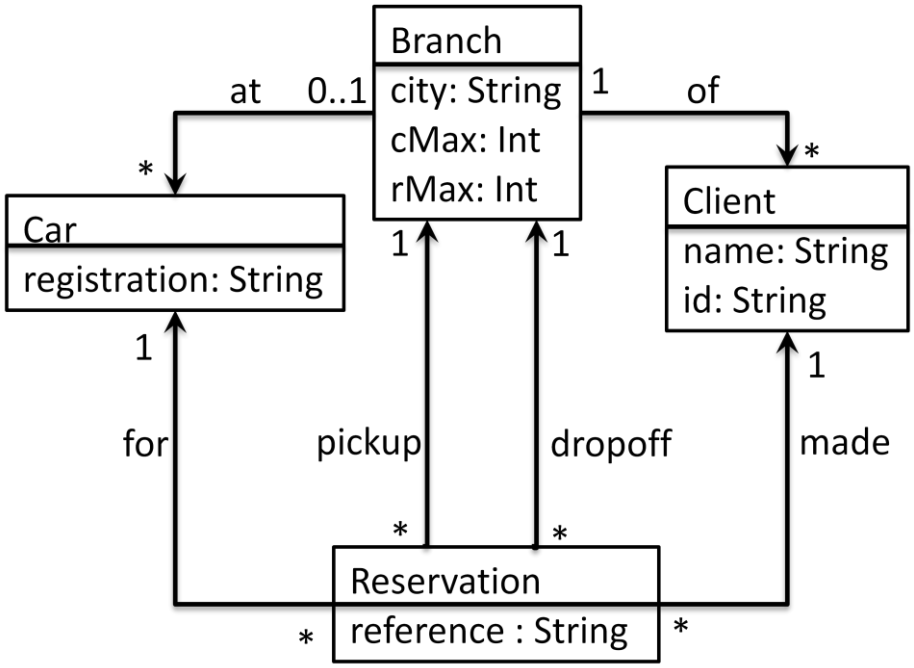
Service Specification

The specification below describing the interface of a car rental agency service consists of a class diagram modelling the available data, a list of operation signatures and a informal description of the preconditions and effects of those operations.

Data Model:



Operation Signatures:

- registerClient(city: String, client: String): String
- makeReservation(client: String, pickup: String, dropoff: String): String
- pickupCar(reference: String)
- dropoffCar(reference: String)
- cancelReservation(reference: String)

and for the queries:

- showClientReservations(client: String): Reservation[]
- showCars(reservation: String): Car[]
- showBranch(city: String): Branch
- showClients(city: String): Client[]

Specification of operations

`String` **`registerClient`** (`String city, String client`)

Creates new *client object* for client and registers it with the branch at *city*. The attribute *branch.cMax* will be increased for each new client added.

Parameters:

`city` – non-null string value used to get branch object by city name.
`client` – non-null string value used to set client name

Returns:

`String` – if the client is registered successfully with the branch, client id of the form city + "_" + Branch.cMax, *`null`* otherwise.

`String` **`makeReservation`** (`String client, String pickup, String dropoff`)

Creates new reservation object for a client that must be registered with *pickup* branch. The *pickup* branch must have at least one Car available to be booked. The attribute *branch.rMax* will be increased by 1 for each new reservation.

Parameters:

`client` – non-null string value used to get client object by name.
`pickup` – non-null string value used to get branch object by city name
`dropoff` – non-null string value used to get branch object by city name.

Returns:

`String` – if the reservation object is created successfully, reservation reference of the form city + "_" + Branch.rMax, *`null`* otherwise.

`Void` **`pickupCar`** (`String reference`)

Removes linkes *pickup* and *for* between reservation object and *pickedup* branch. The reserved *car* can only be picked up once. If there is no suitable reservation, the operation does not have an effect on the state.

Parameters:

`reference` – non-null string value used to get *reservation* object by reference.

Returns:

`no return`

`Void` **`dropoffCar`** (`String reference`)

Creates new link *at* by returning reserved car to the *dropoff* branch, and removes reservation object with all its links, namely: *made*, *pickup*, *dropoff* and *for*. If there is no suitable reservation, the operation does not have an effect on the state.

Parameters:

`reference` – non-null string value used to get *reservation* object by reference.

Returns:

`no return`

`Void` **`cancelReservation`** (`String reference`)

Removes reservation object that matches *`reference`* (if it exists) with all its links, namely: *made*, *pickup*, *dropoff* and *for*. If there is no suitable reservation, the operation does not have an effect on the state.

Parameters:

`reference` – non-null string value used to get *reservation* object by reference.

Returns:

`no return`

```
public class Rental implements IRental{

	private static final long serialVersionUID = 6324598725198583458L;
	...
	public String registerClient(String city, String clientName){

		Branch cBranch = getBranch(city);
		if (cBranch !=null){

			Client newClient = new Client();
			newClient.name =clientName;
			newClient.id = cBranch.city + "_" +  (cBranch.of.size());

			cBranch.of.add(newClient);
			return newClient.id;
		}
		return null;
	}

	public String makeReservation(String ClientID, String pickup, String dropoff){

		Branch pickupBranch = getBranch(pickup);
		Branch dropOffBranch = getBranch(dropoff);

		Client clientMade = getClient(pickupBranch, ClientID);

		if (clientMade==null){
			clientMade = getClient(dropOffBranch, ClientID);
		}

		Car car = getCar(pickupBranch);

		if (pickupBranch==null
				|| dropOffBranch==null
				|| clientMade==null
				|| car==null){
			return null;
		}

		pickupBranch.rMax++;
		Reservation mReservation = new Reservation(
				pickupBranch.city + "_" + pickupBranch.rMax,
				clientMade,
				pickupBranch,
				dropOffBranch,
				car);

		this.reservations.add(mReservation);
		return mReservation.reference;
	}

	public void cancelReservation(String Reference){

		for (int iIndex=this.reservations.size()-1; iIndex>=0; iIndex--){

			Reservation readRes= this.reservations.get(iIndex);
			if (!readRes.made.equalsIgnoreCase(Reference)){
				continue;
			}
			else {
				this.reservations.remove(iIndex);
			}
		}
	}
...
```

```
66          public void pickupCar(String Reference){
67
68                  int iIndex = getReservationIndex(Reference);
69
70                  if (iIndex==-1){
71                          return;
72                  }
73
74                  Reservation getReservation = this.reservations.get(iIndex);
75
76                  // check if it hasn't been picked up already
77                  if (getReservation.pickup==null){ return; }
78
79                  // check if the reserved car still exists in the pick-up branch
80                  iIndex=-1;
81
82                  for (int iCarIndex=0; iCarIndex <getReservation.pickup.at.size(); iCarIndex++){
83
84                          if (getReservation.pickup.at.get(iCarIndex)
85                                  .getRegistration().equalsIgnoreCase(
86                                  getReservation.for.getRegistration())){
87
88                                  iIndex=iCarIndex;
89                                  break;
90                          }
91                  }
92
93                  if (iIndex==-1){return; }
94
95                  // remove car from pickup branch
96                  getReservation.pickup.at.remove(iIndex);
97          }
98
99          public void dropoffCar(String Reference){
100
101                 int iIndex = getReservationIndex(Reference);
102
103                 if (iIndex==-1){ return;}
104
105                 Reservation getReservation = this.reservations.get(iIndex);
106
107                 // check if it has been picked up already
108                 if (getReservation.pickup!=null || getReservation.made==null){
109                         return;
110                 }
111
112                 // add car to drop-off branch
113                 getReservation.dropoff.at.add(getReservation.for);
114
115                 getReservation.dropoff=null;
116                 getReservation.for=null;
117         }
118
119 }
120
```