Lecture Contact Hours: 48-54; Homework Hours: 96-108; Total Student Learning Hours: 144-162

CUYAMACA COLLEGE
COURSE OUTLINE OF RECORD

COMPUTER SCIENCE 119 – PROGRAM DESIGN AND DEVELOPMENT

3 hours lecture, 3 units

Catalog Description
Introductory course in program design and development using Java or other object-oriented programming language to serve as a foundation for more advanced programming, computer science or networking courses. Emphasizes the development of problem-solving skills while introducing students to computer science through the use of a modern object-oriented programming language. Devotes attention to the development of effective software engineering practices emphasizing such principles as design decomposition, encapsulation, procedural abstraction, testing and software reuse. Students will learn and apply standard programming constructs, problem-solving strategies, the concept of an algorithm, fundamental data structures, the machine representation of data, and introductory graphics and networking.

Prerequisite
None

Corequisite
CS 119L

Recommended Preparation
"C" grade or higher or "Pass" in CIS 110 or equivalent

Entrance Skills
Without the following skills, competencies and/or knowledge, students entering this course will be highly unlikely to succeed:
1) Demonstrate a general understanding of the components in a computer hardware system:
   a. Distinguish between the different sizes/types of computers.
   b. Identify the components of a computer: input, process, output, storage.
   c. Explain how data are represented within a computer.
   d. Identify typical input, output and storage units.
2) Knowledge of the facts about computer software:
   a. Explain the difference between systems and applications software.
   b. Demonstrate the use and functions of an operating system.
   c. Explain the data structures within the computer (field, file, etc.).
   d. Explain systems analysis methods and implementation.
3) Manage files and folders in a Windows operating system.
4) General experience and proficiency using word processing or spreadsheet software.

Course Content
1) History of computing
2) Basic computability
3) Machine level representation of data
4) Assembly level machine organization
5) Software tools and environments
6) Software requirements and specifications
7) Software design (object-oriented and procedural)

8) Software validation
9) Overview of programming languages
10) Object-oriented programming (data hiding, encapsulation, inheritance, polymorphism, design and implementation of user-defined classes)
11) Abstraction mechanisms
12) Fundamental programming constructs
13) Algorithms and problem solving
14) Fundamental data structures and computing algorithms
15) Declarations and types
16) Fundamental techniques in graphics

Course Objectives

Students will be able to:

1) Use software engineering methodologies and practices to:
   a. Decompose a complex problem into modular components using both procedural and object oriented design practices.
   b. Define and provide examples of procedural abstraction.
   c. Define and provide examples of encapsulation.
2) Produce flowchart and pseudocode solutions to solve introductory level programming problems.
3) Design programs that utilize classes from the standard library (in Java this might include JTextField, DecimalFormat, Math, for example) to solve a variety of programming problems.
4) Design user-defined methods to augment the standard library in solving a variety of programming problems.
5) Discuss the difference between public and private access modifiers and their implications in programs.
6) Distinguish between local variables and class attributes and their accessibility implications.
7) Utilize all three fundamental programming structures–sequence, decision/selection, repetition–in solving programming problems.
8) Define the concept of "inheritance" in a programming perspective and provide examples of where it might be used.
9) Provide examples of "polymorphic" methods.
10) Identify the role and variety of constructor methods in programs.
11) Apply time management strategies critical for success in the business world.
12) Demonstrate critical problem-solving skills in the debugging of programs.
13) Trace program flow and predict outcomes.

Method of Evaluation

A grading system will be established by the instructor and implemented uniformly. Grades will be based on demonstrated proficiency in subject matter determined by multiple measurements for evaluation, one of which must be essay exams, skills demonstration or, where appropriate, the symbol system.

1) Quizzes and exams that measure students' ability to use programming terminology and explain technical concepts related to program design and development.
2) Practical exams requiring students to trace programming code to predict outcome. For example, a 10-line section of code is provided and students must be able to identify how all of the values in all variables are altered by that code and predict the final output as specified in the code.
3) Project requiring students to write one complete application program demonstrating effective use of object-oriented principles and methodology.

Special Materials Required of Student

1) Electronic storage media
2) Access to web-based course material/software

Minimum Instructional Facilities
Computer lab with Internet access including FTP, current web browser, and software to create Java development environment compliant with Sun Microsystems specifications

Method of Instruction
1) Lecture and demonstration
2) Hands-on exercises

Out-of-Class Assignments
Design, code and debug multiple programs and objects that demonstrate class objectives

Texts and References
1) Required (representative example): Farrell, Joyce. *An Object-Oriented Approach to Programming Logic and Design*. 9th edition. Cengage, 2020.
2) Supplemental (optional): Murach, Joel. *Murach's Java Programming*. 5th edition. Murach & Associates, 2017.

Exit Skills
Students having successfully completed this course exit with the following skills, competencies and/or knowledge:
1) Demonstrate knowledge of software engineering methodologies and practices.
2) Identify common tools, functions, classes and utilities available in modern programming languages.
3) Define and utilize structured programming principles.
4) Identify logical "objects" and their "attributes" and "behaviors."
5) Design "behaviors" that follow all structured programming guidelines.
6) Effectively utilize the three basic logic control structures of sequence, decision and repetition.
7) Conceptualize abstraction, polymorphism, inheritance and encapsulation.
8) Manipulate one- and two-dimensional arrays.
9) Contrast procedural vs. object-oriented languages.
10) Trace the flow of execution and values of all variables through a complex object-oriented program.

Student Learning Outcomes
Upon successful completion of this course, students will be able to:
1) Decompose problems and design program solutions using flowcharts, pseudocode, models, or other tools.
2) Properly code applications using the fundamental coding structures: sequence, selection, and loops.