

# 01 Introduction ▾



- [Introduction](#)
  - [History of these notes](#)
- [The Unit Outline](#)
- [How the unit has changed](#)
- [Assessment](#)
  - [Task and Test status](#)
- [Classes & Other Activities](#)
  - [Attendance Requirements](#)
- [Resources](#)
  - [Avoiding plagiarism](#)

[ [Close all sections](#) ] Only visible sections will be included when printing this document.

References: *Unit Outline*; *Essential Information* > *How your final grade will be determined*

## ▽ Introduction

Programming is the skill of instructing the computer what do to. Because computers are general-purpose information processing machines they are also *not* innately intelligent, so when you give them instructions you need to do so in a way that is more formal than the way you would give instructions to another person, and constrained by how the computer represents information and how it can process it.

The purpose of this unit is to develop your skills in developing solutions to computing problems in a way that is suitable for a computer to solve, and then to translate that into a program that it can execute, in this case in the Java programming language.

### History of these notes

The notes were created in 2014 when I reduced the amount of text in lecture slides but wanted to still make those details available for study and revision. They're a combination of brief points (to serve as a reminder) and longer blocks of text, but were always intended to be *brief*, so they are not a textbook, nor a Java language reference, nor full of illustrative graphics (those are still in the lecture slides). However, they do cover a lot of the essentials and if there's anything you'd like covered in more detail then let me know.

–James Montgomery  
February 2018

## ▽ The Unit Outline

- Is available on this MyLO site, in the Unit Information box
  - Describes the Intended Learning Objectives, which can be summarised as:
    1. Read code and diagnose errors
    2. Think algorithmically (that is, more like the computer)
    3. Write small programs (in Java)
    4. Manage complexity (by breaking problems into smaller pieces)
    5. Make maintainable code
- and some additional graduate attributes of an ICT professional that you will develop:
- Take initiative and work independently
  - Communicate effectively
  - Use abstraction and computational, creative and critical thinking to problem solve

## ▽ How the unit has changed

The unit underwent large-scale redevelopment in 2017. The previous structure had weekly lectures on content, weekly tutorials on that content (only), two major assignments and a 60% final open-book hand-written exam. Students found the jump in difficulty of the second major assignment to be quite steep, and a hand-written exam is clearly an inauthentic means of assessing programming skill.

In the new structure you work on a collection of in-semester tasks done on a computer, with documentation available, as well as two written tests on the absolute basics. Tasks are linked to a particular grade and passing only requires that all pass-level tasks be completed successfully. Most content-based lectures are now be pre-recorded (in shorter pieces) so you can access them when you need to. The flow-on effects are:

- Fewer hard deadlines (although the pass-level tasks *do* have deadlines for feedback during semester, in Weeks 6 and 11)
- Resubmission is possible
- You don't have to complete all tasks to finish successfully, just the pass-level ones
  - But you really *should* aim for a higher grade at the beginning






## ▽ Assessment

See the *Unit Outline* for a fuller description, as well as *How your final grade will be determined* under *Essential Information* on this site for a graphical summary.

- There are 18 pass-level tasks of varying difficulty, including the final task, a learning reflection report
- There are also two pass-level *tests*, that are assessed as Fail-Fix-Completed.
- There are also:
  - 6 credit-level tasks
  - 5 distinction-level tasks
  - 2 high distinction-level tasks
- **Passing the unit** requires that all pass-level tasks are completed as well as the two tests and learning reflection report.
- Qualifying for a grade requires that all tasks for that grade are completed *to a reasonable standard* and that all tasks of any lower grade are also completed. Passing the two tests and submitting the learning reflection report +
  - All PP tasks = PP (53–57)
  - All PP & CR tasks = CR (63–67)
  - All PP & CR & DN tasks = DN (73–77)
  - All PP & CR & DN & HD tasks *and* a brief face-to-face interview on your project work = HD (83–100)
- Borderline grades are possible for pass through to distinction (i.e., 50, 60 and 70); see *How your final grade will be determined* or the *Unit Outline* for details
- The quality of your learning reflection report (and of your work during semester) determines your numerical mark within a grade band: a poor report or consistent problems with the quality of your portfolio work mean a lower mark, while a better quality learning report and work meeting the minimum acceptable standard result in a higher mark.
  - If you made some progress on tasks for a higher grade then make sure you talk about the useful learning from that in your reflection report.

### Task and Test status





Each time we assess a task submission we will provide some feedback and assign it a status to indicate progress toward completion:



Task Status	Meaning
 <b>Overdue</b>	Due date has passed with no submission; get face-to-face help and submit final version later
 <b>Fail</b>	Does not satisfy criteria and there is no time to resubmit; status is used after semester when assessing tasks submitted very late in semester
 <b>Redo/Resubmit</b>	There are issues to address before resubmitting
 <b>Discuss</b>	Discuss with your tutor or demonstrate software
 <b>Completed</b>	Task is complete to required standard

Tests are pass/fix/fail, meaning that while you are expected to get them close to 100% correct, if you have only a few errors you will be allowed to correct them and explain your corrections to your tutor. If the errors are too serious (or you cannot explain your corrections to your tutor's satisfaction) you will be allowed to resit the test later in semester.

**Note:** If you're sick on the day of a test then please obtain a medical certificate so we can have you sit the test another time.

Tests are also assigned a status:

Test Status	Meaning
 <b>Completed</b>	Test is completed and demonstrates required learning
 <b>Fix</b>	Fix your mistakes and explain them to your tutor. If you are unable to satisfactorily correct your mistakes (or never show us) then you will need to resit
 <b>Resit</b>	You will need to resit the test later in semester. Review your test paper to see what you need to improve. The resit will be your final opportunity to pass the test
 <b>Fail</b>	There were remaining, serious errors when you resat the test (or you did not attend the resit)

**Note:** If you take your test paper home to  **Fix** it or to review before you  **Resit** then you *must* return your test paper to us (no later than Week 9 for Test 1 and Week 12 for Test 2).


## ▽ Classes & Other Activities

- Face-to-face lectures & demonstrations: 1–2 hours each week
- Online mini-lectures: 1–2 hours each week (depending on what you need to learn about)
- Tutorials: 2 hours each week
  - work on the portfolio tasks and receive assistance
  - have some tasks marked as Complete
- Tests: in Week 6 and 11 lecture time
- Private Study:
  - read online notes, review lectures
  - complete task work
  - self-study activities (on MyLO)

### Attendance Requirements

See also the *Unit Outline* for a more detailed description

You **must** attend:

- lectures when tests are scheduled
- tutorials when tasks are marked as  **Discuss**

I hope you **choose to attend**:

- all other lecture & demonstration times
- most tutorials

Your active engagement during the first part of semester will be monitored by your tutorial attendance, submission of some of the earliest pass-level tasks and by viewing these notes. So you've already done one of those! (But don't rush to check the [zero-weighted] grade item associated with that action as MyLO can't automatically fill it in; we'll update those periodically).

## ▽ Resources

- MyLO site: <https://mylo.utas.edu.au>
  - lecture slides
  - lecture example programs
  - links to the externally-hosted video recordings (Echo360) for Hobart lectures
  - the introductory programming notes (where you are now)
    - self-study activities covering basic skills (beneath this section of the site)
  - the portfolio tasks and submission folders
- Software: links available on unit's MyLO site under Essential Information
- Text Book (recommended)
  - *Java Software Solutions : Foundations of Program Design (8th edition)* by Lewis and Loftus. (Earlier editions can be used but are less ideal)
- Computers
  - Computing laboratories
  - Own machine
- PASS Leader: David Winkler (Hobart), Joeby Neil (Launceston)
- Teaching Staff
  - Lecturer(s):
    - Dr James Montgomery (HBT), Dr Winyu Chinthammit (LTN)
    - Available for face-to-face help during advertised consultation times
    - And via email, to James (course queries) or Winyu (specific Launceston queries)
  - Tutors: available to assist you during tutorials or their consultation times
- Your peers: May discuss general issues and any non-assessable study work collectively

### Avoiding plagiarism

The University of Tasmania [Academic Integrity](#) site summarises plagiarism as:

Plagiarism is a form of cheating. It is taking and using someone else's thoughts, writings or inventions and representing them as your own...

Academic integrity is broader than that, since sharing your work with someone else is also inappropriate and may be penalised.

**Did you know?** The main reasons students plagiarise in programming are not that they are 'lazy' but because:

- **programming is hard** (you can be really close to a solution that will work but the computer just won't run it);
- students may not recognise *code* as *work*, equivalent to copying passages from an essay written by someone else; and
- source code is natively electronic and so easily copied (and 'hiding' its origins looks easy)

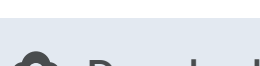
We'll help you to avoid plagiarism by:

- Giving you multiple opportunities to act on feedback to get your solutions working
- Asking you to explain your programs to your tutor
- Using plagiarism detection software designed for program source code

**Tip:** To avoid inadvertently plagiarising someone else's work you should **avoid**:

- Discussing low-level details of how you will solve a problem
- Looking at someone else's solution, since it will make it *really* hard to imagine anything different when you go to write it yourself (this also means don't show your solution to others)

Use the right arrow above this document to continue to the next section, which accompanies the second half of the Week 1 lecture.



Download



Print



#### Activity Details

✓ You have viewed this topic