Changes proposed to DRPS of Inf1-OP for session 2013/14

0.1 Section "Assessment Information"

Current version:

<table>
<thead>
<tr>
<th>Written Examination</th>
<th>95</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessed Assignments</td>
<td>5</td>
</tr>
<tr>
<td>Oral Presentations</td>
<td>0</td>
</tr>
</tbody>
</table>

Assessment

Formative assessment will be used to provide feedback and guidance to students and will take the form of quizzes, exercise sheets, practical exercises and coursework assignments, covering areas from across the syllabus. A summatively assessed class test (worth 5% of the final mark) will be held mid semester and will test students basic programming competence.

If delivered in semester 1, this course will have an option for semester 1 only visiting undergraduate students, providing assessment prior to the end of the calendar year.

Proposed new version:

<table>
<thead>
<tr>
<th>Written Examination</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessed Assignments</td>
<td>0</td>
</tr>
<tr>
<td>Oral Presentations</td>
<td>0</td>
</tr>
</tbody>
</table>

Assessment

The summative assessment will be by an open-book practical programming examination. Formative assessment will be used to provide feedback and guidance to students and will take the form of weekly guided programming exercises designed to progressively cover the curriculum and to offer a range of difficulty levels. Students will also be encouraged to develop larger programs of interest to them, in order to challenge themselves to extend their programming skills beyond the basics and to begin to learn to design software.

0.2 Section ”Study pattern”

Current version:

<table>
<thead>
<tr>
<th>Lectures</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tutorials</td>
<td>10</td>
</tr>
</tbody>
</table>
Proposed new version (this is to bring it in line with what’s been happening in the last few years anyway - no new changes):

Lectures: 10  
Tutorials: 8  
Timetabled Laboratories: 20  
Non-timetabled assessed assignments: 0  
Private Study/Other: 62  
Total 100

0.3 Section “Reading list”

Current version:

Introduction to Programming in Java, Robert Sedgewick & Kevin Wayne (2008), Addison-Wesley
Java Concepts, Cay Horstmann,. Wiley

Proposed new version:

Addison-Wesley The Java Tutorial: A Short Course on the Basics, 5th Edition, Sharon Zakhour et al. (2013),
Introduction to Programming in Java, Robert Sedgewick & Kevin Wayne (2008),
1 Reasons for proposing the changes

Note first that most of the changes are actually to reflect the last few years’ reality, from which the DRPS had become unstuck – I am not proposing any change to the number of lectures, for example, and the practical work has consisted of labs plus (not for credit) group project work – no quizzes, exercise sheets etc., no class test – for at least the last two years.

1.1 Assessment information

The proposal is to remove the 5% credit currently attached to the lab exercises. Note: in contrast to what the old DRPS says, the actual current practice is that the 5% is awarded to a pseudo-random selection of the automatically marked weekly lab exercises, chosen at the end of the course. Students get feedback on the exercises as they do them, by running tests with which they are provided (and I plan to keep providing and incrementally improve the tests, so that they get as good or better feedback in future) – at the end of the course, three of their exercises are selected and they get a mark reflecting how well they did on the best of the three.

The lab exercises examine exactly the same skills as the programming examination. The main difference is that the labs are not time-limited – students who need more than one lab session to do them can come back for another or do them in their own time. However, in practice the examination is not designed to, and apparently does not, place students under time pressure. Also, the exam is open-book, so there isn’t a memorisation difference, either. In my view this is appropriate – that is, I am content that the exam should be open-book and not overly time pressured, because this best examines aspects of student competence that they will need later – but it reinforces the lack of differentiation between the two forms of assessment.

Labs are inevitably scheduled across a whole week. I would like students to be able to solve a programming problem, and then, immediately after they’ve written their own program, see another solution (e.g., make a model answer automatically available after their own program passes the tests, or perhaps even encourage a group of students to test and comment on one another’s solutions: I’d like to experiment here). But this would mean that some students were seeing a model answer before other students had attempted the exercise. I think delaying the release of a model answer until after all students have done the exercise is less effective pedagogically; students who felt they had more or less done it don’t bother to look, so they never see a perhaps different and perhaps better way to do it.

Developing appropriate graded lab exercises is too time-consuming to be repeated every year: inevitably there is at least a large and typically a practically 100% overlap between last year’s exercises and this year’s. This increases the chance that solutions to the exercises will be available to students who wish to use them, even if we tried to prevent it. And of course, and appropriately, students help one another in the lab. The mark we assign to lab exercises
is not a credible measure of any aspect of student attainment.

Currently the system, Infandango, that presents lab exercises stores students’ details including marks – that is, “high risk data” in the university’s terms – on a self-managed machine which has an associated firewall hole to allow it to be accessed. In the light of increasing concerns about data security, **this cannot go on**: if we store such data it must be on our managed systems, i.e., on DICE, so that we can guarantee that security patches are always applied promptly, etc.

It is not clear how much effort would be required to transfer the software and associated data to DICE, but Infandango does rely on several packages not currently available on DICE so this would at least not be trivial.

If we make the lab exercises purely formative, we remove the need to store student details and marks and remove this problem. In fact, rather than using a version of Infandango, I would like to try giving students the current exercises, together with tests and instructions on how to run them – as opposed to instructions to feed their solutions into the Infandango “black box” which runs tests – as I think basic testing should be given a somewhat higher profile in the course from the beginning.

Acknowledging that there may be a small number of students who would work diligently every week for a total of 5% of the mark but not for 0%, I nevertheless feel that the advantages of removing the credit for lab exercises outweighs the disadvantages.

1.2 Study pattern

This is a tidy-up exercise: the new version is what has been happening for the past several years.

1.3 Reading list

I updated the edition of Zakhour. I have also swapped the order – putting Zakhour ahead of Sedgewick and Wayne – as I think the more comprehensive book by Z et al. is probably better for the majority of our students, although it is useful to be able to recommend S&W to students who really want its very gentle introductory approach.

I suggest removing Horstmann because I don’t know it personally, it isn’t referred to in the existing materials, and there are so many Java books these days that it seems better to mention one or two for specific reasons and then give general advice in Lecture 1, as I did this year.