Lassonde School of Engineering

Department of Electrical Engineering and Computer Science

Course

EECS 1020 3.0 Introduction to Computer Science I

Course Webpage

moodle.yorku.ca

Term

Fall 2014

Prerequisite / Co-requisite

One of (1)–(3) below must be met:

(1) (New high school curriculum): Two 4U Math courses including MHF4U (Advanced Functions), with no grade below 65%.
(2) Completion of 6.00 credits from York University MATH courses (not including AK/MATH1710 6.00 or courses with second digit 5) with a grade point average of 5.00 (C+) or better over these credits.
(3) Completion of AK/MATH1710 6.00, or 6.00 credits from York University mathematics courses whose second digit is 5, with an average grade not below 7.00 (B+).

Strongly Recommended: Previous programming experience; for example, a high school programming course or EECS 1530 3.00.

Course Instructor

Steven Castellucci
Location: Lassonde Building, room 3048
Office hours: Mondays and Thursdays, 15:30-16:30 (or by appointment)
Email: steven_c@yorku.ca

Time and Location

Section A
Lectures: Lassonde Building, Lecture Hall C on Thursdays, 19:00-22:00
Labs: Lassonde Building, rooms 1002, 1004, and 1006 on Thursdays, 17:30-19:00

Section B
Lectures: Vari Hall, Lecture Hall B on Mondays, Wednesdays and Fridays, 14:30-15:30
Labs: Lassonde Building, rooms 1002, 1004, and 1006 on Wednesdays, 10:30-12:00

Section E
Lectures: Lassonde Building, Lecture Hall B on Mondays, Wednesdays and Fridays, 10:30-11:30
Labs: Lassonde Building, rooms 1002, 1004, and 1006 on Wednesdays, 14:30-16:00

Expanded Course Description

Many processes can be viewed as a sequence of interactions between a client who requests a service and an implementer who provides it. The concerns of these two parties, albeit complementary, are completely separate because one deals with the "what" while the other deals with the "how". It is widely recognized that separating these concerns leads to reliable, scalable, and maintainable software. Based on this, EECS 1020 deals exclusively with the client who needs to be able to look for services; read their API (Application Programming Interface) specifications; create programs that use them; and determine if they are operating correctly relative to their specifications. Topics include delegation and contracts, encapsulation and APIs, aggregation and the collections framework, and inheritance and polymorphism. The course emphasizes the software development process and introduces elements of UML (Unified Modelling Language) and software engineering. It involves three-hours of lecture and weekly laboratory sessions.

The course uses the Java programming language throughout. Its assessment is based on a series of programming exercises and a number of written tests. The two components have approximately equal weights and are intended to measure the student's understanding of theoretical concepts and ability to build applications.

This course is an introduction to the discipline; it is not a survey course. As such the emphasis is on the development of a theoretical conceptual foundation and the acquisition of the intellectual and practical skills required for further courses in computer science. The course is intended for prospective computer science and computer engineering majors, i.e. those with a well-developed interest in computing as an academic field of study and with strong mathematical, analytical and language abilities; it is not intended for those who seek a quick exposure to applications or programming (for this purpose any of EECS 1520, EECS 1530 or EECS 1540 would be more appropriate).

Warning: The work for this course includes a substantial number of exercises that require problem analysis, program preparation, testing, analysis of results, and documentation and submission of written reports. The course is demanding in terms of time, and requires the student to put in many hours of work per week outside of lectures.

Recommendation: You will benefit if you have prior practical experience with programming as well as using a computer. Students who wish to take a one-course exposure to the practical aspects of computing should consider enrolling in EECS 1520 3.00 and EECS 1530 3.00 instead.

Course Objectives

Introduction and Chapter 1

- Use the EECS Linux computing environment to issue commands using the terminal, navigate the file system, and start applications
- Create, compile, and run Java programs
- Interpret and correct compile-time errors
- Describe data types and give example values
- Declare variables and assign values to them
- Identify when casting between types is required

Chapter 2

- Read and interpret UML class diagrams
- Discriminate between client responsibilities and implementer responsibilities
- Use existing classes and their features to solve simple problems
- Interpret and correct run-time errors

Chapter 3

- Locate specific classes in the Java API
- Read a given API and extract information about the class, attributes, and methods (including return types, required arguments, preconditions, and postconditions)
- Format output according to program specifications

Chapter 4

- Describe how an object is created in memory
- Identify how an object's state can be accessed or modified
- Demonstrate how objects can be tested for equality

Chapter 5

- Control a program's flow of execution using if-statements or switch statements
- Use a loop structure to repeatedly execute a block of code
- Use nested control structures to tackle complex problems
- Perform file input and output

Chapter 6

- Create, transform, and compare String objects
- Iterate over the characters in a String
- Locate a character or substring within a String
- Define and use a mutable representation of characters
- Use regular expressions for pattern matching

Chapter 7

- Describe and explain the waterfall model and iterative methodology of software design
- Identify an appropriate test vector to evaluate correctness of a program
- Perform unit testing using JUnit

Chapter 8

- Understand the difference between aggregation and composition
- Identify the difference between an alias, a shallow copy, and a deep copy
- Identify the difference between indexed and iterator-based traversal of a collection
- Describe and rank algorithm complexity using big-O notation

Chapter 9

- Describe the relationship between a superclass and subclass
- Understand the Substitutability Principle
- Use polymorphism to simplify coding
- Differentiate between a class, an abstract class, and an interface

Chapter 10

- Describe how the list, set, and map collections differ in data organization
- Use generics to perform compile-time type checking
- Demonstrate how to add elements to a collection, remove elements from a collection, and iterate over all elements in a collection

Chapter 11

- Understand the Throwable Hierarchy
- Use a try-catch block to handle thrown exceptions
- Create exception objects and throw them

Course Text / Readings

Additional readings may be assigned or recommended during the course.

Java By Abstraction: A Client-View Approach. Hamzeh Roumani, Pearson Publishing.

Students may use the third or fourth edition. The second edition can also be used provided that students consult a copy of the third edition in the Steacie Science and Engineering Library for Chapter 1 and 2.

Evaluation

The final grade of the course will be based on the following items weighted as indicated:

5 tests each 15%
Final exam 25%

Grading, Assignment Submission, Lateness Penalties, and Missed Tests

Grading

For each test, students receive a score in the range 0-15. For the final exam, students receive a score in the range 0-25. The final grade for the course is obtained by adding the scores of the five tests and the final exam and converting this total to a letter grade according to the following table.

F	E	D	D+	C	C+	B	B+	A	A+
<40	≥40	≥50	≥55	≥60	≥65	≥70	≥75	≥80	≥90

Final course grades may be adjusted to conform to Program or Faculty grades distribution profiles.

For a full description of York grading system see the York University Undergraduate Calendar - http://calendars.registrar.yorku.ca/2010-2011/academic/index.htm

Missed tests
No make-up tests will be given. If you miss a test for reasons beyond your control, inform your instructor as soon as possible. If the reason is medical in nature, you must also submit a completed **Attending Physician's Statement** (see link below) to your instructor. **An ordinary medical note from a doctor is not sufficient. The form must be completed by you and by your attending physician.** If approved by your instructor, the weight of the missed test will be distributed to a later component in the course. Otherwise, you will receive a mark of zero for the missed component.

The Attending Physician's Statement is part of the Registrar's Petition Package, available at http://www.registrar.yorku.ca/pdf.php?pdf=petition_package.pdf

IMPORTANT COURSE INFORMATION FOR STUDENTS

All students are expected to familiarize themselves with the following information, available on the Senate Committee on Academic Standards, Curriculum & Pedagogy webpage (see Reports, Initiatives, Documents) - http://www.yorku.ca/secretariat/senate/committees/ascp/documents/CourseInformationForStudentsAugust2012.pdf

- Senate Policy on Academic Honesty and the Academic Integrity Website
- Ethics Review Process for research involving human participants
- Course requirement accommodation for students with disabilities, including physical, medical, systemic, learning and psychiatric disabilities
- Student Conduct Standards
- Religious Observance Accommodation

September 2014