Comments on Teaching Committee Proposal 19.13 – Informatics 1B course.

I support the idea of expanding the time available for students to learn Java, including an increased focus on code quality and design issues.

However, the proposed assessment mechanism is unacceptable. Assessing this course purely on the basis of coursework would be worse than worthless.

- Basing the course around the same development which is assessed for credit will inhibit conscientious students from freely discussing their work with one another (even in ways that the staff would actually be happy with), thereby damaging a major learning modality.

- It will be utterly trivial for any student who wishes to obtain a higher mark than their competence merits to obtain one (by means ranging from being over-supported by well-intentioned members of the same team, who naturally want good overall behaviour of their application, all the way to commissioning work from any of the many commercial providers);

- regardless of the extent to which this actually happens, it will be obvious to everybody involved that we cannot prevent it;

- thereby rendering the marks of students who don't cheat, as much as of those who do, unreliable;

- and undermining a major aim of the curriculum reform, that is, the aim of ensuring that all students entering the later years of our degree have strong programming skills.

There is a very serious risk of bringing the School and the University into justified disrepute. This is not a theoretical possibility. Older members of the school will remember the Great CS1 Scandal. For those who do not, it should suffice to observe that even now, almost 20 years later, if you google "university Edinburgh plagiarism scandal" you will find press articles about it on your first page of hits.

We could expect the next scandal to be considerably worse, because the scale is now larger, because students today are more worried about their precise marks than students were two decades ago, and because the marketplace for buying coursework has burgeoned.

If this course went ahead in this form I, for one, would be very unwilling to have any involvement with its teaching or assessment: I would do so only under protest.

Assessing a course taught at this scale, using resources we can muster, is difficult and there is no perfect answer. I think either of these two approaches would be strictly better than the proposal:

1) Do not assess the course. Let it be pass/fail based on attendance.

2) Assess the course by programming exam(s), even if the exam(s) cannot assess everything that we aim for students to learn during the course.

(Bear in mind the possibility of offering several programming exams, either as a ladder where passing a basic one is required to enter a more advanced one, or in the manner of tiered GCSEs, by the way.)

I do not think coursework should contribute to a student's mark in more than a trivial way (10% say).

Turning to the teaching aspects: there are naturally few details here. Assuming that the element of summative assessment is removed from the groupwork, I think it will be very challenging, but perhaps not impossible, to run the course this way. Perhaps one might use the teaching studios for large meetings of many teams together, one per table, which would enable them to have timetabled work together supported by fewer tutors than one per group. With groups of 4, though, capacity will be an issue, and I would not suggest groups larger than 4.

Two things that might be one another's solutions:

1) Even with the addition of the slow start at the end of Inf1A, there will still be a very wide range of programming ability in the class. How is this to be handled?

2) The idea that you can have sufficiently precise interface specifications to allow implementations to be swapped between groups, while still allowing flexibility of design, needs to be demonstrated.

An interesting approach might be to have a first phase with very precise interfaces, then actually do a bunch of swapping between groups, encouraging students to evaluate and assess one another's code, and using the results of that to re-form different groups, with some groups comprising students who struggled at the early phases, and others comprising students who produced perfect implementations. The former groups could then be supported into completing a relatively easy application, while the latter were given their head and encouraged to invent variants (perhaps with a culminating competition).

Further points for consideration:

- What will happen when one or more members of a group do not engage?

- Will outside students still be welcome on the course, or will it be restricted to Informatics students only?

- The development of a suitable framework, together with appropriate interface specifications and tests, will be a lot of work -- far more work than we will be able to throw away and re-do every year, so the balance of extension/redevelopment/reuse will be tricky. This is yet another reason for separating it from assessment.