Design a car rental system

User can

1. log in

2. search car based on parameters (carType, money)

3. rent a car

4. cancel renting

-------------------

Class: User

States: userId, name, phoneNumber, loginPassword

Action:

login(userId, loginPassword){

}

searchCars(SearchParams){

List<Cars> cars = CompanyService.search(searchParams);

}

Action:

RentCar(){

RentingDetails = CompanyService.rentCar(carType, pickupTime, returnTime)

}

Action:

CancelRent(){

CompanyService.cancelRent(bookingId)

}

-------------------------

Class: CarCompany

States: carPrice, carName, carType, totalCarNumber

Action:

checkAvailbility(pickupTime, carType){

//check availability of a specific cartype at pickuptime

}

Class: Car

States: name, type

--------------------------

Class: CompanyService

Action:

Recommend(carType){

}

search(searchParams){

list<Car>availableCars

list<Car>cars

for(i=0; i<cars.size; i++){

boolean isAvailable = true;

while(searchParams.carType.number !=0){

if(!cars.get(i).checkAvailability(searchParams.pickupTime)){

isAvailable = false;

break;

}

CompanyService.Recommend(carType)

}

if(isAvailable && cars.get(i).carType == searchParams.carType

&& cars.get(i).price <= searchParams.maxPrice

&& cars.get(i).price >= searchParams.minPrice

&& cars.ger(i).name == searchParams.name){

availableCars.add(cars.get(i));

}

}

Return availableCars;

}

Action:

rentCar(carName, pickupTime, returnTime){

Car car:

Int numberOfDays;

Int price= carPrice \* numberOfDays

paymentService.makePayment(userId, price)

return bookingDetails, bookingId;

}

Action:

cancelBooking(bookingId){

paymentService.refund(userId, price)

}