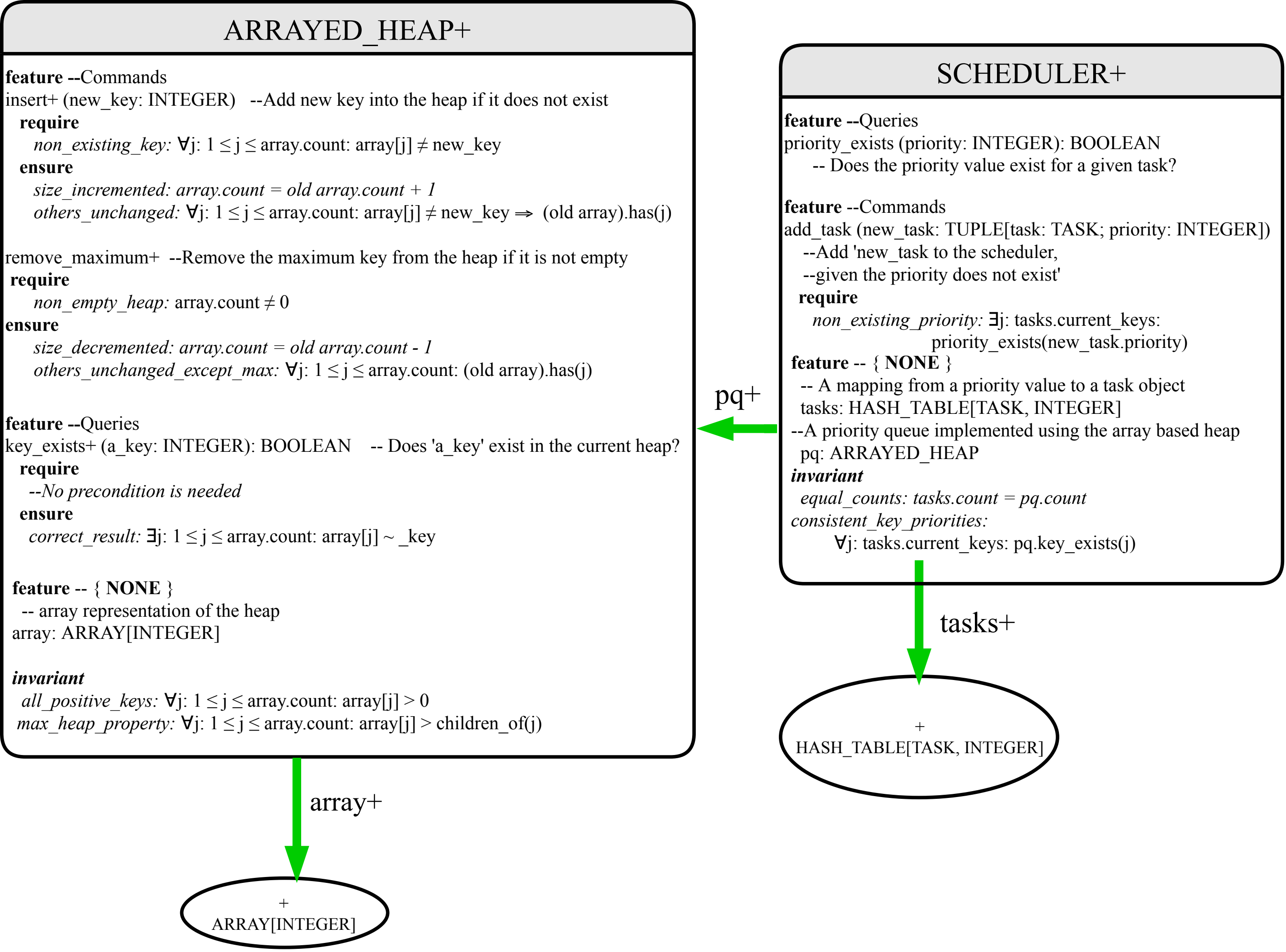

```
ARRAYED_HEAP+

feature --Commands
insert+ (new_key: INTEGER)   --Add new key into the heap if it does not exist
  require
    non_existing_key: ∀j: 1 ≤ j ≤ array.count: array[j] ≠ new_key
  ensure
    size_incremented: array.count = old array.count + 1
    others_unchanged: ∀j: 1 ≤ j ≤ array.count: array[j] ≠ new_key ⇒ (old array).has(j)

remove_maximum+  --Remove the maximum key from the heap if it is not empty
 require
    non_empty_heap: array.count ≠ 0
ensure
    size_decremented: array.count = old array.count - 1
    others_unchanged_except_max: ∀j: 1 ≤ j ≤ array.count: (old array).has(j)

feature --Queries
key_exists+ (a_key: INTEGER): BOOLEAN    -- Does 'a_key' exist in the current heap?
  require
   --No precondition is needed
  ensure
   correct_result: ∃j: 1 ≤ j ≤ array.count: array[j] ~ _key

 feature -- { NONE }
  -- array representation of the heap
 array: ARRAY[INTEGER]

 invariant
  all_positive_keys: ∀j: 1 ≤ j ≤ array.count: array[j] > 0
 max_heap_property: ∀j: 1 ≤ j ≤ array.count: array[j] > children_of(j)
```

```
SCHEDULER+

feature --Queries
priority_exists (priority: INTEGER): BOOLEAN
    -- Does the priority value exist for a given task?

feature --Commands
add_task (new_task: TUPLE[task: TASK; priority: INTEGER])
  --Add 'new_task to the scheduler,
  --given the priority does not exist'
 require
   non_existing_priority: ∃j: tasks.current_keys:
                          priority_exists(new_task.priority)
 feature -- { NONE }
  -- A mapping from a priority value to a task object
  tasks: HASH_TABLE[TASK, INTEGER]
 --A priority queue implemented using the array based heap
  pq: ARRAYED_HEAP
 invariant
  equal_counts: tasks.count = pq.count
 consistent_key_priorities:
       ∀j: tasks.current_keys: pq.key_exists(j)
```