College of San Mateo
Official Course Outline

1. **COURSE ID:** CIS 254 **TITLE:** Introduction to Object-Oriented Program Design **C-ID:** COMP 112
   Units: 4.0 units **Hours/Semester:** 48.0-54.0 Lecture hours; 48.0-54.0 Lab hours; and 96.0-108.0 Homework hours
   Method of Grading: Grade Option (Letter Grade or Pass/No Pass)
   Recommended Preparation:
   Eligibility for MATH 120

2. **COURSE DESIGNATION:**
   Degree Credit
   Transfer credit: CSU; UC
   AA/AS Degree Requirements:
   CSM - GENERAL EDUCATION REQUIREMENTS: E2b. Communication and Analytical Thinking

3. **COURSE DESCRIPTIONS:**
   Catalog Description:
   Introduction to object-oriented computer programming for computer science majors and computer professionals. Includes simple data types, control structures, an introduction to array and string data structures and algorithms, debugging techniques, history of computer science, systems and environments, and the social implications of computing. Emphasizes object-oriented design, good software engineering principles and developing fundamental programming skills in a high-level programming language such as Java, C++ or Python, for example. This course conforms to the ACM CSO standards.

4. **STUDENT LEARNING OUTCOME(S) (SLO'S):**
   Upon successful completion of this course, a student will meet the following outcomes:
   1. Analyze and explain the behavior of programs involving the fundamental program constructs
   2. Write short programs that use the fundamental program constructs including standard conditional and iterative control structures
   3. Identify and correct syntax and logic errors in short programs
   4. Write short programs using arrays
   5. Design and implement a class based on attributes and behaviors of objects
   6. Construct objects using a class and activate methods on them
   7. Use static and instance members of a class properly
   8. Identify and describe value, scope and lifetime of a variable.
   9. Describe the parameter passing mechanisms and method overloading.

5. **SPECIFIC INSTRUCTIONAL OBJECTIVES:**
   Upon successful completion of this course, a student will be able to:
   1. Analyze and explain the behavior of programs involving the fundamental program constructs
   2. Write short programs that use the fundamental program constructs including standard conditional and iterative control structures
   3. Identify and correct syntax and logic errors in short programs
   4. Write short programs that use arrays
   5. Design and implement a class based on attributes and behaviors of objects
   6. Construct objects using a class and activate methods on them
   7. Use static and instance members of a class properly
   8. Identify and describe value, scope and lifetime of a variable
   9. Describe the parameter passing mechanisms and method overloading

6. **COURSE CONTENT:**
   Lecture Content:
   1. Introduction to the history of computer science and programming languages
   2. Ethics and responsibility of computer professionals
   3. Introduction to computer environments and language translation
      A. Comparison of interpreters and compilers
      B. Portability
   4. Introduction to object-oriented paradigm

    A. Abstraction
    B. Objects
    C. Classes
    D. Methods
    E. Parameter passing
    F. Encapsulation
    G. Inheritance
    H. Polymorphism
5. Fundamental programming constructs
    A. Basic syntax and semantics of a higher-level language
        a. data types and variables
        b. arithmetic and Boolean expressions
        c. assignment
    B. Simple I/O (command line and simple dialog box)
    C. Conditional and iterative control structures
    D. Class Types and Object Creation and Usage
6. Fundamental data structure
    A. Primitive types
    B. Arrays
    C. Strings and String processing
7. Algorithms and problem-solving
    A. Problem-solving strategies
    B. The role of algorithms in the problem-solving process
    C. Implementation strategies for algorithms
    D. Debugging strategies
    E. The concept and properties of algorithms

Lab Content:

1. Fundamental programming constructs
    A. Basic syntax and semantics of a higher-level language
        a. data types and variables
        b. arithmetic and Boolean expressions
        c. assignment operators
    B. Simple I/O (command line and simple dialog box)
    C. Conditional and iterative control structures
        a. single and double selection
        b. switch statements
    D. Class types and object creation and usage
    E. Repetition control structures
        a. while statements
        b. do-while statements
        c. for statements
2. Fundamental data structures
    A. Primitive types
    B. Arrays
    C. Strings and string processing
3. Methods
    A. Instance
    B. Static
4. Algorithms and problem-solving
    A. Problem-solving strategies
    B. Algorithm development
    C. Implementation strategies for algorithms
    D. Debugging strategies
5. Documentation and APIs

7. **REPRESENTATIVE METHODS OF INSTRUCTION:**

Typical methods of instruction may include:

A. Lecture
B. Lab
C. Activity

D. Discussion
E. Observation and Demonstration
F. Other (Specify): Lectures, to introduce new topics in design and code syntax; Design development examples, illustrating process and good practice Code development examples on board and live, illustrating process, problem solving techniques and debug strategy. Class participatory problem solving, each person contributing a potential "next step"; Short in-class projects and solution presentation; Q/A sessions where students both ask AND answer the questions; Student groups cooperating to solve significant programming assignments.

8. **REPRESENTATIVE ASSIGNMENTS**
Representative assignments in this course may include, but are not limited to the following:
Writing Assignments:
Textbook exercises and weekly or bi-weekly programming assignments. At least one of the programming assignments is a small-group project to provide experience in a realistic program development environment. Specifically, the intent is to provide an opportunity for students to improve their communication skills and learn to work in a cooperative environment. The primary writing opportunity for students in this course is documentation supporting their programming projects. This includes both technical documentation targeting a peer audience, and user documentation targeting those who will be using the software the student develops. The technical documentation will describe the problem to be solved, the scope of the project, an overview of the solution, and any limitations of the solution. The user documentation will be primarily instructional. The purpose of the written assignments is to help students clarify their ideas and then to articulate them clearly.
Reading Assignments:
Weekly textbook reading assignments.

9. **REPRESENTATIVE METHODS OF EVALUATION**
Representative methods of evaluation may include:
A. Class Participation
B. Class Work
C. Exams/Tests
D. Group Projects
E. Homework
F. Lab Activities
G. Oral Presentation
H. Quizzes
I. Written examination
J. Quizzes to provide feedback to student and teacher on recently presented material; Assessment of student contributions during class discussion and laboratory time; Individual programming assignments; Midterm and Final exams- short answer, general problem solving, short program segments; Assessments of group participation on course projects, including peer-assessment of participation and contribution to the group effort.

10. **REPRESENTATIVE TEXT(S):**
Possible textbooks include:
A. Downey, A. and C. Mayfield. *Think Java: How to Think Like a Computer Scientist*, 2nd ed. ed. O'Reilly Media, 2020
B. Murach. *Murach's C++ Programming*, 1st ed. ed. Murach, 2018
C. B. Hetland. *Beginning Python: From Novice to Professional*, 3rd ed. ed. Apress, 2017

Origination Date: November 2021
Curriculum Committee Approval Date: December 2021
Effective Term: Fall 2022
Course Originator: Mounjed Moussalem