COMP-202: Foundations of Programming

McGill University, Summer 2004

Time and Location: Tuesdays 1:35-3:25pm at ARTS W-215, and Thursdays 1:35-3:55pm at STBIO N2/2.
Web page: https://mycourses2.mcgill.ca

Instructor: Anil Ada
Email: `aada@cs.mcgill.ca`
Office Hours: To be announced.

Teaching Assistant: Xi Chen
Email: `xi.chen11@mail.mcgill.ca`
Office Hours: To be announced.

Teaching Assistant: Mohammad Ishlam Patoary
Email: `mohammad.patoary@mail.mcgill.ca`
Office Hours: To be announced.

Teaching Assistant: Liana Yepremyan
Email: `liana.yepremyan@mail.mcgill.ca`
Office Hours: To be announced.

Contacting the Instructor 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 are encouraged to answer other students' questions. It is important that you do 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 the 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 and myCourses 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 this, 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

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 Textbooks

The order in which material is presented in course lectures *does not* match any textbook in particular. However, the books below can be used as external resources.

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

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

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

- *Introduction to Programming in Java*, Robert Sedgewick, Kevin Wayne.

  `http://amzn.com/0321498054`

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 5.0 Documentation: `http://download.oracle.com/javase/1.5.0/docs/`
  - Java 6.0 Documentation: `http://download.oracle.com/javase/6/docs/`
  - Java 7.0 Documentation: `http://download.oracle.com/javase/7/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

Your final grade in the course is calculated as follows:

- **Assignments:** 38% (6 in total. Lowest assignment gets 3% weight, the rest are 7% each.)
- **Quizzes:** 6% (4 in total, 1.5% each.)
- **Midterm Examination:** 15%
- **Final Examination:** 36% (If the Final mark is higher than the Midterm mark, the Final will get weight 44% and the Midterm will get weight 7%.)
- **Participation:** 5% (attending classes, asking/answering questions in class and on myCourses.)

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 **6** assignments, each of which will require programming. 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 Policy: You are allowed 2 free (no penalty) late days for the course. You can use these late days whichever way you want (e.g. you can use both of them on a single assignment or you can use them on 2 different assignments). The late days cannot be broken down further, i.e., even if you submit 5 minutes late, it will count as 1 late day. If you are 25 hours late, that will count as 2 late days. After you have used up both of your late days, late submissions will not be accepted and will result in 0% for that assignment. Exceptions to the lateness policy will not be granted without appropriate justification submitted in writing and supported by documentary evidence.

Assignment submission will always take place on myCourses. 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 instructor within **5 days** of the assignment deadline in question to provide evidence of your correct submission.

If you encounter technical difficulties with submitting your assignment on myCourses, try using a different browser than the one you are using. Recommended browsers are Safari, Chrome and Firefox.

Programming assignments are notoriously time-consuming. **Plan appropriately and do not submit to myCourses only minutes before the assignment deadline**.

Quizzes

There will be 4 quizzes, each worth 1.5% of your grade. You will complete each quiz on myCourses. The quizzes are tentatively scheduled for May 16th, May 23rd, June 13th and June 20th. You will have a window of 8 hours (from 9am to 5pm of the day of the quiz) to start and complete the quiz in 45 minutes. Even though you are given 45 minutes to do the quiz, it is not expected to take more than 30 minutes to complete. The purpose of these quizzes is to get you to review the material seen in class and provide you with some example multiple choice questions. Similar questions will appear in the Midterm and Final exams.

You are encouraged to review the material seen in class together with your friend(s). However, you are required to do the quizzes by yourself.

There will be a practice quiz on Friday May 9th. This quiz will not be graded. It will be there for you to check that you can complete a quiz without any technical difficulties. Even though you are not graded for this practice quiz, you are required to complete it (your participation grade will go down otherwise). It will take at most 5 minutes to do so.

Midterm and Final Examinations

The final examination will be held June 26th during lecture time. The midterm examination is tentatively scheduled for May 29th, also during lecture time. Both of these exams will be composed of multiple choice questions. As practice, you will be given some of the exams from previous semesters so that you'll know what to expect in these kinds of exams.

Participation

You are expected to attend every lecture and you are encouraged to participate in lectures and online discussion boards by asking and answering questions. Attendance will not be taken, but it will be evident if you miss a significant portion of the lectures and this will result in a very low participation grade.

Tutorials

There will be about 5 optional tutorials. In these tutorials, one of the TAs will go over a selected topic or go over examples. The tutorials will be held at the Trottier Building. The time, room number and topic of the tutorials will be announced on myCourses.

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. There is usually a consultant around if you have any questions or problems.

In order to enter the Trottier building during weekends or late at night, you will need to scan your McGill ID at the building entrance . 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, 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. 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 5.0 or 6.0 is installed by default on most Mac computers. It is available as a Mac OS software update.
  - GNU/Linux users: JDK 5.0 or 6.0 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*)

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.

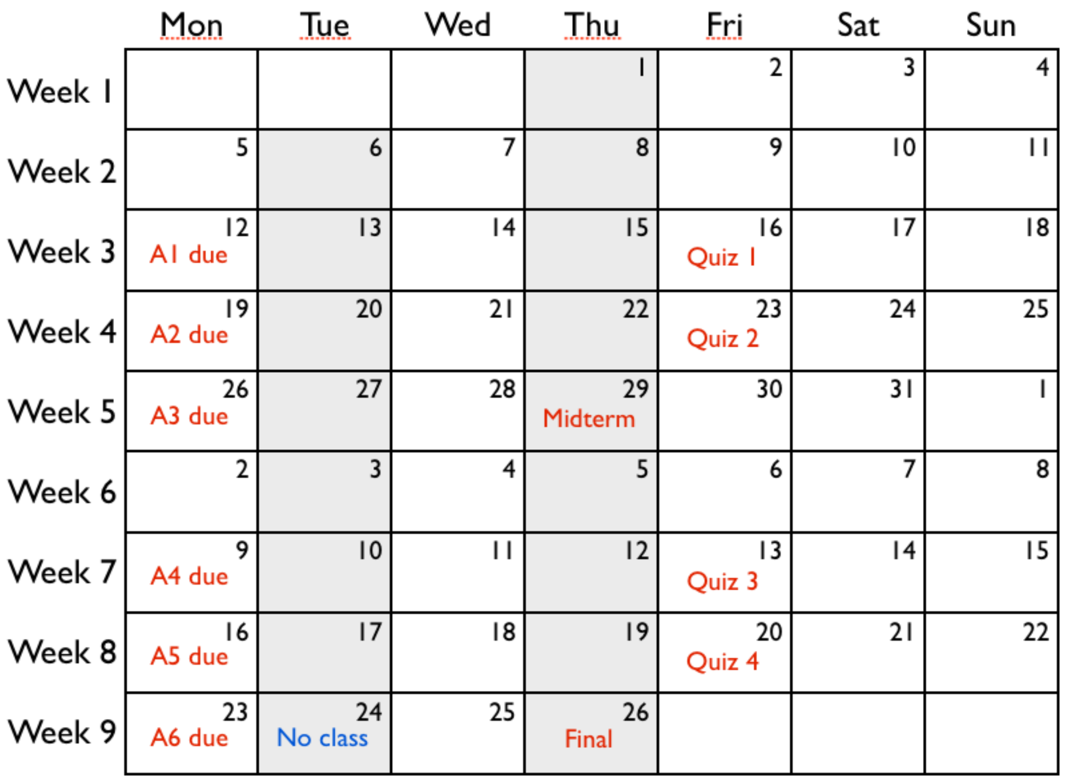
We will be using automated software similarity detection tools to compare your assignment submissions to that of all other students registered in the course, and these tools are very effective at what they have been designed for. However, note that the main use of these tools is to determine which submissions 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.

Calendar with Tentative Dates

The dates below are not finalized except for the date of the Final examination.



	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Week 1				1	2	3	4
Week 2	5	6	7	8	9	10	11
Week 3	12 A1 due	13	14	15	16 Quiz 1	17	18
Week 4	19 A2 due	20	21	22	23 Quiz 2	24	25
Week 5	26 A3 due	27	28	29 Midterm	30	31	1
Week 6	2	3	4	5	6	7	8
Week 7	9 A4 due	10	11	12	13 Quiz 3	14	15
Week 8	16 A5 due	17	18	19	20 Quiz 4	21	22
Week 9	23 A6 due	24 No class	25	26 Final			