Curriculum Committee Approval: 02/20/2024

Lecture Contact Hours: 48-54; Homework Hours: 96-108;
Laboratory Contact Hours: 48-54; Homework Hours: 0;
Total Student Learning Hours: 192-216

CUYAMACA COLLEGE
COURSE OUTLINE OF RECORD

COMPUTER SCIENCE 182 – INTRODUCTION TO JAVA PROGRAMMING

3 hours lecture, 3 hours laboratory, 4 units

Catalog Description
Introductory course in the basics of the Java programming language focusing on object oriented methodology. Topics include classes, methods, parameters, arrays, modularity, abstraction, exception handling, and stream and file I/O. In addition to writing and using new classes, students will utilize the AWT and/or Swing libraries of classes. Basic inheritance and mobile application programming are introduced.

Prerequisite
Appropriate placement or intermediate algebra

Recommended Preparation
"C" grade or higher or "Pass" in CS 119 or equivalent or experience programming in C++ or Java

Entrance Skills
Without the following skills, competencies and/or knowledge, students entering this course will be highly unlikely to succeed:
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 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.
11) Identify the properties of integers including commutativity, associativity, identities and inverses.
12) Solve polynomial equations and systems of linear equations.
13) Find the general terms of sequences.
14) Understand simple mathematical proofs.

Course Content
1) Review and Practice
    a. Java programming environments
    b. Operators
    c. Data types, strings and structures
    d. Decisions
    e. Loops
2) Multidimensional Arrays
3) Classes and Methods
    a. Information Hiding and Encapsulation
    b. Overloading
    c. Constructors

   d. Packages
   e. Inheritance
   f. Dynamic Binding and Polymorphism
4) Sorting and Searching
5) Exception Handling
6) Stream and File I/O
7) Window Interfaces
8) Applets and HTML
9) Mobile Applications

Course Objectives

Given a programming project scenario, students will be able to:

1) Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today.
2) Write well-designed, modularized Java programs to solve a variety of scientific problems as applets or standalone programs.
3) Select appropriate storage class characteristics for variables, objects and methods and justify the reason(s) for the selection.
4) Solve a variety of programming problems using sound structured, top-down, object-oriented design principles.
5) Use inheritance to minimize reinvention of similar objects.
6) Access standard Java library modules.
7) Explain the difference between (and implications of) reference vs. primitive data types.
8) Design, create and utilize complex classes and their associated objects.
9) Input and output data to and from standard devices and files.
10) Visualize array subscripting as memory offset calculations and explain some of the implications. Represent this visualization in drawings.
11) Utilize fundamental program control structures (sequence, decision, and repetition).
12) Utilize the try-throw-catch approach to exception handling.
13) Code efficient sort and search algorithms.
14) Identify the scope of a Java variable in a program.
15) Demonstrate in a program the use of a method that returns an array.
16) Trace program flow and predict outcomes.
17) Program mobile applications

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 development and implementation in Java.
2) Practical exams that measure students' ability to trace programming code to predict outcome.
3) Projects that measure students' ability to write complete application programs demonstrating effective use of object-oriented principles, methodology and concepts.

Special Materials Required of Student

USB flash drive

Minimum Instructional Facilities

Computer lab with network computers and software to create Java development environment

Method of Instruction

1) Lecture and demonstration
2) Hands-on practice
3) Lab problems

Out-of-Class Assignments

1) Design, code and debug multiple Java programs and objects that demonstrate class objectives
2) Analyze instructor-assigned pre-coded Java programs, post analysis comments on the class discussion board

Texts and References

1) Required (representative examples):
   a. Savitch, Walter. *Java, An Introduction to Problem Solving and Programming*. 8th edition. Pearson, 2021.
   b. Savitch, Walter. *Absolute Java*. 6th edition. Pearson, 2016.
2) Supplemental: None

Exit Skills

Students having successfully completed this course exit with the following skills, competencies and/or knowledge:

1) Write well-designed Java code to solve a variety of scientific problems as applets or standalone programs.
2) Select appropriate storage class characteristics for variables as well as methods.
3) Solve a variety of programming problems using sound structured, top-down, object-oriented design principles.
4) Effectively use inheritance to minimize reinvention of similar objects.
5) Effectively access standard Java library modules.
6) Recognize the difference between (and implications of) reference vs. primitive data types.
7) Design, create and utilize complex classes and objects.
8) Input and output data to and from standard devices and files.
9) Visualize array subscripting as memory offset calculations and recognize some of the implications.
10) Describe and effectively utilize fundamental program control structures.
11) Utilize the try-throw-catch approach to exception handling.
12) Describe and analyze the relative efficiency of the various sort and search algorithms.
13) Untangle poorly designed code.
14) Define the requirements for and program a functional mobile application.
15) Utilize basic database SQL commands to access and retrieve data.

Student Learning Outcomes:

Upon successful completion of this course and given a scientific problem-based scenario, 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.
3) Test and debug applications using debugging tools such as trace execution.