Smart Estate Specification

COMPSCI 2XB3 L09 Group 9

March 29, 2019

This Module Interface Specification (MIS) document contains modules, types and methods for implementing Smart Estate.

StateInfo Type Module

Module

StateInfo

Uses

N/A

Syntax

Exported Constants

None

Exported Types

StateInfo = ?

Exported Access Programs

Routine name	In	Out	Exceptions
new StateInfo	String	StateInfo	none
getState		String	none
getHPI		$\mathbb{R}$	none
setHPI	$\mathbb{R}$		none
getCrimeRate		$\mathbb{R}$	none
setCrimeRate	$\mathbb{R}$		none
getHousingPrice		$\mathbb{R}$	none
setHousingPrice	$\mathbb{R}$		none
toString		String	none

Semantics

State Variables

state: String
hpi: $\mathbb{R}$
crime_rate: $\mathbb{R}$
housing_price: $\mathbb{R}$

State Invariant

None

Assumptions & Design Decisions

- The StateInfo constructor is called for each object instance before any other access routine is called for that object. The constructor can only be called once.
- Once state info is gathered for each StateInfo object methods setHPI, setCrimeRate, and setHousingPrice are only called once.

Access Routine Semantics

new StateInfo(s):

- transition: $state := s$
- output: $out := self$
- exception: none

getState():

- output: $out := state$
- exception: none

getHPI():

- output: $out := hpi$
- exception: none

setHPI(v):

- transition: $hpi := v$
- exception: none

getCrimeRate():

- output: $out := crime\_rate$
- exception: none

setCrimeRate(v):

- transition: $crime\_rate := v$
- exception: none

getHousingPrice():

- output: $out := housing\_price$
- exception: none

getHousingPrice(v):

- transition: $housing\_price := v$
- exception: none

toString():

- output: $out :=$ "$state$: HPI: hpi Crime Rate: $crime\_rate$ Housing Price: $housing\_price$"
- exception: none

PopulateStateInfo Module

Module

PopulateStateInfo

Uses

ReadHPI
ReadCrimeRate
ReadHousingPrices
StateInfo

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
initStates			none
populateHPI			none
populateCrimeRate			none
populateHousingPrice			none
populateStateInfo		seq of StateInfo	none

Semantics

State Variables

states: seq of StateInfo
state_names: seq of String = [”Alabama”,”Alaska”, ... , ”Wyoming”]

State Invariant

None

Assumptions & Design Decisions

- The result of populateStateInfo must be stored in a StateInfo list of 50 length.

Access Routine Semantics

initStates():

- transition: $states$:=($\forall s$:String $|s \in state\_names$. s = StateInfo(s))
- exception: none

populateHPI():

- transition: $states$:=
  ($\forall i$:int $|0 \leq i \leq 50$. $states[i]$.setHPI(ReadHPI.read_data("data/hpi.csv").value()))
- exception: none

populateCrimeRate():

- transition: $states$:=
  ($\forall i$:int $|0 \leq i \leq 50$. $states[i]$.setCrimeRate(ReadCrimeRate.CRList().value()))
- exception: none

populateHousingPrice():

- transition: $states$:=
  ($\forall i$:int $|0 \leq i \leq 50$. $states[i]$.setHousingPrice(ReadHousingPrices.
  readPrices("data/housingPrices.csv").value()))
- exception: none

populateStateInfo():

- transition: initStates(); populateHPI(); populateCrimeRate();
  populateHousingPrice();
- output: out := $states$
- exception: None

Binary Search Module

Module

binSearch

Uses

StateInfo

Syntax

Exported Types

fieldT = hpi, crime_rate, housing_price

Exported Access Programs

Routine name	In	Out	Exceptions
binSearch	seq of StateInfo, fieldT, $\mathbb{R}$	StateInfo	none

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

- The result of populateStateInfo must be stored in a StateInfo list of 50 length.

Access Routine Semantics

initStates():

- transition:
- output: *out* :=
- exception: none