SE SOFTWARE ENGINEERING

RWTH AACHEN

Exercise

Class Diagrams

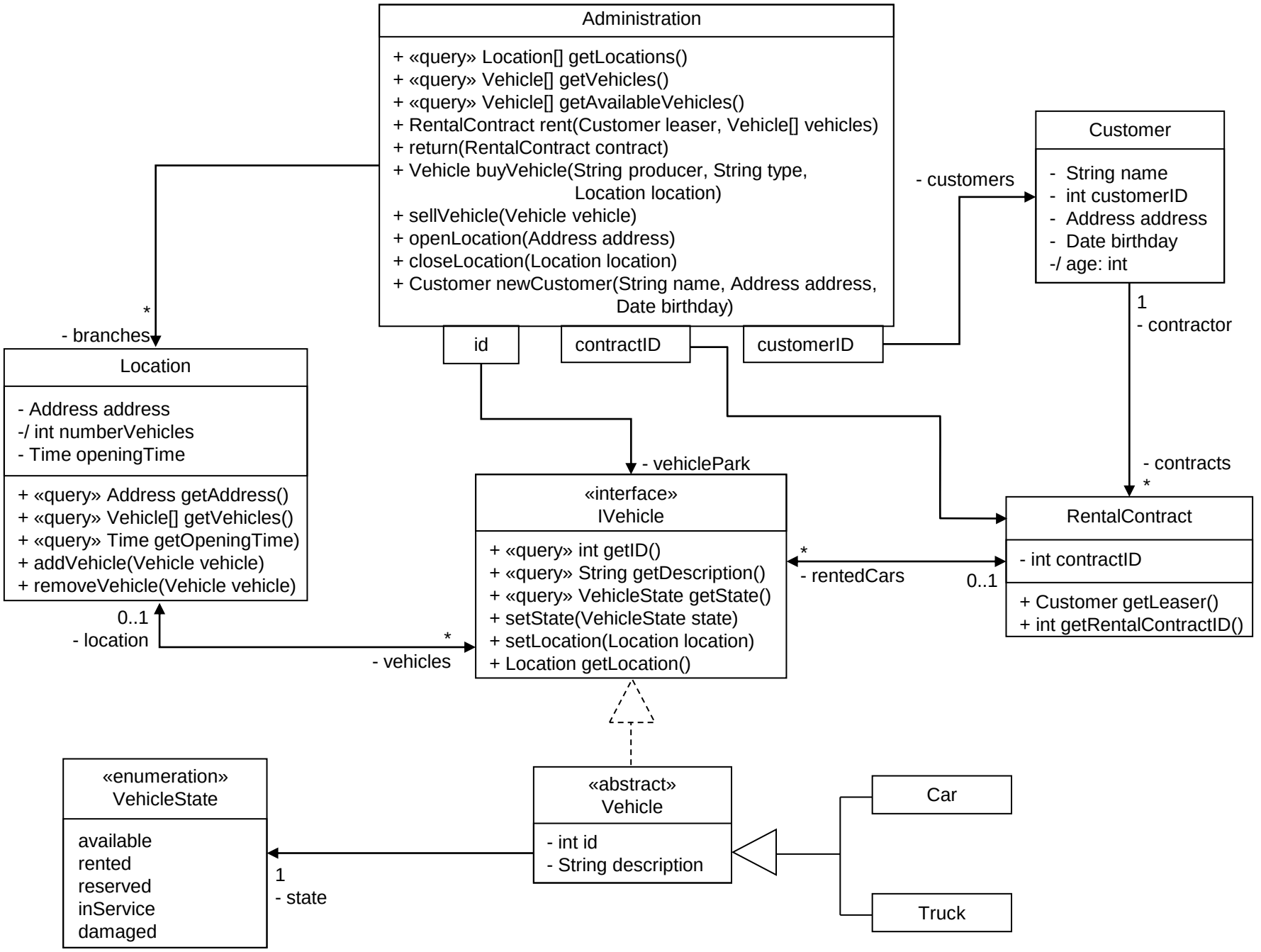
Prof. Dr. Bernhard Rumpe
Lehrstuhl für Software Engineering
RWTH Aachen

http://www.se-rwth.de/

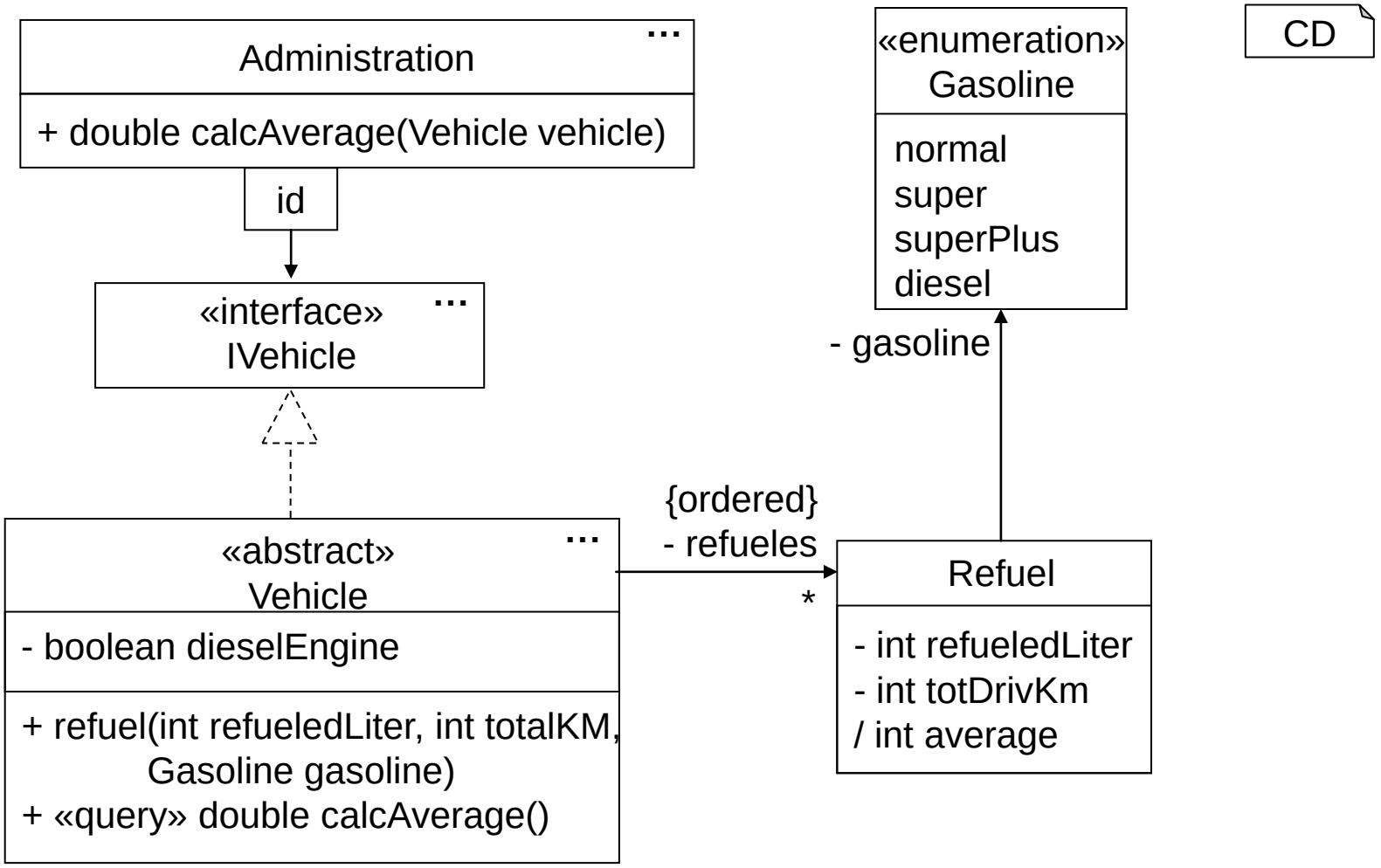
Exercise 9.1

The customer now wants the system to be usable for managing the fuel consumption of his car fleet. It should be possible:

- to add the total driven kilometers together with the number of refueled liter to the system
- for each refuel the kind of gasoline is selected too
  - the following kinds can be selected *normal, super, superplus* and *diesel*
- for every vehicle in the fleet, it is defined weather it is a diesel engine or a gas engine
- the total driven kilometers of a vehicle and every refuel should be stored in the system
  - so it is possible to calculate the average consumption of every vehicle.


Administration
+ «query» Location[] getLocations()
+ «query» Vehicle[] getVehicles()
+ «query» Vehicle[] getAvailableVehicles()
+ RentalContract rent(Customer leaser, Vehicle[] vehicles)
+ return(RentalContract contract)
+ Vehicle buyVehicle(String producer, String type, Location location)
+ sellVehicle(Vehicle vehicle)
+ openLocation(Address address)
+ closeLocation(Location location)
+ Customer newCustomer(String name, Address address, Date birthday)
- customers
Customer
- String name
- int customerID
- Address address
- Date birthday
-/ age: int
1
- contractor
*
- branches
id
contractID
customerID
Location
- Address address
-/ int numberVehicles
- Time openingTime
+ «query» Address getAddress()
+ «query» Vehicle[] getVehicles()
+ «query» Time getOpeningTime)
+ addVehicle(Vehicle vehicle)
+ removeVehicle(Vehicle vehicle)
- vehiclePark
- contracts
*
«interface»
IVehicle
+ «query» int getID()
+ «query» String getDescription()
+ «query» VehicleState getState()
+ setState(VehicleState state)
+ setLocation(Location location)
+ Location getLocation()
*
- rentedCars
0..1
RentalContract
- int contractID
+ Customer getLeaser()
+ int getRentalContractID()
0..1
- location
*
- vehicles
«enumeration»
VehicleState
available
rented
reserved
inService
damaged
1
- state
«abstract»
Vehicle
- int id
- String description
Car
Truck

Solution 7.1


CD
Administration
...
+ double calcAverage(Vehicle vehicle)
id
«interface»
IVehicle
...
«abstract»
Vehicle
...
- boolean dieselEngine
+ refuel(int refueledLiter, int totalKM,
Gasoline gasoline)
+ «query» double calcAverage()
{ordered}
- refueles
*
Refuel
- int refueledLiter
- int totDrivKm
/ int average
- gasoline
«enumeration»
Gasoline
normal
super
superPlus
diesel

Exercise 7.2

1. Specify an OCL-Condition for the new method, which guarantees that after a car is refueled, the total driven kilometers are greater than before the previous refuel. *(The case that the counter of the vehicle has an overflow can be ignored)*

Solution 7.2

1. Specify an OCL-Condition for the new method, which guarantees that after a car is refueled, the total driven kilometers are greater than before the previous refuel. *(The case that the counter of the vehicle has an overflow can be ignored)*

```
context Vehicle.refuel(int refuelLiter, int totalKM,
                       Gasoline gasoline):
  pre: let oldTotDrivKm = refuels.last.totDrivKm
  post: oldTotDrivKm < refuels.last.totDrivKm

context Vehicle.refuel(int refuelLiter, int totalKM,
                       Gasoline gasoline):
  pre: true
  post: refuels@pre.last.totDrivKm
        < refuels.last.totDrivKm
```

Exercise 7.2

2. Cars which run on gasoline can be refueled with normal, super and superplus. But gasoline cars cannot be refueled with diesel. Analogous constraints hold for cars which run on diesel.

Solution 7.2

2. Cars which run on gasoline can be refueled with normal, super and superplus. But gasoline cars cannot be refueled with diesel. Analogous constraints hold for cars which run on diesel.

```
context Vehicle inv:
  if Vehicle.dieselEngine then
    refuels.gasoline.asSet == {Gasoline.diesel}
  else !(refuels.gasoline.contains(Gasoline.diesel))
```