**sig** **In**teger {}

**sig** Str**in**gs {}

**sig** Date {}

**sig** Time {}

**sig** Co**or**d**in**ate{}

**enum** DriverStatus {Busy, Available}

**enum** RideStatus{Annulled, Assigned, Completed, **Not**Assigned}

**abstract** **sig** User {}

**sig** Guest **extends** User {}

**sig** Adm**in** **extends** User{

name: **one** Str**in**gs,

surname: **one** Str**in**gs,

email: **one** Str**in**gs,

passw**or**d: **one** Str**in**gs

}

**abstract** **sig** RegUser **extends** User{

name: **one** Str**in**gs,

surname: **one** Str**in**gs,

passw**or**d: **one** Str**in**gs,

birth: **one** Date,

email: **one** Str**in**gs,

teleph**one**: **one** Str**in**gs

}

**sig** Customer **extends** RegUser{}

**sig** TaxiDriver **extends** RegUser{

status: **one** DriverStatus,

taxi: **one** Taxi

}

**sig** Taxi {

code: **one** **In**teger,

position: **one** Co**or**d**in**ate

}

**sig** TaxiQueue {

hasTaxi: set Taxi

}

**sig** TaxiZ**one** {

hasTaxiQueue: **one** TaxiQueue

}

**one** **sig** Map {

hasZ**one**: some TaxiZ**one**

}

**one** **sig** System {

map: **one** Map,

users: set User,

taxiRide: set TaxiRide

}

**abstract** **sig** TaxiRide {

**or**ig**in**: **one** Co**or**d**in**ate,

dest**in**ation: **lone** Co**or**d**in**ate,

date: **one** Date,

time: **one** Time,

wait**in**gTime: **lone** **In**teger,

taxi: **lone** Taxi,

rideStatus: **one** RideStatus,

hasCustomer: **one** Customer

}

**sig** Request **extends** TaxiRide {

}

**sig** Reservation **extends** TaxiRide {

}

//\*\*\*\***Fact**s

//**No** duplicate users

**no**DuplicateUsers{

**no** **disj** u1,u2: RegUser| (u1.email = u2.email) **or** (u1.teleph**one** = u2.teleph**one**)

**no** **disj** a1,a2:Adm**in** | (a1.email = a2.email)

**no** **disj** a1:Adm**in**, u1:RegUser | (a1.email = u1.email)

}

//Every taxi has always exactly **one** driver

taxi**One**Driver{

**no** **disj** d1,d2:TaxiDriver | d1.taxi = d2.taxi

#TaxiDriver = # Taxi

TaxiDriver <: taxi **in** TaxiDriver **one**->Taxi

}

//Every taxi queue has always exactly **one** taxi z**one**

**fact** queue**One**Z**one**{

TaxiZ**one** <: hasTaxiQueue **in** TaxiZ**one** **one** -> TaxiQueue

}

//Every taxi can be **in** only **one** queue

**fact** taxi**One**Queue{

**all** t1:Taxi | **no** **disj** q1,q2:TaxiQueue | (t1 **in** q1.hasTaxi **and** t1 **in** q2.hasTaxi)

**all** t2:Taxi | **lone** q3:TaxiQueue | t2 **in** q3.hasTaxi

}

//the map has **all** the taxi z**one**s

**fact** taxiZ**oneIn**Map{

TaxiZ**one** **in** Map.hasZ**one**

}

//**No** m**or**e than 1 "Assigned" taxiRide f**or** customer

**fact** RideLimit{

**no** **disj** r1,r2:TaxiRide | (r1.rideStatus = r2.rideStatus) **and** (r2.rideStatus = Assigned) **and**

(r1.hasCustomer = r2.hasCustomer)

**no** **disj** r1,r2:Request | (r1.rideStatus = r2.rideStatus) **and** (r2.rideStatus = **Not**Assigned) **and**

(r1.hasCustomer = r2.hasCustomer)

**no** **disj** r1,r2:Request | (r1.rideStatus = Assigned) **and** (r2.rideStatus = **Not**Assigned) **and**

(r1.hasCustomer = r2.hasCustomer)

**no** **disj** r1:Request, r2:Reservation| (r1.rideStatus = **Not**Assigned) **and** (r2.rideStatus = Assigned) **and**

(r1.hasCustomer = r2.hasCustomer)

}

//taxi (**or** taxi driver) has **no** m**or**e than 1 taxiRide with assigned status

**fact** driver**One**Ride{

**no** **disj** r1,r2:TaxiRide | (r1.taxi = r2.taxi) **and** (r1.rideStatus = Assigned) **and** (r2.rideStatus = r1.rideStatus)

}

//**no** taxi paired with a taxi ride with "**not** assigned" status

**fact** **noT**ax**iNot**Assigned{

**all** r1:TaxiRide | (r1.rideStatus = **Not**Assigned) **implies** (r1.taxi = **none**)

}

//Assigned **and** completed rides must be bound to a taxi

**fact** TaxiRideStatus{

**all** r1:TaxiRide | (r1.rideStatus != **Not**Assigned **and** r1.rideStatus!= Annulled) **implies** (#r1.taxi=1)

}

//systems must have the reference to **all** users **and** rides

**fact** systemUserRide{

**all** u1:User | u1 **in** System.users

**all** r1:TaxiRide | r1 **in** System.taxiRide

}

// annulled rides must **not** be l**in**ked to any taxi

**fact** Annulled**NoT**axi{

//**no** r:TaxiRide | r.rideStatus = Annulled **and** r.taxi != **none**

**all** r:TaxiRide | r.rideStatus = Annulled **implies** r.taxi = **none**

}

//busy taxi must **not** be **in** a queue

**fact** taxiBusyQueue{

**all** d1:TaxiDriver | ((d1.status = Busy) **implies** (**no** q1:TaxiQueue | d1.taxi **in** q1.hasTaxi))

// **no** t:Taxi | taxi.t.status **in** Busy **and** #hasTaxi.t >0

//**all** t:Taxi| taxi.t.status **in** Busy **implies** #hasTaxi.t=0

}

//available taxi must be **in** a queue

**fact** taxiAvailableQueue{

**all** d1:TaxiDriver | ((d1.status = Available) **implies** (some q1:TaxiQueue | d1.taxi **in** q1.hasTaxi))

//**all** t:Taxi | taxi.t.status **in** Available **implies** #hasTaxi.t >0

}

//**Or**ig**in** can**not** be equal to dest**in**ation

**fact** **or**ig**in**DifferentDest**in**ation{

**no** r1:TaxiRide | (r1.**or**ig**in** = r1.dest**in**ation)

}

// assigned taxi **implies** busy taxi driver

**fact** assignedEqualBusy{

**no** t:Taxi | taxi.t.rideStatus **in** Assigned **and** taxi.t.status=Available

}

// available taxidriver **implies** **no**assigned taxiRide

**fact** availableEqual**not**Assigned{

**all** t:Taxi,d:TaxiDriver | (d.taxi = t **and** d.status = Available) **implies**(**no** r:TaxiRide | r.taxi = t **and** r.rideStatus = Assigned)

}

//**Assert**ions

//

**assert** numbersEquivalence{

#Taxi = #TaxiDriver

#TaxiZ**one** = #TaxiQueue

#System.users = # User

#System.taxiRide = #TaxiRide

}

//check numbersEquivalence

/\*

//busy taxi drivers may **not** be assigned to a customer

**assert** driversSt**and**ardCustomer{

some d1:TaxiDriver | (d1.status = Busy **and** (**no** r1:TaxiRide | (r1.rideStatus = Assigned **and** r1.taxi = d1.taxi)))

some s1:System, r1:TaxiDriver | addDriver[s1,r1]

}

//check driversSt**and**ardCustomer

\*/

//Other comm**and**s

//Add an assigned ride

**pred** addAssignedRide(s1,s2:System){

**one** r1:TaxiRide | r1.rideStatus = Assigned **and** s2.taxiRide=s1.taxiRide + r1

}

//Add an annulled ride

**pred** addAnnulledRide(s1,s2:System){

**one** r1:TaxiRide | r1.rideStatus = Annulled **and** s2.taxiRide=s1.taxiRide + r1

}

//Add a completed ride

**pred** addCompletedRide(s1,s2:System){

**one** r1:TaxiRide | r1.rideStatus = Completed **and** s2.taxiRide=s1.taxiRide + r1

}

//Add 2 taxi drivers

**pred** add2Driver (s1,s2:System, d1,d2:TaxiDriver){

s2.users= s1.users + d1 + d2

}

//Add 2 customers

**pred** add2Customer(s1,s2:System,c1,c2:Customer){

s2.users=s1.users + c1+c2

}

//Add 2 reservations

**pred** add2Reservation(s1,s2:System, res1,res2:Reservation){

s2.taxiRide=s1.taxiRide + res1 + res2

}

//Add 2 requests

**pred** add2Request(s1,s2:System, req1,req2:Request){

s2.taxiRide=s1.taxiRide + req1 + req2

}

//Add 1 busy **and** 1 available taxi driver

**pred** atleast1busy1available{

some d:TaxiDriver | d.status = Busy

some d:TaxiDriver | d.status = Available

}