Title (Units): **COMP2026 Problem Solving Using Object Oriented Programming (4,3,3)**

Course Aims: To study the object-oriented programming principles and techniques. Upon completion, students should be able to use an object-oriented language to develop computer programs for problem solving.

Prerequisite: COMP 1005 Essence of Computing or COMP1007 Introduction to Python and Its Applications

Course Intended Learning Outcomes (CILOs):
Upon successful completion of this course, students should be able to:

No.	Course Intended Learning Outcomes (CILOs)
	Knowledge
1	Describe the elements and principles of object-oriented programming
2	Apply the object-oriented concepts to software design
3	Describe the importance of programming styles, implementation and testing
	Professional Skill
4	Design, develop and test object-oriented computer programs
5	Formulate problems as steps so as to be solved systematically
	Attitude
6	Integrate robustness, reusability, and portability into software development

Calendar Description: This course introduces the object-oriented programming concepts, principles, and techniques, including classes, objects, inheritance, and polymorphism. All these concepts are illustrated via a contemporary object-oriented programming language.

Teaching and Learning Activities (TLAs):

CILOs	Type of TLA
1, 3, 5	Students will learn the elements of an object-oriented programming language and the object-oriented principles via lectures and tutorials.
1-6	Tutorials and machine problems are designed for students to incorporate object-oriented techniques into their programs.

Assessment:

No.	Assessment Methods	Weighting	CILOs to be addressed	Description of Assessment Tasks
1	Continuous Assessment	60%	2-6	Continuous assessments are designed to measure how well the students have learned the fundamentals and major concepts of object-oriented programming. A number of machine problems will be given to students to train them to design programs via the object-oriented approach. Tests will be used to test their programming capabilities.
2	Examination	40%	1-6	Final examination questions are designed to see how far students have achieved their intended learning outcomes. Questions will primarily be concepts and skills based to assess the student's ability in object-oriented programming.

Assessment Rubrics:

	Excellent (A)	Good (B)	Satisfactory (C)	Marginal Pass (D)	Fail (F)
Principles of object-oriented programming	The student acquires excellent knowledge in the principles of object-oriented languages, namely, data encapsulation, inheritance, and polymorphism.	The student acquires sufficient knowledge in the principles of object-oriented languages, namely, data encapsulation, inheritance, and polymorphism.	The student acquires average knowledge in the principles of object-oriented languages, namely, data encapsulation, inheritance, and polymorphism.	The student is able to describe the meanings of data encapsulation, inheritance, and polymorphism, and to give simple examples on them.	The student is unable to describe the meanings of data encapsulation, inheritance, and polymorphism, and to give simple examples on them.
Applying object-oriented techniques to software packages	The student is able to extensively apply object-oriented techniques to write software applications with multiple classes, e.g., enforcing data hiding as much as possible via class privacy.	The student is able to sufficiently apply object-oriented techniques to write software applications with multiple classes, e.g., enforcing data hiding via class privacy.	The student is able to apply object-oriented techniques in some key elements of software applications with multiple classes, e.g., enforcing data hiding via class privacy.	The student can apply some object-oriented techniques to write software applications with multiple classes, e.g., enforcing data hiding via class privacy.	The student cannot apply object-oriented techniques to write software applications with multiple classes, e.g., enforcing data hiding via class privacy.
Design and implement object-oriented software for problem solving	The student demonstrates a strong ability in designing and implementing programs to solve moderately complex problems.	The student demonstrates a considerable ability in designing and implementing programs to solve moderately complex problems.	The student demonstrates an average ability in designing and implementing programs to solve moderately complex problems.	The student demonstrates some ability in designing and implementing programs to solve moderately complex problems.	The student does not demonstrate any ability in designing and implementing programs to solve moderately complex problems.
Exception handling	The student correctly writes object-oriented programs with complicated exception handling facilities.	The student correctly writes object-oriented programs with considerable exception handling facilities.	The student correctly writes object-oriented programs with an average amount of exception handling facilities.	The student correctly writes object-oriented programs with some exception handling facilities.	The student cannot write object-oriented programs with any exception handling facilities.

Course Content and CILOs Mapping:

Content		CILO No.
I	Object-oriented Programming: Basic Elements	2-6
II	Object-Oriented Programming: Advanced Concepts	1-2, 5-6
III	Exception Handling and Advanced Features	3, 5-6

References:

- C. S. Horstmann and G. Cornell, Core Java 2 (Volume I-Fundamentals), Prentice Hall, 9th Edition, 2012.
- H. M. Deitel and P. J. Deitel, Java How to Program, Prentice Hall, 9th Edition, 2012.
- A. Kak, Programming with Objects: A Comparative Presentation of Object Oriented Programming with C++ and Java, Wiley-IEEE Press, 2003.
- D. Liang, Introduction to Java Programming, Prentice Hall, 9th Edition, 2014.

- G. Booch, R. A. Maksimchuk, M. W. Engel, and B J. Young, Object-oriented Analysis and Design with Applications, Addison-Wesley, 3rd Edition, 2007.

Course Content:

Topic

I. Object-oriented Programming: Basic Elements
   A. Programming methodologies (Design, Flowchart, Pseudo code)
   B. Lexical elements, data types, operators and expressions
   C. Control structures
   D. Classes and objects
   E. Methods
   F. Classification, generalization and specialization
   G. Constructs of an OOP language
   H. Problem solving

II. Object-Oriented Programming: Advanced Concepts
   A. Inheritance
   B. Interfaces and abstract classes
   C. Polymorphism
   D. Modularity

III. Exception Handling and Advanced Features
   A. Exception handling
   B. Advanced features