COMP 202 - Foundations of Programming

McGill University, Summer 2019

Course Details

Section 1	Instructor:	Tzu-Yang (Ben) Yu
	Office:	McConnell 224
	Office hours:	MW 13:30-14:30
	Contact info:	`tzu-yang.yu@mail.mcgill.ca`
	Lecture room:	MDHAR G-10
	Class times:	MW 10:35–12:55

myCourses Webpage

- myCourses: `http://www.mcgill.ca/lms/`

Teaching Assistants (T.A.)

TAs will be available for office hours, on the third floor of the Trottier building room 3110, to help you with your assignments and answer questions about the course material. You can also contact TAs by e-mail. Each TA's office hours and e-mail address will be posted on myCourses.

Contacting Instructors and Teaching Assistants

Post all your questions about the course (including assignments and the midterm/final) on the myCourses message boards so that everyone can see both the questions and the answers. You may freely answer other students' questions as well, with one important exception: you may not provide solution code (although you are permitted to provide one or two lines of code to illustrate a point). Of course, you can send e-mail to a teaching assistant or instructor directly for private matters.

Students are expected to monitor both their McGill e-mail account and myCourses for course-related news and information.

Course Description

Welcome to COMP-202! Please read this document carefully and keep it for reference throughout the term.

This course introduces students to computer programming and is intended for those with little or no background in the subject. No knowledge of computer science in general is necessary or expected. On the other hand, basic computer skills such as browsing the Web, sending e-mail, creating documents with a word processor, and other such fundamental tasks will be necessary in this course.

The course uses the Java programming language. Java is an example of an *object-oriented* language (as are Python, C++, and many others). We will see what this term means later in the course. Other kinds of programming languages include functional and logical programming languages. Despite these differences, there are some basic building blocks in all languages that are fundamental to programming and software development. A large part of this course will focus on these basic building blocks before we move to object-oriented or other language-specific concepts.

Learning how to program is not easy; it is not a set of facts that one can simply memorize. In principle, a computer program is simply a set of instructions that tells a computer to perform a task. However, finding the right set of instructions can be quite challenging. For that, one has to learn how to structure a larger problem into small subsets, and then find the solution to each particular subset. This course aims to teach students *a way of thinking* that will enable them to build non-trivial programs.

Primary Learning Objectives

By the end of this course, you will be able to:

- Design and describe precise, unambiguous instructions that can be used to solve a problem or perform a task;
- Translate these instructions into a language that a computer can understand (Java);
- Write programs that solve complex problems by decomposing them into simpler subproblems;
- Apply programming-style conventions to make your programs easy to understand, debug and modify;
- Learn independently about new programming-language features and libraries, as you encounter them, by reading documentation and by experimenting.

What This Course is *Not* About

This course is not about how to use a computer. It will not teach you how to send e-mail, browse the Web, create word processing documents or spreadsheets, set-up and configure a computer, use specific software applications (except those needed to complete coursework), design Web pages, or deal with operating system or hardware problems. However, the course offers introductory tutorials that provide instruction in aspects of computer usage necessary to complete coursework.

Course Prerequisites

- A CEGEP-level mathematics course or equivalent. For students who did not attend CEGEP, any upper-level mathematics course is sufficient. However, attention to detail, rigour, and the ability to think in an abstract manner is much more important than knowledge of calculus, algebra, or trigonometry.

Recommended Textbook:

- *How to Think Like a Computer Scientist: Java Version*, 6th edition. Allen B. Downey.

  Available at no cost under the GNU Free Documentation License at:

  `http://greenteapress.com/thinkjava6/thinkjava.pdf`

Other References

- *Java Documentation.* You can browse or download this from Oracle's Web site. Use the documentation appropriate for the Java version you are using.
  - Java 8.0 Documentation: `http://download.oracle.com/javase/8/docs/`
- *The Java Tutorial.* You can also browse or download this from Oracle's Web site.
  - `http://download.oracle.com/javase/tutorial/index.html`
- There are many books available to learn Java. For instance:
  Java Software Solutions: Foundations of Program Design, 7th/8th Edition.
  John Lewis and William Loftus. Addison-Wesley. 2012. ISBN: 0132149184.

Grading Scheme and Deadline Policy

Your final grade in the course is calculated by the following:

- **Assignments:** 40%
- **Midterm Examination:** 20%
- **Final Examination:** 40%

Assignments

There will be **4** assignments consisting of writing Java programs. It is *very important* that you complete all assignments, as this is the best way to learn the material. By working hard on the assignments, you will gain essential experience needed to solve problems on the midterm and final examinations.

To receive full grades, assignments (as well as all other course work) **MUST** represent your own personal efforts (see the section on Plagiarism Policy and Assignments below).

Late Policy: Late assignments will be deducted 10 points each day or fraction thereof for which they are late, including weekend days and holidays; that is, assignments that are between 0 and 24 hours late will be deducted 10 points, assignments that are between 24 and 48 hours late will be deducted 20 points, and so on. Assignments submitted more than 2 days after the deadline will not be accepted, nor graded, and will therefore receive a grade of 0.

Assignment submission will always take place on myCourses. Instructors and TAs will discuss how to use it during the lectures and tutorials, but every student is responsible for verifying that their submissions are successful. If you believe the content of your myCourses submission box is different from what you have submitted, you must e-mail your section instructor within **5 days** of the assignment deadline in question to provide evidence of your correct submission.

Assignment marks will also be posted on myCourses. It is your responsibility to check that the marks are correct and to notify your section instructor of any errors or missing marks. If you believe that your assignment was graded incorrectly, you should first email the TA who marked your assignment. Their email should be in the feedback left on your assignment. If you and the TA cannot resolve the discussion, then you should contact your instructor.

The instructors reserve the right to modify the lateness policy for a particular assignment; any such modifications will be clearly indicated at the beginning of the relevant assignment specifications. **Plan appropriately and do not submit to myCourses only minutes before the assignment deadline**. Take care, programming assignments are notoriously time-consuming and individual exceptions to the lateness policy will not be granted without appropriate justification submitted in writing and supported by documentary evidence.

Midterm Examination

The midterm examination will take place in the class time at the same class on the following date:

- Monday, May 27th, 10:35am - 12:55pm

The final exam will take place on the last date of the class which is..

- Wednesday, June 19th, 10:35am - 12:55pm

Campus Computer Laboratories

Using the SOCS computer laboratory facilities: All students registered in COMP-202 may use the SOCS computer laboratory facilities to do their work regardless of the program in which they are registered. These facilities are located on the third floor of the Trottier building.

Refer to `https://www.cs.mcgill.ca/about/facilities/` for more information on the SOCS computer laboratory facilities.

Other computer laboratory facilities: You may also use other computer laboratory facilities on campus to do your work. Most facilities are available to all McGill students, but there are facilities which grant usage privileges only to students registered in a course or program offered by the faculty or department which manages the facility.

Students should contact the work area of their choice to inquire about access requirements, opening hours, or any further information such as software availability.

Required Software

You will use the Java compiler on personal computers to compile the programs you are required to write for the assignments. The Java compiler is included in a larger software package called the Java Development Kit (JDK). You can use any **plain-text editor** of your choice to write your programs, and then use the tools included with the JDK to compile and run them. There are several of these plain-text editors such as Notepad++ and RText. Note that Microsoft Word will NOT work properly for writing Java code.

Typically, though, programmers nowadays use an integrated development environment (IDE) to write programs. IDEs provide an editor that allows you to type your program, commands to compile and run it, and many other useful tools, all in one application. We recommend a simple and intuitive IDE called *Dr. Java* (`http://drjava.sourceforge.net`). It is a perfect programming environment for solving the assignments of this course.

However, to use a more powerful IDE which can assist you in writing your code, we recommend *Eclipse* (`http://www.eclipse.org/`). All instructors and teaching assistants will provide support for these IDEs.

The JDK is installed on the computers in the SOCS laboratory, as are Dr. Java and Eclipse. You are encouraged to install the JDK and either Dr. Java or Eclipse on your own computer so you do not have to depend on the SOCS computer laboratory facilities to do your work. Installing any of these is fairly straightforward. If you need help, you can consult a TA during office hours.

- **Required:** The JDK.
  - Windows users: You may download the JDK installation program from the following Web site: `http://www.oracle.com/technetwork/java/javase/downloads` (choose *Java - Download* or *JDK* (click on the *Download JDK* button), with no additional software such as Java EE or NetBeans). The JDK is available at no cost, and there is no time limit on its use. **You should install the JDK before any IDE.**

  - Mac users: JDK 6.0, 7.0, or 8.0 is installed by default on most Mac computers. It is available as a Mac OS software update.
  - GNU/Linux users: A JDK is available in the software repositories of most of the major GNU/Linux distributions like Ubuntu or Fedora; you can install it through your package manager.

- **Recommended:** Dr. Java. You should install this **after** you have installed the JDK, as this will enable you to avoid several configuration problems. Note that if you have a Mac you might encounter some issues when trying to install Dr. Java. For Dr. Java to work on a Mac you need a version of JDK earlier than 8. You can ask the TAs for help installing Dr. Java or you can check out the following youtube video: `https://www.youtube.com/watch?v=d_GhOopu0lY`.
- **Optional:** Other IDEs are slightly harder to install and use, but offer fantastic benefits such as automatically checking your code for errors. This can be a great help if you struggle with typos.
  - Eclipse: `http://www.eclipse.org/downloads/` (choose *Eclipse IDE for Java Developers*)
  - IntelliJ IDEA: `https://www.jetbrains.com/idea`

Plagiarism Policy

Official policy: McGill University values academic integrity. Therefore all students must understand the meaning and consequences of cheating, plagiarism, and other academic offenses under the Code of Student Conduct and Disciplinary Procedures (see `www.mcgill.ca/integrity/` for more information).

Plagiarism Policy and Assignments

You must include your name and McGill ID number at the top of each source code file that you implement and submit. By doing so, you are certifying that the program or module is entirely your own, and represents only the result of your own efforts.

Work submitted for this course must represent your own efforts. Assignments **must** be done **individually**; you **must not** work in groups. Do not rely on friends or tutors to do your work for you. You **must not** copy any other person's work in any manner (electronically or otherwise), even if this work is in the public domain or you have permission from its author to use it and/or modify it in your own work (obviously, this prohibition does not apply to source code supplied by instructors explicitly for this purpose). Furthermore, you **must not** give a copy of your work to any other person.

The plagiarism policy is not meant to discourage interaction or discussion among students. You are encouraged to discuss assignment questions with instructors, TAs, and your fellow students. However, there is a difference between discussing ideas and working in groups or copying someone else's solution. A good rule of thumb is that when you discuss assignments with your fellow students, you should not leave the discussion with written notes. Also, when you write your solution to an assignment, you should do it on your own.

Students who require assistance with their assignments should see a TA or instructor during their office hours. If you have only partially finished an assignment, **document the parts that do not work**, and submit what you managed to complete for partial credit. However, the code to answer any question must compile (with the test engine provided to you, if any), or else you will receive a maximum grade of 25% on that question.

We will be using automated software similarity detection tools to compare your assignment submissions to that of all other students registered in the course, and these tools are very effective at what they have been designed for. However, note that the main use of these tools is to determine which submissions should be manually checked for similarity by an instructor or TA; we will not accuse anyone of copying or working in groups based solely on the output of these tools.

You may also be asked to present and explain your assignment submissions to an instructor at any time.

Students who put their name on any code that are not entirely their own work will be referred to the appropriate university official who will assess the need for disciplinary action.

Course Content

Note that minor changes in content, reading material, and times for tutorials and assignments may occur. It is your responsibility to attend class and be aware of what content is being covered.

Tutorials

Throughout the term, there will be several (optional) tutorials. These will be designed to help you with the material and assignments, and to give you a chance to ask questions in a smaller environment than lectures. Further information will be posted on myCourses. It is not necessary to register for tutorials.

The tutorials will review and practice material presented in class. For example, a tutorial in the fifth week might cover the `while` and `for` statements to ensure that everyone is caught up. As well, three special tutorials will be provided:

Tutorial	Title	Contents
T0	Basics of Course Software Tools	The JDK, Dr. Java and Eclipse: creating, loading, editing, saving, compiling, and running programs
TM	Midterm Review	Review of all material for the midterm, including questions selected from previous midterms
TF	Final Exam Review	Review of all material for the final, including questions selected from previous finals

Approximate Schedule of Topics

The references to chapters in the table below are from the recommended online textbook (`http://greenteapress.com/thinkjava6/thinkjava.pdf`). Although our lectures will not follow the textbook exactly, especially later in the semester, reading the textbook is highly recommended. **The following schedule is only approximate and may/will change depending on how the semester unfolds.**

	Week	Topics	Reference	Events
Intro	1– May 1	What is programming?	Chapter 1 – 3	
		What is computer science?	Appendix A.1–A.4	
		How does a computer work?		
		Binary numbers		
		Structure of a Java program		
		Variables and primitive data types		
		Expressions and assignments		
Fundamentals	2–May 8	Input arguments	Chapters 3–6, 9	A1-due
		Methods		(May 12)
		mod operator (%)		
		`if` and `if-else` statements		
		More on methods		
		Strings		
		Primitive type conversions		
	3–May 15	`while` statements	Chapters 7–8, Appendix C	
		`for` statements		
		Type issues (division, casting, overflows)		
		Errors (syntax, run-time, logic, throws)		
		Arrays		
		Reference types		
	4–May 22	null	Chapter 8	A2-due
		Multidimensional Arrays		(May 20)
		Review		
	5–May 29	Scanner and Random	Chapter 3, 10 – 11	Midterm
		Intro to Objects		(May 27)
		Classes		
		Instantiating and using objects		
		Modifiers		
		Constructors		
	6–June 5	Arrays of Objects	Chapter 11 – 12	A3-due
		Try-catch blocks		(June 5)
		Checked vs Unchecked Exceptions		
	7–June 12	ArrayLists		A4-due
		File I/O		(June 17)
Not cover in Exam	8–June 19	HashMap/HashSet	Chapter 13, 14	Final Exam
		Inheritance		(June 19)
		Review		