Lab assessment for Software Design and Modelling

Week 6, Semester 1, 2018

Introduction

This assessment is to be done individually on your DICE machines during the lab session, making use of the tools you have used in the lab sessions so far.

In order to be able to give students with schedules of adjustments extra time, the exercise has been designed to be done in 75 minutes. Those given extra time in exams may use the remaining time, according to their usual arrangements.

The assessment will be marked out of 20. It is in three parts.

- Part A is worth 10 marks, and is intended to take no more than 30 minutes, provided you understand the work and the tools well.
- Part B is worth 6 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 4 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 well.

To submit

Some questions ask you to submit specific files. To do this, use a terminal, navigate to the directory containing the file, and use examsubmit to submit it, e.g.

```
examsubmit Foo.java
```

Others ask you to submit your entire Eclipse project. To do this, in Eclipse, use menu File → Export → General → Archive file. (NB do *not* use a Papyrus-specific export menu!) Use the Select All button to select all your files. (It does not matter if some of them are irrelevant.) Use the Browse button to choose where to save your archive; be sure to give it the required name. Then use examsubmit to submit it, as above, e.g.

```
    examsubmit A1.zip
```

Part A

1. Using Papyrus, and calling your project A1, draw a UML class diagram showing:

   (a) A class `Artist`, with private attribute `name` of type `String` **(use the UML primitive types in this assessment, not the ecore ones)**;

   (b) A class `Recording`, with private attribute `title` of type `String`;

   (c) A class `MP3`, specializing `Recording`, with private attribute `bytes` of type `Integer`;

   (d) An association between `Artist` and `Recording`, named `performs`, navigable both ways;

   (e) Multiplicities indicating that each `Recording` is linked to between zero and 5 (inclusive) `Artists`, and each `Artist` is linked to one or more `Recordings`.

   Export your project as A1.zip as described above. **Submit A1.zip**.

   [6 marks]

2. Using Papyrus, and calling your project A2, draw a UML state diagram showing:

   (a) two states, called S1 and S2;

   (b) where S1 is the initial state;

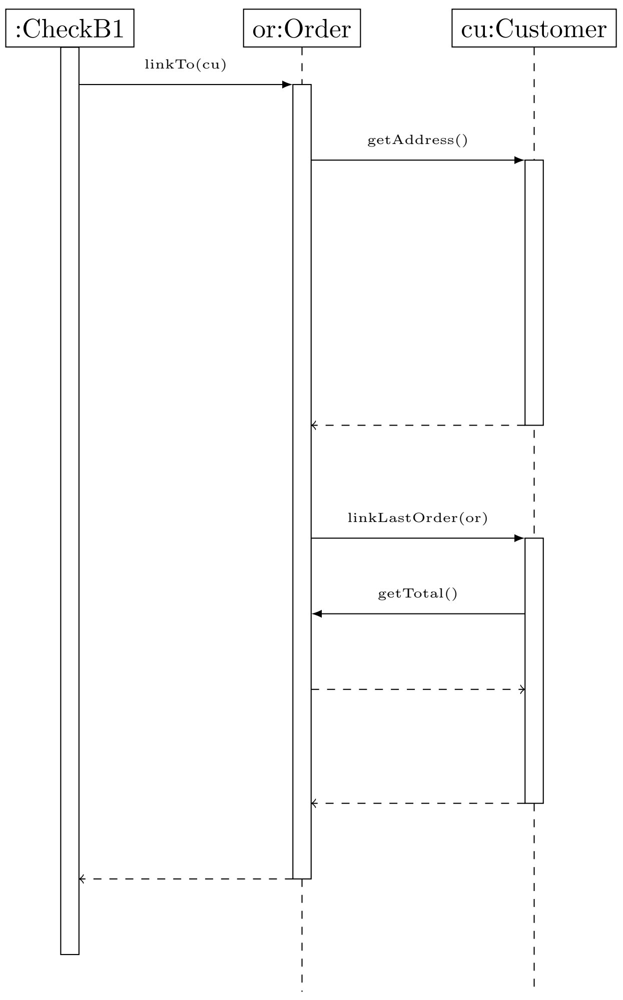
   (c) and the event `bang` causes a transition from S1 to S2, but only if the guard condition `today is Tuesday` holds. (Model this as a constraint, in natural language.)

   Export your project as A2.zip as described above. **Submit A2.zip**.

   [4 marks]

Part B

1. Consider the Java code in files Order.java and Customer.java, in conjunction with the sequence diagram shown below. The Java and the UML are inconsistent in several ways. Assuming that both the UML and the class Customer are correct, modify the Java code for Order to be consistent with it. Include comments to explain, briefly, what you have changed and why.



Submit your modified version of Order.java.

[4 marks]

2. Consider the following description of a business process.

   A lecturer sets an exam and then uploads it to secure storage. Then it must be sent to an external examiner for comments, and also inspected by someone in the School. When both of these things have happened, the lecturer revises the paper.

   In Papyrus, develop an activity diagram to model this process. Do not concern yourself with object flow, and do not include swimlanes. Wherever the order of activities is not clear in the description, show them happening in parallel.

   Export your project as B2.zip as described above. **Submit B2.zip**.

   [2 marks]

Part C

Your answers to Part C should be typed into a text file called C.txt.

1. Consult the UML2.5.1 standard, provided at `file:///group/examreadonly/sdm/`. You are familiar with the idea of one class being a generalization of another; you are also familiar with protocol state machines. In UML, is it legal for one protocol state machine to be a generalization of another protocol state machine? Explain precisely how you can tell that this is, or is not, permitted, giving the relevant section (e.g. 1.2.3.4) and/or figure (e.g. 1.2) numbers. (If you need to, you may state, without justifying it from the standard, that a Class is a Classifier.) [2 marks]

2. Regardless of whether or not such a generalization is actually permitted in UML2.5.1, we may consider what it should mean, if it were permitted. Briefly describe what you think should be true about two protocol state machines, for it to be sensible to regard one as a generalization of the other, justifying your answer. (There is more than one right answer to this question! Marks will be given for anything sensibly justified.) [2 marks]

Submit file C.txt.

Summary

If you did every part of this assessment, the files you should have submitted are:

- A1.zip
- A2.zip
- Order.java
- B2.zip
- C.txt

Later submissions override earlier ones, so if in doubt, resubmit.