## EECS 3311 - LAB 2 DATABASE Varuhn Ruthirakuhan - 215634140

## feature {DATABASE} --Initialization Design Diagram for Database structure in eiffel studio -- make empty linear database for 'keys' and 'values' feature -- Abstract Function count model+: REL[K, V] -feature -- Deferred Routines INTEGER count+: INTEGER --Return total number of elements in keys array has\_key+(p\_key: K): BOOLEAN -- Return true if p\_key is in array 'keys' and false otherwise **DATABASE\*** search+(p key: K): detachable V --checks if p\_key is located in hashtable 'values' has\_key insert+(p\_key: K; p\_value: V) feature -- Abstraction function -- inserts a ['p\_key', 'p\_value'] mapping into the model\*: REL[K, V] BOOLEAN ensure delete+(p\_key: K) unchanged count: count = old count -- removes mapping if p key is located unchanged implementation: model = old model invariant: $all_{key_value_tuples_exist_as_model_pairs: [K,V] = TUPLE$ key data pair count same: all model pairs exist as model pairs: -- i: 1 < i < count: [k, v] exists in TUPLE keys.count = values.count feature -- Deferred Routines all\_key\_exists\_in\_data: count\*: INTEGER ensure nothing changed: count = old count consistent counts: Result = model.count has\_key\*(p\_key: K): BOOLEAN ensure nothing\_changed: key exists: -search\*(p key: K): detachable V count INTEGER nothing\_changed: checks if all variables remained case\_of\_key\_found: key has been found insert\*(p\_key: K; p\_value: V) has kev BOOLEAN no\_previous\_entry: checks key does not exists entry added: insert value with corresponding key delete\*(p\_key: K) require existing\_entry: checks if key exists ensure entry\_deleted: if yes, then delete values attached to key feature -- Basic override+(p\_key: K; p\_value: V) -- updates entry with p key so p key maps with p value ensure overriden result: old model = current model common key db+(p second: DATABASE[K, V]): DATABASE[K, V] -- Result database created for union of entries nothing\_changed: old DATABASE = DATABASE result\_has\_common\_key\_mappings\_from\_current\_database\_and\_p\_second: --TREE DB+ feature -- Intermediate interval\_image +(p\_first, p\_finish: K): LIST[V] inherit -- Returns list of values mapped from keys count DATABASE[K, V] feature {ITERATION CURSOR, ES TEST} INTEGER p first smaller than p finish: p first < p finish bst: BALANCED\_BST[K, V] feature {DATABASE} - Initialization nothing changed: {DATABASE} = old {DATABASE} -- create empty tree database via an empty splay correct\_values\_are\_included\_in\_result: has key --∃i: 1<i<count: key exists & value matches key feature -- Abstraction Function result\_includes\_correct\_values\_only: BOOLEAN model+: REL [K, V] --∃i: 1<i<count: Result = [key, value] feature -- Deferred Routines feature -- Advanced count+: INTEGER innerjoin+(other: DATABASE[K, STRING]): REL[K, PAIR[V, STRING]] -- return number of mappings in DATABASE TUPLE[keys\*, values\*] has\_key+(p\_key: K): BOOLEAN ensure: -- Return true if mapping with p\_key exists and nothing changed: old {DATABASE} = {DATABASE} false otherwise result exists in current database: -- \( \frac{1}{2} \) i < i < count: \( [k, v] \) exists in \( [k, (v, s)] \) search+(p\_key: K): detachable V result exists in other database:--\(\frac{1}{2}\)i: 1<i<count: other [k, v] exists in [k, (v, s)] -- check if p\_key exists in binary search tree common\_key\_mapping\_exists\_in\_result: --\(\frac{1}{3}\)i: 1<i<count: other [k, v] = current [k, v] insert(p\_key: K; p\_value: V) -- Inserts a ['p\_key', 'p\_value'] mapping into the database delete(p key: K)

LINEAR DB+

feature {ITERATION CURSOR, ES TEST} --

-- deletes mapping that has key 'p\_key'

LINEAR IT

TREE IT

inherit

DATABASE [K, V]

Restricted Attributes keys: ARRAY[K] values: HASH\_TABLE[V, K]