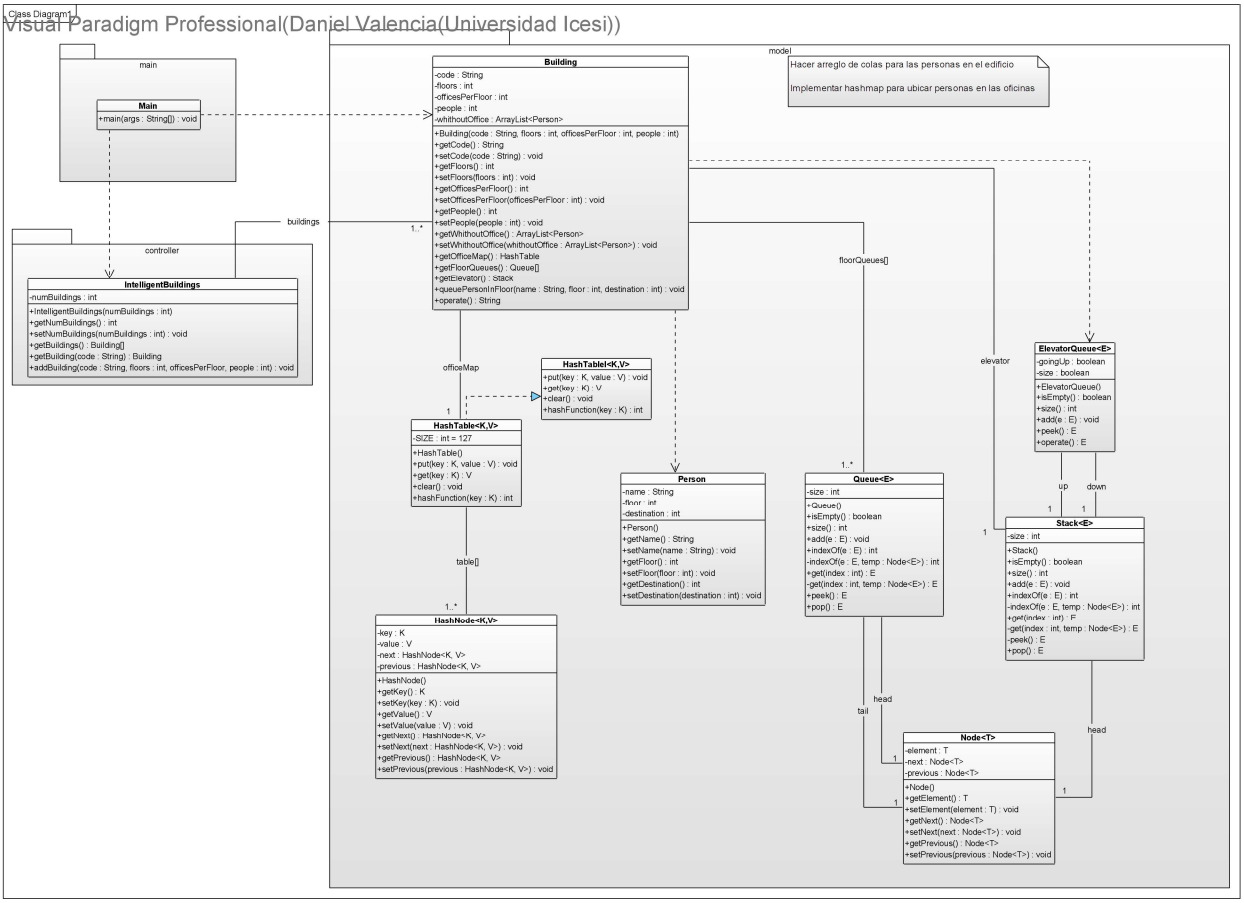
Class Diagram1
Visual Paradigm Professional(Daniel Valencia(Universidad Icesi))
main
Main
+main(args : String[]) : void
controller
IntelligentBuildings
-numBuildings : int
+IntelligentBuildings(numBuildings : int)
+getNumBuildings() : int
+setNumBuildings(numBuildings : int) : void
+getBuildings() : Building[]
+getBuilding(code : String) : Building
+addBuilding(code : String, floors : int, officesPerFloor, people : int) : void
buildings
1..*
model
Hacer arreglo de colas para las personas en el edificio
Implementar hashmap para ubicar personas en las oficinas
Building
-code : String
-floors : int
-officesPerFloor : int
-people : int
-whithoutOffice : ArrayList<Person>
+Building(code : String, floors : int, officesPerFloor : int, people : int)
+getCode() : String
+setCode(code : String) : void
+getFloors() : int
+setFloors(floors : int) : void
+getOfficesPerFloor() : int
+setOfficesPerFloor(officesPerFloor : int) : void
+getPeople() : int
+setPeople(people : int) : void
+getWhithoutOffice() : ArrayList<Person>
+setWhithoutOffice(whithoutOffice : ArrayList<Person>) : void
+getOfficeMap() : HashTable
+getFloorQueues() : Queue[]
+getElevator() : stack
+queuePersonInFloor(name : String, floor : int, destination : int) : void
+operate() : String
officeMap
1
HashTable<K,V>
-SIZE : int = 127
+HashTable()
+put(key : K, value : V) : void
+get(key : K) : V
+clear() : void
+hashFunction(key : K) : int
HashTable<K,V>
+put(key : K, value : V) : void
+get(key : K) : V
+clear() : void
+hashFunction(key : K) : int
table[]
1..*
HashNode<K,V>
-key : K
-value : V
-next : HashNode<K, V>
-previous : HashNode<K, V>
+HashNode()
+getKey() : K
+setKey(key : K) : void
+getValue() : V
+setValue(value : V) : void
+getNext() : HashNode<K, V>
+setNext(next : HashNode<K, V>) : void
+getPrevious() : HashNode<K, V>
+setPrevious(previous : HashNode<K, V>) : void
Person
-name : String
-floor : int
-destination : int
+Person()
+getName() : String
+setName(name : String) : void
+getFloor() : int
+setFloor(floor : int) : void
+getDestination() : int
+setDestination(destination : int) : void
floorQueues[]
1..*
Queue<E>
-size : int
+Queue()
+isEmpty() : boolean
+size() : int
+add(e : E) : void
+indexOf(e : E) : int
-indexOf(e : E, temp : Node<E>) : int
+get(index : int) : E
-get(index : int, temp : Node<E>) : E
+peek() : E
+pop() : E
head
tail
1
Node<T>
-element : T
-next : Node<T>
-previous : Node<T>
+Node()
+getElement() : T
+setElement(element : T) : void
+getNext() : Node<T>
+setNext(next : Node<T>) : void
+getPrevious() : Node<T>
+setPrevious(previous : Node<T>) : void
elevator
1
ElevatorQueue<E>
-goingUp : boolean
-size : boolean
+ElevatorQueue()
+isEmpty() : boolean
+size() : int
+add(e : E) : void
+peek() : E
+operate() : E
up
down
1
Stack<E>
-size : int
+Stack()
+isEmpty() : boolean
+size() : int
+add(e : E) : void
+indexOf(e : E) : int
-indexOf(e : E, temp : Node<E>) : int
+get(index : int) : E
-get(index : int, temp : Node<E>) : E
-peek() : E
+pop() : E
head