Public Course Outline

CS 111 - Introduction to Computer Science I: Java

Units Lecture 2.00	**Units Lab** 1.00	**Units Total** 0.00 - 3.00
Lecture Weekly Contact Hours 2.00	**Lab Weekly Contact Hours** 3.00	**Total Weekly Contact Hours** 5.00
Lecture Weekly Out of Class Hours 4.00	**Lab Weekly Outside of Class Hours** 0.00	**Total Weekly Outside of Class Hours** 4.00
Total Contact Hours 80.00 - 90.00	**Total Outside of Class Hours** 64.00 - 72.00	**Total Course Hours** 144.00 - 162.00

Typically Offered: Fall, Spring, and Summer - F,SP,SU

COURSE DESCRIPTION

This course introduces object-oriented programming and concepts designed primarily for students majoring in computer science and engineering who have some programming fundamentals. The course uses topics of personal and social relevance to investigate the impacts of computing through exploring language basics, including control structures, data types, input/output, operators, classes, methods and parameters, basic inheritance, and documentation practices as well as testing and verification techniques. UC CREDIT LIMITATION: No credit for CS 111 if taken after CS 112. C-ID COMP-122, ITIS-130.

ENROLLMENT RESTRICTIONS

Advisory
CS 101 and MATH 64 or MATH 64S

OUTLINE OF COURSE LECTURE CONTENT

The course lecture will address the following topics:
I. Introduction to object-oriented programming (OOP) design and analysis
A. Classes, objects, methods, or functions
B. Encapsulation, inheritance and class hierarchies, and polymorphism
C. Constructors.

II. OOP basic language elements
A. Declarations, initializations, primitive data types, strings, and string processing: basic language elements
1. Syntax (programming grammar)
2. Integer arithmetic
3. Typecasting
4. And/or logic.

B. Decision structures
1. If/else
2. Switch.
C. Loop structures
1. While
2. Do-while
3. For and for_each.

III. Software engineering design: concepts and principles
A. Algorithms
B. Unified modeling language (UML) diagrams
C. Testing and debugging strategies
D. Multi-source files, reusable code.

IV. Introduction to data structures
A. Arrays
B. Searching and sorting.

OUTLINE OF COURSE LAB CONTENT

The course lab will address the following topics:
Individual and group projects in the lab are designed to be hands-on activities that support, compliment, and extend the material and theory presented in the lectures. Students gain significant project experience.

PERFORMANCE OBJECTIVES

Upon successful completion of this course, students will be able to do the following:
1). Use object-oriented design and programming concepts to solve problems and accomplish tasks.
2). Edit, compile, link, run, and debug programs written in an object-oriented language using at least one development environment.
3). Explain and use programming language constructs, including selection, iteration, and methods.
4). Explain the memory allocation and scope/lifetime of a value container.
5). Identify appropriate types and design compound types to organize data.
6). Use industry standard documentation techniques.
7). Debug and resolve errors that occur while programming.
8). Maintain integrity and check the validity of data input.
9). Design and implement user-friendly solutions for interactive programs.
10). Explain how computer science relates to the students' lived experiences and their future.

READING ASSIGNMENTS

Reading assignments will be consistent with, but not limited by, the following types and examples:
1). Read about object-oriented programming using the Java language in the class text as well as using and understanding specialized terms, concepts, and theories.
2). Re-enforce text topics by researching the material on the Web and other texts for further explanation.
3). Read instructor-distributed handouts on specialized topics relevant to successful programming in industry using Java and extending this knowledge to class projects.

WRITING ASSIGNMENTS

Writing assignments will be consistent with, but not limited by, the following types and examples:
1). Create and write a requirements specification that defines what the program will accomplish and how the user will interact with it for each project done as homework and in class.
2). Write an essay on a current topic assigned in class, such as "What's new in Java in 2015," and what it means and provides for computer scientists.
3). Create an essay and bullet points on a current topic of interest in the computer science field, such as "Programming Tools: Are they doing too much?" and debate the topic with classmates.

OUTSIDE-OF-CLASS ASSIGNMENTS

Outside-of-class assignments will be consistent with, but not limited by, the following types and examples:
1). Complete weekly programming assignments including a requirements specification.
2). Complete individual and group programming projects.
3). Prepare for individual presentations on specific topics discussed in the text.

STUDENT LEARNING OUTCOMES

1. The student will be able to critically analyze designs incorporating basic control structures and object-oriented design techniques which will result in appropriate solutions.

2. The student will be able to apply critical thinking to implement effective solutions to simple programming problems. These solutions may utilize primitive data types and objects of pre-defined classes and may employ basic control structures and class library objects and functions to control program execution, event-handling and provide a simple graphical user interface.

3. The student will be able to convey the basic concepts of the object-oriented paradigm and to describe basic syntactic and program control structures verbally and in writing. The student will also be able to write simple program designs using algorithms and class diagrams as well as explain a program design verbally and in writing.

4. The student will be able to work collaboratively and cooperatively in teams to develop simple software designs using algorithms and class diagrams as well as develop simple computer programs that implement these designs.

METHODS OF INSTRUCTION

Instructional methodologies will be consistent with, but not limited by, the following types or examples:
1). Instructor lecture and guided discussion about computer science and programming using an object-oriented language.
2). Use of group projects in class provides students with the opportunity to practice critical thinking skills in solving a problem as they evaluate their peers' code.
3). Use of group projects in class also provides students with the opportunity to see how other students approach and solve a problem, which enhances their programming skills.

4). Use of peer learning on coding, practice quizzes, and tests is also an effective way to help students learn the material.
5). Policies and practices that promote ownership of active learning, such as the following:

- Late policy that allows flexibility for various student circumstances.
- Opportunities to resubmit assignments to demonstrate continuous improvement.
- Personalizing choices of assessments to learn computer science and apply it to student's life.
- Regular interactions with students to provide support and feedback.

METHODS OF EVALUATION

Evaluation methodologies will be consistent with, but not limited by, the following types or examples:
1). Discussions evaluated for the student's demonstrated ability to relate computer science to their lives.
2). In-class activities evaluated for the student's ability to explain and use object-oriented and programming concepts.
3). Participation in campus events evaluated for the student's ability to relate computer science to their lived experiences.
4). Programming labs and exercises evaluated for demonstrated improvement in the student's ability to debug and resolve errors that occur while programming.
5). Unit deliverables evaluated for the student's ability to design and implement user-centered solutions that relate to their lives and interests.
6). Exams and quizzes to align with industry certifications.

REQUIRED TEXTBOOKS

Examples of typical textbooks for this course include the following:

1. **Author** Gaddis, Tony

   Title Starting Out with Java: From Control Structures Through Objects

   Edition 7th ed.

   Publisher Pearson

   Year 2018

   ISBN 978-0134802213

   This is the most current, in-print edition. No

COURSE REPEATABILITY

Total Completions Allowed: 1

Rationale for multiple enrollments:

Courses Related in Content (CRC) in Physical Education, Visual Arts, and Performing Arts:

DISTANCE ED (FORM A)

Type of Approval: 100% Online or Hybrid

You may indicate here which component(s) of the course should never be conducted online (e.g. proctored exams, labs, in-person orientation, etc.):

ARTICULATION

Transfer Status: Acceptable for Credit: CSU, UC -

CSU/IGETC GE Area(s): 103 - CSU, UC

THIS COURSE IS INCORPORATED INTO THE FOLLOWING PROGRAM(S)

Tech Support *CURRENT* Certificate of Achievement
Tech Support *CURRENT* AS Degree
Game Developer *CURRENT* Certificate of Proficiency
Software Development *CURRENT* AA Degree
Software Development *CURRENT* Certificate of Achievement
Mobile Application Development *CURRENT* Certificate of Proficiency
Mathematics for Transfer *FUTURE* AS-T Degree
Software Development *FUTURE* Certificate of Achievement
Software Development *FUTURE* AA Degree