**BusInfo Contracts**

|  |  |
| --- | --- |
| Function | addNewBus(numberPlate : String, seatingCapacity: Integer, standingCapacity: Integer, inService : Boolean, model : String, year : Integer) |
| Pre-condition | * The bus being added is not present on the system already. * A physical bus exists that needs to be added. |
| Post-condition | * b, a new ‘Bus’ object is created * b.busID = new\_ID (a new unused bus ID is generated and assigned) * b.numberPlate = numberPlate (b.numberPlate assigned from parameters) * b.seatingCapacity = seatingCapacity (b.seatingCapacity reassigned from parameters) * b.standingCapacity = standingCapacity * b.year = year * b.model = model (b.model assigned from parameters) * b.inService = inService (b.inService assigned from parameters) * b.doingJourney = false (bus is not doing journey) * b.lastCalledStop = null (bus has not called before) * b.lastCalledTime = null * Null route is assigned by default * Null driver is assigned by default * Null journey is assigned by default * b is added to list of Buses on the system (BusInfo) via association ‘HasBuses’ |

|  |  |
| --- | --- |
| Function | modifyBus(busID : busID, inService : Boolean, numberPlate : String, seatingCapacity: Integer, standingCapacity: Integer, model : String, year : Integer) |
| Pre-condition | * The bus object being edited must be present on the system. * The bus ID must have an existing corresponding bus object on the system. * The bus must not be currently doing a journey. |
| Post-condition | * b, bus object with corresponding busID is found * b.numberPlate = numberPlate (b.numberPlate reassigned from parameters) * b.seatingCapacity = seatingCapacity (b.seatingCapacity reassigned from parameters) * b.standingCapacity = standingCapacity * b.year = year * b.model = model (b.model reassigned from parameters) * b.inService = inService (b.inService reassigned from parameters) * b.doingJourney, b.lastCalledStop, b.lastCalledTime, b.route, b.driver, b.journey all remain the same. |

|  |  |
| --- | --- |
| Function | viewJourneysOnRoute(routeID : routeID) : Journey [1..\*] |
| Pre-condition | * The route ID must have a corresponding route object on the system. * The route object must have at least one journey running on it (stored in Journey list) |
| Post-condition | * R, route object with corresponding routeID is found * J [], journeys list from route object R is gotten * J [] journey list is returned |

|  |  |
| --- | --- |
| Function | viewRouteTimetable(routeID : RouteID) : Timetable |
| Pre-condition | * The routeID must have a corresponding route object on the system. * A timetable object for the route must exist. |
| Post-condition | * R, route object with corresponding RouteID is found. * T, timetable on route R is gotten. * Timetable object T is returned. |

|  |  |
| --- | --- |
| Function | getBusState(busID : BusID) : Bus |
| Pre-condition | * The bus ID passed must have a corresponding Bus object present on the system. |
| Post-condition | * B, bus object with an ID the same as busID is found in Bus[] list in BusInfo * Bus object B is returned |

|  |  |
| --- | --- |
| Function | cancelJourney(journeyID : Integer) |
| Pre-condition | * The journeyID passed must have a corresponding journey object present on the system. * The journey must currently be running and present in journeys list in route. * There must be a bus object assigned to doing that journey. |
| Post-condition | * J, journey object with corresponding journeyID is gotten. * B, bus object linked to journey J is gotten. * B.doingJourney = false (Bus is no longer doing journey) * B.journey = null (Remove canceled journey from bus) * Businfo.journeys.remove(J) (Journey J is removed from journeys list in BusInfo system) * J = null (Journey assigned null to cause destruction via garbage collection after all references to the object have been removed) |

|  |  |
| --- | --- |
| Function | assignDriverToBus(driverID : Integer, busID : BusID) |
| Pre-condition | * The driverID passed must have a corresponding Driver object present on the system. * The busID passed must have a corresponding Bus object present on the system. * The driver must not currently be driving another bus. * The bus must be in service. * The bus must not currently be doing a journey. |
| Post-condition | * D, driver object corresponding driverID is gotten. * B, bus object corresponding to busID is gotten. * D.bus = B (Bus assigned to driver via association ‘HasBus’) * B.driver = D (Driver assigned to bus via association ‘HasBus’) |

|  |  |
| --- | --- |
| Function | addNewDriver(name : String, phoneNumber : String, address : Address) |
| Pre-condition | * The driver being created must not be the same as another driver on the system. * There must exist a physical driver that needs to be added. |
| Post-condition | * D, driver object is created. * D.driverID = new\_ID (New unused ID is generated and assigned to the driver) * D.name = name (Name and other values assigned from passed parameters) * D.phoneNumber = phoneNumber * D.address = address * D.bus = null (Null bus assigned by default) * Driver D added to Driver list in system (‘BusInfo’) via association ‘HasDrivers’ |

|  |  |
| --- | --- |
| Function | setInOrOutOfService(busID : BusID, inService : Boolean) |
| Pre-condition | * The busID must correspond to an existing Bus object in the system. * The bus must not be doing a journey. |
| Post-condition | * B, bus with ID of busID is gotten from buses list in (‘BusInfo’) * B.inService = inService (Bus service status is set to passed boolean) |

|  |  |
| --- | --- |
| Function | assignBusToRoute(busID : BusID, routeID : RouteID) |
| Pre-condition | * The busID must correspond to an existing Bus object in the system. * The routeID must correspond to an existing Route object in the system. * The bus must not be doing a journey during route reassignment |
| Post-condition | * B, bus with ID of busID is gotten from buses list in (‘BusInfo’) * R, route with routeID is gotten from routes list in (‘BusInfo’) * B.route = R (Bus’ route assigned as route R) * R.bus = B (Route assigned as bus B) |