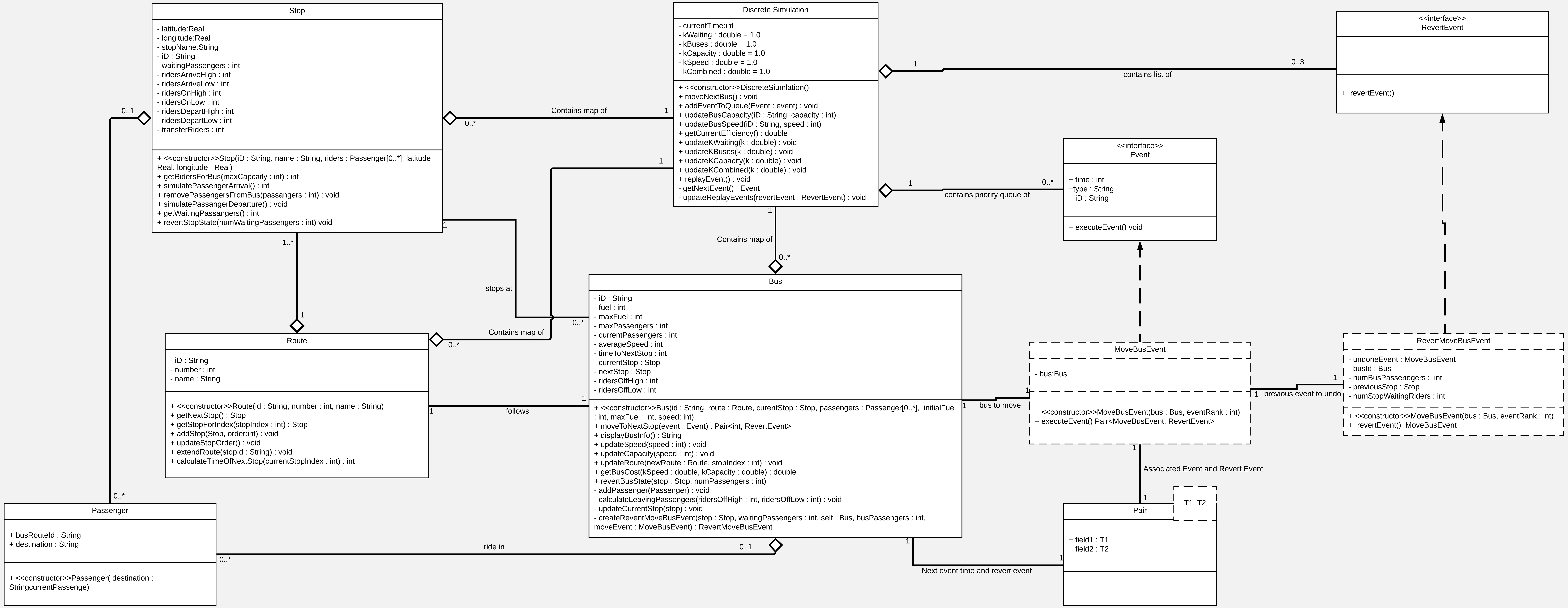
UML CLASS DIAGRAM MARTA SIMULATION

Alex Tulenko | UML 2 | October 28, 2018


Stop
- latitude:Real
- longitude:Real
- stopName:String
- iD : String
- waitingPassengers : int
- ridersArriveHigh : int
- ridersArriveLow : int
- ridersOnHigh : int
- ridersOnLow : int
- ridersDepartHigh : int
- ridersDepartLow : int
- transferRiders : int
+ <<constructor>>Stop(iD : String, name : String, riders : Passenger[0..*], latitude : Real, longitude : Real)
+ getRidersForBus(maxCapcaity : int) : int
+ simulatePassengerArrival() : int
+ removePassengersFromBus(passangers : int) : void
+ simulatePassangerDeparture() : void
+ getWaitingPassangers() : int
+ revertStopState(numWaitingPassengers : int) void
Discrete Simulation
- currentTime:int
- kWaiting : double = 1.0
- kBuses : double = 1.0
- kCapacity : double = 1.0
- kSpeed : double = 1.0
- kCombined : double = 1.0
+ <<constructor>>DiscreteSiumlation()
+ moveNextBus() : void
+ addEventToQueue(Event : event) : void
+ updateBusCapacity(iD : String, capacity : int)
+ updateBusSpeed(iD : String, speed : int)
+ getCurrentEfficiency() : double
+ updateKWaiting(k : double) : void
+ updateKBuses(k : double) : void
+ updateKCapacity(k : double) : void
+ updateKCombined(k : double) : void
+ replayEvent() : void
- getNextEvent() : Event
- updateReplayEvents(revertEvent : RevertEvent) : void
<<interface>>
RevertEvent
+ revertEvent()
1
0..3
contains list of
<<interface>>
Event
+ time : int
+type : String
+ iD : String
+ executeEvent() void
1
0..*
contains priority queue of
0..1
0..*
Contains map of
1
1
1
1..*
1
Contains map of
0..*
stops at
Contains map of
0..*
0..*
Route
- iD : String
- number : int
- name : String
+ <<constructor>>Route(id : String, number : int, name : String)
+ getNextStop() : Stop
+ getStopForIndex(stopIndex : int) : Stop
+ addStop(Stop, order:int) : void
+ updateStopOrder() : void
+ extendRoute(stopId : String) : void
+ calculateTimeOfNextStop(currentStopIndex : int) : int
1
follows
1
Bus
- iD : String
- fuel : int
- maxFuel : int
- maxPassengers : int
- currentPassengers : int
- averageSpeed : int
- timeToNextStop : int
- currentStop : Stop
- nextStop : Stop
- ridersOffHigh : int
- ridersOffLow : int
+ <<constructor>>Bus(id : String, route : Route, curentStop : Stop, passengers : Passenger[0..*], initialFuel : int, maxFuel : int, speed: int)
+ moveToNextStop(event : Event) : Pair<int, RevertEvent>
+ displayBusInfo() : String
+ updateSpeed(speed : int) : void
+ updateCapacity(speed : int) : void
+ updateRoute(newRoute : Route, stopIndex : int) : void
+ getBusCost(kSpeed : double, kCapacity : double) : double
+ revertBusState(stop : Stop, numPassengers : int)
- addPassenger(Passenger) : void
- calculateLeavingPassengers(ridersOffHigh : int, ridersOffLow : int) : void
- updateCurrentStop(stop) : void
- createReventMoveBusEvent(stop : Stop, waitingPassengers : int, self : Bus, busPassengers : int, moveEvent : MoveBusEvent) : RevertMoveBusEvent
1
bus to move
1
MoveBusEvent
- bus:Bus
+ <<constructor>>MoveBusEvent(bus : Bus, eventRank : int)
+ executeEvent() Pair<MoveBusEvent, RevertEvent>
1
previous event to undo
1
RevertMoveBusEvent
- undoneEvent : MoveBusEvent
- busId : Bus
- numBusPassenegers : int
- previousStop : Stop
- numStopWaitingRiders : int
+ <<constructor>>MoveBusEvent(bus : Bus, eventRank : int)
+ revertEvent() MoveBusEvent
1
Associated Event and Revert Event
1
Pair
T1, T2
+ field1 : T1
+ field2 : T2
0..*
Passenger
+ busRouteId : String
+ destination : String
+ <<constructor>>Passenger(destination : StringcurrentPassenge)
0..*
ride in
0..1
1
Next event time and revert event
1

Updates to UML Class Diagram

Corrections to previous submission:
Using aggregation connector between: event class and simulation class, passenger class and bus class, passenger class and bus class, and route class and stop class. Id attribute was added to stop class and event class. Bus class was added a method to display bus information.

Discrete Simulation Class:
Simulation class was given private variables to hold the values of the efficiency coefficients as well as public methods to update these values. Simulation class was also given getCurrentEfficiency function in order to use its list of stops and buses in conjunction with its efficiency coefficients to calculate the efficiency. Simulation was also given methods to update a bus of a given id parameters in correspondence to the Bus Changes requirement. Simulation class now contains a list of up to 3 RevertEvents to provide for replay functionality.

Stop Class:
Stop class was updated with a number of fields to hold the values required for the passenger exchange, Specifically, ridersArriveHigh, ridersArriveLow, ridersOnHigh, ridersOnLow, ridersDepartHigh, ridersDepartLow, and transferRiders we added to facilitate the passenger interchange steps. Correspondingly methods were added to facilitate the steps of the passenger exchange. Important to note that the simulatePassengerArrival/Departure functions take no parameters as the high/low variables are already class parameters. RevertStopState method was added for RevertMoveBusEvent to call to provide the replay functionality asked for. getWaitingPassangers method added for efficiency requirement, so that Simulation class can retrieve number waiting passengers for its calculation.

Bus Class:
Bus class was given riders off high and low parameters and add/remove passenger methods in order to facilitate the passenger exchange requirement. getBusCost method added for simulation class to call to facilitate the system efficiency requirement. To facilitate the bus changes requirements the updateRoute, updateCapacity, and updateSpeed methods were added to the bus class. moveToNextStop method was update to return not only the time of the next moveBusEvent, but also return a RevertMoveBusEvent that would undo the executed event, in order to provide replay functionality.

MoveBusEvent Class:
MoveBusEvent class was updated to return a Pair class of types MoveBusEvent and RevertMoveBusEvent in order to provide replay functionality.

Pair Template Class:
In order to provide replay functionality executed MoveBusEvents needed to return two types, a subsequent MoveBusEvent and a RevertMoveBusEvent that could undo the just executed event. A template class was added to return the two associated events.

RevertEvent Interface Class:
The Event class was implemented as an interface in order to allow for extension in the types of events executed within this system. Correspondingly, the RevertEvent class is implemented as an interface that can be extended to match the Event class.

RevertMoveBusEvent Class:
RevertMoveBusEvent Class is a concrete implementation of the RevertEvent class that contains the state to reset a given Bus and Stop to, in order to undo the changes of a given MoveBusEvent. The RevertMoveBusEvent also contains the MoveBusEvent it is undoing in order to be able to re-add it to the priority queue of events contained by the Discrete Simulation.

Note: Passenger and Route class along with the Event interface had no significant modifications.