CS1101 STUDENT HANDBOOK

Programming is FUN! No kidding. Despite what you might have heard from your friends and seniors, programming is actually fun.

It is FUN because you will be working in a tractable medium and in an environment that provides instant feedback on your work. [1]

It is FUN because you can make your own creation that is useful to others.

It is FUN because it is an activity which you can perform on your own, or one in which you can enjoy working with others in a team.

It is FUN because it teaches you life-long independent learning and problem-solving skills. Every problem you solve is a new problem, so you are constantly learning something new.

CS1101 is only the first step in the journey of learning programming and beyond. It is an introductory course in which we show you the basic elements of programming. Yet, it could be a challenging module.

And precisely because it is challenging, it is FUN! You will find that the challenges in CS1101 are not insurmountable as long as you:

- ✓ Adopt a positive attitude
- ✓ Know the expectation
- ✓ Set your goal
- ✓ Practice, practice and practice
- ✓ Ask questions

This CS1101 Student Handbook tells you how much we want you to succeed. It tells you about our expectation of you. It tells you about what you should pay attention to. It tells you about what we have done behind the scene to ease your pain in your journey of learning.

Pre-allocation of module – Sectional Groups and Discussion Groups

To relieve you of the task of registering for the module and discussion group, the School has pre-allocated the sectional group and discussion group to you.

CS1101 is split into a number of sectional groups. Each sectional group is small enough to fit into a programming lab. This allows us to conduct lab-based lectures in an environment conducive to learning

[1] From *The Mythical Man-Month: Essays on Software Engineering* by Fred Brooks.

programming, with the computer right in front of you. The lectures are no longer monologues, but consist of hands-on exercises and a lot of interaction between the lecturer and the students. We have done this for a few years and the students simply loved it! You might have heard from your seniors that it is quite common for students to skip lectures, but you will regret if you skip CS1101 lectures!

Besides attending 3-hour weekly lectures (we still stick to the term “lectures” although I have explained to you that they are not quite like the lectures you have experienced), you also need to attend 2-hour weekly discussion sessions. Discussion sessions provide even more interaction, as they are held in smaller programming labs, simply because the group size is even smaller. The person who conducts discussion sessions is your discussion leader (DL in short), who is usually a senior undergraduate very skillful in programming. Being an undergraduate, your DL is fully aware of your angst and concerns. Your DL is also the person who grades your assignments, so he or she knows your strength and weaknesses, and is thus able to provide guidance tailored to your needs.

Hands-on Practice

It’s impossible to learn programming by just reading a book. (That said, you still need to read the book!) That’s the main reason we conduct lectures and discussion sessions in a lab-based environment. And that’s why we keep saying: Practice, practice, practice!

An important aspect of learning is feedback. After your practice, you certainly would like to get as much feedback as possible, so that you know which areas you should improve on.

We provide take-home lab exercises so that you can focus on solving these exercises on your own (as much as possible). These are marked by your DL, but the marks are not counted towards your final grade. This way, we hope to achieve two objectives:

- ✓ To provide you with a less stressful environment, with your mind on enjoying the learning experience rather than worried about the marks;
- ✓ To provide you with feedback on your programs, for corrective actions to be done if necessary.

Although we said the marks for these take-home assignments are not counted towards your final grade, we want to encourage you to attempt them. To this end, we will reward you with a token mark for your attempt. As long as you submit your work (with reasonable effort) on time, you will get the mark!

We used to give out take-home lab assignments which were marked and counted towards the final grade with a substantial weightage, and we discovered that some students copied programs from others due to stress. The students were penalized, and they got even more stressed out. We hope this arrangement will eradicate such incidents of plagiarism which we absolutely frown upon.

Hands-on practice should not be limited to those exercises given out in class. You should try as many programs as possible on your own. Practice, practice, practice!

Assessments

At the end of the day, we still need to assess your programming skills (that is why you have to take the take-home labs and other practice exercises seriously, though they are not assessed; the more seriously you take it, the more you learn). This is done through the final examination and a series of continual assessments, which include a mid-term written test, three sit-in labs, and a practical exam.

The mid-term test and practical exam are conducted on Saturdays, while the sit-in labs are conducted during your discussion sessions. Please refer to the module website for the schedule and the weights of these assessments.

Remember: you reap what you sow. If you take your practice seriously, your effort will pay off and the result will show.

Important Links

These are the links or documents you must not miss:

- ✓ Module website
- ✓ IVLE forums
- ✓ API documentation
- ✓ Java Programming Style, Design and Marking Guidelines
- ✓ CourseMarker (CM)

The first place you should go to is the all-important **module website** at http://www.comp.nus.edu.sg/~cs1101 which contains everything you need to know about CS1101. Visit it now! The softcopy of this document is on it. It is constantly being updated, so please check it out regularly.

It is very important to ask questions. Ask questions in lectures, ask questions in discussion sessions. If you have a question when you are outside the class, instead of emailing us – reserve that for private matters – post your question on the **IVLE discussion forums**. You will get much quicker response that way because the forums are open to a wider audience, and the answer can be read by everybody hence benefitting all students. We love to see active participation on the discussion forms.

In this module, you need to do a lot of exploration on your own. The **API documentation page** http://java.sun.com/j2se/1.5.0/docs/api/ is another website you should visit frequently after we have introduced API to you.

We introduce and emphasise good programming practice in this module. This involves style issues such as program layout, comments, variable naming, and design issues such as user-friendliness and efficiency. The write-up **Java Programming Style, Design and Marking Guidelines** includes tips on how you should write your programs that are considered good. It also includes information on how your programs are graded. Refer to this write-up from time to time as you pick up more programming skills. This write-up is on the module website under the “CA”, “Labs” page: http://www.comp.nus.edu.sg/~cs1101/3_ca/labs/styleguide/styleguide.html.

We use an autograder called **CourseMarker** for you to submit your lab assignments. More about it in the next section.

Links of useful websites, which include those mentioned above, are found on the module website.

CourseMarker (CM)

You submit your lab exercises to an online system called CourseMarker, or CM in short. CM assesses your program using the following three “tools” and provides instant feedback from the respective tool:

- ✓ *Typographic Tool:* Checks your program for typographical items such as layout, indentation, whitespace, comments.
- ✓ *Dynamic Tests Tool:* Runs your program against some test data sets.
- ✓ *Feature Tool:* Checks your program for certain programming constructs that you should use in your program. For example, the existence of ‘for’ loops.

Of the three CM tools, we rely on dynamic tests tool the most. When you submit your program, CM runs your program on a few test data sets (we set it to 3). For each data set, it checks the output of your program against the correct output, and awards marks accordingly. This provides instant feedback to you on the **correctness** of your program. If you get a poor result, you may debug your program and re-submit it.

Typographic tool is turned off (we set it to 0%) because we found that the rules it uses are too rigid. Sometimes, a program with pretty good layout receives rather bad feedback from CM when the program does not conform to CM’s expectation. We think that this is best left to your DL to assess. Note that though we turn this tool off, it continues to provide its feedback to you; you may just ignore the feedback.

Feature tool is handy at the beginning when we look for the presence of a certain programming construct in your program. As you learn more programming constructs, this tool become less useful. So usually we will turn this tool off too in the later part of the module.

So in conclusion, when you submit your program to CM, you may ignore feedback from the typographic tool and (most of the time) the feature tool. Look at the feedback from the dynamic tests, which indicates whether your program generates the correct output or not on the test data.

The big question: does this mean that your program is only assessed based on the few test data sets? The answer is an emphatic NO! The fact is that we do not leave all the grading to the CM. Your hardworking DL will go through your program and grade its style (typographic) and design, and provide comments on your program. You will get to read your DL's comments. We will also run your program again on a bigger and more comprehensive set of test data (usually 10).

Program Correctness – Testing and Debugging

Here is an important message for all of you. Some students think that once their program passes all the test data in CM, the program deserves full mark for correctness. This cannot be further from the truth. The truth is: successful testing is not a proof of correctness, unless you try every possible input exhaustively, but that is impossible for most cases.

We make all test data unknown to you because we want you to test your program thoroughly yourself before submitting it. We may release the test data to you after the assignment is graded, but not before the submission deadline. Releasing the test data to you early only makes you focus your attention on satisfying these test data and you may tweak your program to run only on these data but not others. Even if your program does pass all the test data it is subjected to, your program may still be incorrect due to the possibility that the test data we use are not comprehensive enough to catch the flaw in your program.

Devising your own test data to test your program is a very important aspect of programming which many students either take it too lightly or are not adept at it. Such skill will be very useful beyond CS1101, so this is the good time to start.

Equally important is knowing how to debug your own programs. This involves programming techniques as well as the use of tools such as the debugger. Pay attention to these topics when they are covered in class. Sometimes you may find that you spend more time testing and debugging your program after writing it.

Do not ask your friends, DL, or lecturer to test and debug your program for you. Do it yourself. However, if you are unsure about how to do that, do bug us to explain the techniques again.

Differences in Marking and Lecturing

Consistency in marking can be achieved if we leave it entirely to the CM to grade your programs. However, I can tell you in absolute certainty that no student will like it. CM, being a program, is good at telling you that your program does not work, but is unable to detect logic errors in your program, much less to offer suggestion on how to correct the errors or improve your program. Only a human grader, like your DL, is able to do this job well. And we do exactly this.

However, this poses another potential problem. We cannot have a single DL grade all 300 programs as that would probably take too long. Instead, we get the DLs to grade only their own students' programs, so that they can get it done quickly in a few days, so that you can get the feedback promptly. Now, if we have 20 DLs, there is bound to be some differences in their marking. We do have a marking scheme for all DLs to follow so as to minimize differences due to personal style and preferences, and the marking principle is made known to you as well in the document "Java Programming Style, Design and Marking Guidelines", but no matter how detail the marking scheme is, there are bound to be differences especially on areas where judgement has to be made, simply because DLs are humans and not machines! Hence, sometimes we encountered complaints from students that their DL is stricter than another.

We are well aware of this (after all, we are in this business for many years) and we will take appropriate steps to address this issue, such as comparative studies of the DLs' marking and moderation when necessary. We just want to seek your understanding and your tolerance to accept the small differences, and we hope that you do not get unduly worried about this.

Differences exist not only among the DLs, but also among the lecturers. Every lecturer has his or her own style, and he or she may give different examples and exercises in class, even though all lecturers are following the same syllabus. Again, we ask that you accept such differences, or even embrace such differences in the name of diversity.

Progress Chart – Self-monitoring

We provide a Progress Chart on the next two pages for you to keep tab of your own progress. Fill it up at least once a week. When you feel the need to approach your lecturer or DL for consultation, bring along this chart. If we know what specific areas you are having problem with, we are in a better position to help you.

I think I have addressed almost all the important issues we want you to know at this point of time. From time to time, do go over this write-up again as you gain more knowledge and insight.

Enjoy CS1101 and have FUN!

Aaron Tan
CS1101 coordinator
7 August 2009

CS1101: My Progress Chart

About Myself

My name: _________________________________ My sectional group: ____ My discussion group: ____

Do I have programming experience? ☐ No/very little ☐ Some ☐ Yes

My target grade for this module: ______

Week	Topics covered	**Do I understand?** 1: Do not understand at all ☹ 5: Understand completely ☺	Notes (What actions will I take?)
1		1☐ 2☐ 3☐ 4☐ 5☐	
2		1☐ 2☐ 3☐ 4☐ 5☐	
3		1☐ 2☐ 3☐ 4☐ 5☐	
4		1☐ 2☐ 3☐ 4☐ 5☐	
5		1☐ 2☐ 3☐ 4☐ 5☐	
6		1☐ 2☐ 3☐ 4☐ 5☐	
Recess	Review: Am I meeting my target?	1☐ 2☐ 3☐ 4☐ 5☐	
7		1☐ 2☐ 3☐ 4☐ 5☐	
8		1☐ 2☐ 3☐ 4☐ 5☐	
9		1☐ 2☐ 3☐ 4☐ 5☐	
10		1☐ 2☐ 3☐ 4☐ 5☐	
11		1☐ 2☐ 3☐ 4☐ 5☐	
12		1☐ 2☐ 3☐ 4☐ 5☐	
13		1☐ 2☐ 3☐ 4☐ 5☐	

Assessments

	Sit-in lab #1	Sit-in lab #2	Mid-term test	Sit-in lab #3	PE
My score					
Did I meet my own expectation?					
What actions will I take?					

Class exercises

From time to time your lecturer and DL may give out quizzes and programming exercises in class. Take note of these activities and how well you fare, and what follow-up actions you think you should take if necessary. You may also fill in any general comment or reminder for yourself even if there is no particular activity associated with it.

Date	Activity	Score (if appropriate)	Comments or actions to be taken