

Computer Science I

CMSC 120 • Fall 2009

-Background

When	<i>Class</i> Monday and Thursday afternoons 12:30pm through 1:45pm <i>Lab</i> Section 113 Tuesdays at 3:30pm / Section 114 Wednesdays at 3:30pm	
Where	Lowell Thomas, room 135	
Required Text	<i>Starting Out with Java, 4th edition</i> by Tony Gaddis ISBN 978-0-13-608020-6	
Optional Text	<i>Code Complete, 2nd edition</i> by Steve McConnell ISBN 978-0735619678	
Web Site	www.3nfconsulting.com/students-cs1.aspx	
Instructor	Alan G. Labouseur LT 101 (office hours posted)	Alan.Labouseur@Marist.edu alan@3NFconsulting.com 845-575-3000 x2831 <i>Marist phone</i> 845-440-1102 <i>alternate phone</i>

-Grading

Letter Grades											
You can earn up to 1000 points over the course of the semester, broken down over the following areas:	Quizzes	10%	100 points: 10 at 10 points each								
	Homework	20%	200 points: 10 at 20 points each								
	Test 1 (incremental)	10%	100 points, covers material up to Test 1								
	Test 2 (incremental)	10%	100 points, covers material after Test 1								
	Final Project - game	20%	200 points, requires all material								
	Lab Work	15%	150 points, see lab syllabus for details								
	Attendance	5%	50 points for consistently showing up								
	Participation	5%	50 points for constructive participation								
	Laziness Adjustment	3%	30 points for not being lazy								
	Whining Adjustment	2%	20 points for not whining								

-Objectives and Assessment

Assessment methods include assignments, quizzes, exams, discussions, presentations, and projects.	<p>This course will teach the student how to design, develop, test, debug, and document a program with good programming style. The students will:</p> <ul style="list-style-type: none"> • understand how data is represented in a computer • know and use correctly data types, operators, and control structures of Java • be able to correctly use selection and repetition control structures • understand the nature of objects as consisting of data and methods • be able to design and implement simple classes for problem solving • be able to declare and manipulate arrays • get practice in finding some answers for themselves, because capable problem solvers never stop learning.
---------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Computer Science I

CMSC 120 • Fall 2009

-Proposed Schedule

#	Week	Chapter	Topic	Required
1	31-Aug	1	<i>Monday:</i> Introduction, expectations, background, HelloWorld() <i>Thursday:</i> Base 2, 10, and 16 / Memory / Compiling and the JVM	<i>Presence</i>
2	7-Sep	2	<i>Thursday:</i> Program Structure, Esc Seq / Identifiers, Vars, and Literals	Twitter Homework
3	14-Sep	2	<i>Monday:</i> Primitive Data Types, Strings, ASCII, and Unicode <i>Thursday:</i> Assignment and Arithmetic operators, Code Style	Cake Homework
4	21-Sep	2, 5.1	<i>Monday:</i> Primitive vs. Ref types, type conversions, keyboard input <i>Thursday:</i> the Math class, Random(), intro to Methods, Code Style	Golf Homework
5	28-Sep	2, 5.1 3.7	<i>Monday:</i> Constants, Strings, peek at if and while for more Adventure <i>Thursday:</i> Scope, More Adventure, Comments and Code Style	Zork Homework
6	5-Oct	-	<i>Monday:</i> Review for test one <i>Thursday:</i> Incremental Test One in class	Game Loop Homework
7	12-Oct	3, 4	<i>Monday:</i> Control flow in our Adventure Game: chaining or dispatch? <i>Thursday:</i> More control flow, Review and analyze Test One	<i>Game Progress</i>
8	19-Oct	3, 4	<i>Monday:</i> Selection, Relational operators, Control flow <i>Thursday:</i> Repetition - Inc and Dec, Looping, Logical operators	<i>Game Progress</i>
9	26-Oct	5	<i>Monday:</i> Value-returning Methods, Parameter passing <i>Thursday:</i> Scope, pass by “value” and pass by “reference”	Game Nav Homework
10	2-Nov	5, 6	<i>Monday:</i> Reference type parameter passing, Classes and Objects <i>Thursday:</i> Classes and Objects, properties, “Has-a”	Params Homework
11	9-Nov	6	<i>Monday:</i> Constructors, Get and Set, OOP Principles <i>Thursday:</i> Classes and Objects in our Adventure game	Game v0 Homework
12	16-Nov	-	<i>Monday:</i> Review for test two <i>Thursday:</i> Incremental Test Two in class	Game v0.5 Homework
13	23-Nov	-	<i>Monday:</i> Review Test Two	<i>Game Progress</i>
14	30-Nov	8	<i>Monday:</i> Enums <i>Thursday:</i> Arrays of ints, Strings, and objects (oh my); iterating over	<i>Game Progress</i>
15	7-Dec	11	<i>Monday:</i> Introduction to Inheritance, “is a” / Lab time for project <i>Thursday:</i> Lab time and review for final project, achieve calmness.	Game v0.7 Homework
16	16-Dec	-	Wednesday at 1 pm: Adventure Games - demos in class	Game v1.0

Computer Science I

CMSC 120 • Fall 2009

-Policies

Tests

Tests cover material presented up to the class in which the test is administered. No makeup tests will be given. If you anticipate missing a test, make arrangements with me in advance to hand in the exam on or prior to its due date.

Homework

All assignments must be handed in at the beginning of class on the day they are due. Since all homework assignments are outlined in this syllabus, arrange to submit homework on schedule, even when a class will be missed.

Late Submissions

No assignments will be accepted late because we will be going over the assignment in class on the day it's due. This is an important part of the learning process, and once we cover the assignment in class, you clearly cannot hand it in after that. (Inconceivable!)

Academic Honesty

As a part this class, I will uphold and **vigorously enforce** the general policies of this institution on academic honesty and plagiarism. All examinations, papers, projects, and homework assignments are subject to the usual standards of academic honesty as described in the Student Handbook and/or other related publications.

Furthermore, I expect my students to behave in a manner appropriate to Computer Science and Information Technology professionals. Professional ethics **demand** that you embrace traditional "thou shall not cheat" behavior, and also that you soundly reject additional forms of dishonesty and abuse which are uniquely possible working with computers.

Remember: Allowing someone to copy your work is every bit as dishonest as copying someone else's, and will be treated just as harshly.

Any violation -- actual or perceived (in my sole discretion) -- of this Academic Honesty policy will result in one or more of the following actions in addition to any other forms of recourse available as specified by the Student Handbook:

- You will be ejected from the course with a failing grade.
- A letter will be sent to your department chair, your Dean, and the president of the college.
- And more. (And worse!)

The bottom line is that I expect you to conduct yourself as a person of integrity. This means that **plagiarism in any form is completely unacceptable**. You are soon-to-be a computing professional, and I encourage you to consult the ACM professional code of ethics. See www.acm.org/about/code-of-ethics.

Computer Science I

CMSC 120 • Fall 2009

-Policies

Appealing Grades

This semester I will implement an appeals process to handle any questions you might have about fairness related to the grading of your work. I will address each and every one of your concerns. To that end, and in order to be fair and efficient, I insist you to write a letter of appeal.

Rules for Submitting an Appeal

- Appeals must be in the form of a neatly written letter.
- Appeals must be on a separate paper and stapled to the work in question.
- Every appeal (if there is more than one) requires its own paragraph.
- Appeals are due the next class period after the work is returned to you.
- Appeals must be very specific.
- Appeals must be content-based, not personal or emotional.
- Insufficient time is not a basis for an appeal.
- You must communicate what action you would like taken, for instance give full credit, add points, etc.

This process empowers students, advances learning, and moves students toward academic maturity. As such, it benefits both the teacher and the student. Further, students are given a method to argue their points in an appropriate manner and explain their reasoning, while the teacher has an opportunity to learn whether or not he has understood students' reasoning.

Lab Work

It's entirely possible that you may encounter some topics in lab prior to our discussing them in class. **Don't panic!** Our lab instructors are fine teachers as well as experts in this field. As such, they are extremely well equipped to introduce you to new topics. Then, when we get there in class, you can amaze me with your knowledge.

Learning to Learn

Capable professionals know how to solve problems, even -- perhaps most especially -- in the absence of complete knowledge. This is a large part of what I want to teach you. To that end, I will encourage and at times require you to practice finding things out for yourself. There will be occasions when you need to look things up and find things out *on your own* to complete an assignment. This is an important skill, and one that will serve you for the rest of your life, so we might as well begin practicing it now.

Students with Disabilities

Any student requesting or wondering about accommodations based on a disability should see the fine folks at the Office of Special Services in Donnelley 226 and online at www.marist.edu/specialservices.

Computer Science I

CMSC 120 • Fall 2009

-Program Evaluation

Guidelines for Grading Programs and Projects

All programs and applications must be free of syntax errors to receive any credit. Programs that ((compile or interpret) and execute) cleanly but contain logic errors will be graded based on the severity of the errors and how well your work demonstrates your approach to solving the problem.

When evaluating your programming assignments I will ask myself the following questions about your program:

- Is it correct? I.e., free from faults in specification, design, and implementation?
- Is it usable? About the interface...
 - ▶ Is it “clean” and well-organized? Would Mr. Monk be proud?
 - ▶ Does it obey the Laws of Least Astonishment?
 - ▶ Is it accurate?
 - ▶ Is it easy to use?
- Is it reliable and robust? I.e., can it perform its functions without breaking down, even given unexpected input or other circumstances?
- About the readability and maintainability of the source code...
 - ▶ Are the comments plentiful, clear, meaningful, and helpful?
 - ▶ Are the identifier names accurate, clear, and meaningful?
 - ▶ How is its object-oriented architecture?
 - ▶ Does it compile or interpret cleanly?
- About the Design...
 - ▶ Is the solution well-designed?
 - ▶ Is it re-usable?
 - ▶ Does it illustrate the points made and principles discussed in class?
 - ▶ Have any lazy shortcuts been taken?
 - ▶ Can the program be unit and system tested?
- About the results, are they accurate? I.e., are the qualitative outputs free of error?
 - ▶ Does it perform as assigned?
 - ▶ Is the output well-formatted and meaningful?

Remember, neatness and style count. If you hand in a program that works, but that does not adhere to reasonable style standards, is inadequately commented, or is poorly designed, you will be penalized. Good habits are important and I want you to develop some.