Solutions to the exercises in the book

This document contains solutions to the exercises in the book. The appendix at the end contains the Plantuml-code used for the figures in this document.

2.1

input steps:	2, 3, unnamed, 7.1
internal steps:	4.1, 7.2, 8.1
external steps:	1, 6.1, 6.2, 10
output steps:	4.2, 5, 8.2, 8.3, 9

2.2 1 * 2 * 2 * 3 * 1 * 3 = 36.
The last step, for instance, has 3 possibilities:
E1 followed by E2, E2 followed by E1, and ‘nothing’

2.3 If H3 is ‘dominant over’ H1 then replace the third line by

```
if H3 then B3 else
  if H1 then B2 else B end
end;
```

The underlying case-analysis we used is:

H1	H3	Result
–	–	B
+	–	B2
–	+	B3
+	+	?

+: holds
–: does not hold

H1	H3	Result
++	+	B2
+	++	B3

+: holds
++: dominant over the other

2.4 Depends on your native language, of course. For Dutch it could be something like below (although other Dutch ‘keywords’ might have been chosen):

1. F(actor1 ➡ **System:** γ)	≝	F(actor1) **vraagt het Systeem om** F(γ)
2. F(actor1 ➡ actor1**:** γ)	≝	F(actor1) **doet** F(γ)
3. F(actor1 ➡ actor2**:** γ)	≝	F(actor1) **zendt** F(γ) **naar** F(actor2)
4. F(e1**;** e2)	≝	F(e1)**.** <newline> F(e2)
5. F(e1**,** e2)	≝	F(e1) **en** <newline> F(e2)
6. F(**begin** e **end**)	≝	**begin** F(e) **end**
7. F(**if** [a:] c **then** e1 [**else** e2] **end**)	≝	**als** [F(a)**:**] F(c) **dan** F(e1) [**anders** F(e2)] **end**
8. F(**while** [a:] c **do** e **end**)	≝	**zolang** [F(a)**:**] F(c) **doe** F(e) **end**
9. F(**repeat** e **until** [a**:**] c)	≝	**herhaal:** F(e) **totdat** [F(a)**:**] F(c)
10. F(**for each** m **do** e **end**)	≝	**voor elk(e)** F(m) **doe** F(e) **end**
11. F(**maybe** e **end**)	≝	**misschien** F(e) **end**
12. F(**either** e_1 **or** … **or** e_n **end**)	≝	**hetzij** F(e_1) **hetzij** … **hetzij** F(e_n) **end**
13. F(**perform** n)	≝	**volbreng** F(n)
14. F(**define** n **as** e **end**)	≝	F(n) **betekent:** F(e) **end**

In order to check your own translation rules, you may apply them to some of the textual SSDs in the book, for instance. Zelf ook doen!

4.1 (a) The questionnaire has the following likely answers:

1a. A member borrows *0 or more* books
1b. For each book, *0 or more* members borrow that book

2a. An author wrote *0 or more* books
2b. For each book, *0 1 or more* authors wrote that book

3a. A publisher published *0 or more* books
3b. For each book, *exactly 1* publisher published that book

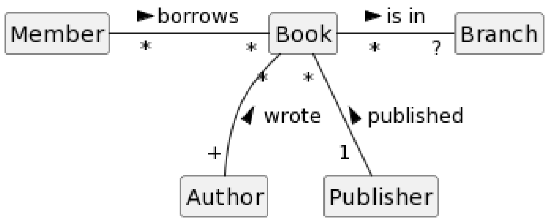
4a. A book is in *exactly 1/at most 1/0 or more* branch(es)
4b. For each branch, *0 or more* books are in that branch

Case 4a is problematic:

- A physical book is in *at most 1* branch (lent out or not)
- A book 'title' (librarian's phrase) is in *0 or more* branches

We will come back to this kind of problem in Section 4.3.3 (*Individual items versus 'catalogue' items*).

(b) The corresponding multiplicities as far as we could determine them:



4.2 Indeed, there are different notions of 'exam'. Something like 'Exam drafted', of which there are 2, and 'Exam made' of which there are 150 in total. As opposed to the concept 'Exam drafted', the concept 'Exam made' probably has a property 'Student' referring to the student who produced the exam work.

In terms of this Section 4.3.3, 'Exam drafted' can be considered as a 'Catalogue' Item and 'Exam made' as an Individual Item.

4.3 (a) According to Table 4.1, the domain model expresses the following:

Association with A = Client, B = Item, xs = buys, m = ‘*’, and n = ‘*’:

1. A Client buys 0 or more Items and
2. an Item is bought by 0 or more Clients

Association with A = Cashier, B = Item, xs = scans, m = ‘1’, and n = ‘*’:

3. A Cashier scans 0 or more Items and
4. an Item is scanned by exactly 1 Cashier

(b) Statements 2 and 4 seem to conflict with each other:

- Since an *Item* can be bought by more than one *Client* (see 2), *Item* seems to be meant as a *‘Catalogue’ item* (unless they mean that in those cases an individual item is bought as a *common* property of those clients)
- Since an Item is scanned by exactly 1 Cashier (see 4), *Item* seems to be meant as an *individual item* (unless they mean that each ‘Catalogue’ item has its own ‘private’ Cashier)

(c) The word ‘Item’ is a homonym here, sometimes used as an *Individual Item* and sometimes as a *Catalogue Item*.

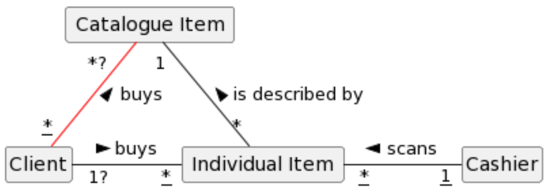
(d) So, you must distinguish between *Individual Item* and *Catalogue Item*.

Then the statements were probably meant as follows:

1. A Client buys 0 or more *Individual* Items and
2. an *Catalogue* Item is bought by 0 or more Clients.

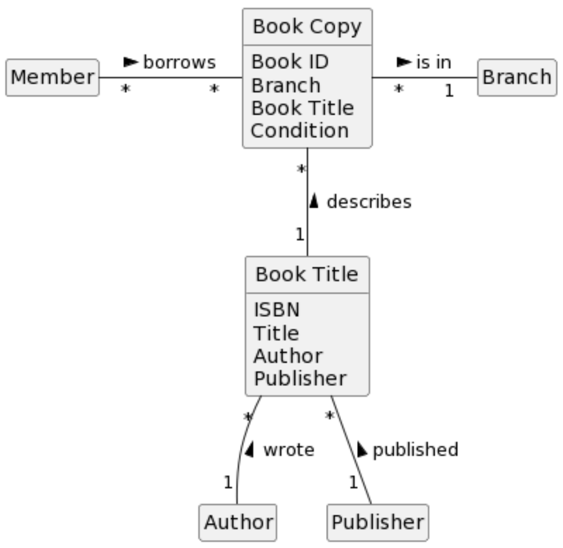
3. A Cashier scans 0 or more *Individual* Items and
4. an *Individual* Item is scanned by exactly 1 Cashier.

This would lead to the following domain model:

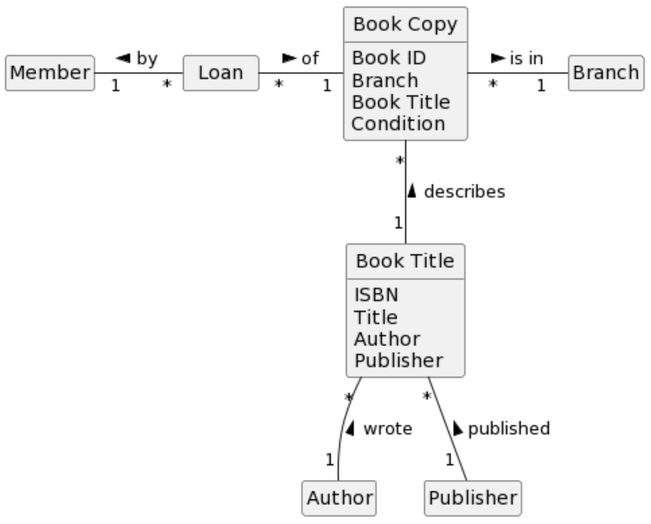


The four underlined multiplicities are expressed in the new statements. The multiplicities with a question mark are not expressed in the new statements. The red coloured association is superfluous because it follows from the other associations.

5.1 (a) Add the (likely) multiplicities



(b) Replace the many-to-many associations (if present):
Well, one many-to-many association must be replaced, leading to:

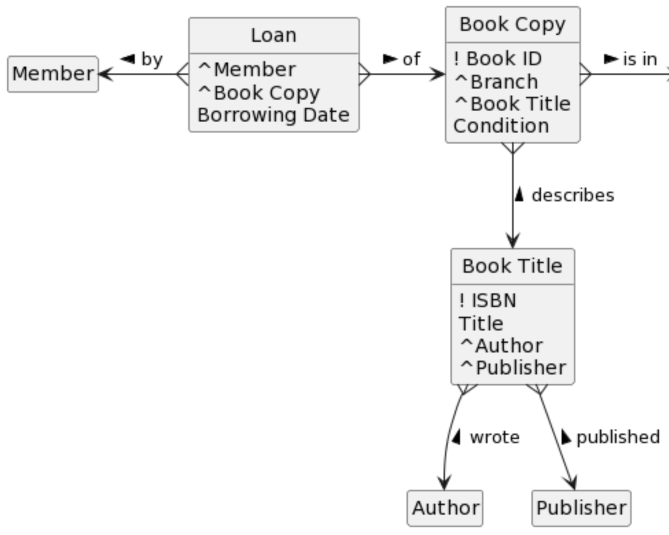


<u>(c), (d), and (e) combined</u>

(c) Replace all association lines by arrows and extend the concepts with the references that follow from the associations (if not present yet)

(d) Add a ‘borrowing date’ at the proper place

(e) Add uniqueness constraints for the two concepts with known properties

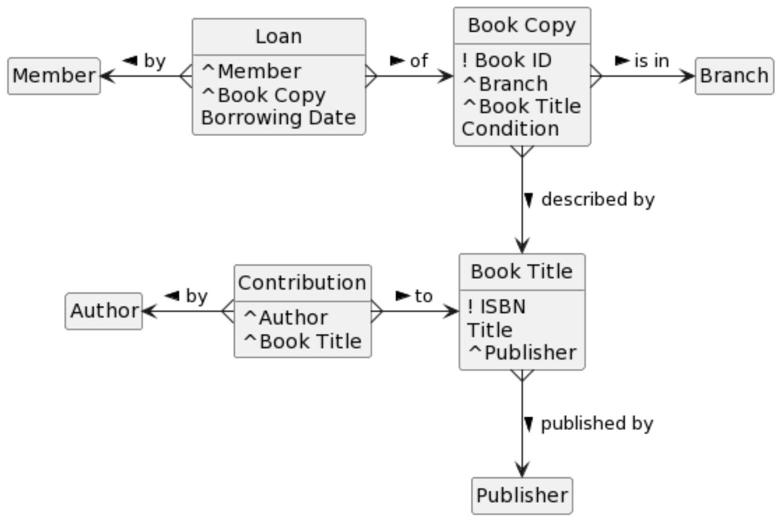


Meanwhile there is a third concept with some known properties (‘Loan’) but its uniqueness constraints are unclear yet.

<u>(f) and (g) combined</u>

(f) Now suppose that a book can have several contributing authors and adapt the conceptual data model

(g) Transform all reading directions such that they follow the arrow (where still necessary)



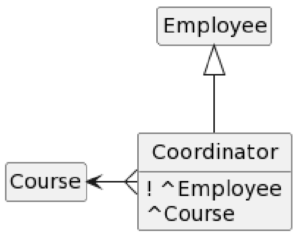
5.2 (a) The proposal expresses that:

- An employee can be the coordinator of at most 1 course

The proposal implies that:

- A course can have more than 1 coordinator
- A course can have no coordinator at all

Therefore, the proposal is not correct.
The proposal illustrated by a graph:



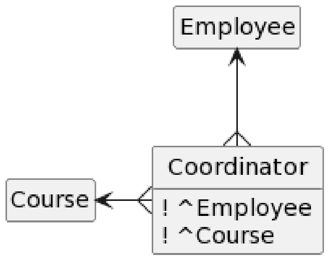
(b) The proposal expresses that:

- An employee can be the coordinator of more than 1 course
- A course can have more than 1 coordinator

The proposal also implies that:

- A course can have no coordinator at all

Therefore, the proposal is not correct.
The proposal illustrated by a graph:



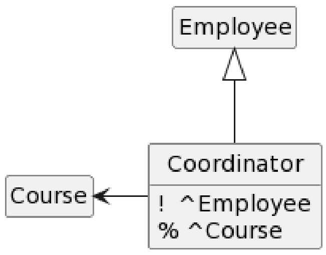
(c) The proposal expresses that:

- An employee can be the coordinator of at most 1 course
- A course can have at most 1 coordinator

The proposal implies that:

- A course can have no coordinator at all

Therefore, the proposal is not correct.
The proposal illustrated by a graph:



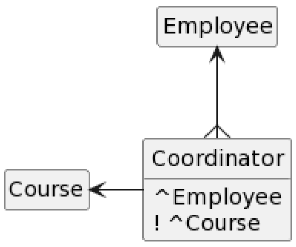
(d) The proposal expresses that:

- A course can have at most 1 coordinator

The proposal implies that:

- A course can have no coordinator at all

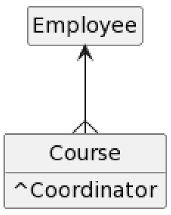
Therefore, the proposal is not correct.
The proposal illustrated by a graph:



(e) The correct proposal is

Course
^Coordinator

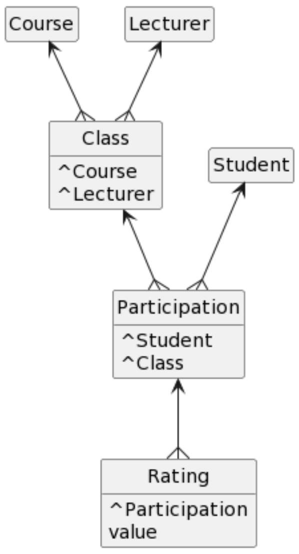
where Coordinator must refer to Employee.
The proposal expressed in a graph:



(f) Only (e) is correct. The course coordinators are those employees who are mentioned in Course.

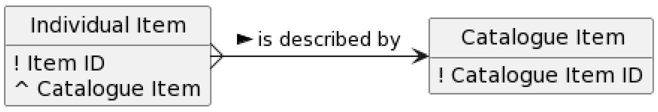
5.3 (a) A Rating refers to a Participation and has a value.
A Participation refers to a Student and refers to a Class.
A Class refers to a Lecturer and refers to a Course.

(b)

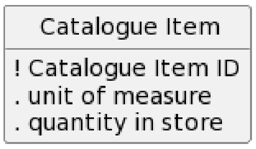


If per Participation only one Rating can be given then the crow's foot arrow from Rating to Participation can be replaced by a normal arrow: $\uparrow$

5.4 (a)



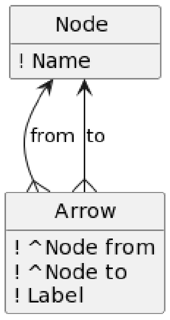
(b) Then there is only the concept *Catalogue Item*, no concept *Individual Item* anymore. We added the two dots for lay-out reasons only:



5.5 Assumptions made:

- Each node has a unique name
- Between two given nodes there can be several arrows,
  but different arrows between those two nodes have different labels.

This results in the following conceptual data model for labelled graphs:



The general Figure 5.2(b) can now be represented as follows:

Node

Name
B
A

Arrow

Node from	Node to	Label
A	B	r1
A	B	r2

For Figure 5.2(a) this results in the following contents:

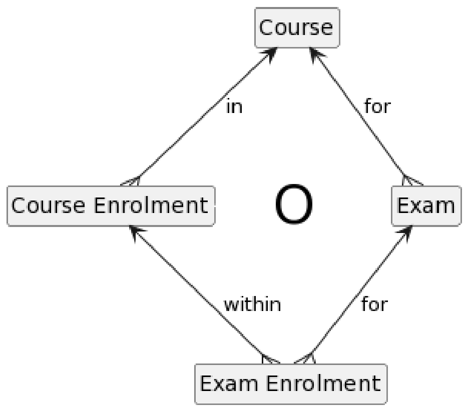
Node

Name
Node
Arrow

Arrow

Node from	Node to	Label
Arrow	Node	from
Arrow	Node	to

8.1

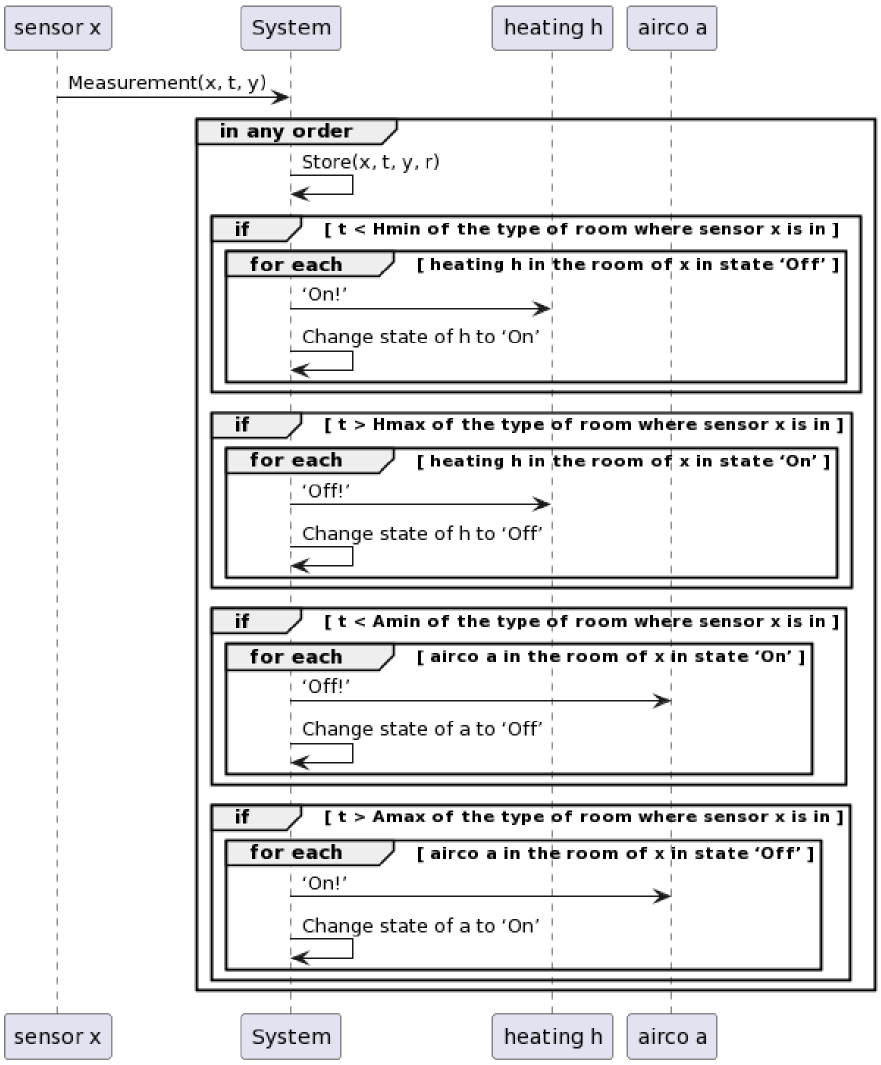


8.2 The commutative diagram in Figure 8.8 says that the r1-value of any A-occurrence is the same as the r2-value of that A-occurrence. So, r2 (or r1) is superfluous.

9.1 4 input steps, 3 internal steps, 4 external steps, and 5 output steps

10.1 Step 2 in the <u>textual</u> SSD is '*System* ➡ *External System: Store(x, t, y, r)*'.
In case there is no storage of the measurements, Step 2 should be left out.
In case of *internal* storage of the measurements, Step 2 should be changed into:
'*System* ➡ *System: Store(x, t, y, r)*'.

Similarly for the corresponding step in the <u>graphical</u> SSD, which is
'*System -> "External System": Store(x, t, y, r)*'.
In case there is no storage of the measurements, this step should be left out. In case of *internal* storage of the measurements, this step should be changed into:
'*System -> System: Store(x, t, y, r)*'.
The resulting graphical SSD in case of *internal* storage of the measurements:


sensor x
System
heating h
airco a
Measurement(x, t, y)
in any order
Store(x, t, y, r)
if
[t < Hmin of the type of room where sensor x is in]
for each
[heating h in the room of x in state 'Off']
'On!'
Change state of h to 'On'
if
[t > Hmax of the type of room where sensor x is in]
for each
[heating h in the room of x in state 'On']
'Off!'
Change state of h to 'Off'
if
[t < Amin of the type of room where sensor x is in]
for each
[airco a in the room of x in state 'On']
'Off!'
Change state of a to 'Off'
if
[t > Amax of the type of room where sensor x is in]
for each
[airco a in the room of x in state 'Off']
'On!'
Change state of a to 'On'
sensor x
System
heating h
airco a

In case there is no storage of measurements, the resulting graphical SSD is the one above but without the step '*Store(x, t, y, r)*' from System to System.

Appendix: Plantuml-code used

<u>For Exercise 4.1(b)</u>

```
@startuml
hide circle
hide members
Member "   *" – "   *" Book : borrows >
Book "*" -- "+" Author: < wrote
Book  "*"-- "1  " Publisher: < published
Book "   *" - "   ?" Branch: is in >
@enduml
```

<u>For Exercise 4.3(d)</u>

```
@startuml
hide circle
hide members
class CI as " Catalogue Item "
class II as " Individual Item "
Client " 1?"  -  "   <u>*</u>" II: buys >
CI " *?   "  -[#red]-  "   <u>*</u>" Client: buys <
II   " <u>*</u>   "  -  "   <u>1</u>"  Cashier: scans <
CI " 1" -- "*" II: is described by <
@enduml
```

<u>For Exercise 5.1(a)</u>

```
@startuml
hide circle
hide empty members
class " Book Copy " {
Book ID
Branch
Book Title
Condition
}
class " Book Title " {
ISBN
Title
Author
Publisher
}
Member        "  *  " – "  *  " " Book Copy " : borrows >
" Book Title " "*" -- "1" Author: < wrote
" Book Title " "*" -- "       1" Publisher: < published
" Book Copy "  "*" -- "1" " Book Title ": < describes
" Book Copy "  "  *  " -  "  1  " Branch: is in >
@enduml
```

<u>For Exercise 5.1(b)</u>

```
@startuml
hide circle
hide empty members
class “ Book Copy “ {
Book ID
Branch
Book Title
Condition
}
class “ Book Title “ {
ISBN
Title
Author
Publisher
}
Member " 1      " - "      *" "  Loan  ": by <
"  Loan  " " *      " - "      1" “ Book Copy “: of >
“ Book Title “ “*” -- “1” Author: < wrote
“ Book Title “ “*” -- “       1” Publisher: < published
“ Book Copy “  “*” -- “1” “ Book Title “: < describes
“ Book Copy “  “  *  ” -  “  1  ” Branch: is in >
@enduml
```

<u>For Exercise 5.1(c, d, e)</u>

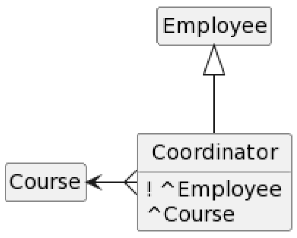
```
@startuml
hide circle
hide empty members
class “ Book Copy “ {
! Book ID
^Branch
^Book Title
Condition
}
class “ Book Title “ {
! ISBN
Title
^Author
^Publisher
}
class "  Loan  " {
^Member
^Book Copy
Borrowing Date
}
Member <-{ "  Loan  ": by <
"  Loan  " }-> “ Book Copy ”: of >
“ Book Title “ }--> Author: < wrote
“ Book Title “}--> Publisher: < published
“ Book Copy “ }--> “ Book Title “: < describes
“ Book Copy “ }->  Branch: is in >
@enduml
```

For Exercise 5.1(f, g)

```
@startuml
hide circle
hide empty members
class “ Book Copy “ {
! Book ID
^Branch
^Book Title
Condition
}
class “ Book Title “ {
! ISBN
Title
^Publisher
}
class "  Loan  " {
^Member
^Book Copy
Borrowing Date
}
class Contribution {
^Author
^Book Title
}
Member <-{ "  Loan  ": by <
"  Loan  " }-> “ Book Copy ”: of >
 “ Book Title “}--> Publisher: > published by
“ Book Copy “ }--> “ Book Title “: > described by
“ Book Copy “ }->  Branch: is in >
Author <-{ Contribution: by <
Contribution }-> “ Book Title ”: to >
@enduml
```

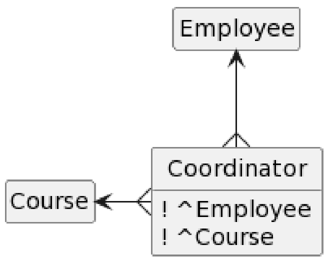
For Exercise 5.2(a)

```
hide circle
hide empty members
Employee <|-- Coordinator
Course <-{ Coordinator
class Coordinator {
! ^Employee
  ^Course
}
```



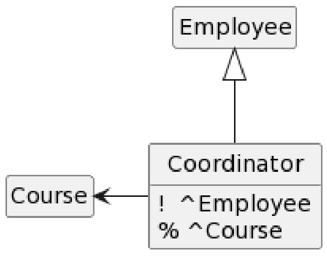
For Exercise 5.2(b)

```
hide circle
hide empty members
Employee <--{ Coordinator
Course <-{ Coordinator
class Coordinator {
! ^Employee
! ^Course
}
```



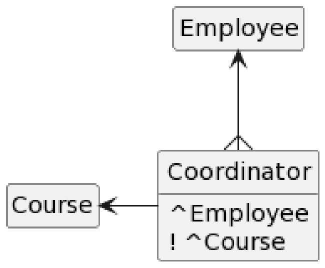
For Exercise 5.2(c)

```
hide circle
hide empty members
Employee <|-- Coordinator
Course <- Coordinator
class Coordinator {
!  ^Employee
% ^Course
}
```



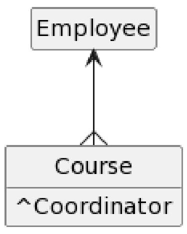
For Exercise 5.2(d)

```
hide circle
hide empty members
Employee <--{ Coordinator
Course <- Coordinator
class Coordinator {
  ^Employee
! ^Course
}
```



For Exercise 5.2(b)

```
hide circle
hide empty members
Employee <--{ Course
class Course {
^Coordinator
}
```



For Exercise 5.3(b)

```
@startuml
hide circle
hide empty members
Class Rating {
^Participation
value
}
Class Participation {
^Student
^Class
}
Class Class {
^Course
^Lecturer
}
Participation <--{ Rating
Student <--{ Participation
Class <--{ Participation
Course <--{ Class
Lecturer <--{ Class
@enduml
```

<u>For Exercise 5.4(a)</u>

```
@startuml
hide circle
hide empty members
"Individual Item" }-> " Catalogue Item": is described by >
Class "Individual Item" {
! Item ID
^ Catalogue Item
}
Class " Catalogue Item" {
! Catalogue Item ID
}
@enduml
```

<u>For Exercise 5.4(b)</u>

```
@startuml
hide circle
hide empty members
Class " Catalogue Item" {
! Catalogue Item ID
. unit of measure
. quantity in store
}
@enduml
```

<u>For Exercise 5.5</u>

```
@startuml
hide circle
hide empty members
class Node as "       Node        " {
! Name
}
class Arrow{
! ^Node from
! ^Node to
! Label
}
Node <--{ Arrow: from
Node <--{ Arrow: to
@enduml
```

<u>For Exercise 8.1</u>

```
@startuml
hide circle
hide empty members
class CE as “Course Enrolment”
class EE as “Exam Enrolment”
Course <--{ Exam: for
Course <--{ CE: in
Exam <--{EE: for
CE <--{ EE: within
class X as “<size:36>O " #white ##white
CE <-[#white] X
@enduml
```

For Exercise 10.1

```
@startuml
“sensor x” -> System: Measurement(x, t, y)
group in any order
  System -> System: Store(x, t, y, r)
participant h as “heating h”
participant a as “airco a”
  group if [ t < Hmin of the type of room where sensor x is in ]
    group for each [ heating h in the room of x in state ‘Off’ ]
      System -> h: ‘On!’
      System -> System: Change state of h to ‘On’
    end
  end
  group if [ t > Hmax of the type of room where sensor x is in ]
    group for each [ heating h in the room of x in state ‘On’ ]
      System -> h: ‘Off!’
      System -> System: Change state of h to ‘Off’
    end
  end
  group if [ t < Amin of the type of room where sensor x is in ]
    group for each [ airco a in the room of x in state ‘On’ ]
      System -> a: ‘Off!’
      System -> System: Change state of a to ‘Off’
    end
  end
  group if [ t > Amax of the type of room where sensor x is in ]
    group for each [ airco a in the room of x in state ‘Off’ ]
      System -> a: ‘On!’
      System -> System: Change state of a to ‘On’
    end
  end
end
@enduml
```