GC01:
Introductory Programming

Lecturer = Lewis Griffin

email: L.Griffin@cs.ucl.ac.uk

office: MPEB 5.02

Course homepage:

http://lewis.d.griffin.googlepages.com/gc01introductoryprogramming

Course Structure

- Primary Aim = learn to program.
  Second Aim = learn to program in Java.

- **Intensive**. Runs for 1st half Term 1 only (plus 4 extra lectures in week 0).

- **Lectures** (6 hours/week) =
  Mon 11-1; Tues 2-4; Wed 9-11

- **Labs** (2 hours/week)

Email Registration

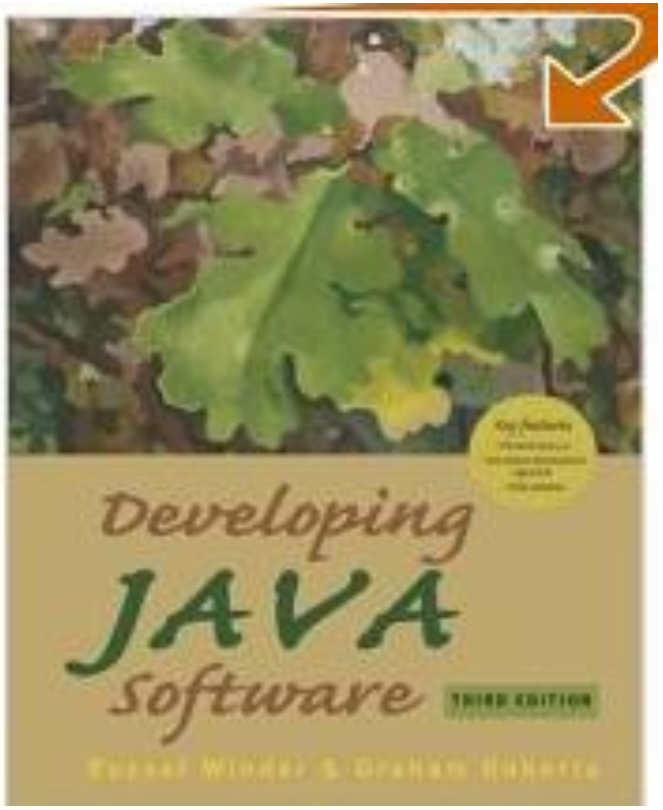
Make sure you register on the GC01 mailing list.

Send an email to GC01-request
Type *join* on the subject line
Only register from a CS dept. machine

- **Check your email daily for changes or announcements. Essential. Not checking is not an excuse.**

The recommended Course Text Book

Winder R & Roberts G
Developing Java Software,
John Wiley & Sons (3rd ed.)



Available at Waterstone's (£37) or Amazon (£20-£30).

Web Resources

- http://lewis.d.griffin.googlepages.com/gc01introductoryprogramming
- http://academicjava.com/
- (excellent, free, and UK. Just the right level for you. Download the interactive tutor. Essential!)
- http://java.sun.com/docs/books/tutorial/java/
  (excellent online tutorials, a highly recommended complement to lectures).
- http://www-128.ibm.com/developerworks/edu/j-dw-java-intjava-i.html
  (a short but substantial tutorial, requires registration (easy))
- http://hebe.cie.uce.ac.uk/soc/InfoC/SelfTestInfoC.nsf/Classroom%5CBy%20Category?OpenView&Start=1&Count=30&Expand=7#7
  (excellent series of programming exercises, starts at very basic level, gentle ramp).
- Type ‘learning java’ into Google
  (many other sites out there, worth exploring).

If you find a good site, send me an email and I will disseminate it.

On the course home page

(as the course goes on)

- ***GC01 - Introductory Programming***
- *Exercises*
- 1
- *Lecture Notes*
- 01 - Introduction
- 02 - Programming I
- 03 - Imperative Programming I
- 04 - Imperative Programming II
- 05 - Arrays & Containers
- 06 - Scope
- 07 - Strings
- 08 - Classes & Objects - A quick first look
- 09 - Exploiting Abstraction
- 10 - Top Down Programming
- 11 - Creating Classes I
- 12 - Creating Classes II
- *Additional Notes*
- Simple Drawing
- Simple Drawing 2
- Keyboard Input
- File Input
- File Output
- *Additional Resources*
- Java at CS pages

Course Objective: Learning to Program

- The best way to learn how to program is to write programs!
- So, a large part of the course is based on lab classes.
- Lectures will support and expand but also explore a wider range of subjects.

What do you need to know to get started?

- *No previous programming experience* is assumed - we start from the beginning.

- You are assumed to be a bit familiar with using the workstations, because of the Unix introduction course.

Note Taking

- You will get copies of lecture slides, so you don’t have to write down everything displayed.
- BUT, do make brief additional notes to help you remember what was said.
- Not everything I say will be on a slide.

When are the labs?

- MSc CS – Monday 2-4
- MSc FC – Monday 4-6

All labs in MPEB 1.21

You *can* attend the other MSc's lab sessions,
but **only** if a workstation is free.
This will be enforced by demonstrators.
Abuse will lead to removal of option.

What happens in a lab class?

- You work on your programming exercises.

- Demonstrators will be present to:
  - give you help and advice while you program.

Should I write programs only in the lab class?

- No - you can and should use the lab any time it is not being used for timetabled classes (or use your own PC).

- Lab classes are where you can get direct, “in front of the screen”, help.

Why have labs at all?

Because:

- It is *your responsibility* to learn Java and develop your programming skills but some of you will benefit from support.
- Lectures can only give you so much knowledge - the rest has to come from practice and experience.
- Programming is like piano playing – technical understanding + practice + talent.

Using your own PC

- You *can* use your own PC for doing programming exercises.

- It is your responsibility to maintain your PC and backup data.

- “My PC is broken” is not an excuse!!

  (The facilities we provide are entirely adequate - owning a PC is useful but not essential.)

Where does one get Java software?

- We use the version of Java called standard edition 1.4.2
- The software is available on the web (www.sun.com)
- You also want a programming tool. We recommend BlueJ or Eclipse.

Assessment

- The course is assessed by both exam and coursework.
  - The exam counts for 90% of the overall mark.
  - The coursework counts for 10% of the overall mark.

- To pass the module, you must:
  - obtain at least 50% overall.
  - obtain at least 50% on the coursework.

- See http://www.cs.ucl.ac.uk/teaching/schemes_of_award/MScComputerScienceSchemeofAward.pdf for definitive statement.

Coursework Exercises

- Exercise sheets will be handed out every 1 or 2 weeks.
- Typically 8-12 questions.
- *Core questions* are compulsory and must be answered if you are to keep up.
- *Additional questions* should be done to push yourself forward.

Plagiarism

- Plagiarism = Copying someone else's work.
- Copying = Cheating.

DON'T DO IT.

It is dishonest, shameful and risky.

Exercise Marking

- Print out program + cover sheet.
- Fill in cover sheet.
- Binary marking used (OK or not OK).

The Exam

- Held during term 3 - the exam term (May/June).
- Lasts 2½ hours.
- Answer 3 questions from 5.

How do you pass?

- Keep working methodically.
- Keep practising your programming.
- Read the book.
- Use web resources.
- Use multiple sources (lectures, labs, book, web tutorials).
- Remember: this course is central to your entire MSc!

What if you don’t pass the course?

- You can try again next year.
- There are *no* summer resits.

Don’t Panic!™

- We *want* you to pass.

- There are plenty of people to ask for help if the going gets tough - your demonstrator, the lecturers, your tutor, the departmental tutor and others.

You CAN do it.