THE UNIVERSITY of EDINBURGH
informatics

Board of Studies
Course Proposal Template

PROPOSED COURSE TITLE: Introduction to Software Engineering using Object Oriented Programming (INF1B)

PROPOSER(S): Paul Anderson, Volker Seeker

DATE: 26.11.2018

SUMMARY

This template contains the following sections, which should be prepared roughly in the order in which they appear (to avoid spending too much time on preparation of proposals that are unlikely to be approved):

1. Case for Support
– To be supplied by the proposer and shown to the BoS Academic Secretary prior to preparation of an in-depth course description

1a. Overall contribution to teaching portfolio

1b. Target audience and expected demand

1c. Relation to existing curriculum

1d. Resources

2. Course descriptor
- This is the official course documentation that will be published if the course is approved, ITO and the BoS Academic Secretary can assist in its preparation

3. Course materials
- These should be prepared once the Board meeting at which the proposal will be discussed has been specified

3a. Sample exam question

3b. Sample coursework specification

3c. Sample tutorial/lab sheet question

3d. Any other relevant materials

4. Course management

- This information can be compiled in parallel to the elicitation of comments for section 5.

4a. Course information and publicity

4b. Feedback

4c. Management of teaching delivery

5. Comments

- To be collected by the proposer in good time before the actual BoS meeting and included as received

5a. Year Organiser Comments

5b. Degree Programme Co-Ordinators

5c. BoS Academic Secretary

[Guidance in square brackets below each item. Please also refer to the guidance for new course proposals at http://www.inf.ed.ac.uk/student-services/committees/board-of-studies/course-proposal-guidelines. Examples of previous course proposal submissions are available on the past meetings page http://web.inf.ed.ac.uk/infweb/admin/committees/bos/meetings-directory.]

SECTION 1 – CASE FOR SUPPORT

[This section should summarise why the new course is needed, how it fits with the existing course portfolio, the curricula of our Degree Programmes, and delivery of teaching for the different years it would affect.]

1a. Overall contribution to teaching portfolio

[Explain what motivates the course proposal, e.g. an emergent or maturing research area, a previous course having become outdated or inappropriate in other ways, novel research activity or newly acquired expertise in the School, offerings of our competitors.]

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.

Student engagement has historically been a problem in Inf1-OP. This proposal aims to incorporate several best practices for improving engagement and encouraging under-represented students, in particular: collaborative learning and student interaction, opportunities for self-reflection, and offering a range of options.

INF1B will focus on practical programming skills and increase the students' ability and confidence in producing realistic, practical applications. It will foster awareness of good engineering practice in an informal way – including version control, testing and documentation. The course will be presented in a way which is stimulating and challenging for those with more experience, at the same time as being accessible to those with less. We aim to encourage student interaction and collaborative development of larger applications, and show how an object-oriented approach can facilitate this. Furthermore, it will provide the practical motivation for a more formal approach to the topics studied in later years.

This course is supported by the current year 1 organiser.

1b. Target audience and expected demand

[Describe the type of student the course would appeal to in terms of background, level of ability, and interests, and the expected class size for the course based on anticipated demand. A good justification would include some evidence, e.g. by referring to projects in an area, class sizes in similar courses, employer demand for the skills taught in the course, etc.]

This course is aimed at first year students of Informatics programs which have just finished the INF1A course. Next to logic and functional programming, 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.

Next to the aforementioned skills, no further programming background will be required from students.

The first year courses in informatics have traditionally a high demand with numbers as high as 400 students. A similar demand is expected for INF1B.

1c. Relation to existing curriculum

[This section should describe how the proposed course relates to existing courses, programmes, years of study, and specialisms. Every new course should make an important contribution to the delivery of our Degree Programmes, which are described at http://www.drps.ed.ac.uk/17-18/dpt/drps_inf.htm.

Please name the Programmes the course will contribute to, and justify its contribution in relation to courses already available within those programmes. For courses available to MSc students, describe which specialism(s) the course should be listed under (see http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2017/programme-guide/specialist-areas), and what its significance for the specialism would be. Comment on the fit of the proposed course with the structure of academic years for which it should be offered. This is described in the Year Guides linked from http://web.inf.ed.ac.uk/infweb/student-services/ito/students.]

INF1B will be part of each of the current undergraduate degree Programmes offered at the School of Informatics. Since it will replace INF1-OP and INF1-DA in the second semester of the first year, it will be offered at that place.

1d. Resources

[While course approvals do not anticipate the School's decision that a course will actually be taught in any given year, it is important to describe what resources would be required if it were run. Please describe how much lecturing, tutoring, exam preparation and marking effort will be required in steady state, and any additional resources that will be required to set the course up for the first time. Please make sure that you provide estimates relative to class size if there are natural limits to its scalability (e.g. due to equipment or space requirements). Describe the profile of the course team, including lecturer, tutors, markers, and their required background. Where possible, identify a set of specific lecturers who have confirmed that they would either like to teach this course apart from the proposer, or who could teach the course in principle. It is useful to include ideas and suggestions for potential teaching duty re-allocation (e.g. through course sharing, discontinuation of an existing course, voluntary teaching over and above normal teaching duties) to be taken into account when resourcing decisions are made.]

We do not expect the total resource requirements for the 20 point course to be more than twice those for the existing 10 point course. The exact details depends on the final choice of activities. However:

- 400 students implies a minimum of 26 tutorials (15 students).
- Around 130 groups of 3-4 students would be required for the demonstrations. We anticipate these normally being handled by the tutors, which would require an additional 3 hours per group - this includes about 20 mins per demo, 4-5 groups, and some time to provide written feedback.

- The marking is estimated to require about 200 hours to scan the code and read the final reports - allowing about 20-30 minutes per student, and additional time for moderation and training.
- Additional TA and lecturer resource will be required for development.

We are slightly concerned about the ability to find the necessary number of qualified TAs/demonstrators/tutors/markers, and the training necessary to maintain a consistent standard.

SECTION 2 – COURSE DESCRIPTOR

[This is the official course descriptor that will be published by the University and serves as the authoritative source of information about the course for student via DRPS and PATH. Current course descriptions in the EUCLID Course Catalogue are available at www.euclid.ed.ac.uk under 'DPTs and Courses', searching for courses beginning 'INFR'*]*

2a. Course Title *[Name of the course.]*:

Introduction to Software Engineering using Object Oriented Programming

2b. SCQF Credit Points:

[The Scottish Credit and Qualifications Framework specifies where each training component provided by educational institutions fits into the national education system. Credit points per course are normally 10 or 20, and a student normally enrols for 60 credits per semester. For those familiar with the ECTS system, one ECTS credit is equivalent to 2 SCQF credits. See also http://www.scqf.org.uk/The%20Framework/Credit%20Points.]

20

SCQF Credit Level:

[These levels correspond to different levels of skills and outcomes, see http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf At University level, Year 1/2 courses are normally level 8, Year 3 can be level 9 or 10, Year 4 10 or 11, and Year 5/MSc have to be level 11. MSc programmes may permit a small number (up to 30 credits overall) of level 9 or 10 courses.]

10

Normal Year Taken: 1/2/3/4/5/MSc

[While a course may be available for more than one year, this should specify when it is normally taken by a student. "5" here indicates the fifth year of undergraduate Masters programmes such as MInf.]

1

Also available in years: 1/2/3/4/5/MSc

Different options are possible depending on the choice of SCQF Credit Level above: for level 9, you should specify if the course is for 3rd year undergraduates only, or also open to MSc students (default); for level 10, you should specify if the course is available to 3rd year and 4th year undergraduates (default), 4th year undergraduates only, and whether it should be open to MSc students; for level 11, a course can be available to 4th and 5th year undergraduates and MSc students (default), to 5th year undergraduates and MSc students, or to MSc students only]

Year 1 only

[If the course is only available to MSc students, then it must be classified as a Postgraduate course. All other courses, regardless of level, will be classified as Undergraduate]

Undergraduate

2c. Subject Area and Specialism Classification:

[Any combination of Computer Science, Artificial Intelligence, Software Engineering and/or Cognitive Science as appropriate. For courses available to MSc students, please also specify the relevant MSc specialist area (to be found in the online MSc Year Guide at http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2017/programme-guide/specialist-areas), distinguishing between whether the course should be considered as "core" or "optional" for the respective specialist area.]

Computer Science and Software Engineering

Appropriate/Important for the Following Degree Programmes:

[Please check against programmes from http://www.drps.ed.ac.uk/17-18/dpt/drps_inf.htm to determine any specific programmes for which the course would be relevant (in many cases, information about the Subject Area classification above will be sufficient and specific programmes do not have to be specified). Some courses may be specifically designed for non-Informatics students or with students with a specific profile as a potential audience, please describe this here if appropriate.]

All current undergraduate programmes.

Timetabling Information:

[Provide details on the semester the course should be offered in, specifying any timetabling constraints to be considered (e.g. overlap of popular combinations, other specialism courses, external courses etc).]

Should be offered in semester 2. Overlap should be considered in the same way as for Inf1-OP and Inf1-DA.

2d. Summary Course Description:

[Provide a brief official description of the course, around 100 words. This should be worded in a student-friendly way, it is the part of the descriptor a student is most likely to read.]

This course presents a conceptual and practical introduction to object oriented programming and software engineering practices, exemplified by Java. As well as providing a grounding in the use of Java, the course will cover general principles of programming in imperative and object oriented frameworks. After completing the course successfully, students will be able to develop programs that support experimentation, simulation and exploration in other parts of the Informatics curriculum (e.g. the capacity to implement, test and observe a particular algorithm).

Course Description:

[Provide an academic description, an outline of the content covered by the course and a description of the learning experience students can expect to get. See guidance notes at: http://www.studentsystems.is.ed.ac.uk/staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html

An introduction to the concepts of programming and software engineering using an object oriented programming language.

Pre-Requisite Courses:

[Specify any courses that a student must have taken to be permitted to take this course. Pre-requisites listed in this section can only be waived by special permission from the School's Curriculum Approval Officer, so they should be treated as "must-have". By default, you may assume that any student who will register for the course has taken those courses compulsory for the degree for which the course is listed in previous years.

Please include the FULL course name and course code].

Course Title: Introduction to Computation

Course Code: INF1A

Co-Requisite Courses:

[Specify any courses that should be taken in parallel with the existing course. Note that this leads to a timetabling constraint that should be mentioned elsewhere in the proposal. Please include the FULL course name and course code].

Course Title: -

Course Code: -

Prohibited Combinations:

[Specify any courses that should not be taken in combination with the proposed course. Please include the FULL course name and course code].

Course Title: -

Course Code: -

Other Requirements:

[Please list any further background students should have, including, for example, mathematical skills, programming ability, experimentation/lab experience, etc. It is important to consider that unless there are formal prerequisites for participation in a course, other Schools can register their students onto our courses, so it is important to be clear in this section. Also be aware that MSc students are unlikely to have the pre-requisite courses, so alternative knowledge should be recommended. If you want to only permit this by special permission, a statement like "Successful completion of Year X of an Informatics Single or Combined Honours Degree, or equivalent by permission of the School." can be included.]

SCE H-grade Mathematics or equivalent is desirable.

Available to Visiting Students: Yes/No

[Provide a justification if the answer is No.]

Yes

2e. Summary of Intended Learning Outcomes (MAXIMUM OF 5):

[List the learning outcomes of the course, emphasising what the impact of the course will be on an individual who successfully completes it, rather than the activity that will lead to this outcome. Further guidance is available from https://canvas.instructure.com/courses/801386/files/24062695]

On completion of this course, the student will be able to

1. Implement components of an object-oriented program, given a specification, and demonstrate the use of an object-oriented approach to enable group development of larger applications.
2. Justify implementation decisions, compare implementations, and comment on their strengths and weaknesses.
3. Demonstrate an awareness of good software engineering practice, including the use of version control, testing and readable code.
4. Locate and use additional sources of information (to include discussion with peers where appropriate) to facilitate independent problem-solving, and reflect on one's own and others' contribution to a collaborative learning environment.
5. Plan and organize time, working consistently to a goal.

Assessment Information

[Provide a description of all types of assessment that will be used in the course (e.g. written exam, oral presentation, essay, programming practical, etc) and how each of them will assess the intended learning outcomes listed above. Where coursework involves group work, it is important to remember that every student has to be assessed individually for their contribution to any jointly produced piece of work. Please include any minimum requirements for assessment components e.g. student must pass all individual pieces of assessment as well as course overall].

We do not believe that the practical programming objectives of this course can be meaningfully assessed by examination, and we intend the assessment be based entirely on the coursework. We have discussed this approach with the Vice Principal for Feedback and Assessment who is enthusiastic about the proposal and supportive of the proposed assessment. We have also surveyed other programming courses in the University and found that exam-only assessment of programming is a clear outlier: an informal survey of programming-heavy courses in other schools, found 21 such courses: 16 of which (across 5 schools at levels 8-11) are 100% coursework or take-home exam, and the remaining 5 are from 20% to 70% coursework.

We anticipate something similar to the following:

- A series of small weekly formative exercises, similar to the current Inf1-OP lab exercises.

- One exercise each week to be submitted for summative assessment (10 in total, using CodeRunner or similar). Passing the course to require a minimum number of these to be submitted (70%). Examples of students' submitted code will also be used for formative feedback on style and efficiency during tutorials.
- A first, formative assignment requiring the submission of code for a component of the framework, and a group demonstration of the running code to a tutor.
- A second, summative assignment also requiring the submission of code for a component of the framework, and a group demonstration of the running code to a tutor.
- A final 2-3 page document produced by each individual student which describes their implementation decisions and reflectively compares their code with solutions from other groups, and their experience of working within the group.

To pass the course, students would be expected to (a) submit the minimum number of weekly exercises, (b) produce a minimally working implementation for the second assignment which is reasonably well-structured and readable, and (c) to provide a meaningful written discussion of their implementation, and experiences.

Assessment Weightings:

Written Examination: _0_%

Practical Examination: _0_%

Coursework: _100_%

Time spend on assignments:

[Weightings up to a 70/30 split between exam and coursework are considered standard, any higher coursework percentage requires a specific justification. The general expectation is that a 10-point course will have an 80/20 split and include the equivalent of one 20-hour coursework assignment (although this can be split into several smaller pieces of coursework. The Practical Examination category should be used for courses with programming exams. You should not expect that during term time a student will have more than 2-4 hours to spend on a single assignment for a course per week. Please note that it is possible, and in many cases desirable, to include formative assignments which are not formally assessed but submitted for feedback, often in combination with peer assessment.]

For this course students are expected to split their time between coursework and tutorial exercises. The breakdown of hours on this 20-point course includes a total of 45 hours of time spent on assessed coursework assignments as well as 40 hours for tutorial exercises.

Academic description:

[A more technical summary of the course aims and contents. May include terminology and technical content that might be more relevant to colleagues and administrators than to students.]

We propose taking an "Objects First" 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 intend 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.

Syllabus:

[Provide a more detailed description of the contents of the course, e.g. a list of bullet points roughly corresponding to the topics covered in each individual lecture/tutorial/coursework. The description should not exceed 500 words but should be detailed enough to allow a student to have a good idea of what material will be covered in the course. Please keep in mind that this needs to be flexible enough to allow for minor changes from year to year without requiring new course approval each time.]

The syllabus will mainly be based on the book "Objects First with Java: A Practical Introduction using BlueJ" by David J. Barnes & Michael Koelling.

Relevant QAA Computing Curriculum Sections:

[Please see ttp://www.qaa.ac.uk/en/Publications/Documents/SBS-Computing-consultation-15.pdf hto check which section the course fits into.]

TBD

Graduate Attributes, Personal and Professional skills:

[This field should be used to describe the contribution made to the development of a student's personal and professional attributes and skills as a result of studying this course –

i.e. the generic and transferable skills beyond the subject of study itself. Reference in particular should be made to SCQF learning characteristics at the correct level [http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf].

Students completing this course can be expected to have developed the following personal and professional attributes and skills, beyond the study of the subject itself:

- The ability to plan and manage their time for larger programming projects and coursework in general.
- Experience working and communicating in a group of programmers involved in the same or related projects.
- The ability to search for resources and literature required for programming projects.
- Experience with presenting and communicating their own work as well as reflecting on programs written by others.

Breakdown of Learning and Teaching Activities:

[Total number of lecture hours and tutorial hours, with hours for coursework assignments.]

[The breakdown of learning and teaching activities should only include contact hours with the students; everything else should be accounted for in the Directed Learning and Independent Learning hours.

The total being 10 x course credits. Assume 10 weeks of lectures slots and 10 weeks of tutorials, though not all of these need to be filled with actual contact hours. As a guideline, if a 10-pt course has 20 lecture slots in principle, around 15 of these should be filled with examinable material; the rest should be used for guest lectures, revision sessions, introductions to assignments, etc. Additional categories of learning and teaching activities are available, a full list can be found at:

http://www.euclid.ed.ac.uk/Staff/Support/User_Guides/CCAM/Teaching_Learning.htm]

Lecture Hours: _20_ hours

Seminar/Tutorial Hours: _20_ hours

Supervise practical/Workshop/Studio hours: _9_ hours

Summative assessment hours: __30_ hours

Feedback/Feedforward hours: __4_ hours

Directed Learning and Independent Learning hours: _65_ hours

Total hours: _148_ hours

You may also find the guidance on ‘Total Contact Teaching Hours’ and ‘Examination & Assessment Information’ at:
http://www.studentsystems.ed.ac.uk/Staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html

Keywords:

[A list of searchable keywords.]

Software Engineering, Object Oriented Programming

SECTION 3 - COURSE MATERIALS

3a. Sample exam question(s)

[Sample exam questions with model answers to the individual questions are required for new courses. A justification of the exam format should be provided where the suggested format non-standard. The online list of past exam papers gives an idea of what exam formats are most commonly used and which alternative formats have been http://www.inf.ed.ac.uk/teaching/exam_papers/.]

No exam

3b. Sample coursework specification

[Provide a description of a possible assignment with an estimate of effort against each sub-task and a description of marking criteria.]

See attached document Appendix A

3c. Sample tutorial/lab sheet questions

[Provide a list of tutorial questions and answers and/or samples of lab sheets.]

Similar to the questions and exercises currently used in INF1OP but more focus on realistic applications rather than small “toy” exercises.

3d. Any other relevant materials

[Include anything else that is relevant, possibly in the form of links. If you do not want to specify a set of concrete readings for the official course descriptor, please list examples here.]

Document: A Proposal for Informatics 1B

SECTION 4 - COURSE MANAGEMENT

4a. Course information and publicity

[Describe what information will be provided at the start of the academic year in which format, how and where the course will be advertised, what materials will be made available online and when they will be finalised. Please note that University and School policies require that all course information is available at the start of the academic year including all teaching materials and lecture slides.]

TBD

4b. Feedback

[Provide details on feedback arrangements for the course. This includes when and how course feedback is solicited from the class and responded to, what feedback will be provided on assessment (coursework and exams) within what timeframe, and what opportunities students will be given to respond to feedback.

The University is committed to a baseline of principles regarding feedback that we have to implement at every level, these are described at http://www.docs.sasg.ed.ac.uk/AcademicServices/Policies/Feedback_Standards_Guiding_Principles.pdf.

Further guidance is available from http://www.enhancingfeedback.ed.ac.uk/staff.html.]

See attached document.

4c. Management of teaching delivery

[Provide details on responsibilities of each course staff member, how the lecturer will recruit, train, and supervise other course staff, what forms of communication with the class will be used, how required equipment will be procured and maintained. Include information about what support will be required for this from other parties, e.g. colleagues or the Informatics Teaching Organisation.]

TBD

SECTION 5 - COMMENTS

[This section summarises comments received from relevant individuals prior to proposing the course. If you have not discussed this proposal with others please note this].

5a. Year Organiser Comments

[Year Organisers are responsible for maintaining the official Year Guides for every year of study, which, among other things, provide guidance on available course choices and specialist areas. The Year Organisers of all years for which the course will be offered should be consulted on the appropriateness and relevance on the course. Issues to consider here include balance of course offerings across semesters, subject areas, and credit levels, timetabling implications, fit into the administrative structures used in delivering that year.]

5b. BoS Academic Secretary

[Any proposal has to be checked by the Secretary of the Board of Studies prior to discussion at the actual Board meeting. This is a placeholder for their comments, mainly on the formal quality of the content provided above.]