DUMMY Lab assessment for Software Design and Modelling

Week 6, Semester 1, 2016

This dummy version is just to give you an idea of the format of the lab assessment.

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. You can check it yourself using the methods described below. These checks are not definitive – they don't check everything – but if what you submit passes the checks below, you can expect to get at least 8 marks. Part A is intended to take 30 minutes, provided you understand the work and the tools well. You will submit your work, together with the transcript of your checking.
- 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 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 students who find parts A and B easy and do them fast: 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 *some command we'll specify for you*,

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 *some command we'll specify for you.*

Some possible question types

All the actual questions are of forms given in this list, but of course, not all these question types will be asked! There will be 2 questions in each of parts A, B and C. The questions may not be equally weighted, but you will be told how many marks there are for each question (though not given the full markscheme, of course.)

1. Using Papyrus, and calling your project *whatever*, draw a UML class diagram showing *[some stuff, e.g. specified classes, generalizations, associations, multiplicities, attributes].*

   [n marks]

   Each question will end with some text about exactly what to submit, and for part A questions, how to check it

2. Using Papyrus, and calling your project *whatever*, draw a UML state diagram showing *[some stuff, e.g. specified states, transitions, conditions, starting state]*

3. Using Papyrus, and calling your project *whatever*, draw a UML activity diagram diagram showing *[some stuff, e.g. specified activities, flows, forks and joins, decisions and merges]*

4. Consider the following class diagram. Write, as simply as possible, Java code that is consistent with it. Assume *some stuff you might wonder about.*

   some class diagram here

5. Consider the following sequence diagram. Write, as simply as possible, Java code that is consistent with it. Assume *some stuff you might wonder about.*

   some sequence diagram here

6. Consider the following state diagram. Write, as simply as possible, Java code that is consistent with it. Assume *some stuff you might wonder about.*

   some state diagram here

7. Consider the following situation.

   some domain description

   In Papyrus, develop a conceptual class model for this situation, in a model called *whatever*. Include multiplicities and attributes, where you can deduce them from the above, but do not concern yourself with operations.

8. Consider the following lifecycle of a *whatever*.

   some description of things that happen to a whatever and how it changes state as a results

   In Papyrus, develop a state diagram to model a *whatever*, in a model called *whatever*. Choose appropriate states and show transitions between them. Do not concern yourself with conditions or actions.

9. Consider the following description of a business process.

   some description, probably involving a bit of vagueness about order

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

10. Consider the Java code in *whatever*, in conjunction with the class diagram *Or: the protocol state diagram; Or: the sequence diagram; Or: some combination of these wherever, could be here, could be in a project you're given, in which case I'd tell you how to import it to view the diagrams.* The Java and the UML are inconsistent in several ways. Assuming that the UML is correct, modify the Java code to be consistent with it. *Or: Assuming the Java is correct, modify the UML to be consistent with it.* Include comments to explain, briefly, what you have changed and why.

11. *This question type would only appear in Part C* Consult the UML2.5 standard, provided at `file:///group/examreadonly/sdm/`. *Some question about it, could be quite challenging: e.g., is the following UML legal; or, what are the rules about whatever; or, the following is a rule, but where exactly is it specified; or, comparing how Papyrus treats some UML feature with how it is specified in the UML standard, explain what's wrong in Papyrus.*

12. *This question type would only appear in Part C* Consider the following situation... Using your choice of one or more appropriate UML diagrams, design functionality for *doing something*. Briefly explain your work in text file *whatever*, and submit this along with any relevant diagrams.