COMP2911 Engineering Design in Computing

Semester 1, 2014

Course Outline

Contents

- Course Staff
- Course Details
- Course Aims
- Student Learning Outcomes
- Assumed Knowledge
- Teaching Rationale
- Teaching Strategies
- Assessment
- Academic Honesty and Plagiarism
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Staff

Contact Details

Name	Role	E-Mail	Phone
Wayne Wobcke	Lecturer in charge	wobcke@cse.unsw.edu.au	9385 6475
Angela Finlayson	Lecturer	angf@cse.unsw.edu.au	9385 8822

Consultations

Name	Time	Room
Wayne Wobcke	Thu 12.00-1.00	K17-307

Course Details

Units of Credit: 6

Handbook Entry: http://www.handbook.unsw.edu.au/undergraduate/courses/2014/COMP2911.html

Web Site: http://www.cse.unsw.edu.au/~cs2911/14s1/

Timetable:

Lecture Time	Room	Lecturer	Room	Lecturer
Tue 12.00-2.00	Ritchie Theatre	Wayne Wobcke	Red Centre M010	Angela Finlayson
Thu 10.00-11.00	Rex Vowels	Wayne Wobcke	CLB 4	Angela Finlayson

Tutorial Time	Room	Laboratory Time	Room	Tutor/Demonstrator	E-Mail
Mon 12.00-1.00	ASB 114	Mon 1.00-3.00	Sanhu Lab	Adrian Ratter	adrianr@cse.unsw.edu.au
Mon 12.00-1.00	Goldstein G02	Mon 1.00-3.00	Drum Lab	Mina Samy	minah@cse.unsw.edu.au
Mon 5.00-6.00	Goldstein G07	Mon 6.00-8.00	Sanhu Lab	Rob Everest	robertce@cse.unsw.edu.au
Tue 9.00-10.00	Quad 1046	Tue 10.00-12.00	Oud Lab	Adrian Ratter	adrianr@cse.unsw.edu.au
Tue 2.00-3.00	Quad 1049	Tue 3.00-5.00	Viola Lab	Rob Everest	robertce@cse.unsw.edu.au
Tue 2.00-3.00	Quad G055	Tue 3.00-5.00	Cello Lab	Tim Wiley	timothyw@cse.unsw.edu.au
Tue 6.00-7.00	Goldstein G07	Tue 7.00-9.00	Guan Lab	Tim Wiley	timothyw@cse.unsw.edu.au
Wed 9.00-10.00	Quad 1049	Wed 10.00-12.00	Guan Lab	Mina Samy	minah@cse.unsw.edu.au
Wed 9.00-10.00	Quad 1047	Wed 10.00-12.00	Drum Lab	Hayden Smith	hsmi665@cse.unsw.edu.au
Thu 11.00-12.00	Goldstein G05	Thu 12.00-2.00	Guan Lab	Adrian Ratter	adrianr@cse.unsw.edu.au
Thu 1.00-2.00	Webster 252	Thu 2.00-4.00	Guan Lab	Mina Samy	minah@cse.unsw.edu.au

Fri 9.00-10.00	Quad G026	Fri 10.00-12.00	Mabu Lab	Hayden Smith	hsmi665@cse.unsw.edu.au
Fri 9.00-10.00	Quad G045	Fri 10.00-12.00	Guan Lab	Brad Heap	bradfordh@cse.unsw.edu.au
Fri 12.00-1.00	Colombo LG02	Fri 1.00-3.00	Mabu Lab	Rob Everest	robertce@cse.unsw.edu.au

Course Aims

COMP2911 covers the theory and practice of object-oriented design and programming with an emphasis on design patterns and problem-solving algorithms. It is a third course in programming, so students are assumed to be competent C programmers who can understand and use abstract data types and analyse program efficiency. COMP2911 is a prerequisite for some third and fourth stage Computer Science and Software Engineering courses and emphasizes professional issues relevant to working in the software industry.

There are four more specific themes and objectives.

- *Object-Oriented Design.* To introduce the basic principles of object-oriented design, including abstraction, modularity, polymorphism, interface design, design patterns and basic UML.
- *Object-Oriented Programming.* To introduce the object-oriented programming paradigm, including concepts such as objects, classes, inheritance, encapsulation, programming by contract, test driven development and unit testing.
- *Java.* To introduce the Java programming language and Java application development using the Eclipse IDE (Interactive Development Environment).
- *Algorithms.* To consolidate problem-solving methods introduced in previous courses in the context of object-oriented design and programming.

Student Learning Outcomes

The following are the intended learning outcomes of this course.

- *Object-Oriented Design.*
  - Be able to design and interpret CRC cards.
  - Be able to interpret and produce UML class diagrams and sequence diagrams.
  - Understand the principles of object-oriented design.
  - Be able to follow a systematic object-oriented design process.
  - Understand and use object-oriented design patterns.
- *Object-Oriented Programming and Java.*
  - Be able to write medium-scale object-oriented programs in Java.
  - Become familiar with the Java APIs and class libraries.
  - Understand exception handling, event handling, serialization and reflection.
  - Apply systematic methods such as test driven development using JUnit.
  - Understand the importance of documentation and use tools such as javadoc to adequately document software.
  - Be able to use an IDE (Interactive Development Environment) such as Eclipse for software development.
  - Be able to work within a small team in the context of a software development project.
- *Algorithms.*
  - Understand and use a range of problem-solving algorithms.
  - Be able to implement problem-solving algorithms in Java.

This course contributes to the following UNSW graduate attributes.

- *An in-depth engagement with the relevant disciplinary knowledge in its interdisciplinary context.* The development of systems suitable for different user groups encourages consideration of the users of technology.
- *The capacity for analytical and critical thinking and for creative problem solving.* The creation of software applications requires creative approaches to design and implementation.
- *The ability to engage in independent and reflective learning.* Software design and implementation must be done independently and students benefit by reflecting on their experiences.
- *The skills required for collaborative and multidisciplinary work.* Through a small team-based project, students gain experience in working within a collaborative environment.
- *The skills of effective communication.* Through a small team-based project, students develop interpersonal communication skills and presentation and writing skills.

Assumed Knowledge

Students are assumed to have completed two programming courses: the first (typically) an introductory programming course based on a procedural language, the second (typically) a course on abstract data types and algorithms. Students should be able to write and debug medium-sized procedural programs, understand the elements of good programming style, understand and use abstract data types in program design and implementation, be familiar with a range of standard data structures and algorithms (such as lists, trees, graphs, etc.), and be able to analyse the computational complexity of algorithms. Students should also have some experience working in small groupwork activities.

Teaching Rationale

COMP2911 is an important course that bridges the introductory programming courses and further specialist third and fourth stage Computer Science and Software Engineering courses intended for professionals in the software industry. The course builds on students' programming skills and introduces professional issues such as designing and implementing larger-scale software systems through working in teams.

The course is primarily an introduction to object-oriented design and programming. Much of student learning takes place through lab-based exercises, programming assignments and a small team-based project, with lectures and tutorials providing a conceptual framework, practical examples and the opportunity to reflect on design issues.

The course aims to cover object-oriented design, object-oriented programming, the Java programming language and problem-solving algorithms in an integrated fashion. As far as possible, the approach is "problem-centred", where, given an example problem, the issues of design, choice of algorithm and implementation are addressed in the context of the problem. Design is not seen as independent of programming, but as part of a systematic Software Engineering process. The coverage of algorithms consolidates material from the introductory courses.

Teaching Strategies

The course has high contact hours, including 3 hours of lectures and a combined 1 hour tutorial and 2 hour lab session each week (lectures in Weeks 1-12 and tutorial/lab sessions in Weeks 2-13). The tutorial and lab sessions are essential for gaining programming experience and individual feedback from tutors/lab demonstrators. **Java programming (let alone object-oriented design principles) has a high learning curve, and it is critical to attend tutorial/lab sessions to keep up with the material in this course.**

Lectures provide a conceptual framework for the material covered by the course, in an environment that provides a chance for students to ask questions and listen to the answers to other students' questions. Feedback from students to the lecturer on topics causing common difficulty is encouraged.

The *textbook* provides background and examples to reinforce understanding of the course content. *It is recommended to read the related textbook material prior to the lecture.*

Tutorial/lab sessions provide students an opportunity to learn via simple practice examples and to gain personal feedback from their tutor/lab demonstrator and other students. **The lab sessions are also important for guidance during the team-based project.** If anything, the tutorial/lab sessions are more important than lectures.

Assignments, aside from being the primary means of assessment in the course (along with the final exam) give students the opportunity to practise what they have learnt on relatively large problems.

Assessment

The assessment for this course consists of the following weighted components.

- Tutorial/lab participation and lab exercises (10%)
- Two programming assignments, each 10% (20%)
- Team-based project (25%)
- Final exam (45%)

The tutorial/lab participation marks are determined by the tutor/lab demonstrator and consist of 1 mark for each of 10 tutorial/lab sessions. The two programming assignments are marked by machine, with a small part of the marks given for design quality and programming style. The team-based project consists of an individual component (10%) and a team component (15%) making up 25%. The individual mark is based on a reflective diary shown weekly to the tutor/lab demonstrator; the team-based assessment includes a presentation/demonstration of the final system (held in Week 13). Each member of the team generally receives the same mark for the team-based component of the project. The final exam is a 2 hour written examination covering all aspects of the course, including object-oriented design, object-oriented programming, the Java programming language and problem-solving algorithms.

The final mark for the course is determined by adding together these component marks according to the above weighting to give a result out of 100, which is subject to further scaling. **A pass mark (50%) on the final exam is a requirement for passing the course.**

Late submission policy for assignments: Assignments submitted late are subject to the penalty that the mark reduces by 10% of the value of the assignment per day late, up to a maximum of 5 days, after which a mark of 0 is received.

Academic Honesty and Plagiarism

UNSW has instituted severe penalties for academic plagiarism. Therefore it is important for students to understand what is acceptable and unacceptable in the presentation of work for assessment.

You should first read the UNSW policy on plagiarism.

The School of Computer Science and Engineering has a more specific policy on plagiarism in Computer Science.

In essence, as applied to COMP2911, copying or sharing program code for individual assignments counts as plagiarism and is unacceptable. In the team-based programming project, sharing code *amongst team members* is acceptable, but sharing code between (members of) different teams is unacceptable. Copying code from the Internet is unacceptable in both cases. Assignment submissions are checked for similarity, both with one other and with code on the Internet. The penalties for plagiarism range from receiving 0 marks for the assignment, through receiving a mark of 00 FL for the course, to expulsion from UNSW (for repeat offenders). The school maintains a register of students with confirmed plagiarism offences. Note that allowing someone else to copy your work counts as academic misconduct, and makes you liable to a penalty, even if you can prove that the work is yours originally.

Net etiquette: There are many online Java programming forums. Do **not** use these forums to post questions that

essentially ask for solutions to the assignments. These forums are for professional Java programmers, and generally the user community will discourage use of the forum in this way. Use such forums only to ask arcane Java questions, and only then after consulting the Java API documentation.

Course Schedule

The following is the rough sequence of topics by week. More details will be provided during the semester.

Week	Topic	
1	Introduction	
2	Object-Oriented Design Process	
3	Class Design and Programming by Contract	
4	Interface Types and Polymorphism	
5	Patterns and GUI Programming	
6	Problem-Solving Algorithms	Assignment 1 due
7	Informed Search Algorithms	
8	Inheritance and Abstract Classes	
9	Java Object Model, Frameworks	
10	Multithreading	Assignment 2 due
11	More Design Patterns	
12	Review	Project due
13	Project Assessment	

Resources for Students

Textbook

Much of the material on object-oriented design and programming for this course follows the presentation in the textbook.

> Horstmann, C. *Object-Oriented Design and Patterns.* Second Edition. John Wiley and Sons, Hoboken, NJ, 2006.
> Textbook Home Page

Additional References

The following may serve as further reading expanding on the ideas developed in this course.

> Liskov, B. & Guttag, J. *Program Development in Java*. Addison-Wesley, Reading, MA, 2001.
> *Discusses different notions of program abstraction and their use in Java programming.*

> Gamma, E., Helm, R., Johnson, R. & Vlissides, J. *Design Patterns*. Addison-Wesley, Reading, MA, 1995.
> *The basic reference on design patterns that provides many more examples than given in the course.*

Java Resources

This course involves Java programming. There is no need to buy a Java programming reference (there are many!), however the following online Java API documentation is useful. If you do buy a Java reference, make sure that it is for Java Standard Edition Version 6 or 7 (or at least Version 5). Java SE 6 is installed in the labs so that is the preferred option.

> Java Platform SE 6 APIs
> Java Platform SE 7 APIs

The course also involves Java development using Eclipse, an IDE (Interactive Development Environment). The following links are to useful Eclipse resources. The version of Eclipse installed in the labs is Eclipse IDE for Java EE Developers (top of the list of package solutions). Eclipse Standard 4.3.1 can also be used.

> Eclipse Downloads
> Eclipse Help

Students may also want to make use of the Violet UML graph drawing tool (or there are other UML editors).

> Violet UML Editor

Course Evaluation and Development

CSE courses are evaluated by student survey each time they are taught. The survey includes standard questions asked of all comparable courses so that it is possible to compare COMP2911 with other relevant UNSW courses, and also includes

space for free-form comments. Survey responses are anonymous. The completed survey forms are analysed statistically by someone independent of the course staff, and the results, including free-form comments, are made available to the lecturer in charge *after* grades have been reported and released.

Feedback from previous offerings of COMP2911 has indicated that the workload has been considered heavy compared to other CSE courses. Since the course involves learning a new paradigm (object-oriented design and programming) which is typically learnt by practical experience, and also includes a project involving programming in teams, this course does require an extensive time commitment. However, in line with this feedback, the number of programming assignments has been reduced to two, with one small team-based project.

Another difference from previous versions of the course is the use of a textbook, which can be used as a basis for a large part of the material and provides further explanation of the main concepts of object-oriented design and programming than is possible in lectures. This should especially help those students who are unable to attend lectures.

A further difference from previous versions of the course is the coverage of GUI (graphical user interface) development, with the aims of introducing a broader notion of design that incorporates user-system interaction, giving additional motivation for design patterns, strengthening the programming component of the course, and providing more relevant skills typically needed in industry by Java software developers.

COMP2911 was taught using this structure for the first time in Semester 2, 2009. A survey of students was undertaken in the Week 12 tutorial session, capturing almost all students who were still actively engaged in the subject. Results indicated that students generally agreed that the workload was about right (mean 4.6), that the textbook/textbook slides were helpful for understanding the subject (mean 4.3) and that GUI programming should be part of this course (mean 4.9), with responses given on a six point scale where 4 is Mostly Agree, 5 is Agree and 6 is Strongly Agree.

COMP2911 was formerly offered in both semesters. Since 2013, it has been offered only in Semester 1. Therefore the number of students in the course is higher than in previous years. Hence from 2014, the project will have teams of 4-5 students, rather than 3 as previously. More explicit assessment criteria will be given to ensure consistency of marking across this large group.

Wayne Wobcke
Last updated: 19 November 2021
CRICOS Provider No. 00098G