Lab assessment for Software Design and Modelling

Week 6, Semester 2, 2023

- This assessment is to be done **individually**, making use of the tools you have used in the lab sessions so far.
- It is **open-book**: you may consult the course material and any online source you have found useful. However, you may not collaborate with one another or with anyone else, and you may not use any tool other than those specified in the questions.
- Please remember the **good scholarly practice** requirements of the University regarding work for credit. You can find guidance at the School page

  `https://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct`

  which also has links to the relevant University pages. Please do not publish solutions to these exercises on the internet or elsewhere, until I tell you you may.
- The exercise has been designed to be done in **75 minutes**. Those given 25% extra time in exams may use 94 minutes, according to their usual arrangements.
- The assessment will be **marked out of 100**. It is in three parts.
  - Part A is worth 50 marks, and is intended to take no more than 30 minutes, provided you understand the work and the tools well.
  - Part B is worth 30 marks. You are advised not to work on it until you are confident that you have done Part A well. It is intended to take no more than 30 minutes for those who understand the work and the tools very well, but might take up to 45 minutes with a few false steps.
  - Part C is worth 20 marks. It is intended to challenge those students who find parts A and B easy and do them fast. Don't worry if you don't get to it: as you see, a first-class mark can be obtained without attempting it. You are advised not to attempt it *unless and until* you feel you have done Parts A and B very well. **It will not be marked unless your Parts A and B have already earned you at least 70 marks, i.e., a first-class mark.**

To submit

Each question tells you what to submit, giving a filename and format, e.g. A.pdf. Save your files locally. Then when you are ready to submit, make a zip file called `done.zip`, containing them all *at top level*, e.g. using DICE command:

```
zip done.zip A.pdf A.java B.java C.pdf
```

and upload it using the button in the question where you got this paper. *Use exactly the names and formats specified*, otherwise you may lose marks.

Part A

1. In LucidChart draw a UML class diagram showing:

   - a class CrimeStory with a public operation synopsis taking no argument and returning a String
   - a class Detective
   - a class Character with a private attribute isMurderer of type Boolean
   - a class Victim
   - an appropriate relationship between Character and Victim to indicate that every Victim is a Character
   - aggregations indicating that a CrimeStory may contain Detectives, Characters and Victims
   - multiplicities to ensure:
     - every CrimeStory must contain one or two Detectives, at least one Victim, and between 3 and 10 Characters (inclusive)
     - a Detective can be in more than one CrimeStory, but Victims and Characters can be in at most one.

   (40 marks)

2. Add a Note to your UML class diagram. In it, write an OCL expression, with an appropriate context line, to specify that every CrimeStory must contain at least one Character who is *not* a Victim.

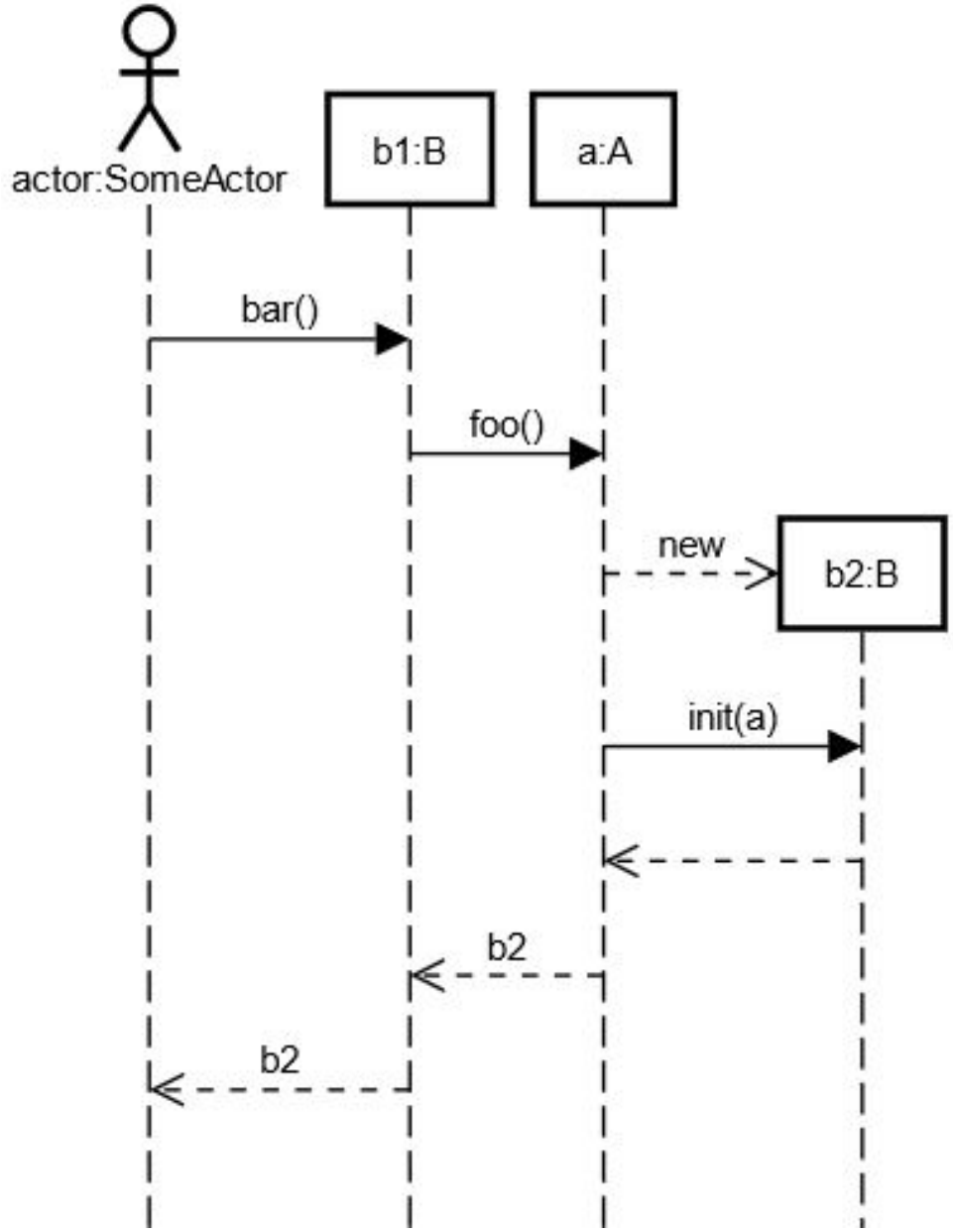
   (10 marks)

   Export your diagram as A.pdf

Part B

Study the following sequence diagram. In Java, implement classes A and B in such a way as to be consistent with the diagram. Keep your code as simple as you can, and do not invent extra functionality – you should not need to write more than 10 lines of code in either class.

You may use a Java compiler and any editor and/or IDE you choose. Partial credit can be obtained by code that does not compile. For full credit your code should compile without error, though you do not need to run it.



Save your code as A.java and B.java

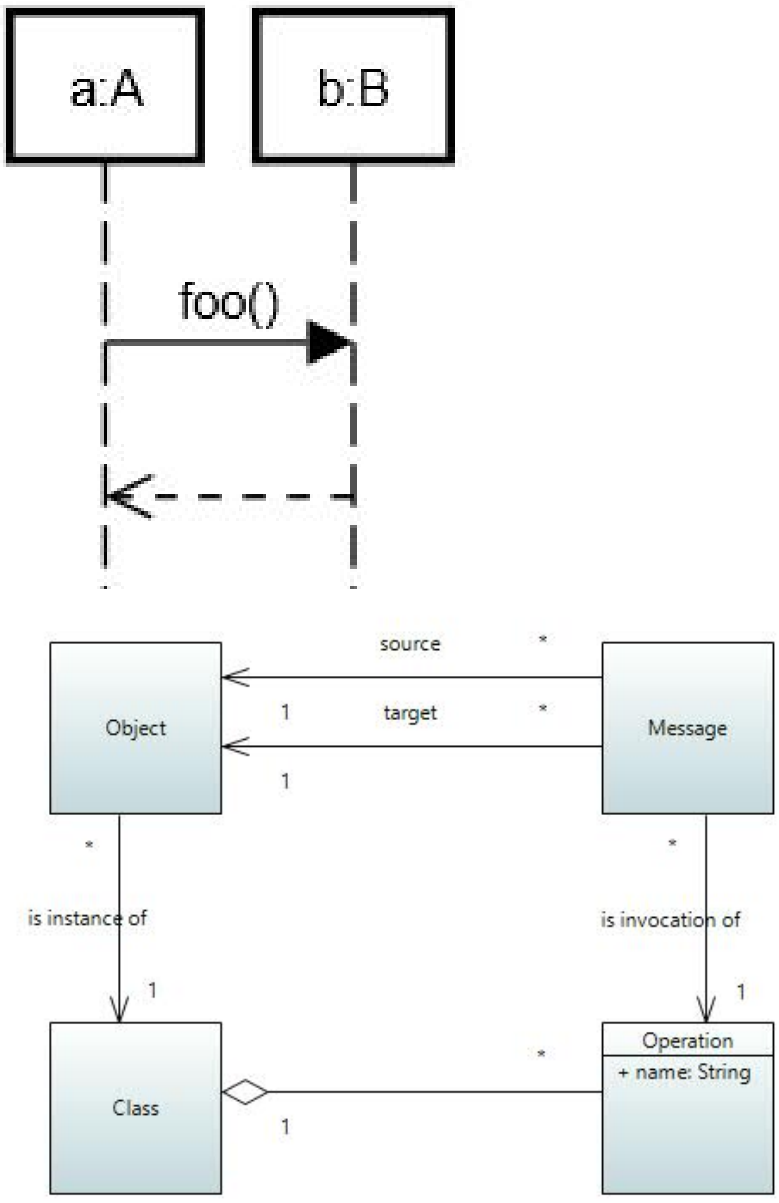
(30 marks)

Part C

REMINDER: This part is intended for students in the upper reaches of the University Standard Marking Scale. It is aimed at students who have found Parts A and B easy and done them quickly. It will only be marked if your answers to Parts A and B have already secured you a first-class mark! So you are strongly advised not to attempt it until you are sure you have done as well as you can on Parts A and B.

This question concerns the formal definition of modelling languages. We will explore sequence diagrams, first with reference to a fictional, naive modelling language definition, and then with reference to UML2.5.1.

Consider the fragment of sequence diagram, and the fragment of a modelling language metamodel, below; then answer the questions on the next page.



Your answer to this question should preferably be submitted as a single PDF named C.pdf. But if you have difficulty including the diagram for part 1 in the same document as the other, textual, parts, you may submit it separately as C1.pdf.

1. As an instance of the metamodel fragment, draw an object diagram to represent the sequence diagram. You may draw this diagram with any tool you like: LucidChart is one sensible option.

2. Express in OCL the rule which ensures that a sequence diagram only shows objects receiving messages that they will understand.

Now consult the UML2.5.1 Specification (as a reminder, it can be found at `https://www.omg.org/spec/UML/2.5.1/About-UML`). I have said in the SDM teaching materials that you can think of the rectangles in sequence diagrams (such as the ones labelled `a:A` and `b:B` here) as objects, but that technically they are not objects.

3. Name the metaclass of which these are, in fact, instances.

4. Give the figure number of the diagram that shows the relevant abstract syntax.

5. Finally, compare the naive metamodel fragment on the previous page with the actual UML metamodel for sequence diagrams. Briefly explain some of the deficiencies of the naive metamodel fragment – why do you think the UML metamodel has become so much more complicated? Do not write more than 200 words.

(20 marks)

Finally: submission

If you have done all parts of the lab assessment you will have files to submit as follows:

- A.pdf
- A.java
- B.java
- C.pdf (if you did Part C)
- C1.pdf (if you did Part C and didn't incorporate the diagram)

Now create a single zip file called done.zip containing all the files you wish to submit. On DICE the command to do this is

`zip done.zip A.pdf A.java B.java C.pdf`

– adjust in the obvious way for files you have/have not created.

Upload it using the button in the question where you got this paper. *Use exactly the names and formats specified*, otherwise you may lose marks.

After you have submitted you may leave, quietly.