A Proposal for Informatics 1B

Discussion Document

Paul Anderson <dcspaul@ed.ac.uk>
Volker Seeker <volker.seeker@ed.ac.uk>
School of Informatics
University of Edinburgh

1 Background

INF1B is a new 20 point course intended to replace the existing INF1-OP[1] and INF1-DA[2] courses (10 points each).

The main motivation is to provide more time and effort to help students to acquire a reasonable level of practical programming ability:

- The material from the existing INF1-DA course will largely be deferred to later years.
- An introduction to practical imperative programming will be included at the end of INF1A for those students with no prior experience. This will allow INF1B to assume some initial (imperative) programming knowledge from all students.

This discussion document presents some thoughts on a possible approach to the course. We would like to gather feedback on this, and agree on the general principles before developing the details.

2 Motivation

We would like to:

- Focus on practical programming skills and increase the students' ability and confidence in producing realistic, practical applications.
- Foster an awareness of good engineering practice in an informal way - including, version control, testing, and documentation.
- Provide a course which is stimulating and challenging for those with more experience, at the same time as being accessible to those with less.
- Encourage student interaction and collaborative development of larger applications, and show how an object-oriented approach can facilitate this.
- Provide the practical motivation for a more formal approach to the topics studied in later years.

[1] https://www.inf.ed.ac.uk/teaching/courses/inf1/op/
[2] https://blog.inf.ed.ac.uk/da18/

3 Proposed Format

We propose taking an "Objects First"[3] approach to the course material by starting with object-oriented concepts and gradually introducing the features of the Java language in this context. We would like to take advantage of the students' previous experience of functional programming by comparing and contrasting the two approaches.

In practical terms, we would provide a realistic framework of pre-written objects which the students could initially connect and modify in small ways, and later augment by writing their own classes from scratch. This framework would provide the background to the entire course.

We would ask the students to work in small groups (probably of four), with each student in the group responsible for a different class (or a small set of classes). The classes could then be combined to produce a group implementation of a realistic application. Each group would have the same set of classes, and the interface specifications would be given in advance. This would allow implementations to be compared and exchanged between groups.

We would hope to choose a suitably flexible application that the same framework could be reused in subsequent years by adding new classes to provide additional functionality. It is also possible to envisage the classes being chosen to highlight various areas of the curriculum - for example databases, graphics, algorithms, natural language, etc.

4 Assessment

We do not believe that the practical programming objectives of this course can be meaningfully assessed by examination, and we propose that the assessment be based entirely on the coursework. We are conscious that this may produce a less reliable individual assessment - however, this is a first-year course, and we are keen to avoid an overly-rigorous assessment which would compromise the learning opportunities. We anticipate using a comparatively coarse-grained, criteria-based marking scheme similar to that used for undergraduate projects.

The assessment might include (for example):

[3] Objects First with Java: A Practical Introduction using BlueJ, David J. Barnes & Michael Kölling.
http://www.bluej.org/objects-first/.

- Group demonstrations of a running application.
- A brief inspection of the student code for non-functional aspects such as readability and structure.
- A written worksheet where students would be expected to discuss their implementation and compare it with implementations of the same class by other groups.

To pass the course, students would be expected to produce a minimally working implementation which is reasonably well-structured and readable, and to provide a meaningful written discussion of their implementation.

We would expect the specification of the individual components to allow considerable flexibility in the functionality of the implementation - for example, different implementations may be differentiated by their performance, error handling, or user interface. In combination with the written discussion of the implementation, this would provide scope for exceptional students to be recognised in accordance with the common marking scheme.

Resourcing of the assessment process will require careful consideration. We would like to consider the use of later-year undergraduates for group mentoring and formative feedback. There are unlikely to be significant opportunities for auto-marking. We would however, expect to automated code-similarity checking (MOSS) to be reasonably effective.