Laboratory Contact Hours: 48-54; Homework Hours: 0; Total Student Learning Hours: 48-54

CUYAMACA COLLEGE
COURSE OUTLINE OF RECORD

COMPUTER SCIENCE 119L – PROGRAM DESIGN AND DEVELOPMENT LAB

3 hours laboratory, 1 unit

Catalog Description
Laboratory tutorials, drills and programming problems designed to help students master the concepts and programming projects presented/assigned in CS 119.

Prerequisite
None

Corequisite
CS 119

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
Mirrors the course content of CS 119; exercises and programs will be developed to supplement those content areas.

Course Objectives
Students will be able to:
1) Demonstrate practical applications of software engineering methodologies and practices as outlined below:
   a. Decompose a complex problem into modular components using both procedural and object-oriented design practices
   b. Effectively invoke user-defined and standard library methods with sole communications between different methods within the program handled through parameters and return values (encapsulation)
   c. Effectively utilize methods from within the standard library, for example, Math.sqrt(), without knowledge of how they work (procedural abstraction)

   d. Demonstrate a variety of debugging techniques, including diagnostic output statements, method stubs, setting breakpoints and tracing program execution
   e. Generate a wide variety of test data as input to programs such that all reasonable testing is done prior to submission of programs
2) Write and test programs that utilize classes from the standard library to solve a variety of programming problems
3) Write and test user-defined methods to augment the standard library in solving a variety of programming problems
4) Write and test user-defined classes to augment the standard library in solving a variety of programming problems
5) Write programs that use structured exception handling
6) Write programs with a simple Graphical User Interface (GUI) using standard library GUI components
7) Utilize public and private access modifiers with recognition of their implications in programs
8) Utilize both local variables and class attributes with recognition of their accessibility implications
9) Utilize all three fundamental programming structures–sequence, decision/selection, repetition–in solving programming problems
10) Provide examples of "polymorphic" methods
11) Write multiple constructor methods in programs
12) Demonstrate writing skills in effective commenting of programming code
13) Apply time management strategies critical for success in the business world
14) Demonstrate critical problem-solving skills in the debugging of programs
15) 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) Exercises that demonstrate effective problem solving and generation of programming code to solve a variety of programming concepts, including all three control structures (sequence, decision, and iteration), use of library classes, inheritance, method creation with appropriate parameters.
2) Project requiring students to write one complete application program demonstrating effective use of object-oriented principles and methodologies.

Special Materials Required of Student
1) Electronic storage media
2) Access to web-based course material/software
3) Java program development environment if students wish to work on programming exercises outside of the laboratory; suitable Java development environments (i.e., NetBeans) are available at no cost

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) 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) Test and debug applications using debugging tools such as trace execution.