All MSc students should be able to program well by the time they leave the School of Informatics. Most students have some previous programming experience, but not everyone is confident of being able to write a complete, realistic application from scratch. IPPO is intended to help develop these skills.

This involves more than just studying a programming language: it requires an ability to make good design choices; use appropriate tools; write code which is reliable and readable; integrate with other systems - which may be large and/or poorly-specified; and to locate all of the necessary information.

Students are expected to work independently, but the course supports this by providing online materials, supported lab sessions and a responsive online forum. Most students succeed in developing a significant application from scratch, and this forms the basis of the assessment.

Many students with no previous experience have taken the course in the past and successfully learned to create real, useful programs - however, this is likely to be challenging and will require additional time.

1 Activities

Lab Sessions: The lab sessions provide an opportunity for on-campus students to work on exercises or assignments with demonstrators on hand to answer questions and help solve problems. The demonstrators won’t always provide direct answers, especially to assessed material. And sometimes, they simply won’t know the answer! You should think of them as a resource to help you make progress, but think of yourself as being in charge of your own learning.

This year, the lab sessions will be managed by the central University system. The allocated sessions should appear directly in personal Outlook calendars, as well as being available on MyEd. A form will be available to request changes. You should contact the teaching office if you have problems with this.

Online Sessions: Support for online students will be provided mostly via the Piazza forum (see below). This allows students to participate at a convenient time, and discuss specific issues privately with a tutor. There will also be an online group session to introduce the course, online demonstration sessions (see below), and further online sessions if there is a general demand for group discussion of a specific topic.

These remote sessions will use Blackboard Collaborate1.

Piazza: The Piazza2 online forum is an important element of the course: it is monitored regularly by the course staff, and this is your opportunity to get answers to questions without waiting for the next lab session. Students are encouraged to reply to questions as well, and it is not uncommon to get answers within a few hours, even during the evenings or at weekends (but don’t rely on this!). Piazza is also used for course announcements, tips on the assignments and general discussions.

In the “real world” forums such as StackOverflow3 are an indispensable resource for developers, so Piazza provides you with a similar experience in a less-exposed setting.

You should make sure that you are registered for the forum as soon as possible after deciding to take the course, and you should check regularly for new announcements. Please sign up for Piazza using your

1https://www.ed.ac.uk/information-services/learning-technology/communication/collaborate
2https://piazza.com/ed.ac.uk/fall2019/ippo
3https://stackoverflow.com/
University email address, so that we know if someone is not signed-up (and therefore not getting the course announcements).

**Lectures:** A full course of lectures would not be helpful for IPPO: the range of possible topics is very large, the students have a very wide range of different starting points, and practical work is a more valuable use of the available time. However:

- There is an initial lecture to introduce the course and discuss the first assignment.
- There is a second lecture to provide feedback on the first assignment, and discuss the second assignment.
- There is a third lecture to discuss the design of the second assignment.

The lecture recordings are available under the IPPO course which should appear in your page for the University’s Learn system. Note that this is the “on campus” version of the course - all students should be able to access this (even online students) and the videos are only available under this course code. Unfortunately, direct links to the videos do not work reliably. There will be a delay of 24hrs between the end of the lecture and the release of the video.

**Demonstrations:** All students will have the opportunity to demonstrate their assignment code to a demonstrator or tutor. This will allow you to explain any problems with the code and to get some verbal and written feedback. For the second assignment, the demonstration will involve small groups of students. This will allow you to see a range of alternative solutions to the problem, and to get extra feedback from your peers.

On-campus students will be asked to demonstrate and describe their running code to a lab demonstrator in the lab session following the assignment submission. We strongly recommend that you do this: if you do not attend the demonstration, code which we are unable to compile and run (for any reason) will not be awarded any marks. These sessions are usually busy and difficult to schedule, so please arrive on time for your allocated session.

The demonstrator will not be marking your assignment. However, we will refer to the demonstrator’s notes if, for example, we cannot get your submitted assignment code to run on our systems. You may be asked to provide a second demonstration to a marker – for example if there appears to be a discrepancy between the demonstration notes and the submitted code.

We recognise that online students may find a live session difficult to schedule, so we will only ask you to do this if we have difficulty with your submitted code. However, we will attempt to accommodate anyone who would like to make an online demonstration to receive some additional feedback.

# Resources

It is an important outcome of the course that students should be able to locate the resources that they need among the vast amount of information which is available. So, there is no attempt to provide detailed material on the Java language, or JavaFx, for example, since these are well-covered elsewhere.

However, we provide some notes, and we recommend a textbook:

**Notes:** We intend to provide a collection of short notes which are largely independent, so that students can find a concise description of a particular topic, or a solution to a specific problem which is not well-explained elsewhere. This collection will grow throughout the course and we would welcome feedback.

---

4[http://www.learn.ed.ac.uk](http://www.learn.ed.ac.uk)

5It is not easy to write good “portable” code and it is fairly common for students to submit applications which work on their own machines, but not on ours.

---

Paul Anderson 06/11/2019
on suggested topics, as well as the content of the existing notes.

**Textbook:** There are a huge number of books on Java. The course textbook takes a particular approach which introduces the concept of “Objects” right at the start. It also provides an environment (BlueJ) which allows you to experiment practically (graphical) with these objects without the need to learn a lot of other details first. We strongly recommend this.

**Videos:** We will be providing some short videos to illustrate particular topics which benefit from an interactive presentation. However, these are expensive to maintain since they need re-recording every time that the software is updated. They are also difficult to search. So we prefer to provide written notes for most topics. However, feedback on this is welcome.

### 3 Assessment & Feedback

The final mark for the course is based on the two assignments, weighted 30% for the first assignment and 70% for the second assignment. The second assignment is submitted in two parts, but these parts are not marked independently.

**The Marking Scheme:** The marking scheme is based on a number of criteria, all of which are important aspects of any real software application. These criteria are explained on each assignment sheet. The final mark is intended to reflect how well the overall assignment meets these criteria. In general:

- The final mark is not derived from summing the components - each grade will require a good standard for *all* of the relevant criteria. For example, code which is difficult to read, or poorly structured will not be awarded a very good mark, regardless of how well the code “works”, or how nice the interface looks.
- Higher overall marks require a deeper understanding to be demonstrated. This will usually involve a good, well-written justification on the worksheet, as well as an implementation in the code.

A number of staff will be involved in the marking process, and the final mark is based on their academic judgement. Reviewing individual marks would be unfair, and marks are not negotiable (unless of course, there is clear evidence of some procedural error).

University regulations require us to use the Common Marking Scheme⁶. This is a frequent source of confusion for students who are used to a different scheme: an excellent (first-class) solution to the assignment will normally be awarded a mark of around 80%. Marks higher than this are rare and can only be awarded for work which is (for example) “beyond the expected level”, “demonstrates professional scholarship” or “significant personal insight”.

Students with some previous programming experience, spending the recommended amount of time on the course should expect a mark in the 60-69 range. In the past, more than half of the students have received marks in this range. Students with more or less experience or effort typically receive marks in the 50-59 or 70-79 range. A very small number of students receive marks which are higher or lower than this.

**Good Scholarly Practice:** Please remember the University requirement regarding all assessed work for credit. Details and advice about this can be found on this page⁷ (and links from there). In particular, you are required to take reasonable measures to protect your assessed work from unauthorised access. For example, if you put any such work on a public repository then you must set access permissions appropriately (generally permitting access only to yourself).

---

⁶[https://web.inf.ed.ac.uk/infweb/student-services/ito/students/common-marking-scheme](https://web.inf.ed.ac.uk/infweb/student-services/ito/students/common-marking-scheme)

⁷[http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct](http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct)
Reading and discussing other people’s code is extremely valuable when learning how to write real programs. However, when you submit code for assessment, your own contribution must be completely clear. For example, if you use code derived from elsewhere, or developed together with a colleague, you should comment the relevant section in the code, and explain this clearly on the worksheet.

We routinely run software on the IPPO assignments which is extremely effective at detecting sections of code or text that have been derived from some other source (even when attempts are made to disguise this). If this software detects any unexplained similarities between assignment submissions, we will pass them directly to the Academic Misconduct Officer as a case of suspected plagiarism.

**Feedback:** The School of Informatics has a Feedback Pledge[^8].

Demonstrators are available throughout the course to provide feedback on your ongoing work during the lab sessions. Please do ask them for their views on your code - even if it apparently works, as it may not always be good in terms of correctness or readability (for example).

Written feedback will be provided on the assignments. This may be brief, but it should be clear from reading this in conjunction with the marking scheme how the final mark has been derived, and what might have been done in order to improve your solution.

### 4 Some Tips

Here are some tips for getting a good mark in this course (and for being a better programmer). In no particular order:

**Write clear code:** Programming languages are a way of communicating algorithms and concepts between humans as well as between humans and machines. Code needs to be well structured and clear and readable if it is to be reliable and maintainable. No matter how clever an application looks when it is running or what “features” it has, it won’t get good marks (and I wouldn’t give you a job!) unless the code is clear.

**Exploit the Java language:** If you have done some programming before, don’t just rely on the paradigms of your previous language. Take time to learn how to use the features of Java appropriately. For example: think carefully before using arrays and integers to represent your data - consider using objects, sets, hashmaps and enumerated types to create better representations. Use iterators where appropriate, instead of for-loops.

**Think about structure and design:** Spend time thinking carefully about the overall structure and design of the classes. If the design is not good, the code will be harder to write, less readable, less reliable, and difficult to reuse. The first assignment is intended to show you how to think about this. A good mark on the the second assignment depends heavily on your own design.

**Read the instructions:** Read the instructions carefully. Read them again. Even if you produce a very good solution you won’t get good marks if you have solved the wrong problem, or done something in the wrong way.

**Trust in your own ideas:** Beware of sharing. Discussing ideas is good, but other students sometimes have really bad ideas - you may get better ideas of your own just by following the course materials! Discuss your ideas with the demonstrators, or post them on the forum. But remember to maintain good scholarly practice - if you use a contribution from someone else, make it absolutely clear.

[^8]: [http://www.inf.ed.ac.uk/student-services/teaching-organisation/for-taught-students/feedback](http://www.inf.ed.ac.uk/student-services/teaching-organisation/for-taught-students/feedback)
Communicate your ideas clearly: The worksheet is just as important as the code for the assessment. Spend time reading the questions carefully and answering thoughtfully and concisely. Glib answers are really bad - for example “my code is cohesive” without any justification.

You must be able to express yourself clearly in English. Language and grammar is important for a good mark, even if this is difficult for non-native speakers - this will be even more important for your MSc dissertation. Even for good English speakers, expressing things concisely can be difficult. Spend time on this. Ask other people to comment on the English language in your written work.

Take advantage of the course support: This course has no (content) lectures. The real value comes from the demonstrators and the staff who respond to your questions on the forum. This allows you to get help with your own individual problems, so take advantage of this.

Have fun: Learn to write good code. Don’t be over-focussed on the assessment ...