This note provides some general information about the IPPO course, including a description of the resources, activities, assessment, support available, and some tips to help you succeed.

1 **What's This Course About?**

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. And these languages and tools all change at a dizzying rate. If you would like to understand how the course relates to real programming practice, you should watch the Video[^1] where professional programmer Jim Paterson discusses these topics with Paul Anderson.

This means that is is not sufficient for us to teach specific languages or tools - even if the time were available to do so. Instead, we try to help students develop the generic skills necessary to locate, evaluate and use the tools and information that they need to complete a realistic development task.

This is challenging, and we understand that it can be quite intimidating. It involves a lot of practical activity (rather than more passive lectures) and independent work. The value of the course is in the support that we provide for this - particularly the responsive forum and individual tutor discussions, but also the course materials and carefully scaffolded assignments.

The course uses the Java language because it encourages a natural use of the object-oriented design principles which we think are important. We use the JavaFx library because it is good example of a complex library which is perhaps less well-known, so it presents all of the challenges typical of a real software development situation. However, the importance of the course is on the generic skills which will apply equally to other languages and toolsets.

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 will almost certainly be challenging and will require significant additional time.

We are definitely open to any suggestions that could make the course more effective for you, so please do raise these on the Piazza forum.

2 **Activities & Support**

Developing software is a practical skill which takes time and experience. So, the course is based around two project assignments which are carefully designed to lead students through the development of a realistic application. There are no traditional lectures or tutorials, but we provide a high level of reactive support to help students work through these assignments at a pace which suits their own experience.

[^1]: [https://media.ed.ac.uk/media/Jim+Paterson+talks+to+Paul+Anderson/1_d98b05wc](https://media.ed.ac.uk/media/Jim+Paterson+talks+to+Paul+Anderson/1_d98b05wc)
IPPO has been a “hybrid” course for several years with some students taking the course remotely, and some students physically on-campus. We will be taking advantage of this experience to offer the course remotely to all students this year.

**Live Sessions:** We will hold (synchronous) online sessions most weeks using Blackboard Collaborate². It is likely that one or two of these sessions will be formal lectures covering a specific technical topic - such as the design of the second assignment - but most sessions will only start with a short presentation and then open for questions. The sessions will be recorded and made available (only) to students on this course.

This is an informal opportunity to cover those issues or topics which you think may benefit from a more interactive discussion or explanation. You should have looked at the week’s material in advance and be prepared to ask any questions that you might have - these can be raised during the session, or submitted to Piazza in advance. If there are no questions, the sessions may be very brief!

You will need to access the live sessions via the Learn² course (available from the menu on the course website). On the Learn page, you should see Live Classroom in the left hand menu. If this entry is not present, it probably because you are not logged in to Learn - you will need to login (top right) and return to the same page.

²https://www.learn.ed.ac.uk/webapps/blackboard/execute/launcher?type=Course&id=_83906_1&url=

We will use the same Course Room for all of the sessions, and this will be available continuously, so you are free to use this at other times to meet with other students.

**The Piazza Forum & Beyond:** We will use the Piazza Forum as the starting point for providing group and individual support on a number of levels. This is a process which has proved popular and successful with our remote students in the past:

1. We usually ask students to start by posting any questions to the public forum. This is likely to receive the fastest response since it is monitored by staff and visible to other students (possibly in different time zones) who may all be able to provide a helpful reply. It also makes the responses visible to other students who may have the same question.

2. Sometimes it is not appropriate to post a public question - if it involves details of an assessment solution for example. In this case, private Piazza messages can be used which are visible only to the instructors. This restricts the number of people who can answer or take advantage of the response, so we prefer to use public messages where possible.

3. Of course, there are times when a more synchronous communication really is helpful - in trying to understand a more general concept, or to trace a specific bug for example. So it will be possible to agree an individual online conversation with a tutor at a time which is convenient, and using a technology which is convenient for the student and the tutor.

The individual sessions are intended to replicate the experience which would be available in a physical lab session, and the amount of time that a tutor will spend with any individual obviously depends on the overall demand. This will be higher in the periods immediately before assessment deadlines. We would also like to make the most efficient use of the individual sessions, so tutors may suggest some further work or investigation from the student before agreeing to an individual meeting.

Notice that there is a range of people to provide you with support, and for any non-trivial question you

²https://www.ed.ac.uk/information-services/learning-technology/communication/collaborate

Paul Anderson 17/09/2020
may get a range of different opinions - some of which may even conflict! This is a very realistic situation and it is your responsibility to interpret the responses critically. We’d also like to help you learn how to diagnose your own problems, and find your own information, so the tutors will often make suggestions of things you should try yourself, rather than simply providing direct answers.

3 Resources

It is an important outcome of the course that students are able to locate the resources that they need among the plethora 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:

Notes: We 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 is expanding and we would welcome feedback 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. For these reasons, we strongly recommend this book although we do recognise that this unfortunately introduces an additional tool.

Videos: In general we (and previous students) prefer textual material to videos - it is much easier to search, easier to update and more accessible. However, there are a few topics, particularly involving the graphical interface builder where is is useful to see the on-screen process, and we provide some videos specifically for this. We will also suggest some videos which are available elsewhere when this is relevant.

4 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 GettingStarted tasks are considered to be part of the first assignment for assessment purposes. The second assignment also includes a preliminary design task and a requirement to submit preliminary code to the git repository: these must be completed by specific dates, ahead of the final assignment submission.

Demonstrations: Even with a “portable” language such as Java, writing code which runs reliably on different systems is not easy. In a small number of cases we may be unable to run the code that you have submitted on our own machines - even though it apparently works well on yours. We will not attempt to “fix” this - instead we may ask you to discuss and demonstrate your code on your own system via an individual online session. We may also request an online demonstration/discussion if there are other aspects of your submission which are unclear.

Note that these demonstration sessions cannot be arranged until we have attempted to mark the assignments, which will be after the final week of the course.

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 by 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
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 Scheme[^3]. 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 page[^4] (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).

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[^5].

Tutors are available throughout the course to provide feedback on your ongoing work via Piazza. 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. We will be happy to discuss this with you where it is not clear.

[^3]: https://web.inf.ed.ac.uk/infweb/student-services/ito/students/common-marking-scheme
[^4]: http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct
[^5]: http://www.inf.ed.ac.uk/student-services/teaching-organisation/for-taught-students/feedback
**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.

**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 tutors 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 ...