Australian National University | PROGRAMS AND COURSES

PROGRAMS AND COURSES / COURSES / COMP6442 / FIRST SEMESTER /

CLASS SUMMARY | BACK TO COURSE DETAILS

Software Construction

A Postgraduate course offered by the **School of Computing**.

COMP6442

Overview | Assessment | Submission | Contacts

CLASS NUMBER 2347 TERM CODE 3330

CLASS INFO

Unit Value
6 units

Mode of Delivery
In Person

COURSE CONVENER
Dr Bernardo Pereira Nunes

LECTURER

CLASS DATES

Class Start Date
20/02/2023

Class End Date
26/05/2023

Census Date
31/03/2023

Last Date to Enrol
27/02/2023

CLASS NUMBER 2347 TERM CODE 3330

CLASS INFO

Unit Value
6 units

Mode of Delivery
In Person

COURSE CONVENER
Dr Bernardo Pereira Nunes

LECTURER
Dr Bernardo Pereira Nunes

CLASS DATES

Class Start Date
20/02/2023

Class End Date
26/05/2023

Census Date
31/03/2023

Last Date to Enrol
27/02/2023

TUTOR
Llew Reilly

This course focuses on construction of medium scale programs, using design patterns and tools that are used in the software development process. Students will gain further experience with industry standard revision control and integrated development environment (IDE) tools.
Students will learn appropriate application of programming abstractions they have learned in previous courses to the structuring of medium scale software: inheritance, generic types, polymorphism, procedural abstraction, and abstract recursive data structures (including abstract syntax trees as a program representation, and tools that manipulate them).

The course also covers more advanced data structures, such as priority queues, B-trees, red-black trees, and AVL trees, and deepens understanding of appropriate algorithmic strategies.

The course also treats intellectual property considerations in software development and deployment.

Learning Outcomes

Upon successful completion, students will have the knowledge and skills to:

1. Apply fundamental programming concepts for medium scale programs
2. Understand basic types and the benefits of static typing, with understanding of generics, subtyping, and overloading, and their roles in structuring programs
3. Map programming language abstractions through to execution environment; use non-source (text) internal representations of programs (e.g., abstract syntax trees); sketch low-level run-time representations of core language constructs (objects and closures)
4. Describe contractual specifications, analyse documentation and specifications against other's code, develop, understand, test, and evolve substantial programs using a modern IDE, and associated configuration tools; explain the importance of correctness for quality software; understand common coding errors and how to avoid them; practice fundamental defensive programming; understand principles of secure design
5. Use, implement, and evaluate more advanced data structures and associated algorithms; discuss factors other than computational efficiency for evaluating software; create,

implement, debug, and evaluate algorithms for solving problems, including recursively, using divide-and-conquer, and via decomposition; implement an abstract data type; analyse design and implementation alternatives

6. Apply basic algorithmic analysis to simple algorithms; use big-O notation formally, upper lower, and expected case bounds; use and solve recurrence relations; use appropriate algorithmic approaches to solve problems (brute-force, greedy, divide-and-conquer, recursive backtracking, heuristic, dynamic programming, branch-and-bound)
7. Explain how system components contribute to performance; understand Amdahl's law and its limitations; design and conduct performance experiments; use software tools to profile and measure program performance
8. Understand, apply, and analyse state and state machines in expressing computations
9. Understand fundamental concepts of GUIs and user interfaces; understand the basics of modeling and simulation
10. Contrast the concepts of copyright, patenting, and trademarks as mechanisms for protecting intellectual property, within the legal context for these mechanisms; understand, analyse, and evaluate ethical/social tradeoffs in technical decisions, evaluating stakeholder positions

Examination Material or equipment

The materials and equipment will be listed prior to the exams on Wattle.

Required Resources

Computer with a working camera; Android Studio; and IntelliJ or Eclipse IDEs.

Recommended Resources

Whether you are on campus or studying remotely, there are a variety of online platforms you will use to participate in your study program. These could include videos for lectures and other instruction, two-way video conferencing for interactive learning, email and other messaging tools for communication, interactive web apps for formative and collaborative activities, print and/or photo/scan for handwritten work and drawings, and home-based assessment.

ANU outlines recommended student system requirements to ensure you are able to participate fully in your learning. Other information is also available about the various Learning Platforms you may use.

Staff Feedback

Students will be given feedback in the following forms in this course:

- written comments for the group and video assignments.
- auto-generated comments based on a set of pre-defined test cases for all other assignments/exams.

Student Feedback

ANU is committed to the demonstration of educational excellence and regularly seeks feedback from students. Students are encouraged to offer feedback directly to their Course Convener or through their College and Course representatives (if applicable). Feedback can also be provided to Course Conveners and teachers via the Student Experience of Learning & Teaching (SELT) feedback program. SELT surveys are confidential and also provide the Colleges and ANU Executive with opportunities to recognise excellent teaching, and opportunities for improvement.

Class Schedule

WEEK/SESSION	SUMMARY OF ACTIVITIES	ASSESSMENT
1	Overview of Software Construction - This lecture overviews the course and introduces the main concepts related to software construction.	Video Assignments are released.
2	Design Patterns - This lecture presents several typical solutions (i.e. design patterns) to recurring problems in software design using real-world examples.	Lab 1: Setting Up the Environment. This lab is dedicated to helping you set up your course environment.
3	Software Testing - This lecture covers essential software testing concepts and techniques for identifying defects and errors in software during its development, as well as measuring software quality in terms of various coverage techniques.	Lab 2 - Design Patterns. This lab offers an opportunity to implement some of the most popular design patterns used in various applications.
4	Data Structures - This lecture covers well-known data structures that are essential for developing efficient algorithms and a variety of applications.	Lab 3 - Software Testing. In this lab, you will design and implement test cases to identify defects and errors in software, as well as use different techniques to measure the code coverage of a small application.
5	Data Structures II - This lecture is a continuation of the previous one with a focus on more challenging and advanced data structures.	Lab 4 - Data Structures. In this lab, you will implement a part of an important data structure for several applications.
6	Tokeniser and Parser - This lecture introduces an important concept for a range of applications such as compilers, search engines, and even natural language processing.	Lab 5 - Data Structures II. In this lab, you will implement a challenging data structure often used in databases. In this week, the Group Assignment will be released and the first part of the video assignment is due.
7	Android Development - This lecture delves into the concepts of mobile development using Java.	Lab 6 - Android. In this lab, you will implement a small application using Android Studio and Java. The midterm exam and the first checkpoint for the group assignment also take place this week.
8	Persistent Data - This lecture introduces several approaches to data persistence (e.g., JSon, XML, Serialisation, ...).	Lab 7 - Tokeniser & Parser. In this lab, you will have the opportunity to implement a tokeniser and parser.
9	Refactoring - It's time to revisit our code and identify and fix common errors in the software we develop. This lecture will give an overview of the most common design and implementation mistakes when developing software, as well as possible solutions.	Lab 8 - Persistent Data. In this lab, you will implement a simple application where you are required to use one or more data persistence techniques.
10	Design by Contract - This lecture introduces the "design by contract" concept, a methodology for delivering high-quality software.	Lab 9 - Refactoring. In this lab, you will refactor a small application to practice the concepts seen in the lecture. The second checkpoint for the group assignment occurs in this week.
11	Intellectual Property - As part of the software development process is the concept of intellectual property. This lecture covers concepts every developer should know about intellectual property and software development.	Group Assignment is due.
12	Minute Madness Presentations - This lecture is an opportunity for all students to showcase the software they have developed throughout the semester.	Video Assignment 2 is due. Group Assignment presentation is due.

Tutorial Registration

ANU utilises MyTimetable to enable students to view the timetable for their enrolled courses, browse, then self-allocate to small teaching activities / tutorials so they can better plan their time. Find out more on the **Timetable webpage.**

Assessment Summary

ASSESSMENT TASK	VALUE	LEARNING OUTCOMES
Video Assignment 1	2 %	1-5
Video Assignment 2	2 %	6-10
Group Assignment	30 %	1-10
Midterm Exam	10 %	1-5
Final Exam	40 %	1-10
Lab Assignment	2 %	1,2
Lab Assignment	2 %	1,4
Lab Assignment	2 %	2,5,6
Lab Assignment	2 %	2,5,6
Lab Assignment	2 %	2,9
Lab Assignment	2 %	2,3
Lab Assignment	2 %	1
Lab Assignment	2 %	3,4

*** If the Due Date and Return of Assessment date are blank, see the Assessment Tab for specific Assessment Task details**

Policies

ANU has educational policies, procedures and guidelines , which are designed to ensure that staff and students are aware of the University's academic standards, and implement them. Students are expected to have read the Academic Integrity Rule before the commencement of their course. Other key policies and guidelines include:

- **Academic Integrity Policy and Procedure**
- **Student Assessment (Coursework) Policy and Procedure**
- **Special Assessment Consideration Guideline and General Information**
- **Student Surveys and Evaluations**
- **Deferred Examinations**
- **Student Complaint Resolution Policy and Procedure**
- **Code of practice for teaching and learning**

Responsible Officer: Registrar, Student Administration / **Page Contact:** Website Administrator / Frequently Asked Questions