Smart Buildings Data Structures

ADT MyHashMap
MyHashMap "hm" < "hm" = (<k_1, v_1>, <k_2, v_2>, … <k_n, v_n>), size = n **>**
{ inv: MyHashMap.size ≥ 0 ⋀ (∀k_i ∈ MyHashMap "hm" ⋀ ∀k_j ∈ MyHashMap "hm" ⋀ i ≠ j ⇒ $k_i \neq k_j$) ⋀ (∀v_i ∈ MyHashMap "hm" ⋀ ∀v_j ∈ MyHashMap "hm" ⋀ i ≠ j ⇒ $v_i \neq v_j$) **}**
Primitive Operations • Put: *Key x Value* → Void • Get: *Key* → Value • ContainsKey: *Key* → Boolean • ContainsValue: *Value* → Boolean • Replace: *Key x Value* → Void • Size: *None* → Int

ADT MyArrayList
MyArrayList "al" < "al" = <e_1, e_2, … e_n>, size = n **>**
{ inv: MyArrayList.size ≥ 0 ⋀ (∀e_i ∈ MyArrayList "al" ⋀ ∀e_j ∈ MyArrayList "al" ⋀ i ≠ j ⇒ $e_i \neq e_j$) **}**
Primitive Operations • Add: *Element* → Void • RemoveAll: *None* → Void • Size: *None* → Int • Get: *Int* → Element

ADT MyPriorityQueue
MyPriorityQueue "pq" < "pq" = <e_1, e_2, … e_n>, size = n **>**
{ inv: MyPriorityQueue.size ≥ 0 ⋀ (∀e_i ∈ MyPriorityQueue "pq" ⋀ ∀e_j ∈ MyPriorityQueue "pq" ⋀ i ≠ j ⇒ $e_i \neq e_j$) ⋀ i > j ⇒ e_i.priority > e_i.priority **}**

Primitive Operations

• Peek:	*None*	$\rightarrow$ Element$_0$
• Offer:	*Element*	$\rightarrow$ Void
• Size:	*None*	$\rightarrow$ Int
• Poll:	*None*	$\rightarrow$ Element

ADT MyQueue

MyQueue "q" $<$ "q" $= <e_1, e_2, \ldots e_n>$, size $= n >$

{ **inv:** MyQueue.size $\geq 0 \wedge (\forall e_i \in$ MyQueue "q" $\wedge \forall e_j \in$ MyQueue "q" $\wedge i \neq j \Rightarrow e_i \neq e_j)$ }

Primitive Operations

• Add:	*Element*	$\rightarrow$ Void
• Element:	*None*	$\rightarrow$ Element$_0$

ADT MyStack

MyStack "s" $<$ "s" $= <e_1, e_2, \ldots e_n>$, size $= n >$

{ **inv:** MyStack.size $\geq 0 \wedge (\forall e_i \in$ MyStack "s" $\wedge \forall e_j \in$ MyStack "s" $\wedge i \neq j \Rightarrow e_i \neq e_j)$ }

Primitive Operations

• Peek:	*None*	$\rightarrow$ Element$_0$
• Pop:	*None*	$\rightarrow$ Element$_0$
• Push:	*Element*	$\rightarrow$ Void
• Size:	*None*	$\rightarrow$ Int