imp_adt_counts: keys.count = count

TUPLE_ITERATION_CURSOR

feature -- Access

item: **TUPLE** [**V**, **K**]
-- Item at current cursor position.

feature -- Cursor movement

forth
-- Move to next position

feature

make (values: **LINKED_LIST** [**V**]; keys: **ARRAY** [**K**])

feature -- Status report

after: **BOOLEAN**
-- Are there no more items to iterate over?

EXAMPLE_DICTIONARY_TESTS

feature

test_array_comparison: **BOOLEAN**

feature -- Add tests

make
-- Run application.

feature -- Setup

d: **DICTIONARY** [**STRING_8**, **INTEGER_32**]

setup
-- Initialize 'd' as a 4-entry dictionary.
-- This feature is executed in the beginning of every test feature.

teardown
-- Recreate 'd' as an empty dictionary.
-- This feature is executed at end of every test feature.

feature -- Tests

test_add: **BOOLEAN**

test_get_keys: **BOOLEAN**

test_imps: **BOOLEAN**
-- Make sure that DICTIONARY is implemented
-- via ARRAY keys and LIST values.

test_iterable_dictionary: **BOOLEAN**

test_iteration_cursor: **BOOLEAN**

test_remove: **BOOLEAN**

test_setup: **BOOLEAN**