Department of Electrical and Computer Engineering

COMPSYS202 / MECHENG270

Course Outline

Semester Two, 2010

COMPSYS 202 Object-oriented Design and Programming (15pts): A project-based course, with extensive hands-on programming experience. Includes: an introduction to object-oriented programming in a modern high level language, algorithms, data abstraction and elementary data structures; an introduction to good programming practise, including an overview of software engineering, software quality, design patterns, testing and user-interface design.

MECHENG 270 Software Design (15pts): Fundamentals of software design and high level programming making use of case studies and programming projects. Includes: requirements analysis, specification methods, software architecture, software development environments, software quality, modularity, maintenance, reusability and reliability; models of software development.

Staff	**Roles**
Dr. George Coghill Room: 303.157 E-Mail: g.coghill@auckland.ac.nz	Course Co-ordinator
Mr. Colin Coghill Room: 303.234 E-Mail: c.coghill@auckland.ac.nz	Lecturer, Project 1, Test 1
To Be Announced	TA and Lab Management
Mr. Jamie Walker Room: 301.231 E-Mail: it-support@ece.auckland.ac.nz	Linux systems administrator, accounts, systems, applications

Assessment

Week	Assessment	Weighting	Due
2-6, 7-11	Lab Exercises	20% (2% each)	During Lab
4	Test 1	20%	Wed 11 Aug
6	Project 1	20%	Mon 23 Aug
11	Test 2	20%	Mon 11 Oct
12	Project 2	20%	Wed 23 Oct

Lectures: There are four regular lecture slots. Some of these slots may be used for revision or tutorial style work. You are expected to attend all lectures if possible, and should take notes.

Lab Exercises: There are regular lab exercises. You are expected to work on these during your assigned lab session on Tuesday or Wednesday, and demonstrate your work to the TA so they can sign off. Each exercise is work 2% of your course mark. A **log book** must be kept to record actual hours you spend on each part of the assessment, and show this to the TAs in your demos. Also, summarise the hours you spend on each exercise or project.

Tests: There are two tests, each worth 20% of your course mark. These will be written tests, one during week 4 and one during week 11. Exact time, day, and location will be announced in lectures, and via Cecil.

Projects: As this is a practical subject, you will be expected to complete two large programming projects. You must work on these individually, although of course you can get help from your TA or other course staff. Project handouts describing these will be made available in lectures and via Cecil at an appropriate time.

Exam: There is no final exam.

	Mon	**Tue**	**Wed**	**Thu**	**Fri**
10am		**Lab Slot A, B** UG3, UG4	**Lab Slot E, F** UG3, UG4		
11am		**Lab Slot C, D** UG3, UG4	**Lab Slot G, H** UG3, UG4		
12pm					
1pm	**Lecture** Eng 1401				
2pm		**Lecture** Eng 1401			
3pm					**Lecture** Eng 1401
4pm				**Lecture** Eng 1401	

Overall goals and learning outcomes

The overall goal is for students to work independently and develop their programming knowledge and skills. The detailed learning outcomes of this course are for:

• Students to gain experience in software engineering design by undertaking individual projects involving a significant programming component using a modern High-level object-oriented programming language, in this case C++.

• Students to be competent using an appropriate set of software development tools, including compiler, editor, debugger, and multiple file build tool(s). In this case, Linux command line tools, which complement the Microsoft Visual Studio tools used in ENGGEN 131, and expands the student experience in programming tools.

• Students to understand how the object-oriented paradigm extends the programming model, via the concepts of data abstraction, inheritance, polymorphism and composition.

• Students to review and implement appropriate algorithms, data abstraction methodologies and data structures and thus gain an appreciation of how concepts can be reliably realised in code.

• Students to understand the activities in a software engineering project.

• Students to understand desirable properties that a computer program should possess.

• Students to be aware of recognised object-oriented design principles including design patterns and architectures and understand why these are important.

• Students to have a basic grounding in user interface design.

• Students to understand the role that testing plays in program development and to be able to follow a test-first approach to program development.

• Students to establish good coding practices including documentation and version control.

Where COMPSYS 202 and MECHENG 270 fits in...

Both courses follow on from the part 1 prerequisite, ENGGEN 131 – Introduction to Engineering Computation and Software Development course, where introductory programming skills were taught in Matlab and C with Microsoft Visual Studio tools.

Programming skills are further developed in the third year: (a) in design papers, where microcomputers are programmed in C, (b) in COMPSYS 302 Design: Software Practice in larger, hands-on programming projects, (c) in SOFTENG 325 Software Architecture, (d) in MECHENG 313 Real time software design, and in the final year: (a) project papers where many projects involve programming in C and C++, (b) in COMPSYS 406 Robotics and Intelligent Systems, (c) in COMPSYS 404 Real Time Systems.

Course Approach

The course approach is "hands on"; design skills and knowledge are acquired by practice on carefully selected exercises and projects. Software design is a skill that can not be taught as a passive, academic subject, nor memorised from notes. You will learn most by embracing this approach, committing your energy to actively solving the project problems, spending time investigating a variety of possible solutions, and exploring the programming tools available on the computer.

It is important to realise the difference between passive and active knowledge. Passive knowledge is the kind of thing you can be told in lectures, read about, remember and repeat when asked in an examination. Active knowledge is something you can acquire only by actively solving problems, continually practicing skills and using the knowledge you acquire by a combination of acquiring information and putting it to use to solve problems. This means you need to practice your programming all the time and solve all the problems for yourself. The skill to create your own programs involves much more than understanding others' programming, and it is this creative skill that we will be assessing in the course.

The main programming ideas, exercises and projects will be explained briefly in lectures, and then it is up to you to take charge of the problem. You will need to follow up by reading the text book carefully, trying out the programming ideas mentioned, and also by finding your own material in the library. Student–driven lab exercises will enable you to ask for help after the initial introduction.

Grade expectations:

Numerical marks will be assigned to each student for each component, and at the end of the paper a total mark will be calculated as a percentage. A letter grade will be assigned to the total mark. For a pass, you should aim for 60% or higher.

Late penalty:

Late work will be assigned 0 marks. Exceptions to penalties because of unforeseen circumstances must be discussed with the course co-ordinator as soon as possible. We can use the Aegrotat process if needed but you will need to supply evidence (eg. a note from your doctor).

Computing resources:

You will be given computer accounts on the ECE Department Linux system and are accessible in the ECE Department labs and Engineering School labs. You can use any of these labs to do your programming work.

Help with the ECE Department Linux system can be found on the following webpage:

https://www.ece.auckland.ac.nz/it-support/wiki/LinuxFAQ

or email:
it-support@ece.auckland.ac.nz

For questions on C++ programming and lab assistance see your TA in your regular lab sessions, or the teaching staff during their office hours.

Accessing the labs between 10pm to 7am, 7 days a week is disallowed; you are required to leave the building. Security staff patrol the labs for no access after hours. In general, the laboratory access times should be displayed at the entrance to the laboratory.

We will be using the GNU C/C++ compiler (the commands are gcc and g++) and other tools with the Linux version of the Unix operating system, for example gedit and eclipse are used to edit files, and make is used to build projects.

UG1 (303.148), UG2 (303.153), UG3 (303.155) and UG4 (303.105) are on the first floor of the Science centre at 38 Princes Street.

For FAQ's of technical problems (logging on, computers, network, Linux etc) use the email it-support@ece.auckland.ac.nz. If there are technical problems that need reporting to IT support please do this straight away using the same email address.

Academic honesty and plagiarism:

In the past, many students have been caught with similar code to each other. So far several students have been taken to the University Discipline Committee and found guilty of academic misconduct (they were fined and lost marks). The University of Auckland will not tolerate cheating, or assisting others to cheat, and views cheating in coursework as a serious academic offence. The work that a student submits for grading must be the student's own work, properly acknowledged and referenced. This requirement also applies to sources on the world-wide web. A student's assessed work may be reviewed against electronic source material using computerised detection mechanisms. Please be aware that we have and will use online software tools for comparing students' C++ software for similarity. In this course you will be required to provide an electronic version of your work, through the dropbox method. Also, see "University Guidelines: Conduct of Coursework," :

http://www.auckland.ac.nz/uoa/about/teaching/plagiarism/plagiarism.cfm.

For any further queries please contact the lecturers.

What Now?

1. Go to the ECE or Engineering School labs and ensure you can log in to remote Linux: restart the computer and select L from the boot menu. Use your NetID and NetPassword to log in. You can email it-support@ece.auckland.ac.nz (perhaps from Windows or ask a friend to send from their account) if you have problems logging in.

2. Log in to Cecil and:
    - view announcements, knowledge map, activities.

3. Send an email to your TA to say hello.

4. Organise lab access with your TA.

5. Start Lab Exercise 1.

6. Attend your regular lab session each week (either Tuesday or Wednesday).