UNIVERSITY OF NEBRASKA AT OMAHA
COURSE SYLLABUS

Department and Course Number	*CIST 1400*
Course Title	*Introduction to Computer Science I*
Course Coordinator	*Robert Fulkerson*
Total Credits	*3*
Repeat for Credit?	*No*
Date of Last Revision	*Jan 17, 2017*

1.0 Course Description Information

1.1 Catalog description:

An introduction to programming within the context of a high level modern programming language. Coverage of fundamental programming concepts and program design; including arrays, user defined types, and objects. This course has a required laboratory component; students must register for a laboratory section when enrolling in lecture.

1.2 Prerequisites of the course:

MATH 1320 and either CSCI 1200 or CIST 1300 (with a grade of "C-" or better)

1.3 Overview of content and purpose of the course:

This course is a foundational course for all other computer science courses and other courses in the College of IS&T. The primary goal of the course is to prepare students for further study in the field of computer science by presenting them with a two-pronged approach to learning basic computer programming. Students will learn fundamental programming concepts such as selection, iteration, modularity and organization of data that apply to many programming languages. At the same time, they will be exposed to a specific current high-level language that demonstrates these concepts. In the laboratory sections, students will gain additional weekly hands-on practice with the programming skills and techniques described in the corresponding lecture.

1.4 Unusual circumstances of the course.

None

2.0 Course Justification Information

2.1 Anticipated audience / demand:

This course is intended for freshman-level students in the College of IS&T. It also serves many other departments, including math, education, geography/geology and engineering.

2.2 Indicate how often this course will be offered and the anticipated enrollment:

This course is offered every semester, including during the summer. Typical enrollments based on previous semesters:

Fall semester: 350 students

Spring semester: 200 students

Summer semester: 50 students

2.3 If it is a significant change to an existing course, please explain why it is needed:

Two major changes are reflected in this syllabus.

First, course learning outcomes have been significantly elaborated to better communicate the purpose of this course. The new learning outcomes are aligned with CC2013 knowledge units and are specified in a language-neutral way so as to not become outdated due to a change in the programming language of instruction. Section 4 remains unchanged as it is a specific instance of meeting the new abstract learning outcomes.

The second change incorporates a mandatory lab component to this course. CIST1400 has historically had both required and optional lab elements over the years. Since moving to an optional CIST1404 lab we have noted persistent difficulties tied to a lack of guided practice with core skills. Student surveys indicate that the bulk of students found lab to be beneficial and students who take lab often perform better in lecture than their peers. Requiring lab for all ensures there is adequate time for hands-on practice, and that all students going into CSCI1620 do so on a more even skill level. Unlike the current set up with CIST1404, the new lab does not carry a separate course number and will appear as a separate lab section with the CIST1400 course number. CIST1404 will be eliminated from the catalog as it will no longer be needed.

3.0 List of performance objectives stated in learning outcomes in a student's perspective:

Top level learning outcomes indicate the level of mastery per ACM CC2013 {Familiarity, Usage, Assessment}. Detailed knowledge and skills statements further elaborate on each top level learning outcome.

3.1 (Usage) Understand and apply concepts of the software development process.

Students will know:

3.1.1 Familiarity with a basic software development process, including concepts of specification, design, implementation, documentation, and iterative debugging and testing
3.1.2 How to decompose a complex problem into smaller parts
3.1.3 Self-awareness of when to seek outside assistance

Students will be able to:

3.1.4 Use a basic programming environment
3.1.5 Implement problem solutions using stepwise refinement
3.1.6 Write appropriately documented code
3.1.7 Identify and resolve compiler, runtime, and logic errors in a program
3.1.8 Generate and utilize test cases to verify program correctness
3.1.9 Ethically use external resources

3.2 (Usage) Understand and apply correct usage of primitive data types including strings

Students will know:

3.2.1 Existing, language-defined data types and their differences

3.2.2 The basic difference between primitive and reference data types
3.2.3 Basic logical, arithmetic, and relational operations involving primitive data types and text strings

Students will be able to:
3.2.4 Appropriately use the correct data type in a given situation
3.2.5 Create and manipulate primitive data type and text strings
3.2.6 Convert data between types using appropriate techniques
3.2.7 Correctly perform basic input and output of primitive data types
3.2.8 Create meaningful logical, arithmetic, and relational expressions

3.3 (Usage) Understand and apply correct usage of control structures

Students will know:
3.3.1 How and when to use selection, repetition and sequence in a program
3.3.2 Basic algorithm development using step-wise refinement

Students will be able to:
3.3.3 Design and implement interactive programs involving sequential, selection, and repetition structures
3.3.4 Be able to express simple algorithms in abstract language followed by implementation

3.4 (Usage) Understand and apply correct usage of procedures (ie., functions, methods) and parameters

Students will know:
3.4.1 Why and how to use procedures
3.4.2 What an API is, and how to utilize it
3.4.3 The principles and concepts of identifier scope
3.4.4 Program execution flow

Students will be able to:
3.4.5 Utilize common procedures provided by language APIs to solve problems
3.4.6 Write a procedure given specifications
3.4.7 Test functional units independently
3.4.8 Use a provided or self-created procedure effectively in a problem solution
3.4.9 Implement a problem solution requiring multiple procedures

3.5 (Usage) Understand and apply correct usage of 1-D arrays (e.g., vectors, lists) and 2-D arrays (e.g., matrices)

Students will know:
3.5.1 Rationale for using collections of data in problem solutions
3.5.2 Underlying implementation of arrays
3.5.3 Concepts of at least two basic searching and two basic sorting algorithms

Students will be able to:

3.5.4 Create, manipulate, and use 1-D and 2-D arrays of fixed size containing primitive data types
3.5.5 Implement basic array processing algorithms such as finding min or max, copying, and equality
3.5.6 Use predefined searching and sorting procedures in a problem solution

3.6 (Usage) Understand and apply correct usage of classes and objects

Students will know:

3.6.1 Differences between classes, objects, and object references and their relationships
3.6.2 Basic principles of encapsulation including visibility

Students will be able to:

3.6.3 Appropriately implement a class from a given specification
3.6.4 Use a provided or self-created class effectively in a problem solution, including accessing data members and behaviors

4.0 Content and Organization Information

4.1 Overview (0.5 weeks)

4.1.1 What is a computer
4.1.2 Machine Language
4.1.3 Assembly Language
4.1.4 High-Level Language
4.1.5 Compiling bytecode

4.2 Introduction to console-based Java Applications (1.5 weeks)

4.2.1 Basic program structure
4.2.2 Console output with System.out.printf()
4.2.3 Console output with System.out.print()
4.2.4 Console output with System.out.println()
4.2.5 Variables
4.2.6 int
4.2.7 Console Input Using the Scanner Class
4.2.8 Arithmetic (+, -, *, /, %)
4.2.9 Precedence of operators
4.2.10 Equality and relational operators (==, !=, <, >, <=, >=)

4.3 Algorithms and Control Structures (3.5 weeks)

4.3.1 Algorithms
4.3.2 if
4.3.3 Logical operators

4.3.4 if/else
4.3.5 switch
4.3.6 while
4.3.7 do/while
4.3.8 for
4.3.9 break
4.3.10 continue

4.4 Arithmetic and Primitive Data Types (0.5 weeks)

4.4.1 Compound assignment operators
4.4.2 Unary increment and decrement operators
4.4.3 Primitive data types
4.4.4 Promotion and demotion

4.5 Methods (3 weeks)

4.5.1 Static methods
4.5.2 Static fields
4.5.3 Math class
4.5.4 Java API packages
4.5.5 Methods with multiple parameters
4.5.6 Argument promotion and casting
4.5.7 Scope
4.5.8 Method overloading

4.6 Arrays, Random Numbers and Strings (3.5 weeks)

4.6.1 Introduction to Arrays
4.6.2 Arrays and References
4.6.3 Programming with Arrays
4.6.4 Multidimensional Arrays
4.6.5 Searching and sorting, Big-O
4.6.6 Random number generation
4.6.7 String methods

4.7 Introduction to Classes and Objects (2.5 weeks)

4.7.1 Classes, objects and instance variables
4.7.2 Declaring a class with a method
4.7.3 Instantiating an object of a class
4.7.4 Declaring a method with a parameter
4.7.5 Constructors

5.0 Teaching Methodology Information

5.1 Methods:

Concepts and skills are introduced to students during interactive lecture sessions. Lecture content is based on a common set of slides, assignments, quizzes and other documents for all instructors.

Weekly laboratory sessions provide students with directed practice on the skills of programming relevant to the current lecture content through hands-on problem solving and discussions, which are informed by formative assessments.

5.2 Student role:

Students are expected to complete class preparation (e.g., assigned readings, viewing web-lectures, etc.) prior to coming to class so that they are able to engage fully. They are encouraged to participate in class discussions and ask questions when the material is unclear since the material is cumulative. They are also encouraged to try out the code presented in class so that they can practice the concepts presented in lecture and then extend those concepts to their assignments. Students are expected to take an active role in laboratory sessions to clarify their understanding of course content. Students will complete both homework assignments and weekly labs to deepen their understanding.

6.0 Evaluation Information

6.1 Describe the typical types of student projects that will be the basis for evaluating student performance:

Exams: Students' mastery of course concepts will be assessed through three semester exams and a cumulative final exam. Exams focus primarily on the knowledge statements in section 3.

Labs: Weekly labs will be completed in the laboratory sections. Approximately 15 labs will be completed by students to assist their mastery of skills identified in section 3. Labs focus on programming in the small; that is, solving multiple-related problems in order to gain practice with specific skills each week. Students must have their labs checked-off by course staff no later than the start of the next lab period.

Homework: Approximately 5-6 integrative homework assignments will be completed during the semester. These assignments provide larger problems for students to solve that allow assessment of cross-cutting skills and synthesis of multiple course concepts.

6.2 Describe the typical basis for determining the final grade (e.g. weighting of various student projects):

40%	Homework assignments: 40%
15%	Lab Work: 15%
30%	3 Term Exams at 10% each
15%	Cumulative Final Exam

6.3 Grading type:

Percent	***Grade***	***Percent***	***Grade***
97 – 100	*A+*	*77 – 79*	*C+*
93 – 96	*A*	*73 – 76*	*C*
90 – 92	*A–*	*70 – 72*	*C–*
87 – 89	*B+*	*67 – 69*	*D+*
83 – 86	*B*	*63 – 66*	*D*
80 – 82	*B–*	*60 – 62*	*D–*
		0—59	*F*

7.0 Resource Material Information

7.1 Textbooks and/or other required readings used in course:

Deitel, Paul J., Paul J. Deitel, and Harvey M. Deitel. *Java: How to Program*. Upper Saddle River, NJ: Prentice Hall, 2015. Print.

7.2 Other student suggested reading materials:

7.3 Current bibliography and other resources:

7.3.1 Cadenhead, Rogers. *Sams Teach Yourself Java 2 in 24 Hours*. Indianapolis, IN: Sams, 2010. Print.

7.3.2 Deitel, Paul J., Paul J. Deitel, and Harvey M. Deitel. *Java: How to Program (late objects)*. Upper Saddle River, NJ: Prentice Hall, 2014. Print.

7.3.3 Fain, Yakov. *Java Programming 24-hour Trainer*. Hoboken, IN: John Wiley, 2011. Print.

7.3.4 Gaddis, Tony. *Starting out with Java: Early Objects, 6th Edition*. Pearson, 2018. Print.

7.3.5 "Java Tutorial." *Programming Tutorials and Source Code Examples*. Web. 19 Apr. 2011. <http://www.java2s.com/Tutorial/Java/CatalogJava.htm>.

7.3.6 "The Java™ Tutorials." *Automatic Redirection*. Web. 19 Apr. 2011. <http://download.oracle.com/javase/tutorial/>.

7.3.7 *Learn Java Online with Java Beginner Tutorial*. Web. 19 Apr. 2011. <http://www.javabeginner.com/>.

7.3.8 Lewis, John, Peter J. DePasquale, and Joseph Chase. *Java Foundations: Introduction to Program Design & Data Structures*. Boston: Pearson Addison Wesley, 2011. Print.

7.3.9 Lewis, John and Loftus, William. *Java Software Solutions, 9th Edition*. Boston: Pearson, 2018. Print.

7.3.10 Liang, Y. Daniel. *Introduction to JAVA Programming: Comprehensive Version*. 11th Edition. Upper Saddle River, NJ: Pearson/Prentice Hall, 2018. Print.

7.3.11 Reges, Stuart, and Martin Stepp. *Building Java Programs: a Back to Basics Approach*. Boston: Addison-Wesley, 2017. Print.

7.3.12 Savitch, Walter. *Java: An Introduction to Problem Solving and Programming, 8th Edition*. Pearson. 2018. Print.

7.3.13 Schildt, Herbert. *Java 7 A Beginner's Guide, Fifth Edition.* McGraw-Hill Osborne Media, 2011. Print.

8.0 Other Information:

8.1 Accommodations statement:

8.2 Other:

8.3 Author(s):

Robert Fulkerson

9.0 Computer Science Accreditation Board (CSAB) Category Content (class time in hours):

CSAB Category	*Core*	*Advanced*
Data structures	6	0
Computer organization and architecture	0	0
Algorithms and software design	0	0
Concepts of programming languages	37.5	0

10.0 Oral and Written Communications:

Every student is required to submit at least 0 written reports (not including exams, tests, quizzes, or commented programs) to typically 0 pages and to make 0 oral presentations of typically 0 minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

11.0 Social and Ethical Issues:
None

12.0 Theoretical content:
Please list the types of theoretical material covered, and estimate the time devoted to such coverage.

None

13.0 Problem analysis:
Please describe the analysis experiences common to all course sections.

Students must be able to identify the desired goals and outcomes from a problem statement.

14.0 Solution design:
Please describe the design experiences common to all course sections.

Although each programming assignment targets some topics more than others, namely those that have been recently presented in lecture, the solutions will be designed using cumulative programming techniques. Students will design solutions by selecting those techniques that are appropriate to the problem at hand.

CHANGE HISTORY

Date	***Change***	***By whom***	***Comments***
12/16/2002	Initial ABET version	Fulkerson	
06/13/2003	Cleanup	Wileman	
12/04/2007	First draft for migration to Java	Dasgupta	
04/21/2008	Second draft for migration to Java	Fulkerson	
04/24/2008	Third draft, minor updates	Fulkerson	
11/17/2008	Update to include CS Program Outcomes table	Fulkerson	
04/19/2011	Convert to new CCMS format	Fulkerson	
03/30/2014	Update for late objects approach	Fulkerson	
01/17/2017	Update for required 0-credit lab and language agnostic learning objectives	Dorn and Fulkerson	Reflects output of the CSEd Working Group during Fall 2016

S – Strong relationship
X – Contributing relationship

Course objective	**(a) knowledge of discipline**	**(b) analyze problem, define requirements**	**(c) design and implement solution**	**(d) function on a team**	**(e) ethical issues**	**(f) communicate effectively**	**(g) analyze impact of computing**	**(h) continued professioanl development**	**(i) Current techniques and tools**	**(j) apply foundations**	**(k) apply design and development principles**
Understand and apply concepts of the software development process	**S**		**X**							**X**	**X**
Understand and apply correct usage of primitive data types including strings	**S**	**X**	**X**						**S**	**X**	
Understand and apply correct usage of control structures	**S**		**X**						**S**	**X**	
Understand and apply correct usage of procedures (ie., functions, methods) and parameters	**S**	**X**	**X**						**S**	**X**	
Understand and apply correct usage of 1-D arrays (e.g., vectors, lists) and 2-D arrays (e.g., matrices)	**S**	**X**	**X**							**X**	
Understand and apply correct usage of classes and objects	**S**	**X**	**S**						**S**	**X**	**X**