ALGONQUIN COLLEGE

MODULE 00: COURSE OUTLINE, REQUIREMENTS, AND RESOURCES

Professor : Dave Houtman

Office: T323

Office Hrs: Wednesday 14:15 – 15:30

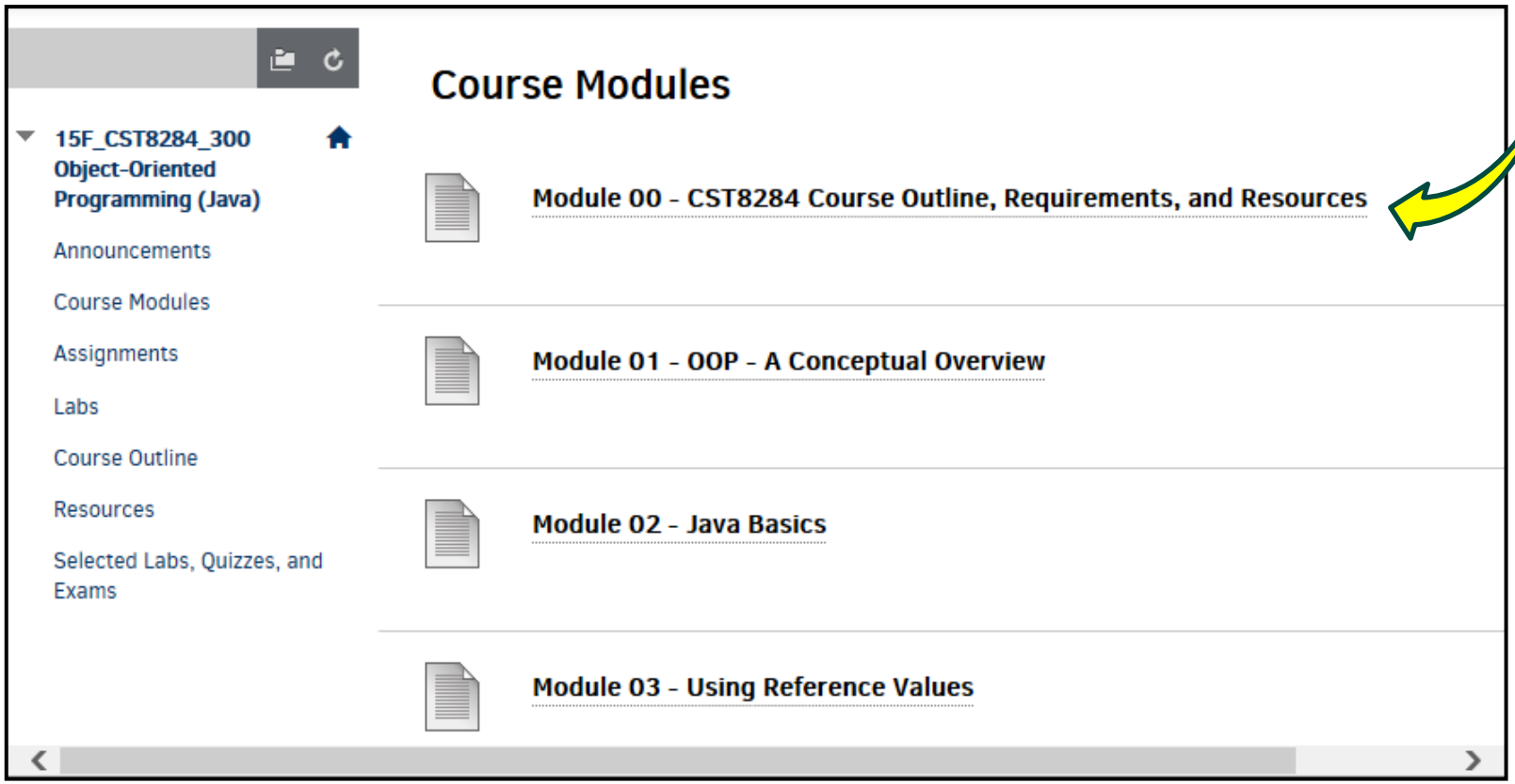
Wednesday (after lecture*)

* confirm beforehand

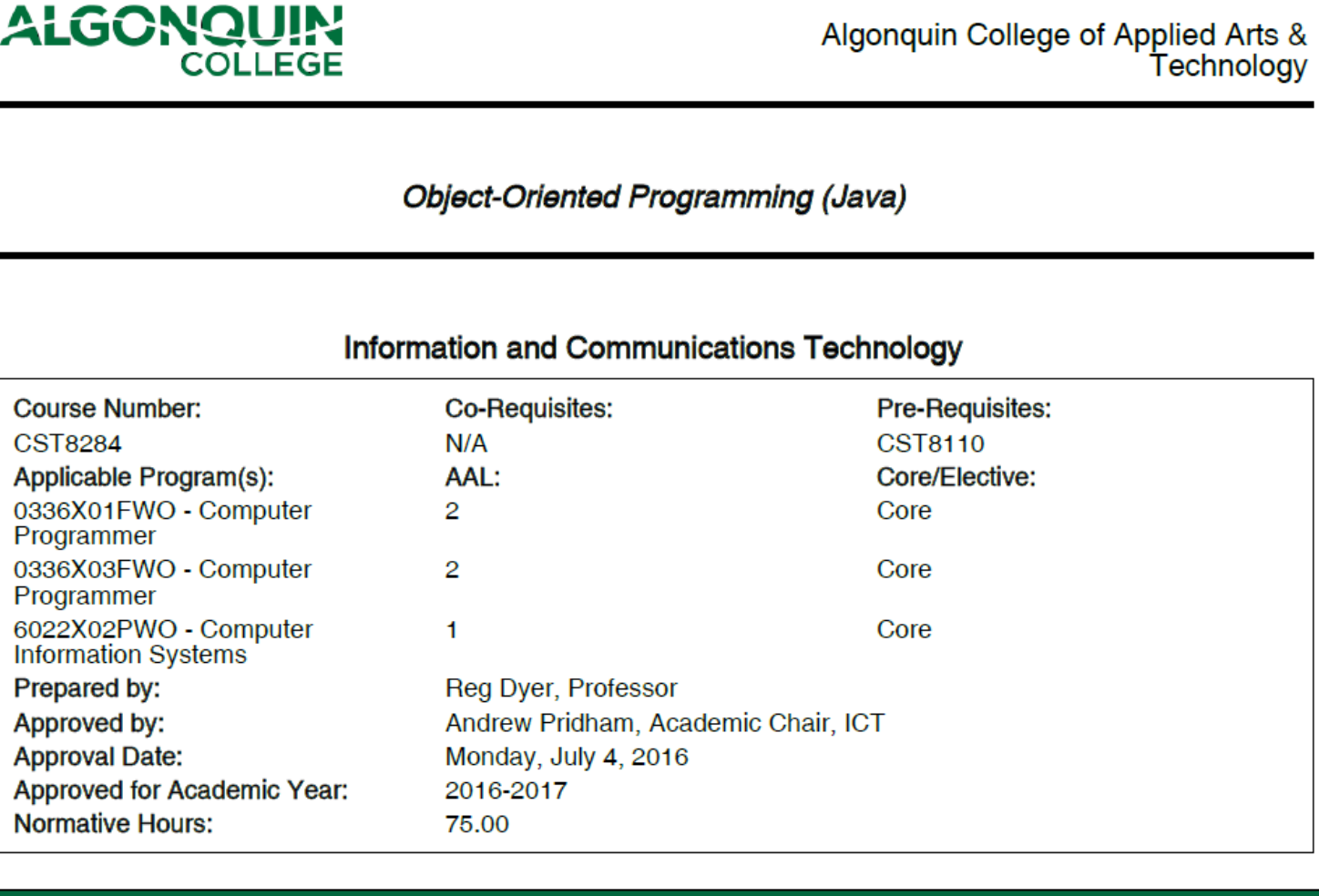
Email: houtmad@algonquincollege.com

Course Lecture Notes

- All course lecture notes, including these slides, along with your grades, labs, assignments, course outline, etc. for CST8284 will be posted on Blackboard



Course Outline



ALGONQUIN COLLEGE

Algonquin College of Applied Arts & Technology

Object-Oriented Programming (Java)

Information and Communications Technology

Course Number:	Co-Requisites:	Pre-Requisites:
CST8284	N/A	CST8110
Applicable Program(s):	**AAL:**	**Core/Elective:**
0336X01FWO - Computer Programmer	2	Core
0336X03FWO - Computer Programmer	2	Core
6022X02PWO - Computer Information Systems	1	Core
Prepared by:	Reg Dyer, Professor	
Approved by:	Andrew Pridham, Academic Chair, ICT	
Approval Date:	Monday, July 4, 2016	
Approved for Academic Year:	2016-2017	
Normative Hours:	75.00	

Course Outline

Course Description

Learn object-oriented programming methodology using the Java programming language. Object-oriented concepts, such as encapsulation, inheritance, abstraction and polymorphism are covered and reinforced with practical applications.

Course Outline

Course Learning Requirements/Embedded Knowledge and Skills

When you have earned credit for this course, you will have demonstrated the ability to:

1.) Install and use the Java Development and Runtime Environment and documentation libraries. Install and use the Eclipse Integrated Development Environment.

Compile and execute java programs consisting of multiple classes in the Eclipse IDE.

2.) Write Java program code to solve a problem, based on a description of the problem context, including UML diagrams, using object-oriented techniques.

Identify objects that are part of the problem domain. Decompose complex objects into component parts and identify the *composition* relationships between those components.

Identify object attributes (which ultimately become *instance variables* that maintain state). Identify object behaviours (which ultimately become *methods* that can be invoked to implement the behaviours).

Define and document the user interface based on the processing requirements.

Create *classes* with *instance variables* and the following categories of methods: *constructor*, *accessor*, *mutator*, *service*, etc. *Instance variables* and *methods* must follow the prescribed naming conventions based on UML class diagram specifications.

Implement non-polymorphic behaviors with *Object* methods.

Implement inheritance hierarchy and polymorphism where derived classes implement specialized behavior using *polymorphic methods*.

Produce appropriate outputs, including: reports, screens, messages, files.

Course Outline

Implement Graphical User Interface (GUI) using JavaFX:

- *Stage / Scene*
- *Node* classes: *Text*, *TextField*, *ListView*, *TableView*, *Button*, *MenuBar*, *Menu*, *MenuItem*,and layout managers such as *HBox*, *VBox*.
- *Transition* classes: TranslateTransition, *FadeTransition*, *ScaleTransition*
- *Effect* classes: *DropShadow*

Write java code which includes the exception handling mechanism, and the use of *throws*, *throw*, *try*, *try-with-resources catch*, *finally;* code that deals with both checked and unchecked excpetions; code that extends base exception classes for custom expections; code that deals with both checked and unchecked exceptions.

3.) Explain and use basic data structures.

Implement arrays of primitive types.

Implement arrays of references to Java objects.

Explain how *Interface* specifications (e.g. *Collection*, *List*, *Set*, etc.) are used to support alternate concrete implementations (e.g. *ArrayList*, *LinkedList*, *TreeSet*, *HashSet*, etc.). Implement Java code to demonstrate this capability.

Implement and manage lists of objects using Java's *ArrayList*.

Explain the core Big-O values (e.g. O(1), O(log *n*), O(*n*), etc.) and their relationship to selected operations on selected concrete *Collection* classes.

Course Outline

4.) Use existing simple Generic classes to manage objects.

Explain the underlying nature of object management and the role of the compiler in handling generic syntax.

Use selected Collection classes.
Use generic Interface reference-to variables to manage concrete generic Collection objects.

5.) Implement program Input/Output operations.

Explain the difference between *binary*, *text*, and *object* streams.

Implement *Serializable* interface to support file I/O (*object* stream). Implement code to support *text* streams.

6.) Produce tested code that executes correctly and consistently. Testing procedures will involve the use of: valid data only, invalid data only, and a combination both valid and invalid data.

Perform iterativetesting to locate and eliminate logic problems until results match expectations ennunciated in the test plan.

Develop a comprehensive test plan with the associated test data.

Ensure that all user messages and help messages correctly reflect program state and expected user actions.

Install and use the JUnit testing framework to implement test classes.

Use jUnit *assertions* to test the *pre-conditions* and *post-conditions* that have been expressed in the Test Plan.

Course Outline

7.) Prepare program documentation using prescribed program specifications.

Use JavaDoc tagging to generate formatted documentation conforming to Oracle Java coding standards.

Create basic UML static class diagrams to represent classes and their structural relationships.

8.) Debug program problems using manual methods and computerized tools in an appropriate manner.

Trace program executation through the debugger, using *breakpoints*, *step-in*, *step-over*. Inspect variables, being able to follow the *reference-object* relationships at run-time.

Describe the differences between the runtime allocation of *stack*-oriented and *heap*-oriented variables.

9.) Identify appropriate strategies for solving a problem.

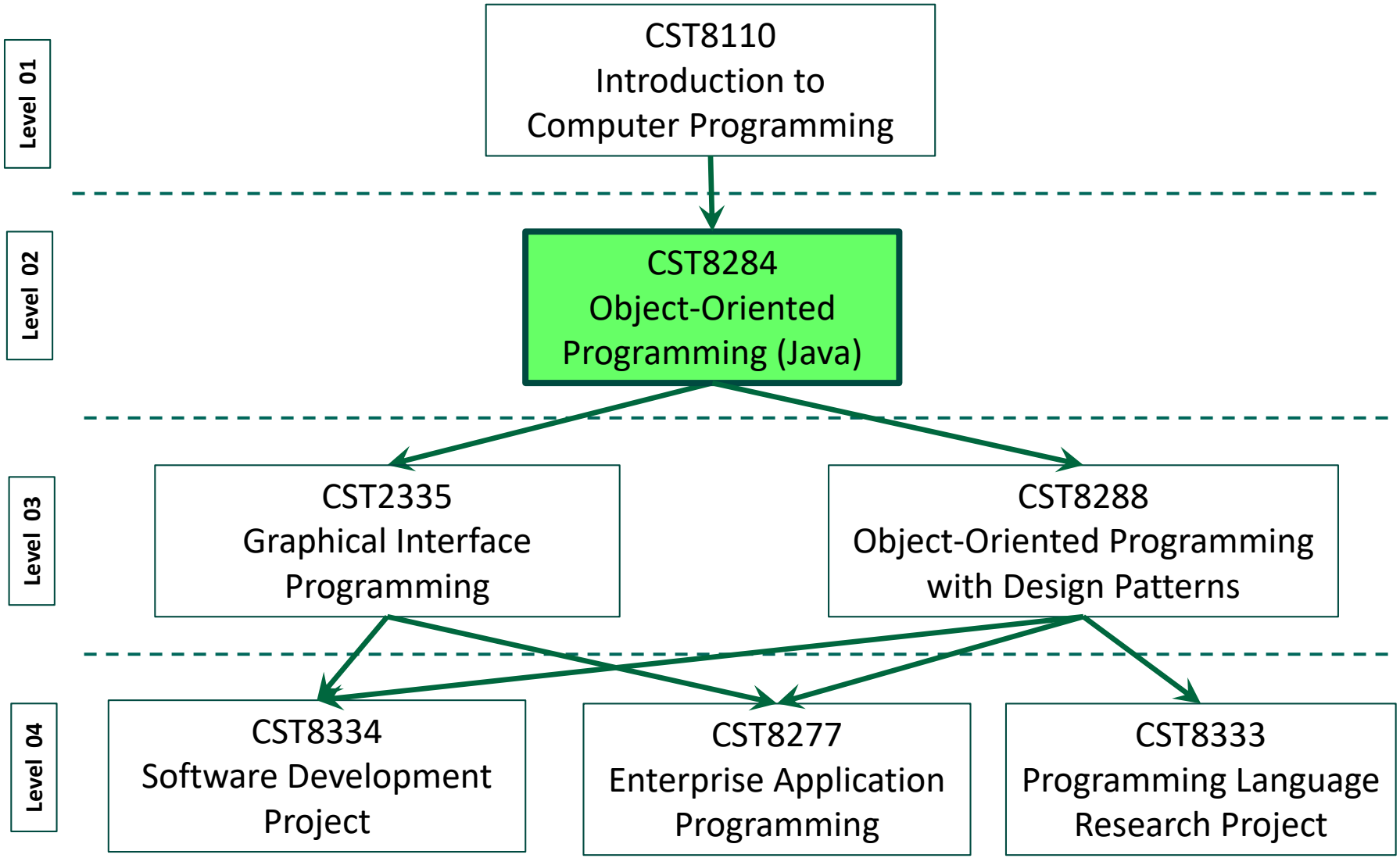
Define a problem; identify sources of help; identify various strategies for problem solving; use technical reference manuals and /or on-line help where applicable.

Course Outline

Note:

1) The focus of the course is on fundamental concepts in object-oriented programming. While JavaFX is part of the course and makes up a good portion of the examples, it is secondary to the OOP concepts that comprise the core of the course.
2) While we generally avoid the more technical details *underlying* programming concepts, it is sometimes necessary to delve into this information in order to make the business of programming more easier in the long run. Programmers who only know *how to write programs*, but don't understand how the underlying mechanisms of Java work, are less likely to get hired, because (1) they don't survive the job interview, which tests their knowledge of how code works, and (2) they make the same fundamental mistakes over and over again, and waste their employer's time. Some understanding of the inner workings of Java *is essential*. Such topics include:
    a. Stack-oriented vs. Heap-oriented variables
    b. Big-O notation
    c. The use of reference values as 'pointers' to objects in Java
    d. The reason why casting works the way it does in polymorphic assignments

How This Course Relates to Your Other Courses



Required Resources

Learning Resources

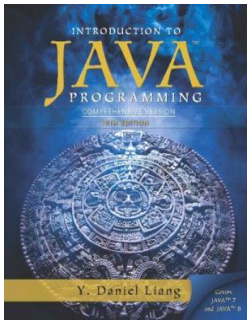
Required Textbook: (also used in pre-requisite course): *Java How to Program, 10th Edition, by Deitel and Deitel, Published by Pearson Education Inc. in 2015 ISBN: 987-0-13-380780-6*

Online Resources

- **Java API (online)**: http://docs.oracle.com/javase/8/docs/api/
- **JavaFX API (online)**: https://docs.oracle.com/javase/8/javafx/api/toc.htm
- **Tutorials**: http://docs.oracle.com/javase/tutorial/
- **API Documentation**: http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html
- **Samples**: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Required Hardware: This course is part of the mobile (laptop) program initiative at Algonquin College. Students are required to have a functioning laptop at all lecture and lab classes. The specifications for the required laptop and additional information about the mobile program initiative can be found at http://www.algonquincollege.com/mlearning.

Required Resources



Note that while the text by Deitel & Deitel is still used, it is slowly being displaced by the text *Introduction to Java Programming*, 10th ed. by Daniel Liang, ISBN: 9780133813432, shown at right.

This text, which is listed as the second textbook for CST8110, is generally considered superior to Deitel & Deitel, and is also free for download to registered College students. It is a much more readable, more thorough text, than D & D, and it has three chapters on JavaFX (D & D has none). Therefore, you are strongly advised to download this text, as you will no doubt find it useful during the course.

Required Resources

Computer Requirements:

- This program is a *mandatory* laptop program (no exceptions) – see the specifications at http://mlearning.algonquincollege.com
- Use Non-Windows-based laptops and tablets at your own risk (you are responsible for solving any problem which results as a consequence of this choice of PC)

Software Requirements:

- Eclipse Neon 4.6.x or higher
- Java 8 SE, Latest version (Update 111 as of this writing)
- JDK 8u111 (a.k.a 1.8.0_111)
- Javadoc version 1.8.0_111 (should be installed with JDK)
- JUnit 4.x

Installed during Lab 1

Teaching / Learning Methods

Lectures

- 2 hours in class per week + 1 hour hybrid online
- Attendance will be taken each class

Labs

- 2 hours per week in lab

Notes:

1) Labs and assignments will be posted online in advance.

2) If you have your lab done before your actual lab period, you still need to stop in to the lab, sign the attendance sheet, and get signed off on the lab. This may only take 5 minutes, but you still need to put in an appearance. In weeks when you've already finished the lab and there's nothing to do, you don't need to drop by to sign attendance.

3) Permanent changes to your CST8284 lab schedule must be cleared with your program coordinator. So if you wish to change your lab section, talk to the program coordinator, Reg Dyer, not the lab/lecture professor.

Teaching / Learning Methods

Notes (con't):

4) Labs are scheduled at the following times throughout the week:

If your lecture section is	and your lab section is	room #	time
300 (Wednesday 4:00-6:00)	301	T317	F: 10:00 – 12:00
Lab prof: Dave Houtman	302	T111	F: 13:00 – 15:00
houtmad@algonquincollege.com	303	T111	F: 15:00 – 17:00
310 (Friday 8:00-10:00)	311	P213B	W: 16:00 – 18:00
Lab prof: Raymond Peterkin	312	T304	Th: 17:00 – 19:00
peterkr@algonquincollege.com	313	B421	M: 17:00 – 19:00

Teaching / Learning Methods

Notes (con't):

4) (con't)

If your lab is on Friday only (highlighted in yellow above), then regardless of your scheduled lab time, you are free to drop by at any of the above-listed times on Fridayt to have your lab assessed, with the following qualifications:

1. Students who are registered for a particular lab section get first priority; students who are 'guests' (i.e. dropping by outside of their normal lab time) will be dealt with once the registered students have been signed off;
2. I can override rule #1 under special circumstance, e.g. I can assign priority to students who are under deadline to hand in a lab that day, regardless of which lab section they are registered in (but see rule #4);
3. 'Guests' who drop by during especially busy times may have to wait until the next available lab section to get signed off;
4. Students should not abuse this offer by arriving too late on Friday afternoon to be properly assessed. Such students may receive a '0'.

Teaching / Learning Methods

Notes (con't):

5) Hybrid time is considered to be an extension of lecture time. Often (too often) students are told to watch videos or read material out of the textbook—which is usually a rehash of material already covered in class—as part of their hybrid activity time. And this is often taken as a suggestion only, which many students feel free to ignore.

   In this course, hybrid time is used to cover material for which there is not enough class time available. You are responsible for keeping up with the hybrid material, which will generally be presented in the form of video lectures. Note that anything presented in a hybrid is fair game for an exam question. And since 1/3 of weekly lecture material is presented *only* in hybrids, you can expect that ~1/3 of all exam material will be introduced in the hybrids.

   Additionally, some of the lab material will appear only in hybrids. So if you encounter something you haven't seen before in a class or a lab, that may be because you haven't kept up with the hybrid material.

Course Evaluation

Theory	**65%**
• Quiz – Feb 15 (Sec. 300), Feb 17 (Sec. 310)	**10%**
• Midterm – March 15 (Sec. 300), March 17 (Sec. 310)	**20%**
• Final Exam	**35%**
Practical	**35%**
• Assignments (3 expected)	**19%**
• Labs (8 total)	**16%**

Note that you must pass the theory portion of this course (i.e. 32.5%/65.0%) for the practical mark to be included in the final grade.

You must pass the practical part of the course (17.5%/35%) as well.

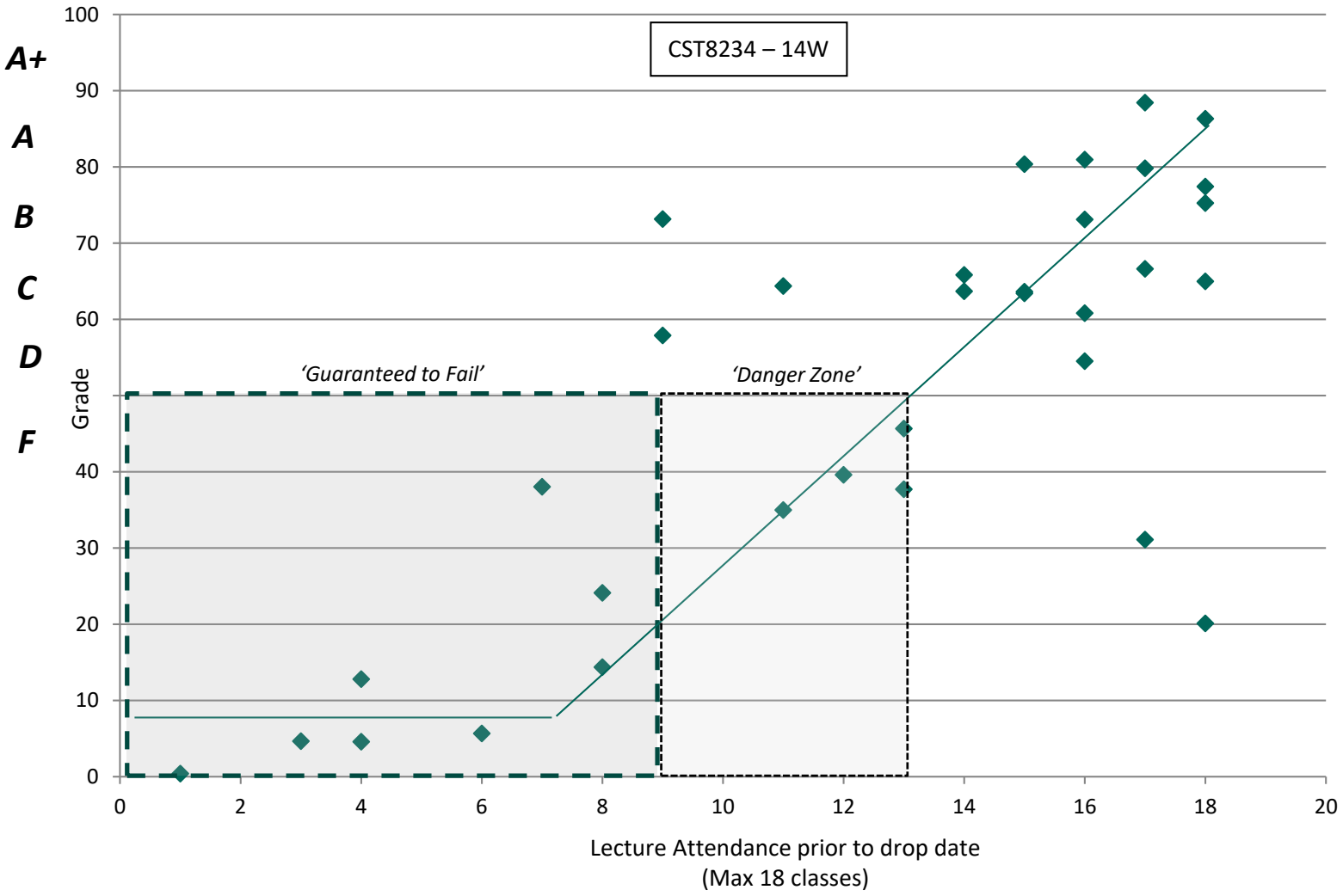
ALGONQUIN COLLEGE

Getting the Most Out of This Course

Many of the marks lost by students during the semester disappear not due to a failure to comprehend course material (resulting in poor marks on exams and assignments), but due to simple negligence. Marks creep away a few percent at a time, and the A- or B+ that a student might have had otherwise, if properly attending to the details of the course—including attending lectures and labs—quickly slips away, until by final exam time the student is scrambling to obtain a passing grade. To avoid this situation:

1. Stay on top of your homework; don't procrastinate and put off deadlines to the last minute;
2. Anticipate the busy times of the semester and try to get ahead of schedule beforehand, whenever possible;
3. Practice good time management skills. Plan your day *at the start of it* by setting realistic goals to achieve by the end of it;
4. When you have some free time, it's tempting to want to go out and party with friends, or enjoy some downtime; but a good night's sleep is more important;
5. **Don't miss labs. Don't miss lectures**—this is where good grades slip away;
6. Students who attend lectures but sit in the back playing computer games typically lose (at least) a half a letter grade as a result. Students who don't bother attending lectures typically lose an entire letter grade. (So a potential B+ becomes a C+.)

How Attendance Affects Your Grade



Student's Academic Responsibilities

Each student is responsible for:

- ***Knowing the due dates*** for assignments and exams.
- ***Attending*** all classes. Anything covered in class, whether in the slides or covered verbally in lectures or hybrids, is fair game for tests, quizzes, etc.
- ***Completing course work on time.***
- ***Maintaining*** a folder of all work done in the course during the semester for validation claims in cases of disagreement with faculty. This means…
- ***Keeping both paper and electronic copies of all assignments***, marked and unmarked, in case papers are lost or go missing;
- ***Regularly checking*** your Algonquin e-mail account for important announcements from both professors and college administration.
- ***Staying on top of the course material each week*** throughout the semester. During those weeks when there's no midterm exam, there's always a quiz, so don't wait until exam time to start studying.

Plagiarism

The School of Advanced Technology's Standard Operating Procedure on Plagiarism and Academic Honesty defines plagiarism as *any attempt to use or pass off as one's own idea or product, work of another without giving credit.* Plagiarism has occurred in instances where a student either directly copies another person's work without acknowledgement; or, closely paraphrases the equivalent of a short paragraph or more without acknowledgement; or, borrows, without acknowledgement, any ideas in a clear and recognizable form in such a way as to present them as one's own thought, where such ideas, if they were the student's own would contribute to the merit of his or her own work.

Note that this has *nothing* to do with copyright infringement or whether the plagiarized material was 'open source' or 'copyright-free'.

Plagiarism is one of the most serious academic offences a student can commit. Anyone found guilty will, on their first offence, have a written warning placed in their file and (typically) be awarded an "F" on the plagiarized work. If the student commits a second offence, an "F" will often be given for the course along with a written warning. A third offence will almost certainly will result in suspension from the program and/or the college.

Plagiarism

Any information which you submit for marks, but which is not your own, must be accompanied by a citation. The correct format for citations is given in the ACOV APA Style Manual (2013), published by the English Faculty of Algonquin College, and available in Blackboard via the *Resources* link. In particular, note that **ALL** web pages used **MUST** be cited according to the following format:

author(s)
date
page title (cap 1st word)
[type label] *Retrieved from*
URL of page that was read (if URL is short)

For example:

Hardcastle, M. (2011). What is cyberbullying? [Webpage]. Retrieved from http://teenadvice.about.com/od/schoolviolence/a/cyberbullying1.htm

From: English Faculty, Algonquin College of the Ottawa Valley. 2013. *ACOV APA Style Manual 2013*, pg. 15

Plagiarism

Additional formatting is available in the APOV APA manual (pgs. 15-18) for the following web-based resources:

- *Blogs*
- *YouTube videos*
- *Social Media*
- *Podcasts*
- *Screen Capture*
- *PowerPoint slide*

Note that any code which is given to you in class, which you use in your programs, must be cited, as follows:

```
/* This code provided by Dave Houtman [2015] personal
communication */
```

Failure to cite sources will result (at best) in lost marks and (at worst) a disciplinary hearing being conducted, with possible course failure.

Miscellaneous

Missed Classes due to Illness

- As per College policy, students are not required to supply a doctor's letter if they are absent due to a cold or flu, and miss a class or an exam as a result. However, students should send me an email (houtmad@algonquincollege.com) as soon as they become aware that they will be missing a class. If missing a lab, contact the lab professor for that section. *However*, if you are sick for more than 48 hours and miss more than two consecutive labs/lectures, you should plan to supply a doctor's letter when you return to class. (Note: if you visit the doctor due to illness, be sure to pick up a letter, even if you don't think your illness will impact your attendance.)
- If you miss an exam as a result of illness, the standard procedure is to pro-rate your remaining exams to account for the missed mark.
- Since students will generally have two weeks or more to do labs, sickness is generally not an excuse for submitting a lab late, unless it is due to a prolonged illness, in which case a doctor's letter will be required.

CAL Students

- Students who require special consideration during exams, as recognized by CAL, should present me with the appropriate paperwork as soon as possible so that proper accommodation can be reached ahead of time.

Miscellaneous

Late Assignments and Labs

- Completed labs should be demonstrated on or before the stated deadline during your lab period; in cases in which this is not possible due to either technical problems or time limitations at the end of the lab period, labs are due no later than the end of the week (you can drop by during office hours and demonstrate your lab then, if necessary). Under special circumstances, the lab submission deadline may be extended into the next week, but in practice this rarely happens.

- Late assignments will be penalized as follows:

> 1 minute late to <= 10 minutes late	**=**	**10.0% off your mark**
> 10 minutes late* to <= *6 hours late	**=**	**20.0% off your mark**
> 6 hours late to <= 12 hours late	**=**	**33.3% off your mark**
> 12 hours late to <= 24 hours late	**=**	**50.0% off your mark**
> 24 hours late	**=**	**Ha! Don't even bother…**

Again, extensions are possible under special circumstances, but unlikely.

- If you make multiple submissions of the same assignment, only the last assignment is counted.

Miscellaneous

Distractions

- Students who engage in activities which are distracting will be asked to leave. 'Distractions' includes:
  - Any conversation with a classmate above a whisper;
  - Anything displayed on a computer which is not related to coursework and which distracts other students, particularly those in the rows behind them. This includes, in particular, playing videos or computer games during class time, or running YouTube (or other) videos, etc.

Electronic Devices

- Turn off all cell phones before class (and anything else that makes a noise, for that matter)

eCigarettes

- are prohibited during labs and lectures.

Weekly Schedule - CST8284

Time	Mon	Tues	Wed	Thurs	Fri
8:00-10:00					**Lec Sec. 310 8:00-10:00 T117**
10:00-12:00					**Lab Sec. 301 10:00-12:00 T317**
12:00-14:00					
14:00-16:00	*Note: office hours are subject to possible change, depending on circumstances, scheduling conflicts, outside obligations, etc…but they will hopefully remain stable for the first few weeks of the semester.		**T323 Office Hrs* 14:15-15:30**		**Lab Sec. 302 13:00-15:00 T111**
16:00-18:00	**Lab Sec. 313 17:00-19:00 B421**		**Lec Sec. 300 16:00-18:00 B170**	**Lab Sec. 312 17:00-19:00 T304**	**Lab Sec. 303 15:00-17:00 T111**
18:00-20:00		**Wed Lab Sec. 311 16:00-18:00 P213B**	**T323 Office Hrs* *with confirmation***		

Exam Schedule \ Important Deadlines & Dates

January				
Monday	Tuesday	Wednesday	Thursday	Friday
2	3	4	5	6
9	10	11 **Sec 300 – 1st Lecture**	12	13 **Sec 310 – 1st Lecture**
16	17	18	19	20 ***Early Withdrawal Date***
23/30	24/31	25	26	27

February				
Monday	Tuesday	Wednesday	Thursday	Friday
		1	2	3
6	7	8	9	10
13	14	15 **Sec 300 – Quiz (10%)**	16	17 **Sec 310 – Quiz (10%)**
20/27	21/28	***Study Break – February 20th -24th***		24

Exam Schedule \ Important Deadlines & Dates

March				
Monday	Tuesday	Wednesday	Thursday	Friday
		1	2	3
6	7	8	9	10
13	14	15 **Sec 300 Midterm – 20%**	16	17 **Sec 310 Midterm – 20%**
20	21	22	23	24 ***Final Drop Date*** - ***last chance for a 'W'***
27	28	29	30	31

April				
Monday	Tuesday	Wednesday	Thursday	Friday
3	4	5	6	7
10	11	12	13	14 **Good Friday**
17	18	19 **Last Lecture**	20	21 **Last Lecture**
24	25	26	**27 Final Exam 3:00-6:00 in the Gym**	28

A Note About the Slides

- The  symbol in the top right corner of a slide means that the information on the slide is provided for general information or context only – you won't be tested on this *specific* information. Use this when studying for exams to determine whether or not you can safely ignore the *details* on the slide.
- Note that, while you will not be *directly* tested on the material contained on these slides, this material is often essential to your overall understanding of the information that follows. So while you *don't need to memorize the specific content of these slides* in preparation for an exam, you're nonetheless responsible for understanding how the material presented fits into the overall context of the course. You can skip the *details* of such slides; that doesn't mean you can skip these slides altogether.

A Note About the Slides

- The first time a term is used in a slide it will usually be highlighted in ***dark blue text***. This also includes terms which you should have seen in previous courses, in particular your Java courses (e.g. ***class***, ***getter***, ***overloading***, etc.). Terminology which you should already have encountered will *not* be defined in the slides, on the assumption that you already understand these words. But if you have any questions, ask for clarification. Regardless of whether you've seen a term before or not, if its related to this course, then you're responsible for knowing what it means on a test, as well as knowing how to use it intelligently in conversation—or during a job interview.

- Also, be aware: each quiz/exam contains a few questions related to terminology, along with Java keywords. Study (and understand) the highlighted words and you'll grab extra marks you wouldn't get otherwise.

A Note About the Slides

- In order to explain why code works, it is sometimes necessary to provide examples of code that contains bugs, and is therefore explicitly designed *not to work*. The following symbol is used to prevent you from accidentally referencing these deliberately-faulty code fragments:

Potential naming conflict

—where the words under the 'X' indicate the source of an error. When an unintended error is *possible* (but unwise), the following symbol is used to caution you against using the code without understanding the nature of the *potential* bug:

Possible unintended rounding error

A Note About the Slides

- When incorrect code has been presented it is sometimes followed by a corrected version. To distinguish the 'wrong' from the 'right' version, the following symbol is sometimes employed:

- Some course components are optional: nice to know about, but not really essential to your understanding of Java. Particularly near the end of the semester, various 'optional' components may appear in the notes, which I may discuss, time permitting—or not. When 'optional' components are discussed, they are fair game for an exam question.

A Note About the Slides

- Because we'll be using both Deitel & Deitel's *Java: How to Program, 10th ed*. and Liang's *Introduction to Java Programming* as references, the slides are sometimes marked with section references to help you locate the material. Whenever you see the following:

D&D 7.6 **D&D A** **Liang 11.9** **Liang C**

this indicates the section or appendix in the text to search out for further information on the subject under discussion.

I frequently borrow code and information from these (and other) sources. The source of the code will be footnoted at the bottom of the slide in which the material first appears, such as :

Liang D. Y. (2014). *Introduction to Java Programming, Comprehensive Version, 10th Ed. Toronto, ON: Pearson.* pp. 507-509.

A Note About the Slides

- Code and keywords in Java are indicated in `Courier New`
- When there's something in a code fragment or sentence that you need to focus on, like a new function, command, etc., it is usually written in **boldface** type, to help set it off from the rest of the code.
- It is frequently necessary to shorten code to fit it on a slide. For convenience, I use the ellipsis (…) to indicate 'continued from before' or 'and so on.' This, of course, is usually not valid code (although the ellipsis *can* legitimately be to enter *varargs* as parameters in method headings). Generally speaking, if you want to test code with a '…' in it, you'll need to figure out what the '…' stands for.
- I test all the code in Eclipse beforehand (time permitting), so it should work as indicated. However, code sometimes needs to be trimmed to fit onto a slide, and this is where errors are most likely to be introduced. If you spot an error, please point it out, and I'll correct the slides ASAP.

A Note About the Slides

- Microsoft Office products have the annoying tendency to insert angular ‵quotes″ in place of normal upright ones, like "this". Java (and C, C++, JavaScript, etc.) do not recognize ‵ and ″, and will flag an error if you try to compile code containing them (Note: they are not in the ASCII character set). If you want to run the sample code found in these slides, make sure it contains only "upright" quotes, and not ‵slanted″ quotes.
- More generally, code copied from .pptx and .pdf documents often contains non-ASCII “hidden” characters which do not show up in text editors but which are forbidden in Java. Oftentimes, the trouble and frustration spent finding these hidden, uncompilable spaces, tabs and carriage returns causes more trouble than just typing the code in manually.

A Note About the Slides

- The inline comments in these slides are intended to explain either *what* the code does, or *how* it works; they are *for educational purposes only*. They are *not* indicative of the kind of comments you are expected to insert into your code.

  Inline comments should focus on *why* a line of code is needed in the context of a class or method. For example, assume `int i=0;` in the following code. Comments such as

```
obj.method(++i)   // incr i by 1 and pass to method
```

Obvious!

  are a waste of space and time, since they contribute nothing that isn't already manifestly obvious. (If anything, they show how little the programmer understands about the purpose of documentation.) However, a comment such as

```
obj.method(++i)   // value passed must be nonzero,
                  // hence prefix required
```

Useful Comment

  is valuable and essential, since this explains something that may not be otherwise obvious, and could be overlooked, creating difficult-to-catch errors in the future.

A Note About the Slides

Comments that explain *how* your code works should be included *only* if the meaning is non-obvious. (And if your code is that convoluted, you should probably consider breaking it down into smaller code fragments.)

In addition to inline comments, you will be expected to supply headers for each method as well as JavaDoc comments (a subject to be dealt with shortly). The College's documentation standard is located in the *Resources* folder in Blackboard

- Finally, code presented in lectures and labs is largely undocumented in order to fit it all onto a slide or two. Fully documented versions will sometimes be made available on Blackboard. Your programs need to be much better documented than I have space for on these slides.