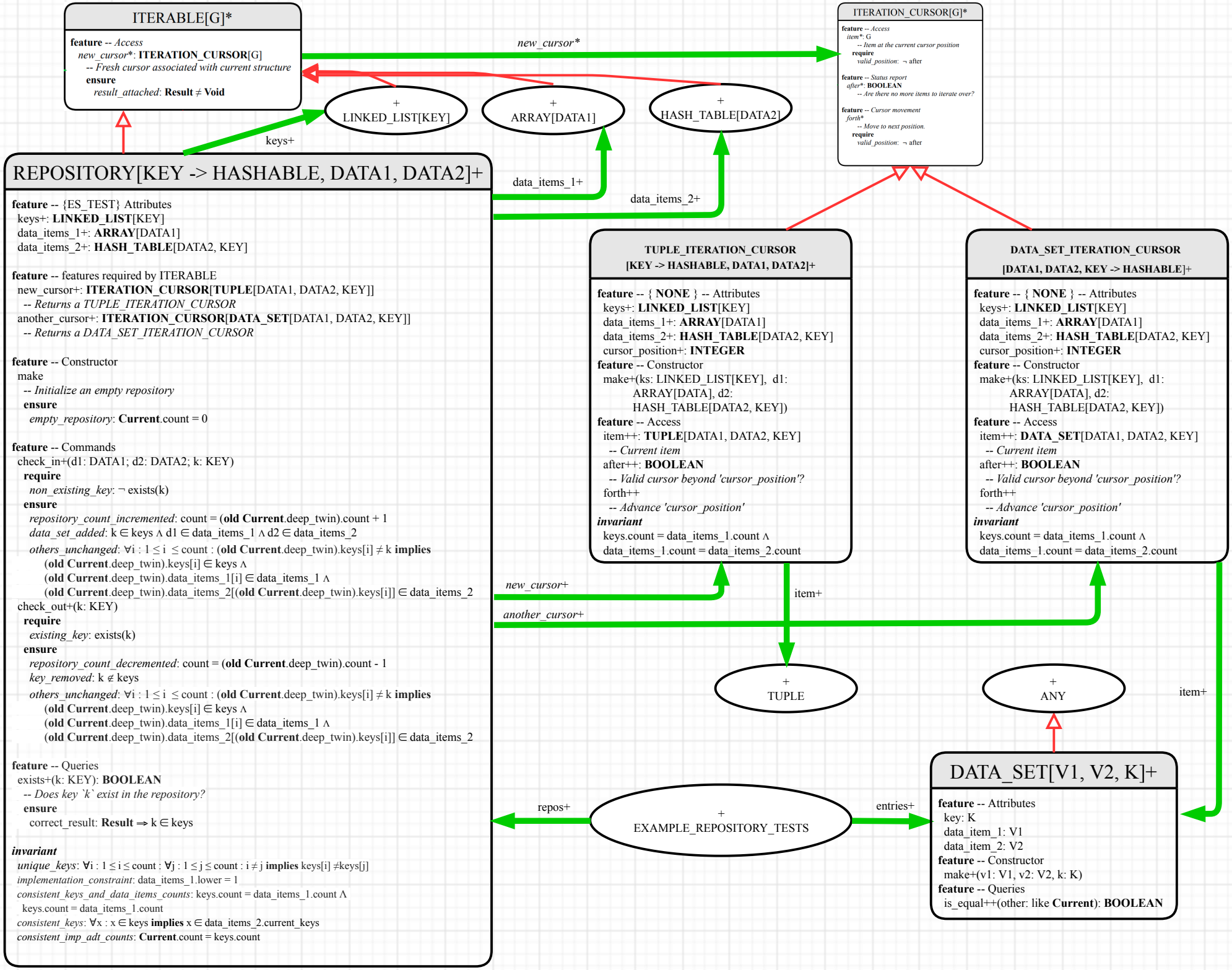
ITERABLE[G]*
feature -- Access
new_cursor*: ITERATION_CURSOR[G]
-- Fresh cursor associated with current structure
ensure
result_attached: Result ≠ Void
new_cursor*
ITERATION_CURSOR[G]*
feature -- Access
item*: G
-- Item at the current cursor position
require
valid_position: ¬ after
feature -- Status report
after*: BOOLEAN
-- Are there no more items to iterate over?
feature -- Cursor movement
forth*
-- Move to next position.
require
valid_position: ¬ after
+
LINKED_LIST[KEY]
+
ARRAY[DATA1]
+
HASH_TABLE[DATA2]
keys+
data_items_1+
data_items_2+
REPOSITORY[KEY -> HASHABLE, DATA1, DATA2]+
feature -- {ES_TEST} Attributes
keys+: LINKED_LIST[KEY]
data_items_1+: ARRAY[DATA1]
data_items_2+: HASH_TABLE[DATA2, KEY]
feature -- features required by ITERABLE
new_cursor+: ITERATION_CURSOR[TUPLE[DATA1, DATA2, KEY]]
-- Returns a TUPLE_ITERATION_CURSOR
another_cursor+: ITERATION_CURSOR[DATA_SET[DATA1, DATA2, KEY]]
-- Returns a DATA_SET_ITERATION_CURSOR
feature -- Constructor
make
-- Initialize an empty repository
ensure
empty_repository: Current.count = 0
feature -- Commands
check_in+(d1: DATA1; d2: DATA2; k: KEY)
require
non_existing_key: ¬ exists(k)
ensure
repository_count_incremented: count = (old Current.deep_twin).count + 1
data_set_added: k ∈ keys ∧ d1 ∈ data_items_1 ∧ d2 ∈ data_items_2
others_unchanged: ∀i : 1 ≤ i ≤ count : (old Current.deep_twin).keys[i] ≠ k implies
(old Current.deep_twin).keys[i] ∈ keys ∧
(old Current.deep_twin).data_items_1[i] ∈ data_items_1 ∧
(old Current.deep_twin).data_items_2[(old Current.deep_twin).keys[i]] ∈ data_items_2
check_out+(k: KEY)
require
existing_key: exists(k)
ensure
repository_count_decremented: count = (old Current.deep_twin).count - 1
key_removed: k ∉ keys
others_unchanged: ∀i : 1 ≤ i ≤ count : (old Current.deep_twin).keys[i] ≠ k implies
(old Current.deep_twin).keys[i] ∈ keys ∧
(old Current.deep_twin).data_items_1[i] ∈ data_items_1 ∧
(old Current.deep_twin).data_items_2[(old Current.deep_twin).keys[i]] ∈ data_items_2
feature -- Queries
exists+(k: KEY): BOOLEAN
-- Does key `k` exist in the repository?
ensure
correct_result: Result ⇒ k ∈ keys
invariant
unique_keys: ∀i : 1 ≤ i ≤ count : ∀j : 1 ≤ j ≤ count : i ≠ j implies keys[i] ≠keys[j]
implementation_constraint: data_items_1.lower = 1
consistent_keys_and_data_items_counts: keys.count = data_items_1.count ∧
keys.count = data_items_1.count
consistent_keys: ∀x : x ∈ keys implies x ∈ data_items_2.current_keys
consistent_imp_adt_counts: Current.count = keys.count
TUPLE_ITERATION_CURSOR
[KEY -> HASHABLE, DATA1, DATA2]+
feature -- { NONE } -- Attributes
keys+: LINKED_LIST[KEY]
data_items_1+: ARRAY[DATA1]
data_items_2+: HASH_TABLE[DATA2, KEY]
cursor_position+: INTEGER
feature -- Constructor
make+(ks: LINKED_LIST[KEY], d1:
ARRAY[DATA], d2:
HASH_TABLE[DATA2, KEY])
feature -- Access
item++: TUPLE[DATA1, DATA2, KEY]
-- Current item
after++: BOOLEAN
-- Valid cursor beyond 'cursor_position'?
forth++
-- Advance 'cursor_position'
invariant
keys.count = data_items_1.count ∧
data_items_1.count = data_items_2.count
DATA_SET_ITERATION_CURSOR
[DATA1, DATA2, KEY -> HASHABLE]+
feature -- { NONE } -- Attributes
keys+: LINKED_LIST[KEY]
data_items_1+: ARRAY[DATA1]
data_items_2+: HASH_TABLE[DATA2, KEY]
cursor_position+: INTEGER
feature -- Constructor
make+(ks: LINKED_LIST[KEY], d1:
ARRAY[DATA], d2:
HASH_TABLE[DATA2, KEY])
feature -- Access
item++: DATA_SET[DATA1, DATA2, KEY]
-- Current item
after++: BOOLEAN
-- Valid cursor beyond 'cursor_position'?
forth++
-- Advance 'cursor_position'
invariant
keys.count = data_items_1.count ∧
data_items_1.count = data_items_2.count
new_cursor+
another_cursor+
item+
+
TUPLE
+
ANY
item+
DATA_SET[V1, V2, K]+
feature -- Attributes
key: K
data_item_1: V1
data_item_2: V2
feature -- Constructor
make+(v1: V1, v2: V2, k: K)
feature -- Queries
is_equal++(other: like Current): BOOLEAN
repos+
+
EXAMPLE_REPOSITORY_TESTS
entries+