VIA University College

Semester Project 1 (SEP1)
Single User System

Steffen Vissing Andersen (sva@via.dk)

First step of analysis was...

- List of requirements — Numbered, Precise, objective and measurable
- Use case modelling
  - Use case diagram → What you can do in the system
  - Use case descriptions → How you do the 'thing' in the system
    - Customer: ensures correct business flows, scenarios
    - Developer: makes sure that all requirements are covered
    - Programmer: gets a recipe for the steps in the program

Reserve a Vehicle: Use case description

- Step 1: Understand how you reserve a vehicle
  - Are you giving all the info like name, address, phone number, email and more - before you know if you can reserve a car?
  - What could go wrong?
  - Play the roles: make scenarios for Bob trying to reserve a car
- Step 2:
  - Make sure that you follow the steps for reserving a vehicle including all exception flows
  - Make sure that you haven't forgotten any requirements
  - Make a recipe for the programmer to implement the system

Agenda – Analysis (part 2)

- Activity diagrams
- Analysis class diagram

Present a use case description as an activity diagram

Use Case description #1

Use Case: Reserve a vehicle

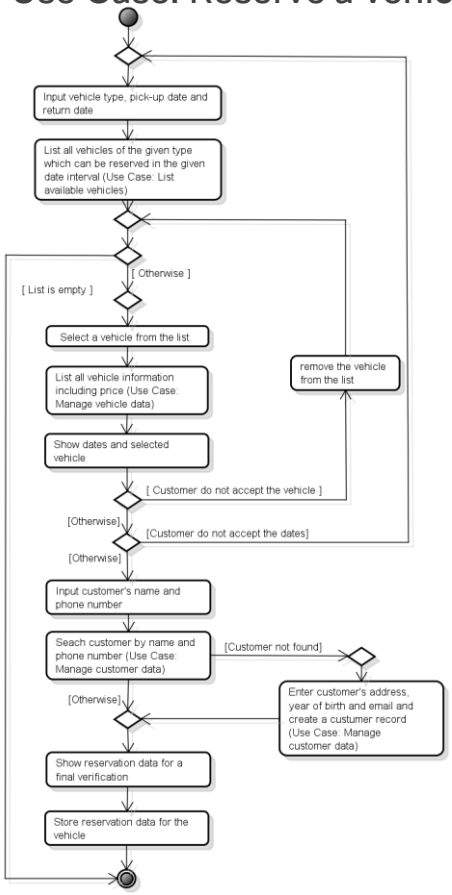
....

1. Enter vehicle type (family car, van, truck or bus) and the two dates; pick-up date and return date
2. System returns a list of available vehicles of the given type in the given date interval (Use Case: List all available vehicles)
3. Select from the list the vehicle to reserve
4. System returns details about the vehicle (Use Case: Manage vehicle data)
5. If vehicle cannot be accepted by the customer then go to step 4 again
6. Verify the dates and vehicle to reserve
7. If dates are not correct then go to step 1
8. Enter name and phone number for the customer
9. System search for the customer by phone number and name (Use case: Manage customer data)

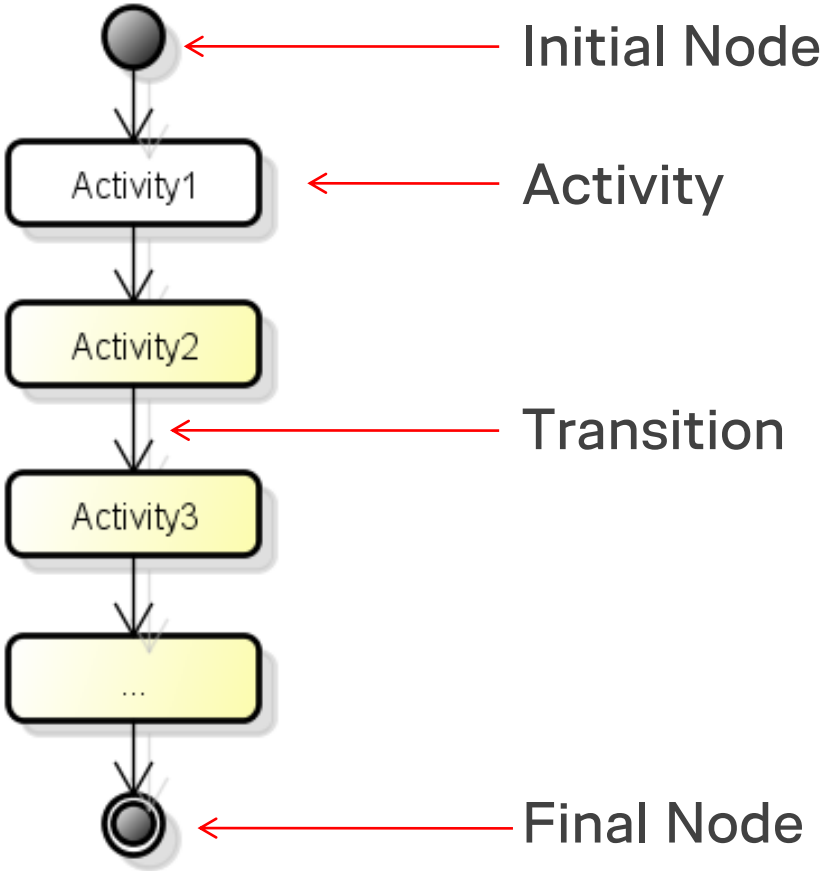
...

Activity diagram #1

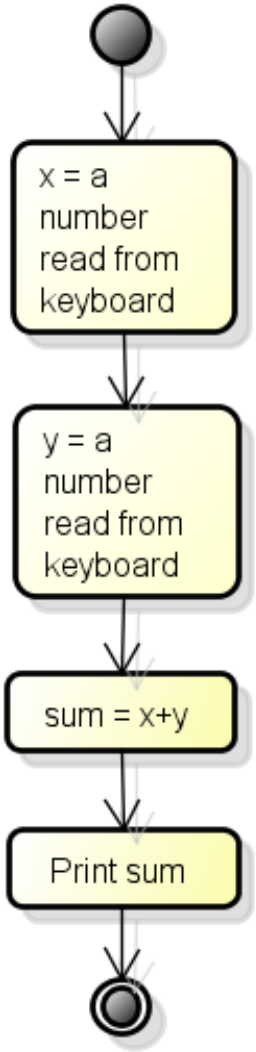
For Use Case: Reserve a vehicle



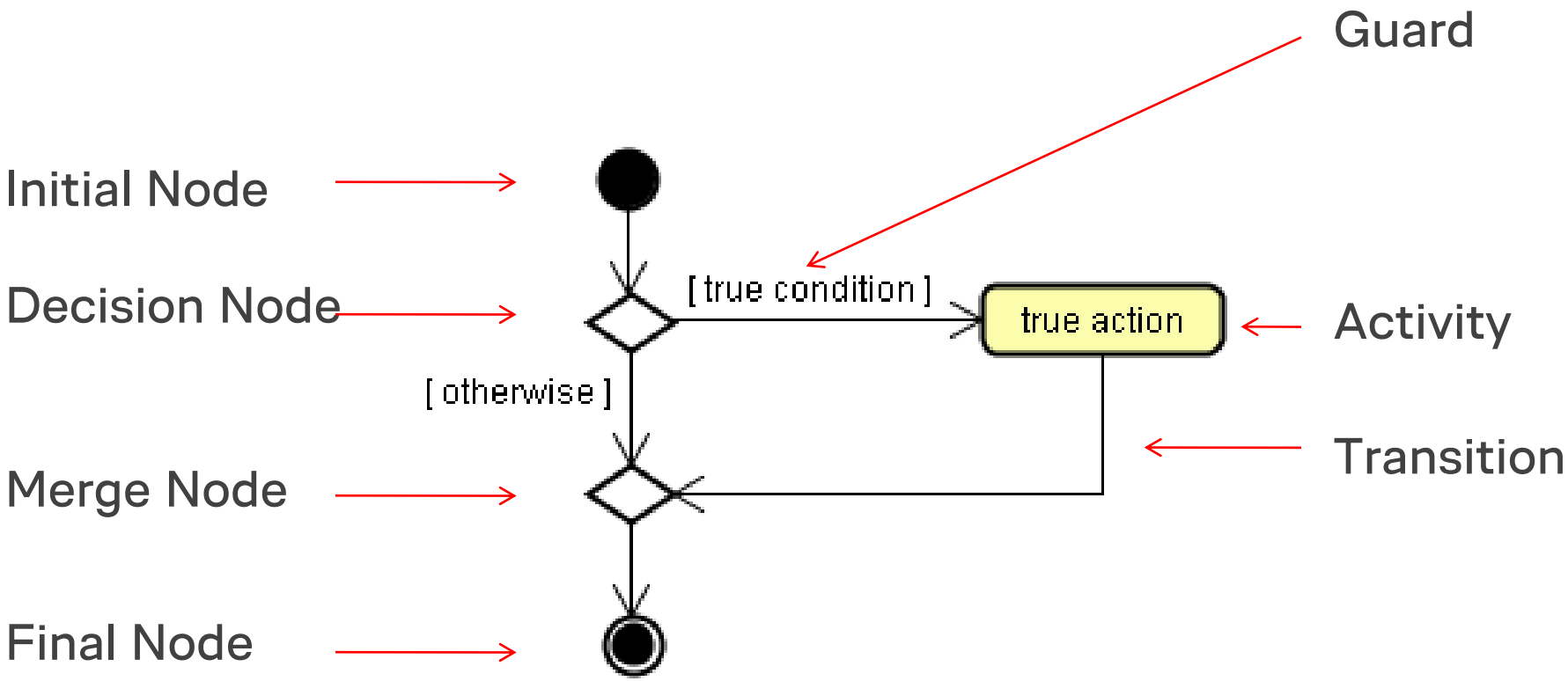
Sequence Structure – UML Activity Diagram



Sequence Structure Example



`If` statement – Activity Diagram



Activity diagrams (dynamic)

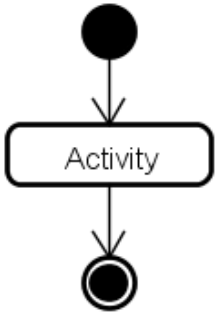
- Activity diagrams are used to:
  - Describe the functionality in a Use Case
  - Model a task (in business modeling for instance)
  - To specify the logic in an operation/method
- Activity diagrams shows:
  - The flow of activities in a process including concurrent activities and branches
  - They are a kind of flowcharts (Gaddis uses flowcharts)

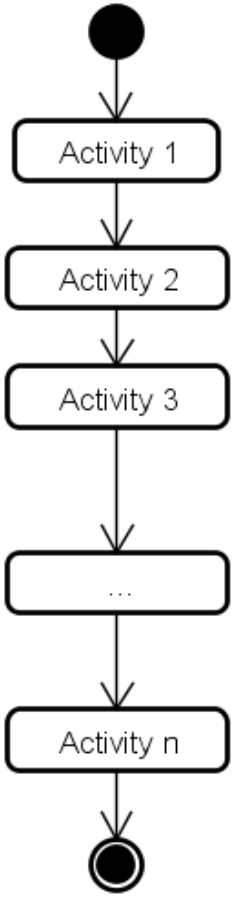
Activity diagrams – “building blocks”

- Any activity diagram can be created to include a combination of the 7 “building blocks”

1. Sequence / single operation
2. if
3. if-else
4. switch
5. while
6. do-while
7. for

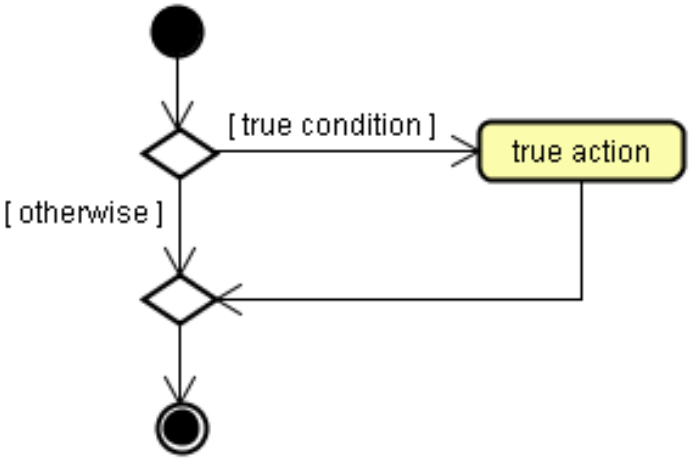
Sequence (building block 1)



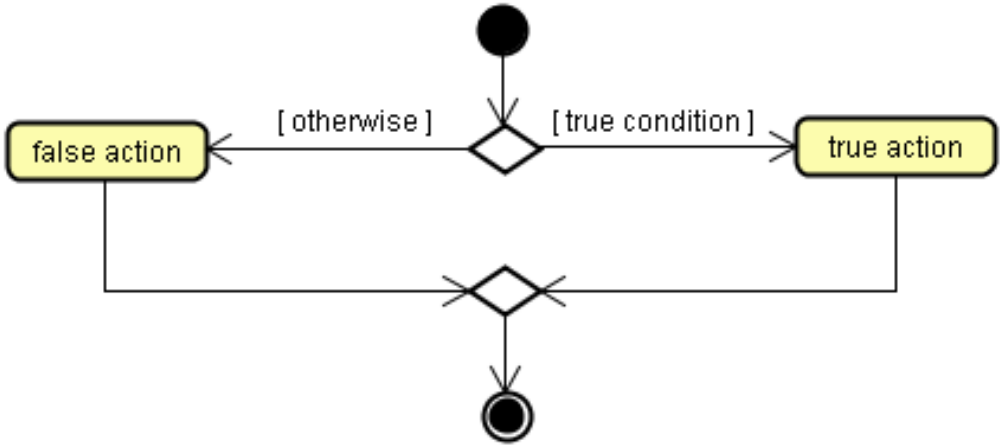


Selection (building block 2-3)

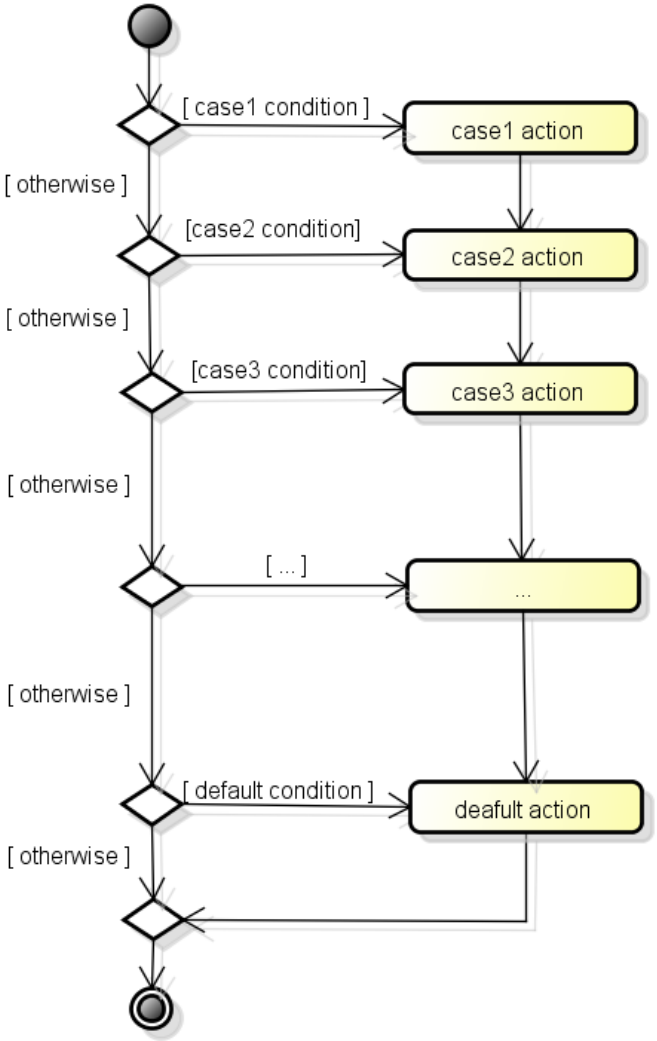
`if`

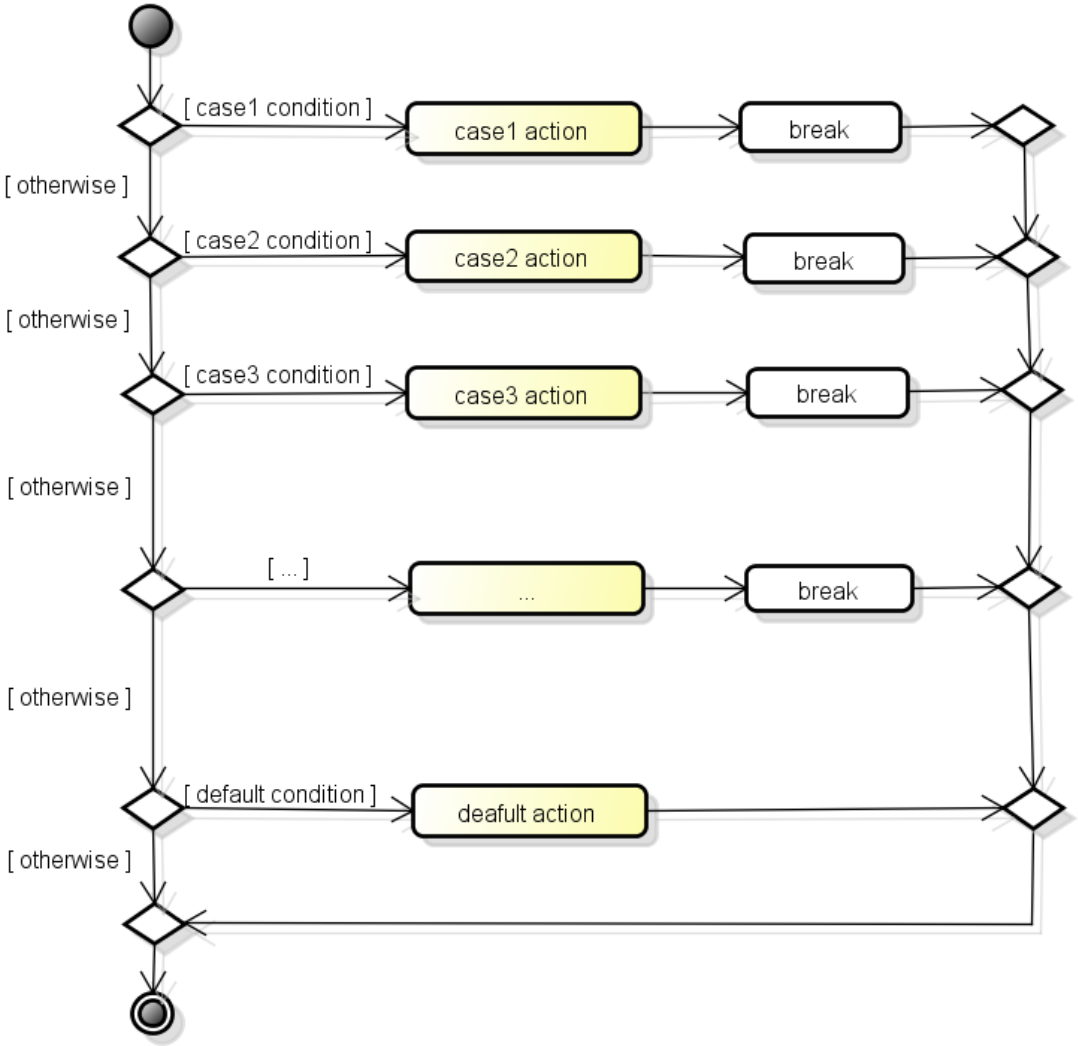


`if...else`



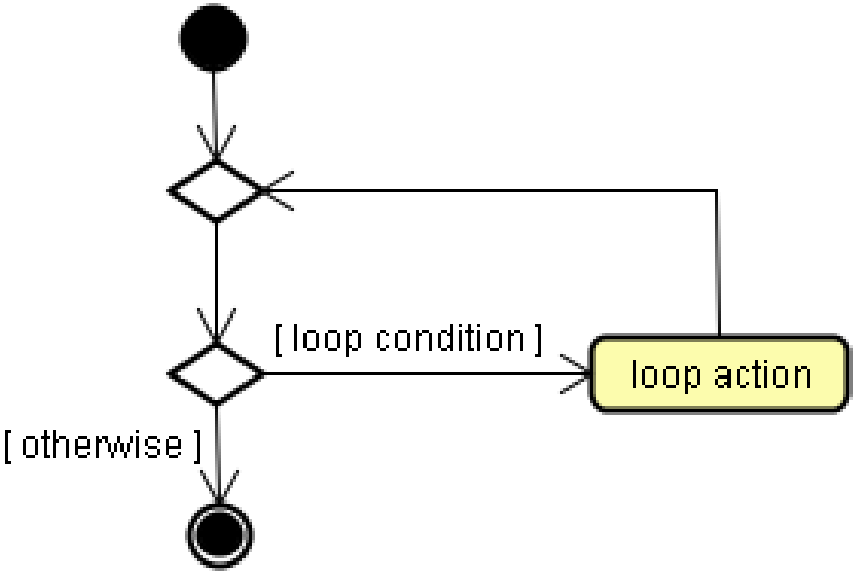
Switch (building block 4)



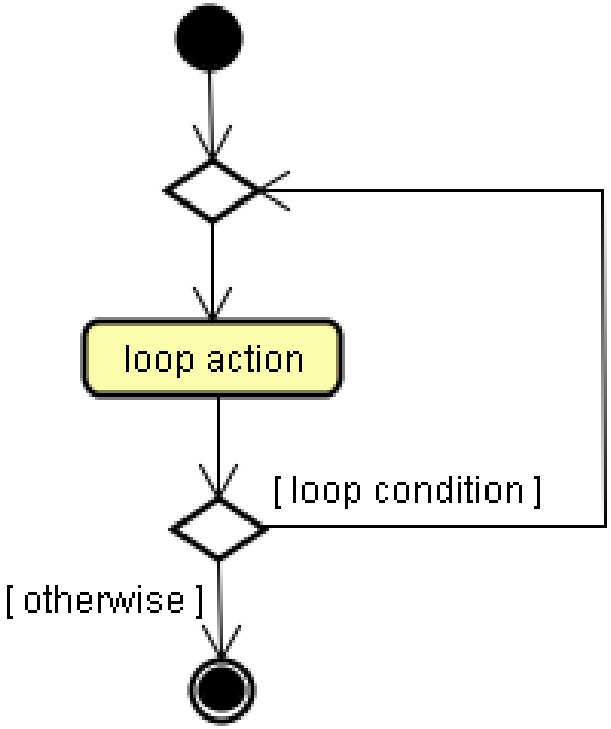


Repetitions / while, do-while (building block 5-6)

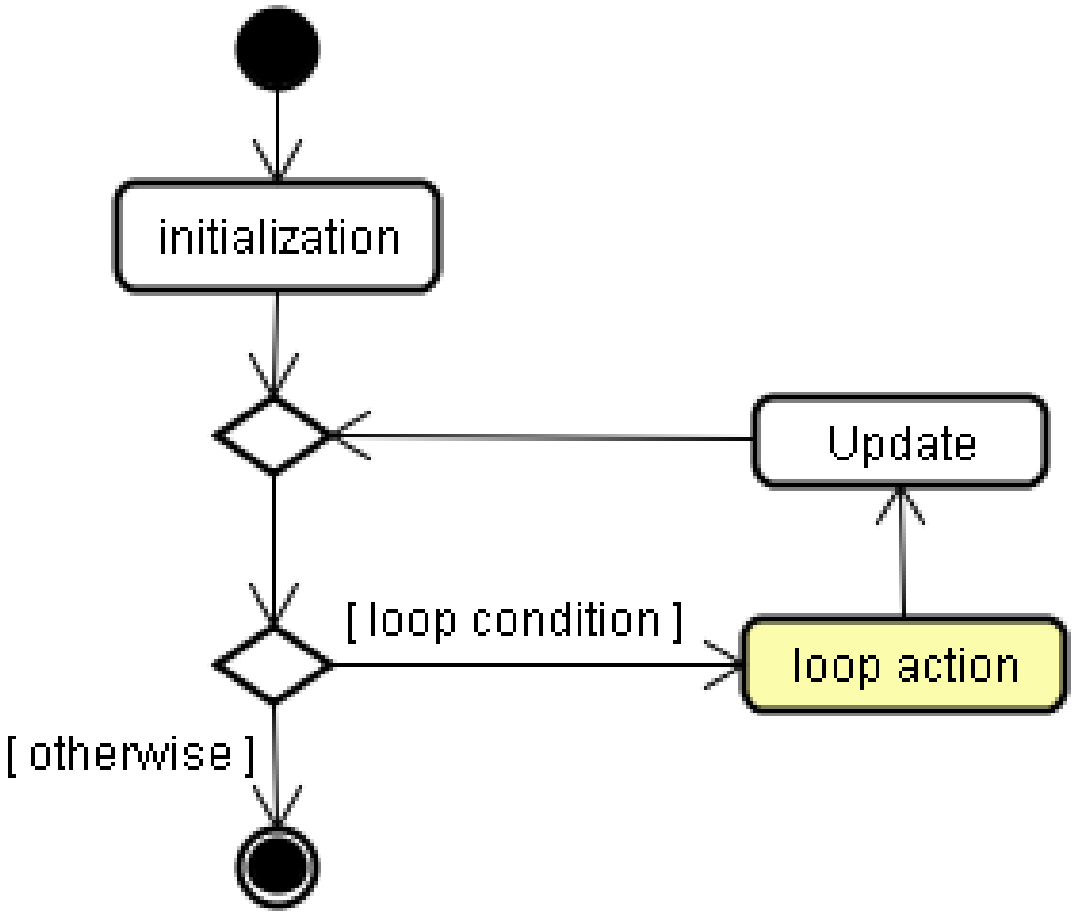
`while`



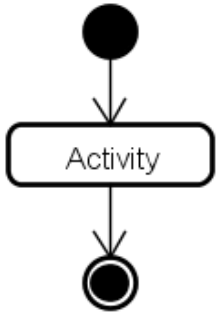
`do...while`



Repetitions / for (building block 7)



Rules for forming structured activity diagrams



1. Begin with the simplest activity diagram
2. Any action state can be replaced by two action states in sequence (stack rule)
3. Any action state can be replaced by any control statement (sequence of action states, if, if…else, switch, while, do…while or for) (nesting rule)
4. Rules 2 and 3 can be applied as often as you like and in any order.

Example: Use Case Description

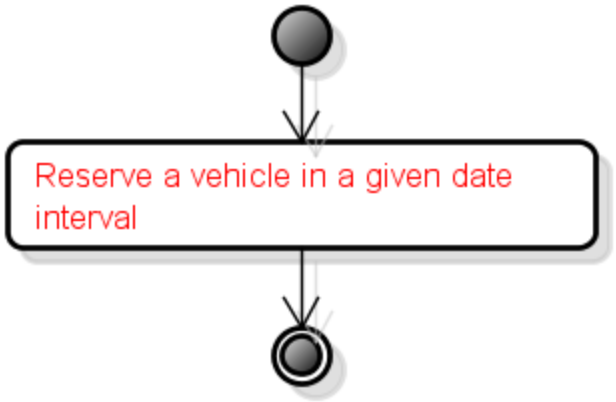
Reserve a vehicle / UseCase Description

ITEM	VALUE
UseCase	Reserve a vehicle
Summary	An employee reserves a vehicle for a customer in a given time period.
Actor	Employee
Precondition	None
Postcondition	A reservation has been made and data for reservation, vehicle and customer has been saved in the system.
Base Sequence	1. Enter vehicle type (family car, van, truck or bus) and the two dates; pick-up date and return date 2. System returns a list of available vehicles of the given type in the given date interval (Use Case: List all available vehicles) 3. Select from the list the vehicle to reserve 4. System returns details about the vehicle (Use Case: Manage vehicle data) 5. If vehicle cannot be accepted by the customer then go to step 4 again 6. Verify the dates and vehicle to reserve 7. If dates are not correct then go to step 1

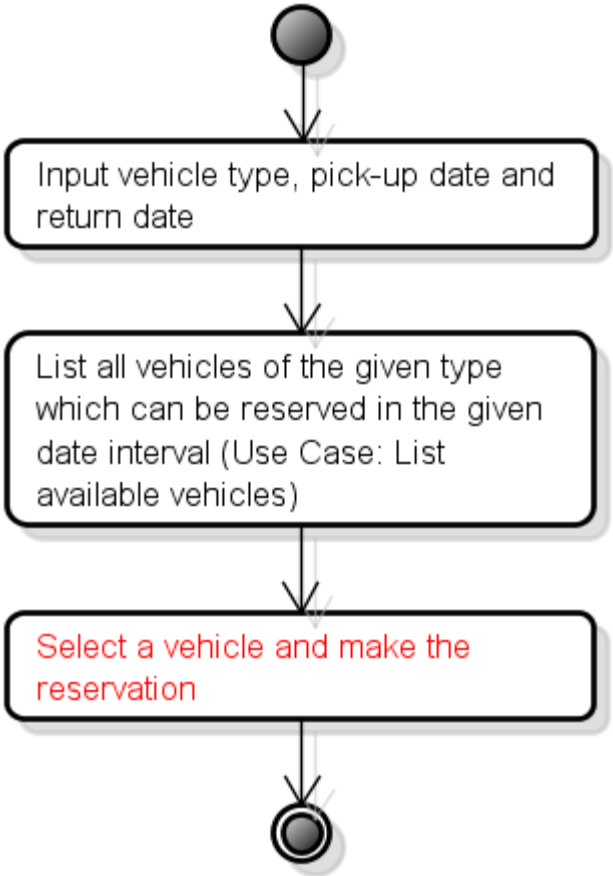
Example: Use Case Description

	8. Enter name and phone number for the customer 9. System seach for the customer by phone number and nam e (Use case: Manage customer data) 10. If customer is not found in the system THEN enter cu stomers address, year of birth and email and create a cu stomer record (Use Case: Manage customer data) 11. Show information of the reservation for a final veri fication 12 Store the reservation data in the system
Branch Sequence	
Exception Sequence	No available vehicles of the given type in the given dat e interval: 1-2 as base sequence The returned list is empty and use case ends. Customer will not reserve one of the available vehicles: 1-5 as base sequence no more vehicles to selct and use case ends.
Sub UseCase	List all available vehicles Manage customer data Manage vehicle data
Note	A reservation can be cancelled any time

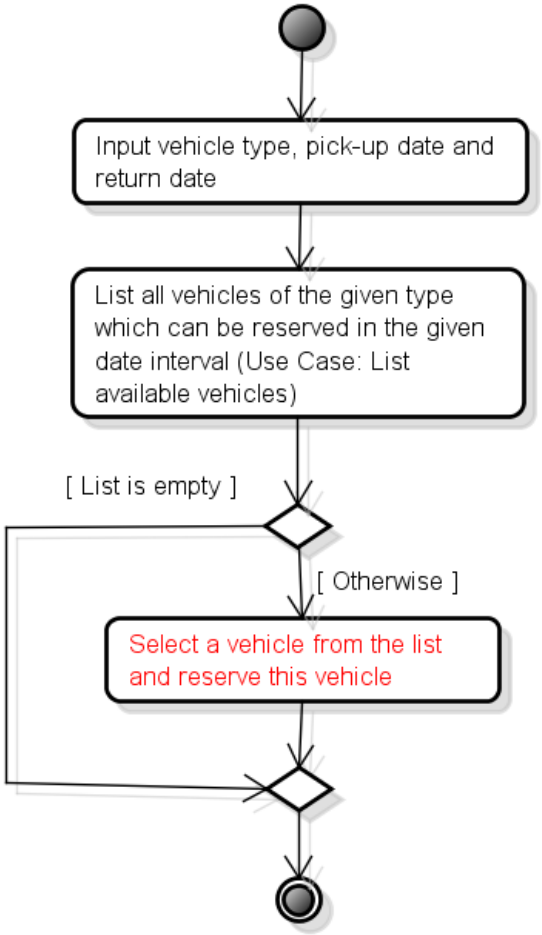
The simplest activity diagram



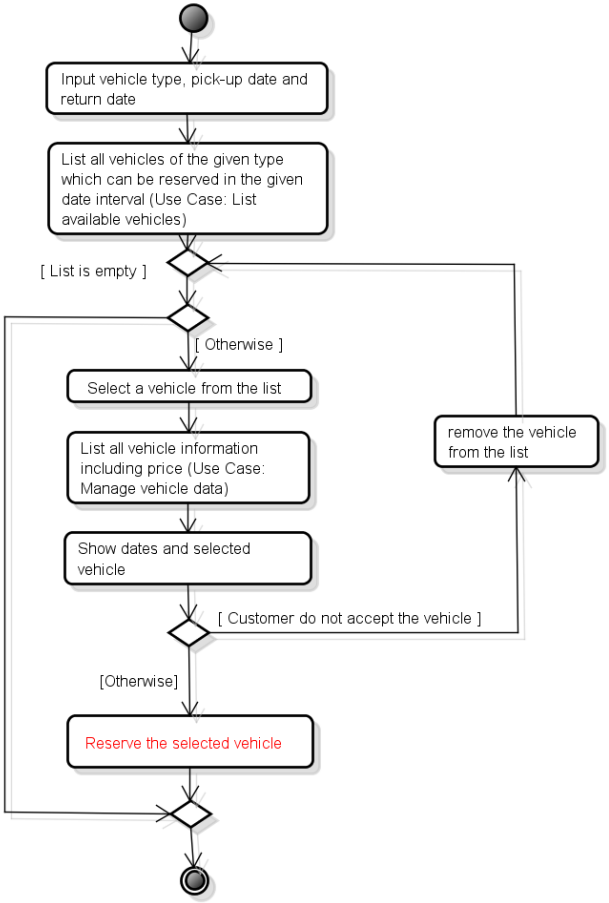
2 stack rules applied



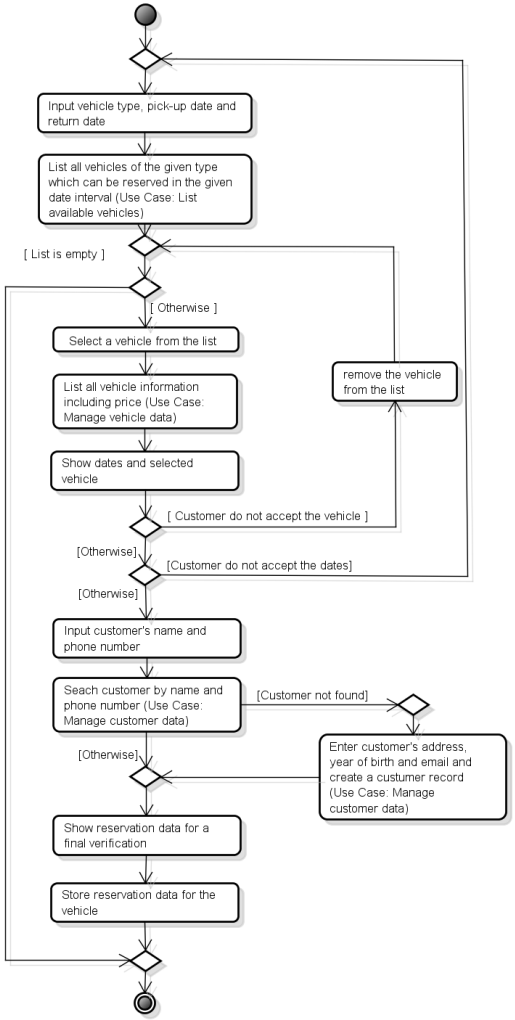
A nested rule applied



Nested rules and a stack rule



And more stack and nested rules



Class diagram (Analysis stage)

Class names are common names

- `Student`
- `Rectangle`
- `Date`
- `DateInterval`

- All classes start with a capital letter
- Each new word starts with a capital letter

Attribute names: common names

- `Student:`
  - `name`
  - `studyNumber`
- `Rectangle:`
  - `width`
  - `height`
- `DateInterval:`
  - `startDate`
  - `endDate`
- All attributes start with a lower-case letter
- Each new word starts with a capital letter

Method names: affirmatives

- `Student:`
  - `getStudyNumber()`
  - `setName(String name)`
- `Date:`
  - `isLeapYear()`
  - `stepForwardOneDay()`

- All method names start with a lower-case letter
- Start each new word with a capital letter

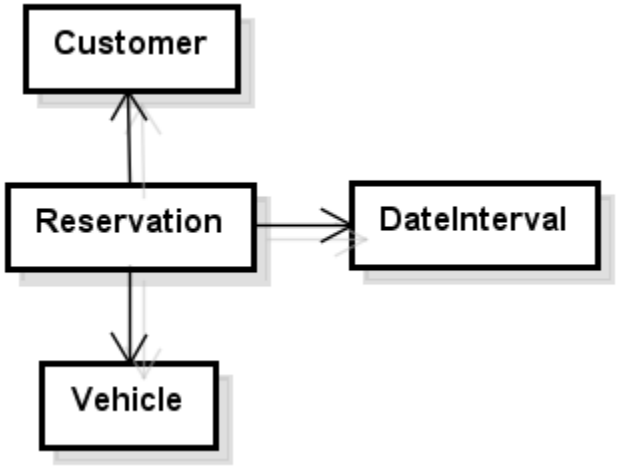
Analysis method

Basis for identifying classes, instance variables and methods:

- Description of the problem / the interview
- List of requirements
  - Includes relevant information from the interview
- Use case diagram and Use case descriptions
  - Includes all requirements

Example: A vehicle rental company

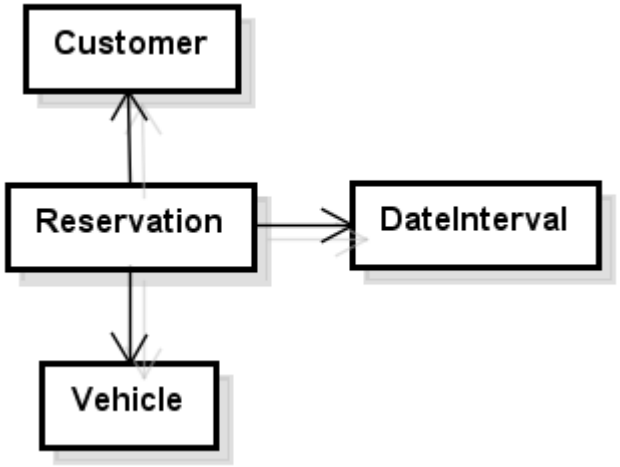
1. A customer can make a reservation of an available vehicle in a given date interval



The company class

1. A customer can make a reservation of an available vehicle in a given date interval

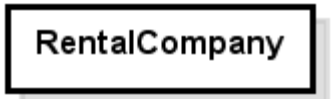
A method: reserveVehicle(Vehicle, Customer, DateInterval)
A method: isAvailable(Vehicle, DateInterval)

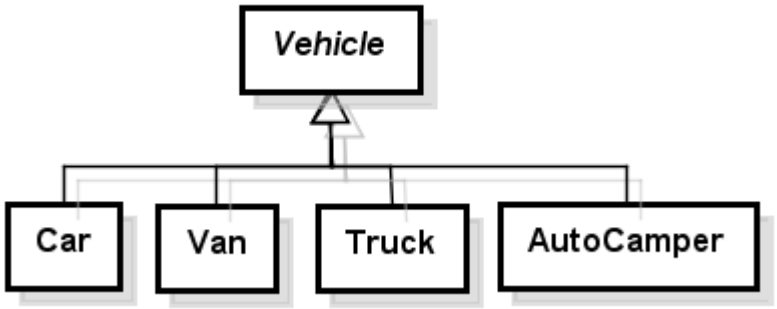


Different kinds of vehicles

2. In the Rental company a customer can rent a car, a van, a truck or an auto camper

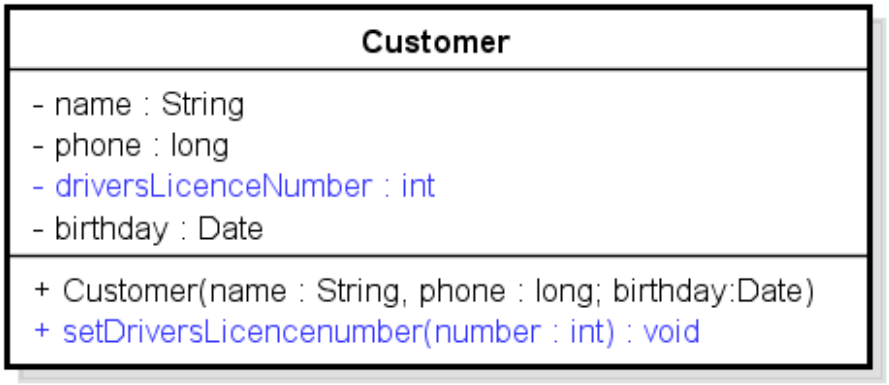
A method: rentVehicle(…)



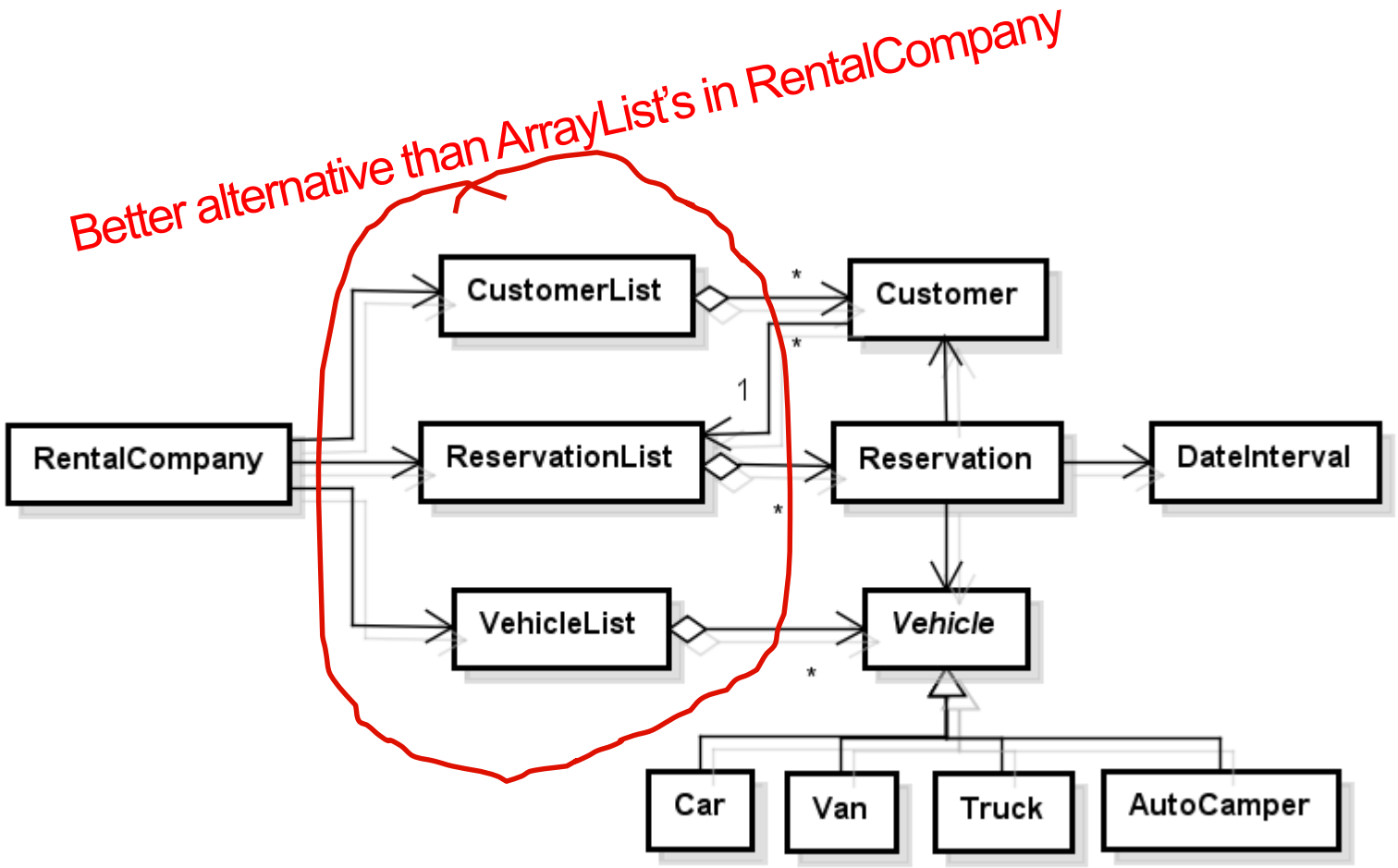


Some instance variables

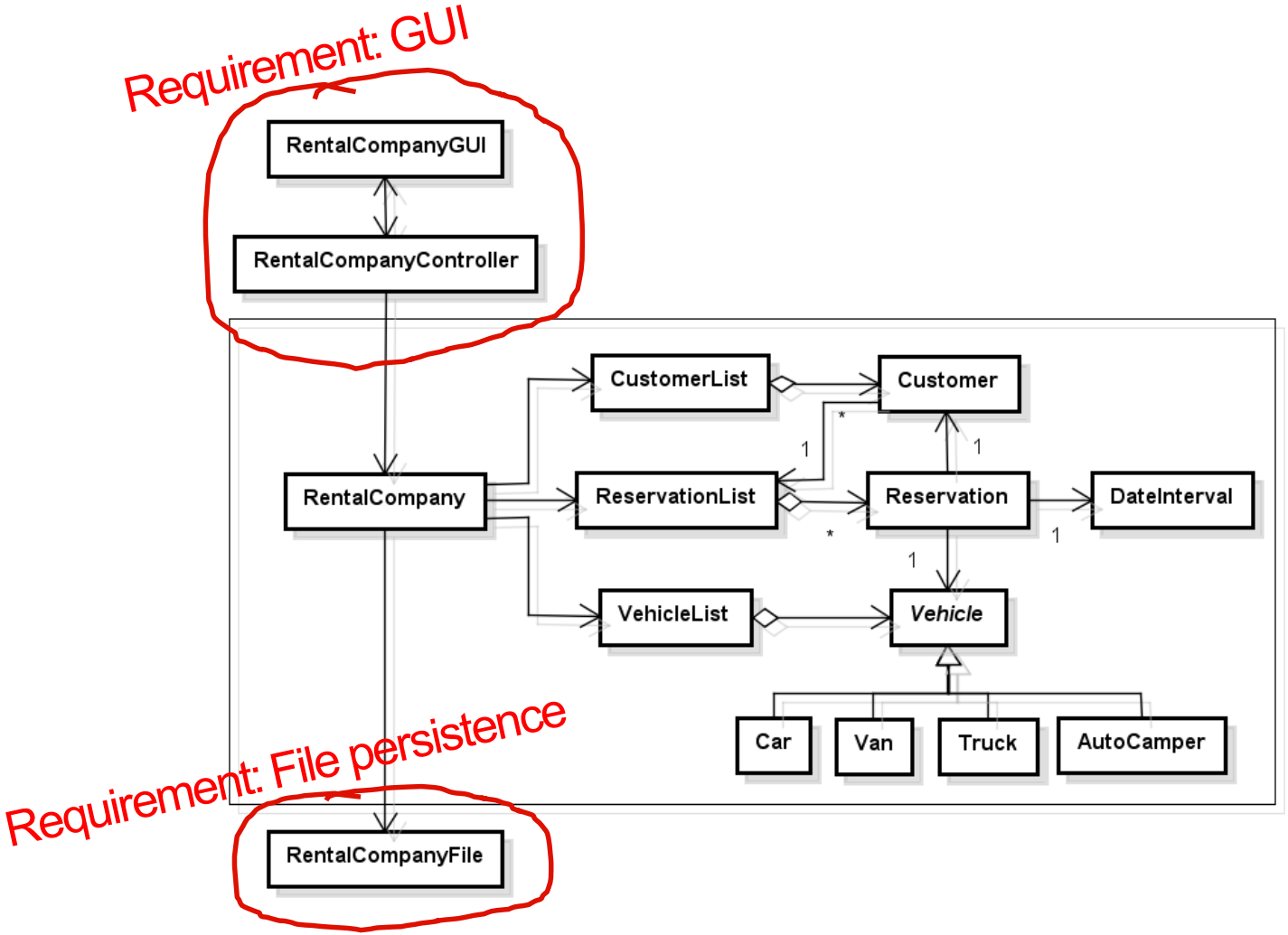
3. When a customer makes a reservation the company stores customers name, phone number and date of birth
4. When a customer rents his/her reserved vehicle, the company needs to store the customers drivers license number



One-to-Many (Design choice)



Classes and relations



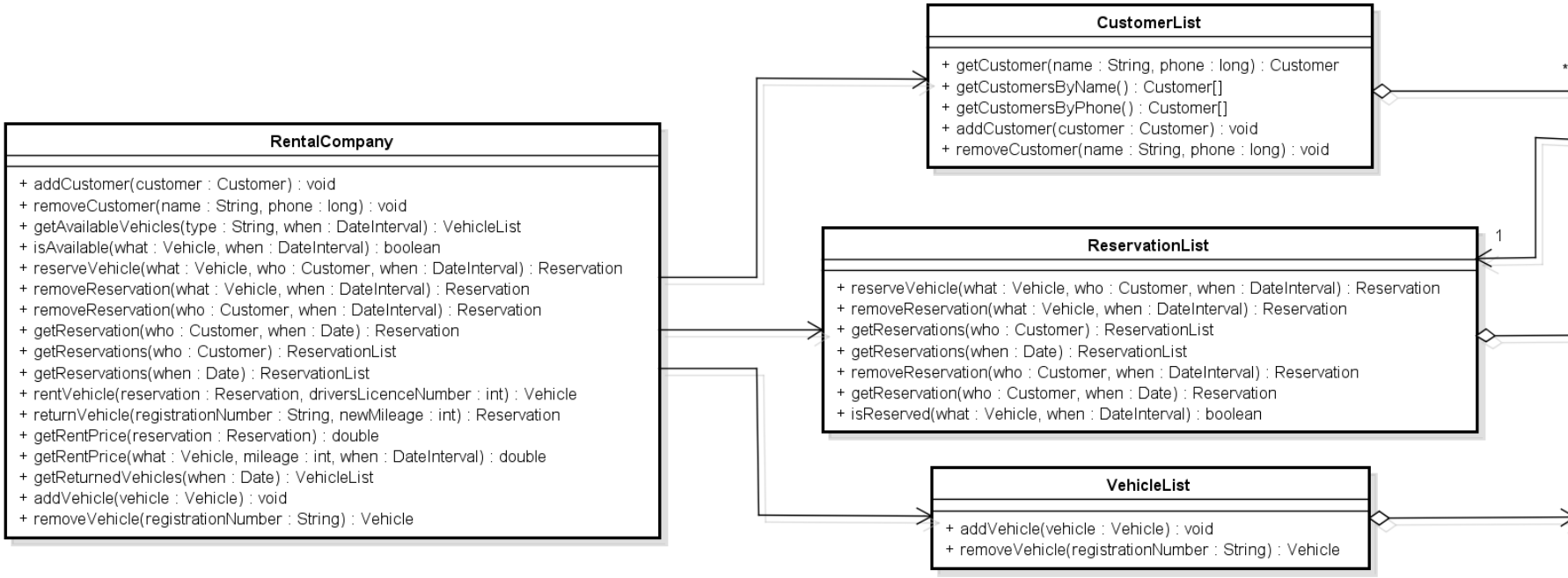
Methods for each use case

Use Cases (or part of a use case description)

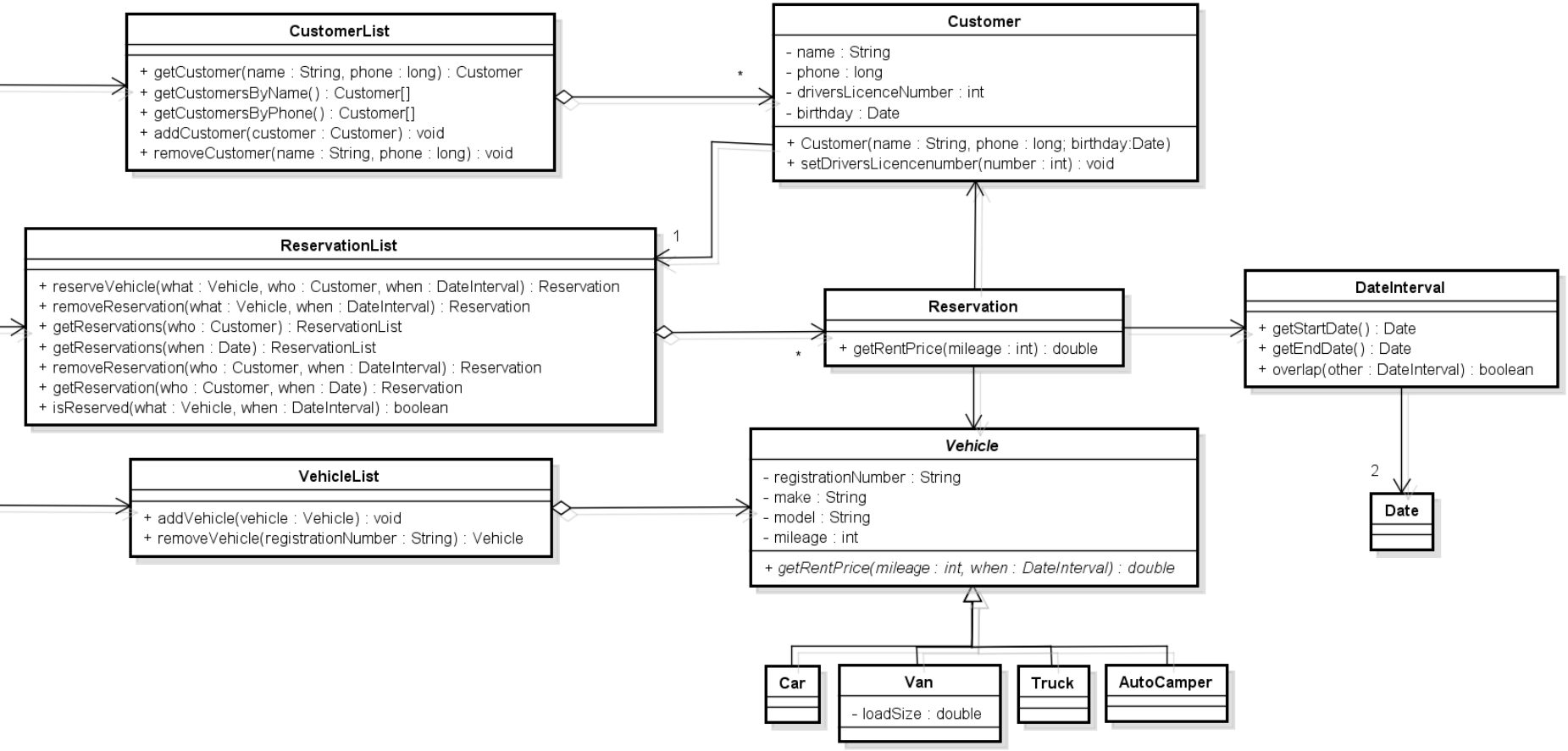
RentalCompany

```
+ addCustomer(customer : Customer) : void
+ removeCustomer(name : String, phone : long) : void
+ getAvailableVehicles(type : String, when : DateInterval) : VehicleList
+ isAvailable(what : Vehicle, when : DateInterval) : boolean
+ reserveVehicle(what : Vehicle, who : Customer, when : DateInterval) : Reservation
+ removeReservation(what : Vehicle, when : DateInterval) : Reservation
+ removeReservation(who : Customer, when : DateInterval) : Reservation
+ getReservation(who : Customer, when : Date) : Reservation
+ getReservations(who : Customer) : ReservationList
+ getReservations(when : Date) : ReservationList
+ rentVehicle(reservation : Reservation, driversLicenceNumber : int) : Vehicle
+ returnVehicle(registrationNumber : String, newMileage : int) : Reservation
+ getRentPrice(reservation : Reservation) : double
+ getRentPrice(what : Vehicle, mileage : int, when : DateInterval) : double
+ getReturnedVehicles(when : Date) : VehicleList
+ addVehicle(vehicle : Vehicle) : void
+ removeVehicle(registrationNumber : String) : Vehicle
```

Delegation to “list-classes”



After inspecting use cases and req.



Analysis class diagram

- Focus is the model (not GUI classes, controller methods or classes for persistence)
- Only show classes, attributes and operations that can be identified in use cases and use case descriptions (and requirements)
  - Note: In Java attributes are called fields or instance variables and operations are called methods

What to do now? (Deadline w. 47)

3. Activity Diagrams
   - Draw an Activity diagram for each Use Case description
   - Follow a guideline to form structured Activity diagrams

4. Analysis class diagram
   - Identify classes from requirements and use case descriptions
   - Draw a class diagram
   - Show classes and relations – not a lot of details
   - Show only fields and methods necessary to understand how each Use Case may be implemented

An activity diagram for each Use Case description

Use Case description #1

Use Case: Reserve a vehicle

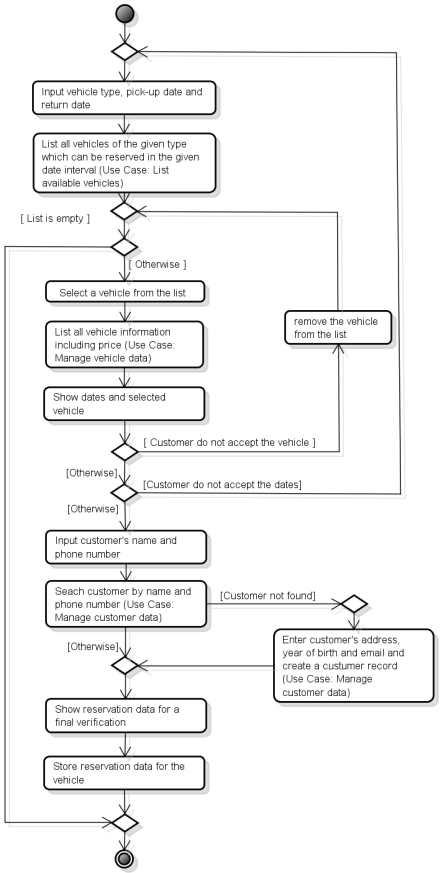
....

1. Enter vehicle type (family car, van, truck or bus) and the two dates; pick-up date and return date
2. System returns a list of available vehicles of the given type in the given date interval (Use Case: List all available vehicles)
3. Select from the list the vehicle to reserve
4. System returns details about the vehicle (Use Case: Manage vehicle data)
5. If vehicle cannot be accepted by the customer then go to step 4 again
6. Verify the dates and vehicle to reserve
7. If dates are not correct then go to step 1
8. Enter name and phone number for the customer
9. System search for the customer by phone number and name (Use case: Manage customer data)

...

Activity diagram #1

For Use Case: Reserve a vehicle



Analysis class diagram