COMP-202A: Foundations of Programming

McGill University, Fall 2014

Course Details

Section 1	Instructor:	Melanie Lyman-Abramovitch
	Office:	McConnell Engineering Building (MC) 312
	Office hours:	F 14:00-16:00 (or by appointment)
	Contact info:	`melanie.lyman-abramovitch@mail.mcgill.ca`
	Lecture room:	ADAMS AUD
	Class times:	MWF 9:35–10:25
Section 2	Instructor:	Juan Camilo Gamboa Higuera
	Office:	McConnell Engineering Building (MC) 403
	Office hours:	TR 14:30-15:30 (or by appointment)
	Contact info:	`gamboa@cim.mcgill.ca`
	Lecture room:	MCMED 504
	Class times:	TR 13:05–14:25
Section 3	Instructor:	Jonathan Tremblay
	Office:	McConnell Engineering Building (MC) 231
	Office hours:	MW 13:30-14:30 (or by appointment)
	Contact info:	`jtremblay@cs.mcgill.ca`
	Lecture room:	SADB M-1
	Class times:	MWF 12:35–13:25

Important Links

- myCourses (WebCT Vista): `http://www.mcgill.ca/lms/`

Contacting Instructors and Teaching Assistants

Post all your questions about assignments on the myCourses message boards so 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; to that end, you may use the e-mail facilities provided by McGill or any e-mail account you have with any e-mail provider.

Students are expected to monitor their McGill e-mail account, myCourses, and the course home page for course-related news and information.

Introduction

The School of Computer Science (SOCS) would like to welcome you to COMP-202. The purpose of this document is to provide you with an overview of what lies ahead in this course. We shall begin with a brief

introduction of the course contents, followed by some important general information about the course. Please read this document carefully and keep it for reference throughout the term.

Course Description

This course introduces students to computer programming and is intended for those with little or no background in the subject. Furthermore, no knowledge of computer science in general is necessary or even 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 a valuable asset in this course.

The course uses the Java programming language. As with human languages, programming languages can be grouped; languages in the same group are conceptually similar, while languages from different groups follow quite different paradigms. Java is an *object-oriented* language (as are C++ and many others). Examples of other language groups are imperative programming languages and functional programming languages.

Despite these differences, there are some basic building blocks in all languages that are fundamental to programming and software development in general. A large part of this course will naturally 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. A large part of this course is dedicated to teaching 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 [by a computer] 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, nor 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 and Textbooks

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, rigor, and the ability to think in an abstract manner is much more important than knowledge of calculus, algebra, or trigonometry.

Recommended textbooks:

- *How to Think Like a Computer Scientist: Java Version*, 4th edition. Allen B. Downey.

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

  `http://www.greenteapress.com/thinkapjava/thinkapjava.pdf`

- *Java Software Solutions: Foundations of Program Design*, 7th Edition. John Lewis and William Loftus. Addison-Wesley. 2012. ISBN: 0132149184.

  You may purchase a copy of this textbook from the McGill bookstore or from amazon.com.

  The order in which material is presented in course lectures *does not* match any textbook in particular. However, the above book is an excellent resource.

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 6.0 Documentation: `http://download.oracle.com/javase/6/docs/`
  - Java 7.0 Documentation: `http://download.oracle.com/javase/7/docs/`
  - 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`

Grading Scheme and Deadline Policy

Your final grade in the course is calculated as follows:

- **Assignments:** 35%
- **Midterm Examination:** 20%
- **Final Examination:** 45%

Students who perform better on the final than on the midterm exam will have the (automatic) option to make their grading scheme 40% assignments and 60% final. However, the assignments are a key part of learning the material, and as such there is no 100 % final option.

In exceptional situations, students may write a supplemental examination. However, ability to do so is not automatic, and depends on your exact situation; contact your Student Affairs Office for further information. The supplemental examination represents 100% of your supplemental grade.

Students who receive unsatisfactory final grades will **NOT** have the option to submit additional work in order to improve their grades.

Official language policy for graded work: In accord [*sic*] with McGill University's Charter of Students' Rights, students in this course have the right to submit in English or in French any written work that is to be graded.

Assignments

There will be **five** assignments, each of which will require programming. The first two assignments will be designed to give you some practice programming and applying the initial concepts you learn in class. The third to fifth assignment will be more involved and will be closer to a small project in length.

- Assignment 1: 7%
- Assignment 2: 7%
- Assignment 3: 7%
- Assignment 4: 7%
- Assignment 5: 7%

It is important that you complete all assignments, as this is the major way in which you will 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 assignments will be deducted 10% 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%, assignments that are between 24 and 48 hours late will be deducted 20%, 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 (see Submitting Assignments, below). 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.

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**; 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 evening at the following date and time:

- Wednesday, October 22nd from 18:00 to 21:00

The room assignments will be announced in class and posted on the course webpage when it is closer to the date.

If you have a scheduling conflict (you are registered in a course for which the midterm examination overlaps with the COMP-202 midterm examination), you **MUST** notify your section instructor in writing by **13th of October**.

All students who are absent from the midterm examination will have the option of writing the examination at home and getting it graded in order to get feedback on their progress in the course; however, in such cases, their midterm examination grades will **NOT** be considered in the calculation of their final grade.

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. All computers are physically accessible on weekdays from 10:00 until 20:00, and on weekends from 12:00 until 20:00; a consultant will be on duty during these times. Outside of these hours, only the computers located in the open area will be physically accessible, and no consultant will be on duty. Please note that these hours may vary for the first week of lectures.

In order to enter the Trottier building before 7:00 or after 21:00 from Monday to Friday, or at anytime during weekends, your McGill ID must be added to the building access list. This will enable card readers located at the entrances of the building to recognize your McGill ID card. Your McGill ID will automatically be added to the building access list if you are officially registered in a computer science course. However, if you registered late or had other registration problems, this might not have been done in your case. If you are officially registered in the course but unable to enter the Trottier building using your McGill ID card outside the building's opening hours, contact the SOCS Systems Staff by e-mail at `help@cs.mcgill.ca`, and request that your McGill ID be added to the building access list.

Students who wish to use the SOCS computer laboratory facilities must first create an account; this can be accomplished by going to any computer on the third floor of the Trottier building, logging in as `newuser`, and supplying `newuser` as the password. You will then be invited to fill out a Web form. Upon completion of this form, you will be provided with the user ID and password with which you will be able to use the SOCS computer systems. Note that if you are not officially registered in this course, you will not be able to create an account for use with the SOCS computer systems. You only need to perform the account creation procedure once.

All computers in the SOCS laboratory facilities run Ubuntu GNU/Linux, which is a Unix-like operating system. Members of the SOCS Systems Staff will hold Unix seminars at the beginning of the term for those who are new to Unix. Information regarding these seminars will be given during the first lectures. **If you are only familiar with a Windows (95/98/Me/NT/2000/XP/Vista/7/8) or Mac OS X environment, it is recommended but not required that you attend these seminars.**

Refer to `http://socsinfo.cs.mcgill.ca/wiki/Main_Page` 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 enquire about access requirements, opening hours, or any further information such as software availability.

Personal Computers and 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. It you want to use a plain text editor, we suggest you use *R*Text (http://fifesoft.com/rtext/).

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, if you are serious about computer science and are planning to write sophisticated programs in the future, you can also directly start using a professional IDE. In this case we recommend *E*clipse (http://www.eclipse.org/)

All teaching assistants will provide support for RText and Dr. Java. However, it is very likely that the TAs also know Eclipse.

The JDK is installed on the computers in the SOCS laboratory, as are RText, Dr. Java and Eclipse. You are encouraged to install the JDK, as well as an IDE if you wish to use one, 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 6 Update 23* (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 or 7.0 is installed by default on most Mac computers. It is available as a Mac OS software update.
  - GNU/Linux users: JDK 6.0 or 7.0 is available in the software repositories of most of the major GNU/Linux distributions like Ubuntu, Fedora, and OpenSUSE; you can install it through your package manager.
- **Optional:** RText, Dr. Java and Eclipse. You should install these packages only after you have installed the JDK, as this will enable you to avoid several configuration problems. You may download these software packages from the following Web sites:
  - RText: `http://fifesoft.com/rtext/`
  - Dr. Java: `http://drjava.sourceforge.net`
  - Eclipse: `http://www.eclipse.org/downloads/` (choose *Eclipse IDE for Java Developers*)

Teaching Assistants (TAs)

Each TA will be available for at least one office hour per week, on the third floor of the Trottier building, 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.

Submitting Assignments

In this course, you will be using the myCourses learning management system for assignment submission; instructors and TAs will discuss how to use it during the lectures and tutorials. Each assignment will contain directions on what to submit. Assignments will also be graded via myCourses.

In addition, myCourses will have a discussion board that you will be able to use to ask questions about assignments. Students, instructors, and TAs will post answers to questions. **Please do not post assignment answers or code (although you may post one or two-line long code snippets to illustrate a point).**

Posting of Course Marks

The course marks will be posted on myCourses. The marks will be updated after each assignment, as well as after the midterm. It is your responsibility to check that the marks are correct and to notify your section instructor of any errors or missing marks.

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 program or module 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 programs or modules 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 generally 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. Further information will be posted on myCourses and the course webpage. It is not necessary to register for tutorials and the exact times will be posted on the course webpage. The tutorials will give you a chance to ask questions in a smaller environment than lectures.

The tutorials will be covering (approximately)

Tutorial time frame	Title	Contents
Week 1	Basics of Course Software Tools	Scratch Dr. Java and the JDK: creating, loading, editing, saving, compiling, and running programs Binary
Week 1–2	Programming Basics	Variables and primitive data types Expressions and the assignment statement Type conversions Input and output Methods `if` and `if-else` statements `while` and `for` statements Syntactic, run-time, and logical errors
Week 3–4	Control Flow	Methods Using arrays and Strings Case studies
Week 5	Algorithms	Binary Search Sorting Array
Week 6–7	Using classes and objects	Defining your own objects Constructors Static vs non-static
Week 7	Midterm review	
Week 8–10	Advanced topics	ArrayList HTML Recursive programming Data structure
Week 11–12	Final Exam Review	

Approximate Schedule of Topics

The references to chapters in the table below are from the recommended online textbook cited above. 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 change slightly depending on how the semester unfolds and when the midterm is scheduled.**

	Week	Topics	Reference	Events
Fundamentals	1	What is programming? What is computer science? How does a computer work? Scratch programming	Chapter 1, A, B	A1 released
	2	Structure of a Java program Variables and primitive data types Expressions and the assignment statement Primitive type conversions Basic input and output Methods `if` `if-else` statements	Chapter 2, A, B	
	3	Methods and Array	Parts of Chapters 3–5	A1 due A2 released
	4	`while` and `for` statements	Chapter 6	
	5	Case Studies	Parts of Chapters 7 and 10	A2 due A3 released
Object Oriented Programming	6	Simple Algorithms Review	Chapter 8	
	7	Classes and objects Instantiating and using objects Java Standard Library classes		
	8	Building your own classes Instance variables and methods Constructors Overloading methods Static variables and methods	Parts of Chapters 9, 11–12	A3 due A4 released
In depth	9	Reference Types and Aliases Passing Reference Types to Methods Encapsulation Techniques Class Relationships		

Week	Topics	Reference	Events
10	*Using provided libraries* The `ArrayList` class Command line arguments		A4 due A5 released
11	Files and file paths I/O streams Reading from and writing to files Basic exception handling		
12	*Review and Advanced Topics (if time allows)* Such as data structure, recursion programming, HTML, *etc.*		
13	Continue review		A5 due