module PowEnj

//SIGNATURES

sig Position

{

latitude:Int //should be float

longitude:Int //should be float

}

abstract sig User

{

rentedCar: one Car,

payInfo: some Payment,

position: one Position

}

sig Passenger extends User{}

sig Driver extends User

{

carRented: one Car;

}

fact driverIsUnique

{

//can’t exist two renters who rent the same car at the same time

all r1,r2:Rent | (r1.Renter!=r2.Renter) => r1.rentedCar != r2.rentedCar and r1.startTime!=r2.startTime

}

assert userMustHaveAtLeastOnePaymentInfo{} //toDo

fact applyDiscountForPassNumb

{

//for every trip that involve a number of passengers >=3, apply a discount for the renter

all r:Rent | (#r.passengers>= 3) => r.applyDiscount

}

fact applyDiscountForBattery

{

//if the battery at the end of the trip is at least at 50%, apply a discount

//not sure if it’s possible to use this notation!!

all c:Car, r:Rent | (c.battery>=50) => r.applyDiscount

//alternatively, we can suppose that the attribute “isCharge” is set to a specific value for every kind of discount appliable

all c:Car, r:Rent | (c.isCharge) => r.applyDiscount

}

fact applyDiscountForArea

{

//if the car at the end of trip is let in a service station, apply a discount

all r:Rent, s:ServiceStation | (r.finalPosition=s.position) => r.applyDiscount

}

fact applyOvertax

{

all c:Car, r:Rent, s:ServiceStation | (!(c.isCharge)) => r.applyOvertax or ((s.position – c.position)>r.maxDistance) => r.applyOvertax

}

sig Rent

{

startTime = Int, //should be Float

rentedCar: set Car,

Renter: one User,

applyDiscount: one User,

applyOvertax: one User,

carsAvailable: set Car,

maxDistance: Int //should be float

passengers: set Passenger,

initialPos: Position,

finalPos: Position }

sig Car

{

driver: one Driver,

position:Position

battery: Int,

passengers: set Passenger;

plate: String

}{#passengers>0}

fact plateIsUnique {

all c1,c2: Car | (c1 != c2) => c1.plate != c2.plate

}

Sig Plug{}

abstract sig safeArea {}

sig serviceStation extends safeArea

{

carsToCharge: set Car,

plugAvailable: set Plug,

position:Position

}

sig Payment{}

fact userIsUnique {

all u1,u2: User | u1 != u2 => u1.email != u2.email

}

fact pathDriverHasAStartAndEnd

{

all r:Rent | (r.RentedCar) => r.InitialPosition and r.finalPosition

}