COMP 2140 Data Structures and Algorithms

Acknowledgements

Content specialist:	John Braico, M.A. Computer Science Department University of Manitoba
Instructional designer:	Bonnie Luterbach, Ph.D. Distance and Online Education Program University of Manitoba

Course Syllabus

Contacting your instructor

For information on contacting your instructor as well as other important information from your instructor, see the Instructor Letter link in your course website.

top

Course description

The University of Manitoba course calendar describes this course as follows:

> (Formerly 074.214) Introduction to the representation and manipulation of data structures. Topics will include lists, stacks, queues, trees, and graphs.
>
> Not to be held with COMP 2061 or 074.206. Prerequisites: COMP 1020 or COMP 1021 (or 074.102 or 074.123).

The purpose of this course is to reinforce the programming concepts you have learned in earlier courses, and apply them to new types of problems. This will largely involve the use of new types of data structures that go further than the simple "list" used in introductory courses. These structures are specialized for particular kinds of problems. There is also a new set of algorithms that accompany each of these data structures.

There are four important learning materials for this course:

- the notes (offered through this web site), which lay conceptual foundations and provide examples;
- the textbook, which fills in the details;
- your programming environment, where you can apply those techniques yourself; and
- the discussion board, where you can communicate with other students and the instructor in an open forum.

All these elements are necessary for a full understanding of the course concepts. Spend some time browsing through both the web site and the textbook, to familiarize yourself with their layouts. Make sure you carefully read through the rest of this introductory section on the site, which acts as your course outline. Then, you can check out the messages on the discussion board, and join in by introducing yourself.

top

Course goals

Upon completion of the course, you should be able to:

- write programs that use abstract data types (ADTs) to store data;
- recognize how the efficiency of operations can be expressed with big-O notation;
- use and implement Stack and Queue ADTs and their variants, when they are appropriate;
- implement linked data structures, such as linked lists and variants;
- use recursion to solve programming problems;
- describe and implement fast sorting algorithms like quicksort;
- use and implement tree data structures, including binary and multiway (2-3) trees;
- use and implement hashing as an alternative means of organizing data in an array;
- use and implement a heap data structure;
- create and perform some simple operations on graph data structures;
- describe and implement some common algorithms for working with advanced data structures; and
- recognize which data structures are appropriate in particular applications.

top

Course materials

In addition to access to this web site, there are a number of items which you must have available to you as you are working on the course.

Textbook

The **required** textbook for this course is *Data Structures and Algorithms in Java™, Second Edition* by Robert Lafore.

> Title: *Data Structures and Algorithms in Java™, Second Edition*
> Author: *Lafore, Robert*
> ISBN: *0-672-32453-9* © 2003

Note that the textbook will be your primary source of learning material for this course. It is not optional.

top

Computer hardware

You must have access to computer hardware capable of running a modern operating system such as Windows 2000, XP, or Vista. It must also run the required software (see below).

You should also have some kind of removable storage for backing up your work, such as a floppy drive, flash drive, or recordable CD/DVD.

If you are near the University of Manitoba campus, there are computer labs which are accessible to any student registered in university courses. A user ID and password are required for these systems, but they may be obtained when using the computers for the first time. These computers have all of the required software installed.

Computer software

You must have a recent version of Windows installed on your computer.

> Alternative operating systems such as Mac OS X or Linux can be used, but neither Distance Education nor your instructor may be able to help you with technical issues that arise as a result of using an alternative operating system. You must still be able to use all of the other required software.

A web browser is also necessary. Internet Explorer is already installed on your Windows computer, but Firefox, Opera, or any other modern browser capable of viewing this site may be used instead.

> On a Mac, Safari is an excellent browser, but this site unfortunately does not work properly with some versions of it. Consider Firefox as an alternative.

The computer that you use for your assignments must have a text editor such as *Textpad* installed. You will need a recent version of the *Java SDK*, including the Java compiler and virtual machine. You may also wish to install a Java IDE such as *Eclipse*, but that is not required.

> There's an assumption that, by this point, you know how to use some Java programming environment. You can speak to your instructor about issues that arise, but it's your responsibility.

top

Internet access

An Internet connection is required to access the web site, though most parts of it can be downloaded and/or printed for later viewing. If you do so, you should still check the web site on a regular basis, as the information on the site (particularly the discussion group) is updated frequently.

An Internet e-mail address is also required. While any kind of address is acceptable, note that Hotmail addresses are especially unreliable. E-mail between Hotmail and the University of Manitoba is frequently lost. A University of

Manitoba webmail address is highly recommended, and is available to all students.

> You can also use the webmail site to change your password, which is also the password you use to log into the course website.

> There is an e-mail function available through the course website, which is different from regular Internet e-mail, but can be used for communication with your instructor or other students in this course.

top

Distance and Online Education (DE) student resources

Contacting Distance and Online Education

Information on how to contact Distance and Online Education staff by email is located online in each course website in the “Contact DE” link under “Welcome, Start Here” For telephone contact information see the *Distance and Online Education Student Handbook* or the *Distance and Online Education Guide*.

Distance and Online Education Student Handbook

The *Distance and Online Education Student Handbook* is located online in each course website and on the Distance and Online Education website. You can bookmark the site for easy access at your convenience. Accessing both the *Handbook* and the *DE Guide* throughout the year provides you with detailed information regarding the management/administrative aspects of this distance education course. The *Handbook* tells you how to access the following:

- Information on using technology;
- The University of Manitoba Libraries;
- Information on ordering your course materials through the University of Manitoba Bookstore; and
- Information on accessing your grades and submitting assignments online.

Distance and Online Education Guide

The *Distance and Online Education Guide* is located online on the Distance and Online Education website. It contains important information on programs and courses offered through DE.

Course overview

The course material is divided into ten units, where every unit should be completed in approximately a week. The first unit is slightly longer, but the material is review and should therefore be easier. There is also a week scheduled for the midterm exam, including preparation and writing, though its date is subject to change based on the current term's calendar.

Each unit consists of a series of learning activities, including reading from the notes and textbook, programming examples, and completion of work for evaluation. You will have to schedule these units to complete the assigned work before certain due dates. A sample schedule is given in the course *Schedule* that can be found on this site.

> Work must be completed and submitted by the due date. Since some assignments take quite a bit of time, you should plan on starting each assignment at least two weeks before its scheduled due date. It is also acceptable to work on assignments even earlier (if they are available) and to submit your solution as early as you like; note that you should still check the discussion board to see if any changes or clarifications posted there could affect your work.

The following sections discuss how you use these materials in greater detail. Please read them carefully before you begin the course.

The ten units of the course cover the following topics:

1. **Review of programming concepts**
   A refresher on some of the key topics from previous courses, including arrays and simple sorting algorithms. Introducing ADTs.
2. **Stacks and queues**
   Simple data structures for particular applications.
3. **Linked lists**
   Creating linked data structures for improved performance with some operations.
4. **Recursion**
   Incorporating self-reference, how it works, and how it can be used.
5. **Advanced sorting**
   More complicated, but more efficient, sorting algorithms.
6. **Binary trees**
   Creating and manipulating unbalanced binary trees, including binary search trees.
7. **Balanced trees**
   Keeping trees balanced, including the use of multiway trees.
8. **Hash tables and hash functions**
   Indexing data by applying functions to key values.
9. **Heaps**
   The heap ADT and the heapsort algorithm.
10. **Graphs**
    Connected graphs, graph traversal, and weighted graph algorithms.

top

Course notes

Part of the required work for each unit will be to read the course notes. They include course material independent of the textbook. Some topics may be covered in both sources, some in only one or the other. You are responsible for learning *all* of the material in the course, whether it is from the notes or the textbook.

Each unit of the notes begins with a brief introduction, followed by a list of learning activities for the unit. The activities will include readings and hands-on work. Even though only some activities (the exercises and the assignments) are submitted for evaluation, you can assume that the other activities are required in order to prepare you for that work. It is strongly recommended that you complete all the activities in the order they are listed.

Included in the notes are *Walkthrough* activities, which guide you through the steps of solving a programming problem, with some additional notes along the way. The solution they present may not be the only way of solving the problem — you might come up with a better way of doing it — but they are designed to help you think of what to try next when you are not sure. They are designed to be worked through, not just read; consider going through them while in front of the computer, and entering (or at least copying and pasting) and running the code in your programming environment.

To make the notes easier to follow, certain typographical conventions have been used:

- ***Definitions*** — words appearing in this format are terms defined in the course glossary (they only appear in this format the first time they are used).
- *Citations* — the names of portions of the textbook or this web site.
- `Code` — brief examples, keywords, or identifiers in the Java programming language.
- *`Output`* — represents things you may see on your screen.
- `Filenames` — the names (or portions of names) of files saved on a computer.

Longer portions of sample Java programs, or complete sample programs, are shown as follows:

```
public class Greetings {
  public static void main(String[] args) {
    // Say hello to everybody out there
    System.out.println("Hello!");

    // The following means that some part of the
    // program has been omitted
    // (it's not relevant to the example):
    // ...

    System.out.println("Goodbye.");
  }
}
```

Some examples and problems are set apart from the rest of the text. These may be examples of problems that will be solved in the notes, or complete problems that you might try on your own:

> Problem: Write a program that will answer the ultimate question of life, the universe, and everything.

At the end of a particularly in-depth or complicated discussion, a highlight box summarizes the main point:

> If you are printing these notes, you may want to consider investing in a colour printer.

Finally, some interesting or relevant facts which do not quite fit with the rest of the text are given as asides:

> You can learn a lot from these asides. Sometimes, they tell you about the requirements of the course, while other times, they point out exceptions to what you've just learned; on some occasions, they just tell a story.

top

Textbook readings

Readings will be assigned from the required textbook. It is recommended that you follow along with the examples by downloading and running the example code as you read them. Make changes, and add output statements, to trace its behaviour. You should be reading the code in the textbook, not to memorize it, but to understand it.

Assigned work

The assigned work in this course is designed to give you an opportunity to apply the techniques you have learned from your readings. Some problems will be small, intended to illustrate a particular concept, while some will be larger projects that incorporate a number of different ideas into a more complete program.

More details about assigned work can be found in the *Evaluation and grading* section of this introduction.

top

Communication

To help you achieve the goals of this course, there are a number of resources available.

The first is the discussion board, where important announcements will be posted, including information about the assignments and exam, and any changes that occur mid-term. You are required to read all of these announcements; you will find they answer many of your questions.

> Some — though not all — of these announcements may be posted on the course home page as well. You should still check the discussion board for more details.

The second is direct communication with your instructor, either through e-mail or telephone. Your instructor can answer any of your questions about the material, provide help or hints on the assignments, or let you know how you are doing in the course. You should assume that your instructor will make reasonable effort to respond to your questions, but without in-person contact, you can expect there will be a delay between having a question and getting it answered. The effect is much less dramatic if you build it into your expectations: start your work early, and you will have plenty of time to ask questions and get feedback.

> See the instructor's letter for more information about contacting your instructor. If you are on campus, your instructor may have office hours for in-person contact as well.

> One thing your instructor may not be able to do is answer technical

questions, either about the computer system you are using for the assignments, or issues with the web site. However, it doesn't hurt to ask.

You can also look for help from your fellow classmates. The discussion board is open to communication from everyone in the class, you can post your own questions and respond to others. In a large class, with good participation, you may be able to get a very fast response to your postings.

The discussion board is as useful as you make it. Everyone who participates contributes to the common good.

The one restriction with the discussion board is that you cannot post any complete or partial answers to the assignments. Hints are OK, though if you aren't sure if your message gives a hint or an answer, it's better to ask your instructor before posting it.

top

Evaluation and grading

Your mark in this course is based on a number of activities which are discussed in greater detail here.

Assignments

There are five assignments in this course, all of which contribute to your final mark. Assignments may consist of one or more questions, which are usually programming questions, but may also include a written-answer portion. Your solutions to programming problems must be submitted as described in the assignment, as complete, compilable, and runnable Java source code. Make certain all necessary parts of your code are included in your submission.

The assignments are equally-weighted in their contribution to your final mark. Each assignment will be marked according to a guide, which will list the specific items that were examined in the marking. It usually includes sections for the overall *Programming*, the following of source code *Standards*, the *Design* of the program (including its efficiency and the appropriate and correct use of data structures and algorithms) and the *Output*. The guide may not have mark deductions for all flaws: a perfect mark does not mean a perfect program. It is your responsibility to check feedback about your marked assignment to see what was missing.

Each assignment must be completed by its due date. Late assignments may not be accepted.

top

Exercises

Between the assignments there are five exercises. Exercises are activities that are much shorter than an assignment, used to reinforce a newly-learned concept or technique. Exercises will generally consist of writing or altering a segment of a program (classes and/or methods) rather than creating a complete runnable program. Your answers should follow the programming standards that relate to formatting code, and writing appropriate code, but do not need to have extensive comments.

Exercises are evaluated based on effort, rather than completion. The solutions you submit should make it clear that you have put thought and effort into the problem. They are not evaluated for correctness.

> Exercises fill the role that labs would in an on-campus course, and are evaluated according to the same standards. You should plan to spend an hour on each.

> As with assignments, exercises must be submitted by their due date.

top

Mid-term test

There will be a single midterm test in the course, scheduled by your instructor. It will come in one of two forms, at the instructor's discretion. It may be a **timed 70 minute online** test that must be written within a prescribed set of hours on the scheduled day. Alternatively, it may be in the form of a **take-home test**, where the questions are released, you may work on the test on your own time, and submit it to the instructor by a particular date and time.

Details about the midterm, including scheduling, format, and material covered, will be provided by the instructor prior to the test.

Final exam

The final exam will be written at the University of Manitoba (UM), Fort Garry campus or at an approved off-campus location. **Students needing to write at an off-campus location must declare a location by the specified deadline date** (see off-campus declaration and policy under Student Resources on course homepage). **Students writing at the UM Fort Garry campus do not need to declare an exam location.**

The Registrar's Office is responsible for the final exam schedule which is available approximately one month after the start of the course.

The exam is written, on paper, without access to a computer. It is closed-book, and no notes or other materials are allowed. The questions are a combination of written answers (possibly including multiple choice or other question formats) and programming questions. Your instructor will provide you with more details about your exam.

The final exam will be cumulative. That is, it will cover all of the material in the course. There may be a greater emphasis on material learned after the midterm.

> The above information about the exam is subject to change; your instructor will notify you early in the term if the exam will be given in a different form.

top

Distribution of marks

Your final mark will be calculated according to the following formula.

Item	Percentage

Assignments (all of 1 through 5, each worth 5%)	25%
Exercises (all of 1 through 5, each worth 1%)	5%
Mid-term test	15%
Final exam	55%
Total	100%

Assignment guidelines

All work submitted must follow these guidelines:

1. **IMPORTANT**: A *Student Honesty Declaration* must be completed before your assignments or exercises will be graded. See the assignments page for more details about the declaration.
2. **Late assignments will not normally be accepted.** Extensions will be granted only in exceptional circumstances. Contact your instructor about any conflicts well in advance of the due date.
3. Information about changes related to assignments and the course will be posted to the course home page and the course discussion boards. You should check this site often (at least twice a week). It is **your responsibility** to become aware of any such changes by checking the site on a regular basis.
4. The document *Programming Standards for COMP 2140* is required reading. These programming standards must be followed in all assignments submitted for this course.

top

Academic honesty

In this course, you are required to do all of the work on all of the assignments independently. All work that you submit for evaluation must be entirely your own. At no point should you share your work, in whole or in part, with any other students in the class.

Violations of this rule are treated very seriously, and have significant consequences. Also note that in the case of violations, both the person who originally did the work and the person who copied that work are subject to the same penalties.

The single exception to this rule is the code provided for you on the course web site, and in the textbook. You can use the programs you are given, including classes and methods, in whole or in part, in your own work. If you do so, you should appropriately credit the source (indicate in comments where the code originated).

> This exception **only** applies to code in the textbook and on *this* web site, not on any other web site, even if it is linked from this one.

All assignments in Computer Science must be accompanied by a signed Honesty Declaration that states that you are aware of the regulations regarding academic honesty. In a distance course, this Honesty Declaration must be signed and submitted before you complete your first assignment; it applies to all of your work in the course. See the *Assignments* page for more details.

The Honesty Declaration is a contract that binds you explicitly to the rules of academic honesty. Assignment marks will be withheld until a completed and signed declaration is received.

The official description of academic dishonesty follows:

> Academic dishonesty is a very serious offence and will be dealt with in accordance with the University's discipline bylaw. Examples of academic dishonesty include:
>
> - submitting assignments which are not entirely your own work;
> - letting others copy work that you created;
> - use of unauthorized material during a test or examination;
> - writing an examination for another person; or
> - having another person write an examination for you.
>
> Please see section 7 of the *General Academic Regulations and Requirements* in the University of Manitoba general calendar for details.

top

COMP 2140 Programming Standards

This document lists the programming standards that you must follow for the programming questions of your assignments. Failure to follow these standards will result in the loss of marks. Some of these are the same as earlier courses, but you should re-read them to be sure!

Commenting files and methods

1. The source code file that contains your main program must begin with a comment block like the following:

```
/**
 * Name of class or program (matches filename)
 *
 * COMP 2140 SECTION D01
 * ASSIGNMENT    Assignment #, question #
 * @author       your name, your student number
 * @version      date of completion
 *
 * PURPOSE: what is the purpose of your program?
 */
```

   Every other source code file must contain an abbreviated version of this comment:

```
/**
 * Name of class or file (matches filename)
 *
```

```
 * @author  your name, your student number
 *
 * PURPOSE: what is the purpose of this class?
 */
```

2. Every method must begin with a prologue comment, similar to the following:

```
/**
 * Describe the purpose of the method. Clearly.
 *
 * PARAMETERS:
 *    firstParam  description of what firstParam
 *                means; if it is a reference, and
 *                its value is modified, discuss
 *    secondParam description of secondParam (as
 *                above)
 *
 * RETURNS:
 *    what is the meaning of the return value?
 */
```

or:

```
/**
 * Describe the purpose of the method. Clearly.
 *
 * @param firstParam  description of what
 *                    firstParameter means;
 *                    if it is a reference,
 *                    and its value is
 *                    modified, discuss
 * @param secondParam (as above)
 *
 * @return            what is the meaning of
 *                    the return value?
 */
```

Parts of the prologue may be omitted if appropriate, e.g. if the method has no parameters. Note that while the use of Javadoc-format comments is not required, it is recommended in Java.

top

Writing readable code

3. Group the contents of your class consistently. Always place data before behaviours. The following order is recommended:

```
class Something {
   // Instance variables
   // Static variables
```

```
   // Class-level constants
   // Constructors
   // Other instance methods
   // Destructors
   // Static methods: main comes first
}
```

4. Use blank lines to separate blocks of code and declarations to improve readability. In particular, use blank lines between declarations and other code, and between methods.
5. Comment blocks of code. Describe why you wrote the code this way, not what each line does. For example:

```
// Summarizes taxes in each category
// (only federal so far)
System.out.println("The taxes payable are:");
System.out.print("Federal tax = ");
System.out.println(federalTax);
```

6. Use meaningful but reasonable variable names. Start in lower case and capitalize only the first letter of each new word.
   - `a` -- Very bad. Too short, not meaningful.
   - `averageMark` -- Good
   - `average_of_all_the_marks_in_the_list` -- Bad. Too long and wordy.
7. Additional documentation of variables should appear in a comment following the variable declaration (a variable dictionary) where appropriate. Use this to express details that are not already included in the variable name (like units or ranges), or to indicate the meaning of initial values. For example:

```
SomeMethod () {
  double maxFlow;           // in river [m3/s]
  int lastMark, averageMark; // marks out of 100
  int swapCount = 0;
              // 0 swaps needed if data is sorted
  int studentNumber, numStudents;
  ...
}
```

8. Use indentation to clarify control structures (e.g., for loops and if constructs). Align **else** with the corresponding **if** for readability. Avoid long lines; where needed, continuations of a statement on a new line should be indented too. Any readable and consistent style is acceptable. The essential features are that all statements that are nested within another statement must be indented, and that the braces { } must be in predictable and consistent positions. Common styles include:

```
if-while-else-etc {
   stuff-inside
```

```
   stuff-inside
}

if-while-else-etc
{
   stuff-inside
   stuff-inside
}
```

---

top

Writing maintainable and extendable code

9. Avoid the use of literal constants ("magic numbers") in your program. Generally use constant (final) identifiers rather than literal constants in your program. Acceptable exceptions are strings that appear only once in output titles and headings, and small fundamental values. For example:
   - `sum = 0;` -- Literal constant 0 is OK
   - `count = count + 1;` -- Literal constant 1 is OK
   - `price = total * 1.07;` -- Not proper. Create a named constant like `PST_RATE = 1.07`.
   - `lastDigit = accountNumber % 10;` -- Literal constant 10 is OK.
   - `if(codeLetter == 'R')` -- Not proper. Create a named constant like `REFUND_CODE = 'R'`.
10. Avoid code duplication. Reuse! Make classes general and in hierarchies where possible, and use polymorphism to reduce duplication.
11. Declare variables with as small a scope as possible. If something is used only with a loop for example, declare and use it there.
12. Values of type float or double should not normally be compared with == or !=, nor should Java Strings!
13. Never change the value of a **for** loop variable inside the loop.
14. Use the best possible construct. For loops are only used when the number of iterations is known in advance; use a while or do/while for non-deterministic loops.

top

Input and Output

15. Validate your input. Do not assume that values will be in range. In Java, use appropriate exception handling.
16. Use prompts when asking for interactive input.
17. All output produced on the console must have appropriate titles and headings. Output that is sent to a file typically does not require titles or headings.
18. Print console output neatly. Use tabular output where appropriate.
19. Print a message on the console at the end of the program that indicates whether or not the program completed successfully.

Object Orientation

20. Each class should have a single purpose. Do not place, for example, multiple

data members in a Node class. Combine the data into a separate class, and make the Node refer to an object of that class.

21. Follow proper object-oriented programming techniques. Make all instance variables private, use class variables only when appropriate, write appropriate methods (rather than arbitrary accessors or mutators).

Consequentially...

22. **Not following the programming standards and the assignment standards will result in a loss of substantial marks. Egregious violations may result in your assignment being rejected entirely.**

top