Course Outline

Contents

- Course Details
- Course Summary
- Assumed Knowledge
- Student Learning Outcomes
- Teaching Strategies
- Teaching Rationale
- Student Conduct
- Assessment
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Details

Course Code	COMP2511
Course Title	Object-Oriented Design & Programming
Convenor	Aarthi Natarajan (a.natarajan@unsw.edu.au) (/users/z2136899)
Admin	Robert Clifton-Everest (/users/z3254687)
Classes	**Lectures** : Wednesday 13:00 - 15:00 Law Theatre G04 (K-F8-G04) 1-9,10-12 Friday 10:00 - 12:00 Ainsworth G03 (K-J17-G03) 1-9,10-12 Timetable for all classes (/COMP2511/18s2/timetable)
Consultations	to be added
Units of Credit	6
Course Website	http://cse.unsw.edu.au/~cs2511 (http://cse.unsw.edu.au/~cs2511)
Handbook Entry	http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2511.html (http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2511.html)

Course Summary

COMP 2511 covers the theory and practice of object-oriented design and programming with an emphasis on teaching students how to apply software design principles and design patterns to the building of flexible, reusable and maintainable systems. The course also introduces user-interface design and programming, concurrency in multi-threaded systems and includes the implementation of a large group project that is implemented based on agile software practices and enables students to have hands-on experience in applying the principles taught, implementing and incorporating the use of good design principles and design patterns in software programs

Assumed Knowledge

Students are assumed to be:

- competent C programmers who can understand and use abstract data types
- understand and know how to use git and Github reasonably
- be familiar with fundamental object-oriented design concepts

This course expects that students have undertaken COMP 1531 (a new prerequisite for COMP 2511 introduced following the curriculum review) prior to taking this course. COMP 1531 introduces use of GitHub and also exposes students to the fundamental principles of object-oriented design. However, we currently have an exceptional situation where there are several students on the old program who have not taken COMP 1531. For these students, the week 1 lab is intended to bring them up to speed with Git and GitHub. Additionally, the fourth hour of lecture in some of the weeks will also be used to provide additional bridging material to help these students gain familiarity with the fundamentals of object-oriented design .

Student Learning Outcomes

The following are the intended learning outcomes of this course:

- *Object-Oriented Design*
  - Understand the principles of object-oriented design
  - Be able to follow a systematic object-oriented design process
  - Be able to interpret and use tools for object-oriented design
  - Learn how to apply design principles and design patterns effectively to design flexible, maintainable and reusable systems
- *Object-Oriented Programming and Java*
  - Understand object-oriented programming languages
  - Be able to write medium-scale object-oriented programs in Java
  - Apply systematic methods such as test driven development
  - Understand the importance of documentation and use tools to document software
  - Be able to use an IDE (Interactive Development Environment) for software development
- *Software Processes*
  - Understand the importance of team organization and communication
  - Be able to work within a small team in the context of a software development project
  - Be able to plan and execute a software project according a systematic software process
- *User Interfaces*
  - Become familiar with principles of effective user interface design
  - Be able to implement a user interface in Java
- *Concurrency*
  - Become familiar with basic issues in multi-threaded systems
  - Be able to use synchronization primitives in Java

This course contributes to the development of the following graduate capabilities:

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 and applying best
- *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 project, students gain experience in working within a collaborative environment.
- *The skills of effective communication.* Through a small team project, students develop interpersonal communication skills and presentation and writing skills.

Teaching Strategies and Rationale

There are some key changes to the structure and content of the course this semester. These are strategic changes necessary to improve the student learning experience and outcomes and to adapt to UNSW3+ trimesters being introduced in 2019.

The revised structure is as follows:

Lectures **- 3 hours each week (instead of 2)**

A total of 36 hours of lectures (2018, S2 : 3 * 12, 2019, UNSW3+: 4 * 9) will be used to present key concepts and practical examples of design techniques taught in this course

Tutorials (new)

A 1 hour tutorial session per week has been introduced this year, which is scheduled before the lab. Tutorials will serve three purposes (1) clarify ideas from lectures and work through exercises based on the lecture material (2) provide practical demonstrations in the Java programming language (3) peer code review. You should make sure that you use them effectively by examining in advance the material to be covered in each week's tutorial, by asking questions, by offering suggestions and by generally participating. The tutorial questions will be posted on the Web in the week before each tutorial.

All tutorials start in week 1.

Laboratory classes (contribute to 12% of overall course mark)

Following the tutorial class each week, there will be a two-hour laboratory class, during which you will work on a variety of small practical problems that will enable you to apply the concepts taught in the lectures.

Each week, there will be one or more exercises to work on. These exercises will be released in the week preceding the lab class. The exercises will need to be submitted and will be assessed by your tutor. This semester, lab exercises will constitute an important part of the course assessment and will contribute to 12% of the overall course mark..

You should normally get your lab work assessed during the week for which it is scheduled (i.e. you must complete the week 1 lab exercise during the week 1 lab). All completed lab exercises must be uploaded to GitHub prior to being marked by your tutor or lab assist. If you are unable to finish the exercises during the lab, you may continue working on it during the week, but **you must submit the completed lab to GitHub by the following Sunday 11:59 pm** in order to get any marks for it. You must then demonstrate your work to your tutor during the first hour of the following week's lab. The code that you submit by Sunday is what will be assessed. You must not make any changes to the completed solution after Sunday. Your tutor will check the time-stamp prior to marking the lab.

Summary: to obtain any lab marks for the Week X lab, you must do 2 things:

- *submit your lab work to GitHub by the following Sunday 11:59pm*
- *demonstrate your work to your tutor in the week X lab class*

OR *demonstrate your work at the start of the lab in week X+1*

You **cannot** obtain marks by e-mailing lab work to tutors. Your tutor will provide feedback during a lab session on your approach to the problem and on the style of your solution. All labs must **normally** be demonstrated OR submitted by the due-date. Your tutor may grant you an extension on the lab exercise if you have been unable to complete the lab exercise due to illness or unavoidable circumstances (e.g, jury duty)

Periodically, the lab sessions will also be used to schedule demonstrations of your ongoing group project. As the implementation of the group project will be based on an **agile software development methodology** , you will required to demonstrate (as a team) progress of your project in iterations. No excuses will be accepted (unless unavoidable circumstances) for some members of the team not being present for the iteration demo.

As this course is has a significant practical component, laboratory classes are important to help you acquire the necessary skills in the relevant tools and frameworks that will be used in the implementation of the group project and also prepare you for the final exam. If you do not put a good amount of effort into the lab classes you risk failing the group project and the final exam.

The practical lab sessions will contribute to **12%** of your overall course mark. The mark for iteration demonstration of the group project will go towards the mark allocated for your group project.

Use of GitHub will be strongly enforced in this course, as using GitHub as the medium of team collaboration and source control is a key learning outcome that will be applied by students in several following courses through their degree.

All labs start in week 1.

Assignment, Group Project and Quiz

There will be one assignment (a design-based) to be completed individually which will be released in week 2 and will be due in week 5. This assignment will constitute **10** % of the overall course mark.

There will one main group project which will run through the rest of the teaching period and contributes to **25%** of the overall course mark. The specification of the group project will be released in week 3. You are required to form groups of 4 for the group project. Details about the group project will be released through the semester.

Periodically, online quizzes will be released and students will have the opportunity to complete the quiz at their own pace by the specified due-date. These quizzes will enable students to test their understanding of the content taught. Quizzes will contribute to **3** % of the overall course mark.

Final Exam

There will be a centrally timetabled final exam which will in your UNSW exam timetable. There is a hurdle requirement on the final exam. If you do not score at least 50% on the exam, you cannot pass this course. If your overall course score exceeds 50%, despite scoring very poorly (<50%) on the exam, the hurdle will be enforced via a grade of UF. Of course, if your overall course score is less than 50%, then your grade will be FL.

Student Conduct

The **Student Code of Conduct** (Information (https://student.unsw.edu.au/conduct) , Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)) sets out what the University expects from students as members of the UNSW community. As well as the learning, teaching and research environment, the University aims to provide an environment that enables students to achieve their full potential and to provide an experience consistent with the University's values and guiding principles. A condition of enrolment is that students *inform themselves* of the University's rules and policies affecting them, and conduct themselves accordingly.

In particular, students have the responsibility to observe standards of equity and respect in dealing with every member of the University community. This applies to all activities on UNSW premises and all external activities related to study and research. This includes behaviour in person as well as behaviour on social media, for example Facebook groups set up for the purpose of discussing UNSW courses or course work. Behaviour that is considered in breach of the Student Code Policy as discriminatory, sexually inappropriate, bullying, harassing, invading another's privacy or causing any person to fear for their personal safety is serious misconduct and can lead to severe penalties, including suspension or exclusion from UNSW.

If you have any concerns, you may raise them with your lecturer, or approach the School Ethics Officer (mailto:ethics-officer@cse.unsw.edu.au) , Grievance Officer (mailto:grievance-officer@cse.unsw.edu.au) , or one of the student representatives.

Plagiarism is defined as (https://student.unsw.edu.au/plagiarism) using the words or ideas of others and presenting them as your own. UNSW and CSE treat plagiarism as academic misconduct, which means that it carries penalties as severe as being excluded from further study at UNSW. There are several on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism and Academic Integrity (https://student.unsw.edu.au/plagiarism)
- UNSW Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism. In particular, you are also responsible that your assignment files are not accessible by anyone but you by setting the correct permissions in your CSE directory and code repository, if using. Note also that plagiarism includes paying or asking another person to do a piece of work for you and then submitting it as your own work.

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW staff and students have a responsibility to adhere to this principle of academic integrity. Plagiarism undermines academic integrity and is not tolerated at UNSW. Plagiarism at UNSW is defined as using the words or ideas of others and passing them off as your own.

If you haven't done so yet, please take the time to read the full text of

- UNSW's policy regarding academic honesty and plagiarism (https://student.unsw.edu.au/plagiarism)

The pages below describe the policies and procedures in more detail:

- Student Code Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)
- Student Misconduct Procedure (https://www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf)
- Plagiarism Policy Statement (https://www.gs.unsw.edu.au/policy/documents/plagiarismpolicy.pdf)
- Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

You should also read the following page which describes your rights and responsibilities in the CSE context:

- Essential Advice for CSE Students (https://www.engineering.unsw.edu.au/computer-science-engineering/about-us/organisational-structure/student-services/policies/essential-advice-for-cse-students)

Assessment

Item	Due	Marks
Quizzes	Weeks 4,8,12	3%
Assignment1	Week 5, Wednesday	10%
Group Project	Weeks 6,9,13	25%
Labs	All Weeks	12%
Final Exam	Exam period	50%

Final Mark

Your final mark for this course will be computed using the above assessments as follows:

- Course_Work_Mark = lab_mark + quiz_mark + assignment_1_mark + group_project_mark (out of 50)
- Exam_Mark = Final Exam Mark (out of 50)
- Exam_OK = Exam_Mark >= 25
- Raw_Final_Mark = Exam_Mark + Course_Work_Mark
- Final_Mark = Raw_Final_Mark with scaling applied
- Final_Grade =UF, if !Exam_OK
- FL, if Final_Mark< 50/100
  PS, if 50/100 ≤ Final_Mark < 65/100
  CR, if 65/100 ≤ Final_Mark < 75/100
  DN, if 75/100 ≤ Final_Mark < 85/100
  HD, if Final_Mark ≥ 85/100

Students are eligible for a Supplementary Exam if and only if:

- they cannot attend the final exam due to illness or misadventure
- their Final_Mark is in the range 48 ≤ FinalMark < 50 and Exam_Mark ≥24/50. The maximum mark awarded to students who have failed the final exam and been awarded a supplementary exam is 50.

A Supplementary Exam will not be awarded for any other reason.

Course Schedule

The following is a **tentative** schedule of when course topics will be covered.

Week	Topic	Assignments Due
1	Course Introduction, Introduction to Java (JDK, J2EE) using Eclipse IDE What is OO design? Object, Classes, Constructors	-
2	Abstraction, Encapsulation and Inheritance, OO Relationships Revision of Requirements Analysis and Domain Modelling (User-Stories, CRC Cards, Class Diagram)	-
3	Interfaces, Abstract classes, Polymorphism What are design smells? Identify good design characteristics Refactoring Techniques Part I	-
4	Exploring Pitfalls of Inheritance - Is Composition and Delegation better than inheritance? Refactoring Techniques Part II Design Patterns - Refactoring targets Introducing Strategy, State Pattern	
5	Refactoring Techniques Part 3	Assignment 1

	Design Patterns - Factory, Abstract Factory	
6	Generics, Collection Frameworks Design Patterns - Iterator, Builder and Composite	Group Project - Iteration 1
7	Exception Handling in Java Design by Contract Test Drive Development	
8	Refactoring Techniques Part 4 Design Patterns: Observer, Decorator, Adapter Grouping Patterns into structural, behavioural, creational	
9	User-centered design and usability heuristics Design patterns: Observer MVC Application framework	Group Project - Iteration 2
10	GUI using JavaFX	
11	Concurrency	
12	Revision, Exam Format	Group Project - Final Demo

Resources for Students

There is no single text book that covers all of the material in this course at the right level of detail. The lecture notes should provide sufficient detail to introduce topics, and you will then study them in further depth in the tutorials, labs and group project.

There are also many online resources available, and we will provide links to the most useful ones. Some are listed below. If you find others, please post links in the Comments section on the Course Outline page.

Some suggestions for books that cover at least some of the topics in this course

- *Head First Design Patterns* , by Elisabeth Freeman and Kathy Sierra, The State University of New Jersey
- *Refactoring: Improving the design of existing code* , by Martin Fowler

Course Evaluation and Development

This course is evaluated each session using the myExperience system.

In the previous offering of this courses, students noted that more learning resources were needed to enhance student understanding of the material presented. Based on their comments, we have introduced tutorial sessions before the lab classes which will help to reinforce the concepts presented in the lectures. The weekly lab exercises will also be given sample solutions after the due date to assist the students. Final periodic online quizzes will help students to revise the lecture content and cement their understanding.

Resource created 3 years ago (Thursday 05 July 2018, 11:16:03 AM), last modified 3 years ago (Wednesday 01 August 2018, 10:12:23 PM).

Comments (/COMP2511/18s2/forums/search?forum_choice=resource/16995) (/COMP2511/18s2/forums/resource/16995)

There are no comments yet.