Introduction to 1007 and 1008
2005/6 Session

Lecturer: Dr. Graham Roberts
(Room 6.17, email G.Roberts@cs.ucl.ac.uk)

Please read all of this carefully

1. Introduction

1007 and 1008 are the first year programming courses. 1007 in term 1 introduces programming using two contrasting programming styles; imperative programming using the Java programming language and declarative or logic-based programming using the Prolog programming language. 1008 follows in term 2 and covers object-oriented programming using Java.

1007 and 1008 are both half-unit courses, giving you a total of one unit of programming in the first year. This means that learning to program counts for one quarter of your work during the year (you are taking four units worth of courses overall this year, most or all of which are normally a half unit in value). 1007 Principles of Programming runs during Term 1 (Oct/Nov/Dec 2005), while 1008 Object-Oriented Programming runs during Term 2 (Jan/Feb/March 2006).

1007 is taught by myself (Java) and Mohamed Ahmed (Prolog), aided by a team including several research fellows who will run problem classes, and research student demonstrators who will run lab classes. The course has 40 timetabled lectures (4 per week) allowing plenty of time to cover the material with some spare. Lectures are supported by the weekly problem classes and labs. Please check the current timetable for details of times and locations (detailed timetables are on the CS teaching web).

1008 is taught entirely by me and is organised in much the same way as 1007. There will also be around 30 lectures, supported by problem and lab classes.

1.1. Course Objectives

The primary objective of the two courses is that you should learn how to program using the Java programming language and also have some experience of logic or declarative programming. The lectures, as well as introducing the two programming languages, will also look at what programming in general is about, cover some underlying ideas and concepts relevant to imperative and declarative programming languages, and demonstrate the process of program design in the small scale.

Both courses are primarily of a *practical* nature, emphasising programming and problem solving skills, so include problems classes, labs and lots of programming exercises.

Specifically, by the end of the academic year you should:

- Have a good understanding of imperative programming and basic object-oriented programming.
- Have an understanding of logic-based programming.
- Be able to design, implement and test a small-sized Java or Prolog program given a simple specification and applying suitable problem solving techniques.
- Have learnt to use a large subset of the Java programming language effectively.

1.2. Your Role

A key assumption underlying the courses is that you must take responsibility for learning to program, learning the programming languages and for getting plenty of programming practice. We can aid you via the lectures, problem classes and labs but you must take the lead and drive your own progress. Unlike the school environment you may be used to, university life requires you to organise and make best use of your own time without someone constantly looking over your shoulder. Also you need to develop the self-discipline to do the detailed study needed to master a subject, particularly something like programming.

As the courses progress, you will need to spend time studying the lecture notes, the course text books and working through the sets of programming exercises provided. The lectures will cover many programming ideas and concepts but are not going to describe everything about the Java language in blow-by-blow detail. You have to fill in the missing pieces. Above all, however, programming is a practical subject and you must practice programming continually in order to master it.

The best way to learn how to program is to write lots of programs!

2.1007 Problem Classes

During the first week of teaching you will be allocated to a 1007 problem class group. Each group has one timetabled class per week that you are required to attend — attendance at your problem class is compulsory. If you look at the timetable you will see a three entries labeled as 1007 problem classes; your group attends one of them (not all of them!).

Problem classes are run by Research Fellows. Each problem class will be based on a collection of problem questions that range from easy through to very hard. You should aim to do as many as possible, continuing as necessary after the problem class. The questions are deliberately chosen to be challenging and make you think. No answers are given; you should develop and use your judgement as to whether your answer is good enough. The people running the class will give you advice, hints and help as necessary but don't expect them to think for you or give you answers! Problem classes are not about writing complete programs — there are no computers in the class rooms — instead they focus on the problem solving skills needed to develop algorithms and solve program design problems.

3. Lab Classes and Programming Exercises

In addition to problem classes, you will also be allocated to a 1007 lab class group. Each group initially has one timetabled lab session per week that you are required to attend. If you look at your timetable you will see that a series of 1007 lab sessions appear; your group attends one of them (again, not all of them!).

All lab classes are held in the 1.05 (1st floor lab) and are run by lab demonstrators. The purpose of a lab class is to work on your programming exercises while sitting in front of a workstation and having direct access to a demonstrator for help, advice and feedback. You should also work on the exercises at any other time the lab is available (i.e., not booked for a lab class), or on your own computer at home or in halls. However, the timetabled lab is a time when you will be able to get help while having access to a workstation.

You will be given sets of programming exercise questions at regular intervals, requiring you to design and implement small-sized programs. Each exercise sheet will contain a set of core questions that you must answer in order to be sure that you are keeping up with the course. The rest of the questions on the sheet will be progressively harder in order to provide a greater challenge. Do as many of these harder questions as you can and that you have time for.

If you complete all the exercises, you are encouraged to either find more (perhaps from the course text book) or create your own programming challenges to work on. You can never get enough programming practice!

Exercise questions are not formally marked and do not count directly towards your final course marks. However, you can get your work reviewed during problem classes by one of the lab demonstrators. The demonstrator can both help you when you get stuck and also tell you when you are doing things correctly. Having your work reviewed by knowledgeable programmers and talking about the design and implementation of your programs is an important aspect of learning to program, so make sure you take advantage of the lab sessions.

A lab class provides a focal point for your exercise work and it is expected that you will attend. Programming problems can also be discussed during tutorials with your academic tutor. As the term goes on, and depending on your progress, additional lab sessions and possibly tutorial sessions may be arranged for you. Also, some lectures will be used as tutorial or problem solving sessions.

The purpose of having lab groups is primarily to avoid everyone turning up at a lab session at the same time. Normally you should stay with your group and don't need to attend other lab sessions. However, if you get stuck and need help to make progress, then go to the next available lab class. Don't simply wait a week without getting anywhere.

Note that the lab demonstrators are not there to do your work for you and simply tell you what to do all the time. When you ask for help, be as precise as you can and try to explain in detail what the problem is. Surprisingly often, in trying to clearly explain a problem to someone else, you realise what the answer is. Don't be shy about asking for advice and feedback, but be prepared to do the thinking. Talking about your work is a good thing.

To learn to program it is essential that you spend plenty of time actually programming. Don't be afraid of making mistakes, you often learn more by making and then correcting a mistake than by getting something right first time. You cannot expect to learn to program and pass 1007/8 just by sitting through lectures or trying to do last minute revision before exams.

Remember — programming exercises are not to be taken as optional. You must make the time to work through at least the core questions. If you don't keep up with the exercises you will fall behind in the course and find it very hard to catch up.

4. Assessment

Both 1007 and 1008 are assessed by a combination of exam and coursework, with most marks coming from the exams.

As programming is considered to be an essential core skill for computer scientists, you must pass 1007 in order to progress to the 2nd year of your degree programme. This means that you must achieve 40% or more as your final course mark and, in addition, you must separately pass the exam and coursework elements of the course.

If you do not pass 1007 then you will either have to transfer to a different, non-Computer Science, degree programme, or take a year out and retake the exam or even the whole course. See http://www.cs.ucl.ac.uk/teaching/ug/assess.htm for the details (this applies directly to CS students and is essentially the same for MACS students in the first year).

Your progress on the course is assessed in various ways; by exercises, examination, programming tests and mini-projects. The following subsections explain what each involves.

4.1. 1007 January Mid-Session Exam

At the start of term 2, in January 2006, a mid-session exam will be held to assess your overall progress during term 1. The mark from the exam is not used as part of the final course mark but is used to review your progress. If you achieve a low or failing mark, then you will be given additional lab and tutorial help in order to recover and catch up. It is our expectation that most students will pass without problem but if you are having difficulties it allows plenty of time to remedy the situation before the final exams.

4.2. Final Exams

1007 and 1008 both have a final 2.5 hour exam that takes place during the exam term (term 3, April/May 2006). The 1007 exam contributes 90% of the overall course mark, while the 1008 exam contributes 85% — doing well on exams is very important. To pass either course as a whole you must pass the exam, which means you need to obtain a minimum of 40% for the exam on its own (the basic pass mark for all exams is 40%). Anything covered in the course syllabus, lectures, problem classes, lab classes and course text book can be considered examinable. Don't just assume that the material covered in lectures only is all you need to know.

The exams will include a compulsory section where you will need to demonstrate your programming and problem solving skills (indeed, all exam questions involve some form of problem solving). If you have successfully completed the programming exercises and other coursework, this section will be relatively straightforward. If you are lazy and skimp on the exercises you will find the compulsory section hard and will seriously risk failing the exam.

4.3. Programming Mini-projects

There will be mini-project courseworks for both courses. A mini-project involves designing and writing a larger program than those needed for the programming exercises, putting into practice all that you have learnt so far. Projects are marked by grading, from A to F, where C is considered satisfactory, A and B progressively better, D and E progressively worse, and F a fail. Satisfactory means that the program works and has an adequate design and implementation.

4.4. Programming Tests

In each half of each term a formal test will be held during one of the lectures. These will be announced before hand and you must attend the lecture. If you are unable to attend or are ill, you must contact the departmental tutor (me, see contact details in the document title) at the earliest possible time to let me know why you were absent. If you have no reason to be absent, you will be given a mark of zero.

A test is 30 minutes long and contains a mixture of multiple choice and short answer questions, some of which will require you to write short sections of program code. Each test will cover the material dealt with by the exercises and lectures to date. If you have done the exercises properly the test will be straightforward.

The primary purpose of these tests is to act as milestones to measure your progress. They give both you and the teaching staff direct feedback of how well you are doing on the course. They also serve as a warm-up for the exams. While failing a test is not a complete disaster, it is a wake-up call that you need to study harder and catch up.

4.5. The Overall Coursework Component

The marks you obtain for the programming tests, mini-projects and any other courseworks will be combined to form a single coursework component mark for each course. This accounts for 10% of your overall final mark for 1007 and 15% for 1008. To pass either course as a whole you must make a proper attempt at doing the coursework. This means that as a minimum you must submit answers to at least 35% of the coursework set. The normal expectation is that you will submit answers to all coursework set, otherwise the Departmental Tutor (me) will want to know why.

Once marked, all coursework is returned to you via your tutor. You should keep all your marked work from all courses, as you may be asked to re-submit it after the final exam so it is available during the examination marking process. Think of this as a building a portfolio to show your progress and what you are capable of. It is a good idea to have a special folder in which you keep your marked work as it is returned to you during the course of the teaching terms.

5. Plagiarism

Plagiarism occurs when you copy another person’s work and submit it under your own name without acknowledging who did the work. In other words cheating.

You must not commit plagiarism.

If you are found to have committed plagiarism you will be reported to the Chair of the Examinations Board who will take strong action against you. You will also get no credit for the work and risk being failed on the coursework component (and, hence, the entire course).

Make sure that you read and understand the official plagiarism policy.

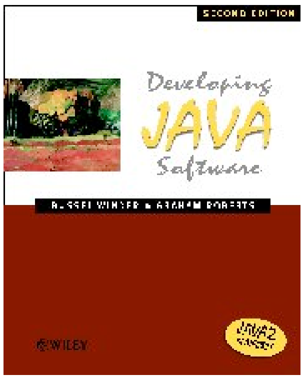
Having said all that, there is no harm in normal discussion with one another and in talking about the programming you are doing, provided any collaborative efforts are properly acknowledged. Moreover, discussing programming ideas and concepts, exchanging tips and talking through design issues are an essential part of the learning process — we certainly don’t want to discourage you from doing that. Good communication skills and the ability to discuss ideas with others are essential abilities for a programmer.

However, don’t be lazy and simply expect to get answers or the key elements that solve a programming problem from someone else. Learning to program requires a lot of practice and time spent developing your problem solving skills. Getting answers from someone else means you will not get the practice you need — and that is something you can’t copy off someone else. Programming uses many of the same kind of skills you use when solving mathematical problems, one of the reasons we require high grades in A-level or equivalent mathematics.

In the end it is your responsibility to do the exercises and coursework, and to learn how to program. If you are lazy and you don't do the work properly you will almost certainly end up failing the coursework tests and exams, as you will won't have the understanding, knowledge and experience needed. In our experience most people who fail programming exams fail for exactly that reason, so this is not simply a scare tactic!

6. The Course Text Book for Java Programming

The recommended Java course text book is:



Developing Java Software, 2nd Edition, by Russel Winder and Graham Roberts, published by John Wiley & Sons Ltd., 2000 ISBN: 0-471-60696-0

Make sure you get the 2nd edition.

There are a number of introductory books on Java programming available, so you can choose a different book if you like but choose carefully. The recommended text or an equivalent should be considered required reading as it contains essential material and a lot of detail that will not be covered in lectures (there is not time for a start). It includes a full introduction to object-oriented programming and a Java language reference. No separate lecture notes will be available during the course as you are expected to have a text book. The recommended book covers all the material for both 1007 (Java part) and 1008, so no additional book is needed next term. It will also be needed in the 2nd year.

Waterstone's Book Shop is a few yards away from the CS department, and usually has a large number of Java text books in stock. There is a good collection of programming text books in the college Science Library (next door to the CS building) but they are also in high demand, so you may have to wait before you can get hold of the ones you want. Do make sure you familiarise yourself with the library and the location of the Computer Science collection.

Expect to do a minimum of 2-3 hours a week reading about programming. In addition, there are many on-line web tutorials, information sources and even complete books, so make good use of them as well. A good place to start is http://java.sun.com or http://java.net. These are websites run by Sun, the company that invented Java. The first site has extensive tutorials on learning and using Java and is also the place to go to for downloading the latest versions of the Java development tools.

7. Your Commitment

To give you an idea of the amount of effort you should put in to 1007 and 1008, each course is valued at 150 hours of work. Some 25-30 of these hours are taken up by lectures (a little more on 1007) and an estimated 60 hours by exercises and coursework. The remaining 60 hours are meant to be used for your own programming practice, reading, revision and so on.

This is a substantial amount of time (at least a day and a half a week) and you need to use it well. It is important to practice programming, read the text book(s), learn about Prolog, Java and develop your understanding of object-oriented programming.

8. Relationship to Other Courses

1007 and 1008 are part of the core Computer Science programme and integrate closely with other courses you take this year and next year. You should look for the connections between courses in order to see the larger picture.

Along side the programming courses you will be taking the theory courses 1B12 and 1B13. These will start to cover the mathematics behind programming, giving you a more formalised picture of what programming is. They also include a study of algorithms and data structures, which will complement the material in 1008 where we look at how algorithms and data structures are implemented using a programming language.

In the 2nd year, 2007 (Concurrent Programming) is the continuation from 1008 and will look at how larger programs are designed and implemented, and how to design and write concurrent programs. It too will involve lots of programming, as will 2010 the Compilers course (CS students only). The Software Engineering and Human Interaction course 2009 is also related to 1007/8 and 2007, looking at how larger software systems are designed and developed. Course 2011 Networks, Databases and Graphics will draw on your programming skills, particularly for the graphics component.

In the 3rd year, CS students will have a further programming course, be doing a group project and, if a BSc (or BScMACS) student, do an individual project. All these require good programming skills, particularly the projects, which require a substantial programming element.

9. Further Information

Further information about the course will be given during lectures as required. In addition, the 1007 Part I web pages are at: http://www.cs.ucl.ac.uk/staff/G.Roberts/1007/index.html, and will be progressively updated during the term.

A few years back first year programming was taught as a single one-unit course numbered 1B11, then more recently as two half-unit course numbered 1B1a and 1B1b. The Java material taught was similar to what will be covered in 1007/8. Past exam papers for 1B11, 1B1a and 1B1b are available via the past exam papers web page, if you want an idea of what a programming exam will look like.