Public Course Outline

CS 112 - Introduction to Computer Science II: 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 uses topics of personal and social relevance to investigate the impacts of computing through exploring advanced object-oriented programming concepts such as abstraction, inheritance, polymorphism, and encapsulation. Topics include recursion, generics, event-driven programming, graphical user interfaces, file input and output, and exception handling.

ENROLLMENT RESTRICTIONS

Prerequisite
CS 111

OUTLINE OF COURSE LECTURE CONTENT

The course lecture will address the following topics:
I. Object-oriented programming (OOP): design and analysis
A. Encapsulation, inheritance, and polymorphism
B. Abstract classes and generics or parameterized types.

II. Software engineering design: techniques and strategies
A. Memory management and garbage collection
B. Visibility, scope, binding, and lifetime.

III. Advanced language elements
A. Copy constructors
B. Advanced input/output techniques
1. Text and binary files

2. Random access files.
C. Recursive methods
D. Fault-tolerance through exception and error handling.

IV. Advanced data structures
A. ArrayLists, vectors, linked lists, and iterators
B. Data structure assessment and appropriate selection.

V. Graphics
A. Graphical user interface (GUI) design
B. Java applets.

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). Analyze program requirements and apply concepts of polymorphism, inheritance, encapsulation, and class hierarchy in developing an object-oriented solution.
2). Explore using different software engineering design concepts and paradigms, such as model-view-controller and event-driven programming.
3). Compare and contrast the advantages and disadvantages among data structures, such as arrays versus ArrayLists.
4). Demonstrate the concept and use of recursive methods.
5). Design abstract classes and methods using generics.
6). Differentiate among, as well as read and write to, various file types.
7). Create exception-handling routines.
8). Demonstrate basic GUI programming concepts using the JavaFX library.
9). 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). Reinforce 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 detailed 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.
3). Create bullet points on a current topic of interest in the CS field, such as "Algorithmic Bias" 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 detailed 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 analyze designs incorporating more complex control structures and object-oriented design techniques which will result in appropriate solutions to specified problems.
2. The student will implement efficient, effective solutions to more complex programming problems. These solutions will incorporate the principles of inheritance and polymorphism, fault-tolerance through exception and error handling, utilize data files and implement basic data structures and associated algorithms.
3. The student will be able to convey more advanced concepts of the object-oriented paradigm, describe more complex syntactic and program control structures, write more sophisticated programs as well as explain program designs verbally and in writing.
4. The student will be able to work in a team, agree upon an organizational model, establish roles, and delegate tasks to develop object-oriented software designs and complete programming projects on time in a cooperative and collective manner.

METHODS OF INSTRUCTION

Instructional methodologies will be consistent with, but not limited by, the following types or examples:
1). Instructor-provided lecture and guided discussion.
2). In-class quizzes and lab assignments to give student programmers the opportunity to practice critical thinking skills as they relate to the efficacy and efficiency of their programs.
3). Group projects completed by teams of four to five students to provide student programmers an opportunity to learn from each other as well as get valuable feedback on their programming skills and their abilities to work in a team environment simulating industry conditions.
4). 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 CS and apply it to their 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 life.
2). In-class activities evaluated for the student’s ability to explain and analyze object-oriented and GUI 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 apply advanced object-oriented principles.
5). Unit deliverables evaluated for the student’s ability to implement software engineering design concepts to create user-centered solutions that relate to their life 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)

Computer Science for Transfer *CURRENT* AS-T Degree
Software Development *CURRENT* AA Degree
Software Development *CURRENT* Certificate of Achievement
Mobile Application Development *CURRENT* Certificate of Proficiency
Software Development *FUTURE* Certificate of Achievement
Software Development *FUTURE* AA Degree