UNIVERSITÄT
DUISBURG
ESSEN

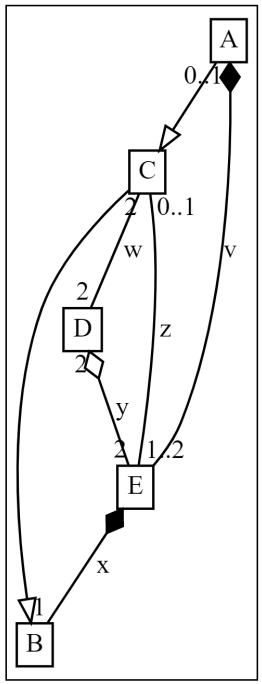
Offen im Denken

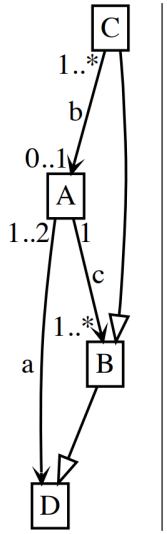
Aspects of "Real-World Injection" in Modelling-Exercises

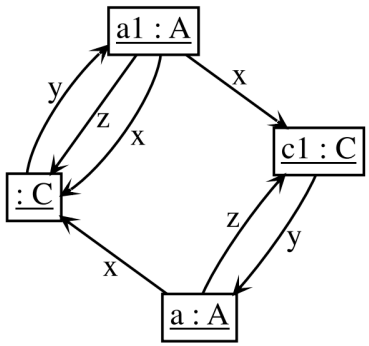
Janis Voigtländer

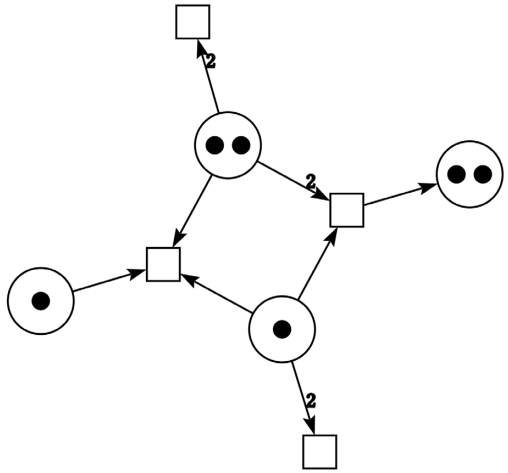
■ 9th December 2025

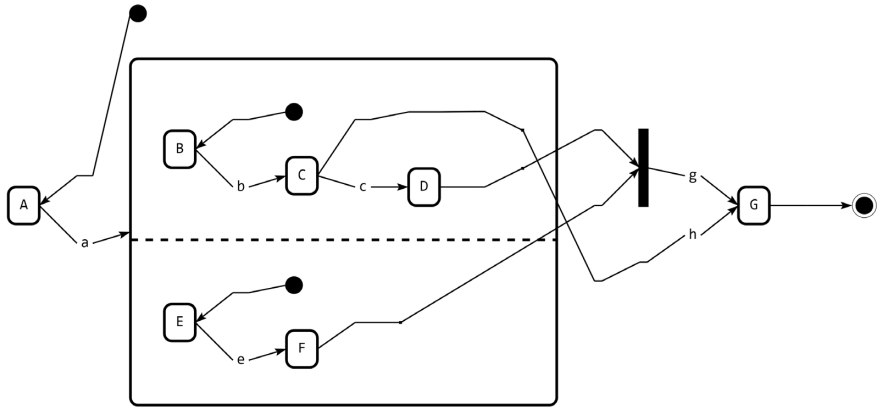
Context: Diagram languages taught











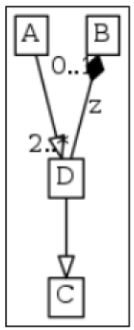
Context: Generated online exercises

UNIVERSITÄT DUISBURG ESSEN

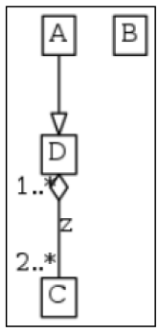
Offen im Denken

Consider the following two class diagrams.

1

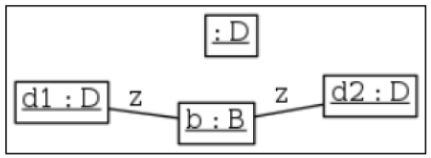


2

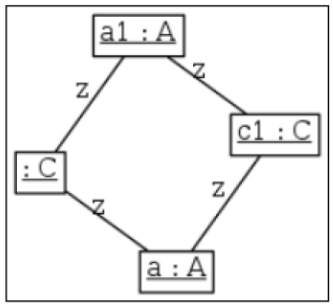


Which of the following five object diagrams conforms to which class diagrams?
An object diagram can conform to neither, either, or both class diagrams.

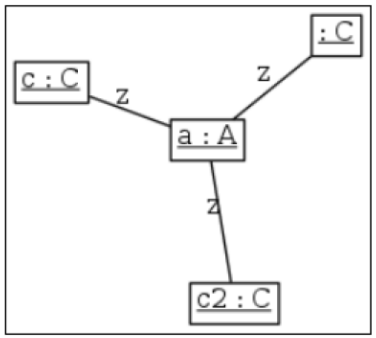
a



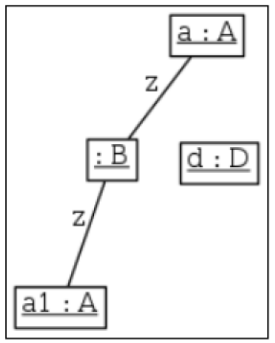
b



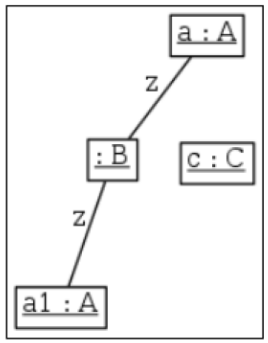
c



d



e



Please state your answer by giving a list of pairs, each comprising of a class diagram number and a string of object diagram letters.
Each pair specifies that the mentioned object diagrams conform to the respective class diagram.

Context: Alloy in the background

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

```
abstract sig Class {}

abstract sig Relationship {
  from : one Class,
  to : one Class
}
```

```
pred noInheritanceCycles [is : set Inheritance] {
  no c : Class | c in c.^((~ from :> is) . (is <: to))
}
```

```
run { cd1 and (not cd2) }
```

```
abstract sig Change {
  add : lone Relationship,
  remove : lone Relationship
}
```

```
pred conflict[t1, t2 : Transitions, p : Places]{
  t1 != t2
  activated[t1]
  activated[t2]
  p.tokens < plus[p.flow[t1], p.flow[t2]]
}
```

```
abstract sig Regions{
        name: lone ComponentNames,
        contains: disj some Nodes
}
```

Janis ❤ Alloy

The role of Alloy

Why formally specify all this stuff at all here?

- guarantee correctness of the task instances
- control difficulty of task instances
- target individual (sub)concepts
- generate useful/interesting distractors
- sneak more formal specification into the course

Some interesting bits that arise:

- different flavours of Alloy usage (shallow vs. deep embedding)
- sometimes useful semantic feedback comes as a byproduct

Our variation strategies

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Why aiming for variation of tasks?

- prevent plagiarism
- provide volume for practising
- address different competencies

Aside: Why does variation of tasks still work for cheat prevention at all?

- because of the graphical nature of the domain here

How do we achieve variation?

- playing with the constraints that are sent to Alloy
- combinatorial space of instances returned by Alloy
- randomization (shuffling names and layout, for example)
- playing hide-and-seek on the data obtained from Alloy

Hide-and-Seek (example)

Based on our modelling of the concepts of class and object diagrams, Alloy might return all of these (together, and more):

1. a valid class diagram
2. an invalid class diagram with a certain set distance to the valid one
3. a representation of what the exact difference between the two is
4. a denomination of the kind of error(s) in the invalid class diagram
5. an object diagram for the valid class diagram

An exercise task design can then be formed by deciding which of the above ingredients to use for:

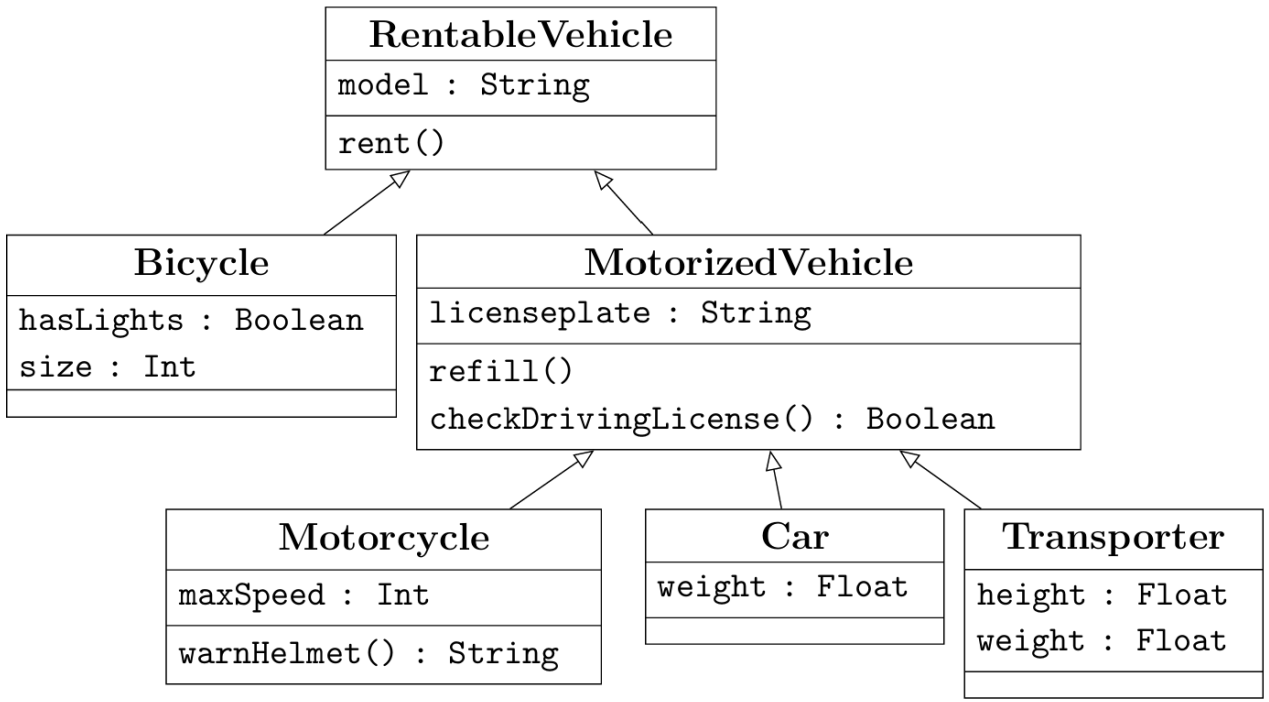
- the task description presented to students
- the expectation of what students should input
- the feedback provided to students after their solution attempt

What happens outside the online exercises?

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Task from an exercise sheet:

Let the following class diagram be given (as part of modelling a vehicle rental company):



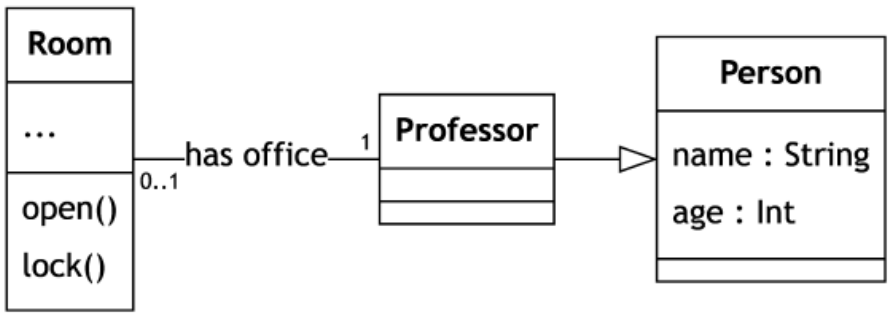
For each of the given classes, construct one object as instance of exactly that class. You can choose the names of objects and the (type-correct) values of attributes. But do not use any value more than once.

What happens outside the online exercises?

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Task from an exam:

Let us now focus on "has office", in the following part of ChatGPT's diagram:



ChatGPT didn't get the multiplicities right in this since the following two requirements from the given scenario text are not respected: "Each professor has a specific room as own office. Not every room is some professor's office."

Your task now is to demonstrate ChatGPT's error by providing two object diagrams below (1. and 2.), in each case only involving classes from the class diagram part given above in (c), not any of the other classes from the complete diagram. Ignore the attributes of **Room**.

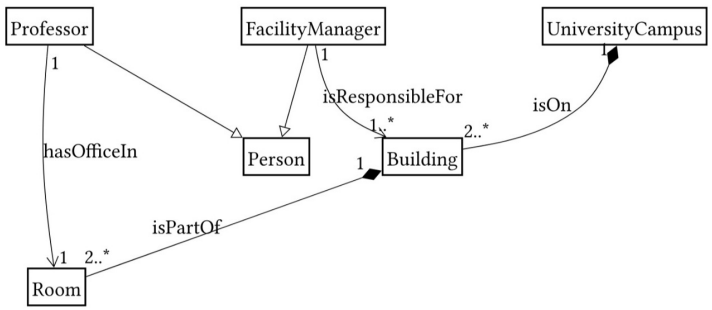
1. An object diagram, with exactly three objects, that conforms to the above class diagram (c), but should have been forbidden according to the given excerpt from the scenario text:
   Ein Objektdiagramm, mit genau drei Objekten, das zum obigen Klassendiagramm (c) passt, aber gemäß dem gegebenen Auszug aus dem Szenario-Text hätte verboten sein sollen:
2. An object diagram, with exactly three objects, that does not conform to the above class diagram (c), but should actually have been allowed:
   Ein Objektdiagramm, mit genau drei Objekten, das nicht zum obigen Klassendiagramm (c) passt, aber eigentlich hätte erlaubt sein sollen:

Real-world scenarios in online form

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Experimenting with hand-crafted non-artificial diagrams online:

Ein Universitätscampus besteht aus verschiedenen Gebäuden. Hausmeister sind für Gebäude zuständig und jedes Gebäude wird von einem Hausmeister betreut. Ein Hausmeister ist eine Person. Eine andere Art von Personen sind die Professoren, die jeweils einen bestimmten Raum als eigenes Büro haben. Ein Gebäude besteht aus verschiedenen Räumen, von denen nicht jeder ein Professorenbüro ist.



Welche dieser Aussagen zum Klassendiagramm ist zutreffend? (Das Klassendiagramm ...)

- ○ a: hält eine Vorgabe des Szenarios nicht ein.
- ○ b: enthält einen Syntaxfehler.
- ○ c: hat gar kein passendes Objektdiagramm.
- ○ d: ist vollständig richtig modelliert.

Im Fall eines Problems, welche Beziehungen sind darin involviert? (alle auswählen, die zu ändern wären, um das Problem zu beheben)

- ☐ 1. die Vererbung, bei der Professor von Person erbt
- ☐ 2. Assoziation isResponsibleFor
- ☐ 3. Komposition isOn
- ☐ 4. Assoziation hasOfficeIn
- ☐ 5. Komposition isPartOf
- ☐ 6. die Vererbung, bei der FacilityManager von Person erbt

Real-world scenarios in online form

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Why try to generate (lots of) tasks with real-world concepts?

- a new dimension of variation (for plagiarism prevention etc.)
- be closer to the exams across the board of all task types
- exposure to lots of “practical” examples
- instil the formal validity aspects also on such examples
- hypothesis: more engaging/motivating for students

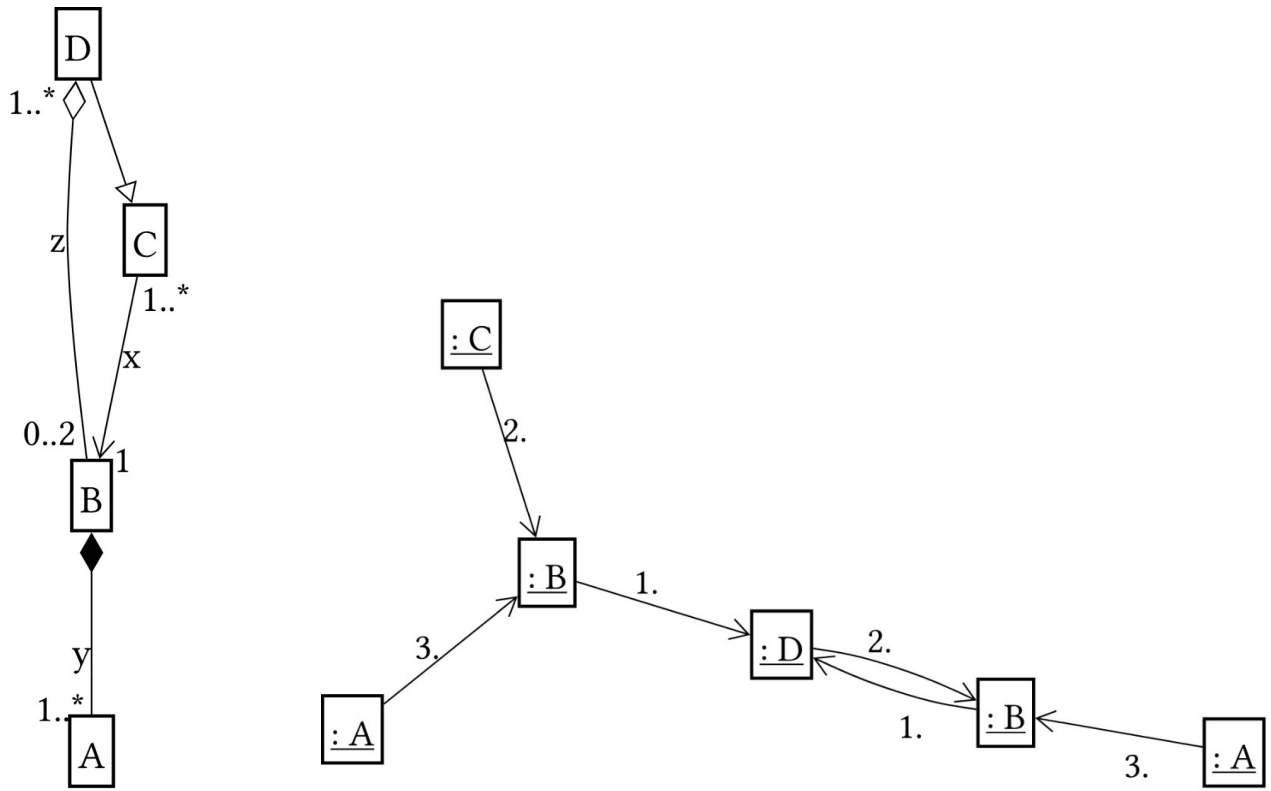
But:

1. How to do it at scale?
2. How does it play with our existing approaches to correctness, variability, feedback?

Let’s look at a few aspects of the 2. point (mainly on one vanilla task type).

Matching class and object diagrams

On this task type, given:



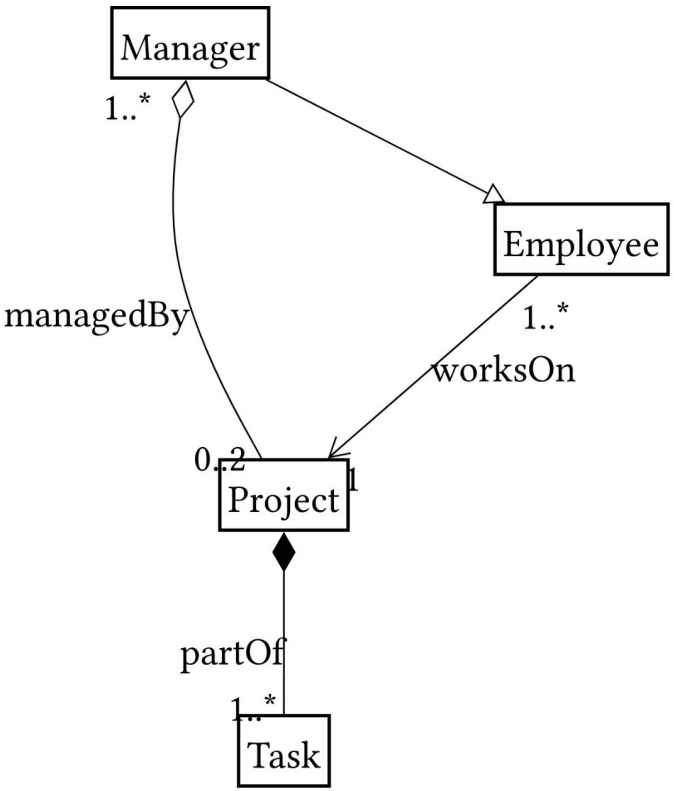
and sought: 1./z, 2./x, 3./y

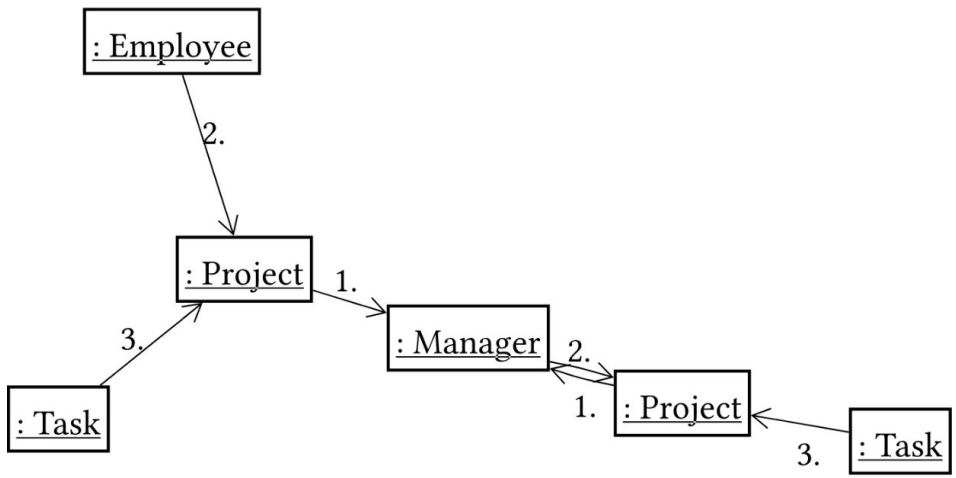
. . . we can use “concept injection” (Zixin’s presentation) to obtain . . .

Matching class and object diagrams

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

. . . this to show to students:



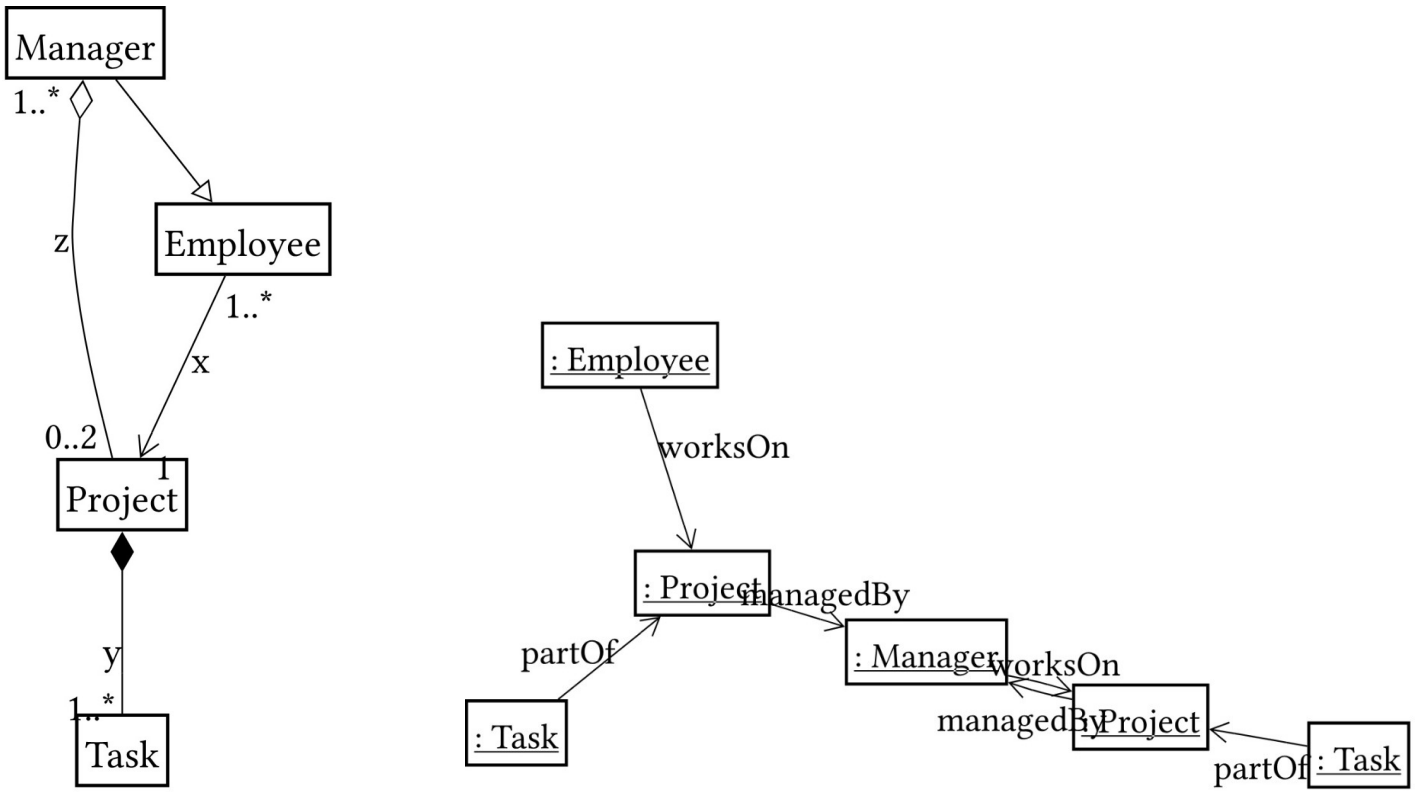


Now sought: 1./managedBy, 2./worksOn, 3./partOf

Matching class and object diagrams

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

But also, with no real additional effort:



Now sought: x/worksOn, y/partOf, z/managedBy

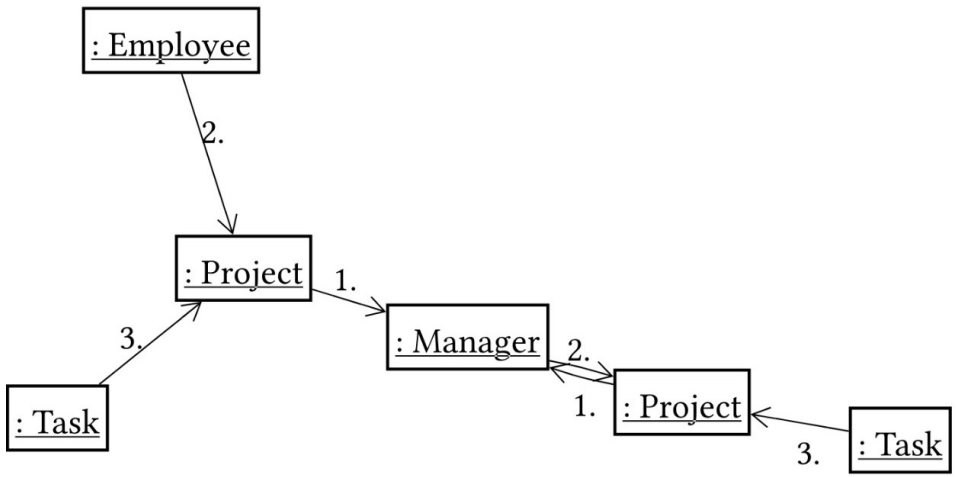
Matching class and object diagrams

Observations:

- All three task variants are different hide-and-seek versions of the same underlying pair of class diagram and object diagram in which only A, B, C, D and x, y, z are used.
- But only in the real-world scenario do we get two truly different variants.
- Just how “truly different” these actually are (in terms of solution strategies and difficulty, say) may be an interesting didactic question (that cannot even be asked/studied in the artificially-named scenario).

Assigning relationships “just so”

With a different hide-and-seek strategy, we could also just show this to students:



and tell them that the relationships “worksOn”, “partOf”, “managedBy” occur, then ask them which belongs where.

That would obviously not be possible with the artificially-named version.

Is it actually possible (in the sense of: uniquely solvable) here?

Assigning relationships “just so”

Well, we could accompany the object diagram by a narrated description (obtained from the hidden class diagram) such as:

> *A company runs projects. Every project is made up of tasks, and it must have at least one task; each task is a component of exactly one project (it can’t exist on its own or belong to multiple projects). People in the company are employees. Some employees are managers (a specialized kind of employee). Managers are responsible for projects in a loose, non-owning way: a single manager can oversee up to two projects, and each project must be overseen by at least one manager (possibly more). Employees work on projects. Each employee is assigned to exactly one project, while every project has one or more employees working on it.*

and then ask students again where which of the relationships “worksOn”, “partOf”, “managedBy” belongs in the object diagram.

Assigning relationships “just so”

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Observations:

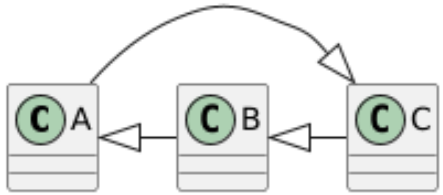
- In the artificially-named case, narrating the class diagram to the same purpose would lead nowhere.
- In the real-world case, we might even try adding some distractors, i.e., offering further relationships alongside “worksOn”, “partOf”, “managedBy”.
- Conversely, we could narrate the object diagram and ask for assigning given relationship names in the class diagram.

- We could also try yet another hide-and-seek strategy, where the class names are the hidden part (in one of the two diagrams, and with the other one displayed or narrated).
- Other of our existing task types offer similar opportunities for repurposing their ingredients in completely new task designs.

Other consequences of real-world injection

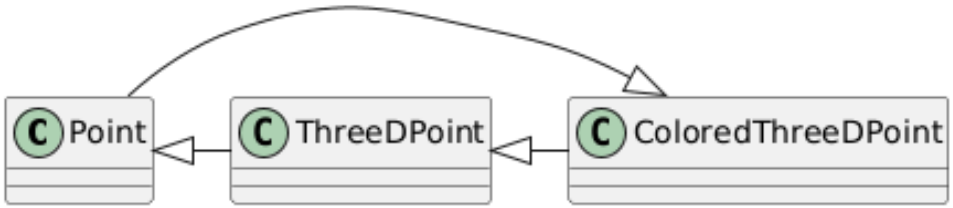
Sometimes what we expect as correct answer in a task type will change!

For example, consider a task type of identifying errors in class diagrams, and let the following constellation appear in a task instance:



Our current input modality would be that we expect students to identify all the inheritances contributing to the cycle.

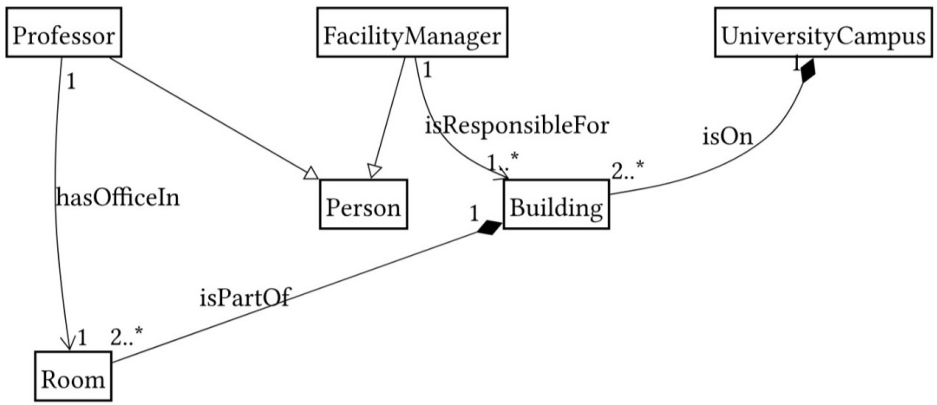
But what if the scenario is presented as follows instead?



Other consequences of real-world injection

Sometimes what we provide as feedback has to be changed!

For example, in the task type underlying this hand-crafted instance for online use:



the feedback so far has been given by displaying the class diagram again without the offending (as determined by the student or by the sample solution) relationship and stating whether the thus reduced class diagram is correct or not.

But here that would defeat the adherence to the narrative in the task text.