SC/CSE1020 E - Introduction to Computer Science I (Summer 2011-2012)

CSE 1020: Introduction to Computer Science I

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, CSE1020 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.

The course uses the Java programming language throughout. Its assessment is based on a combination of programming tests and 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 CSE1520, CSE1530 or CSE1540 would be more appropriate).

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.

Students will benefit if they 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 CSE1520 and CSE1530 instead.

This course lays the conceptual foundation of object-orientated programming. It covers
delegation and contracts

- encapsulation and API programming
- aggregation and the collections framework, and
- inheritance and polymorsphism,

all from the client's perspective. It also covers language-specific topics including

- types
- control structures, and
- exception handling

The coverage is done within the framework of the software development process, and emphasizes software engineering throughout.

Emphasis is placed on the development of theoretical concepts, and the acquisition of intellectual and practical skills. The course is intended for prospective computer science, computer security, and computer engineering majors with strong mathematical, analytical, and language abilities. Students who seek a quick exposure to applications or programming might instead consider CSE1520, CSE 1530, or CSE 1540.

Course exercises require problem solving, coding, program testing, and analysis of results. This course is demanding in terms of time; it requires the student to put in many hours of work per week outside of lectures.

Instructor

- Michael Jenkin (jenkin@cse.yorku.ca)
- Office hours: Tuesdays 5-6pm in Sherman 1028
- In order to ensure timely responses to emails, please include CSE1020 in the email subject line and include your CSE account number and student number in the body of the email. Emails lacking such information are unlikely to receive timely or useful responses.
- Students are expected to monitor the course moodle page.

Lecture Times

- Tuesdays and Thursdays 6pm-7:30pm in SC 302

Laboratory Times

- Tuesdays or Thursdays 7:30pm-9:00pm in LAS 1006. You will have signed up for a specific laboratory. You must attend the laboratory in which you have registered.

Textbook

The required textbook for this course is "Java by Abstraction: A Client-View Approach 3rd Edition" by Hamzeh Roumani. It is available at the textbook store, and also through amazon.ca. It will be expected that you will have completed the readings for the lecture prior to the lecture. This includes the first week (see the tentative course outline below).

Tentative Course Outline

- Week 1 May 8 and 10. Chapter 1. Introduction to programming.
- Week 2 May 15 and 17. Chapter 2. Programming by delegation.
- Week 3 May 22 and 24. Chapter 3. Using API's.
- Week 4 May 29 and 31. Chapter 4. Using objects.
- Week 5. June 5 and 7. Chapter 5. Control structures.
  - Labtest #1 will be held in-lab this week
- Week 6. June 12 and 14. Chapter 6. Strings.
- Week 7. June 19 and 21. Chapter 7. Software development.
  - Midterm will be held in-class on June 19.
- Week 8. June 26 and 28. Chapter 8. Aggregation.
  - Labtest #2 will be held in-lab this week
- Week 9. July 10 and 12. Chapter 9. Inheritance.
- Week 10. July 17 and 19. Chapter 10. The collection framework.
- Week 11. July 24 and 26. Chapter 11. Exception handling
- Week 12. July 31 and Aug 2. Chapter 12. A multi class-application
  - Labtest #3 will be held in-lab this week

Grading

- Lab test #1. 20% Held in-lab the week of June 5
- Lab test #2. 20% Held in-lab the week of June 26
- Lab test #3. 19% Held in-lab the week of July 31
- Midterm. 20% Held in-class on June 19
- Final. 21%. Date and location TBD, but during the examination period

Each piece of work will be assigned a numeric grade. A final numeric grade will be computed using the weighting given above. The final letter grade will be determined from the numeric score using the standard Computer Science and Engineering mapping.