Pokhara University
Faculty of Management Studies

Course Code: CMP 172 | Full marks: 100
Course title: **Object Oriented Analysis and Design** | Pass marks: 45
Nature of the course: Theory + Practical | Credit Hrs: 3
Year: Second | Total periods: 48 hours
Level: Bachelor | Program: BCSIT
Semester: III

1. Course Description

This course introduces the fundamental concepts of object-oriented (OO) analysis and design (OOD) for software development. Students will gain an understanding of the OO paradigm, learn to identify objects and their relationships, and create models using the Unified Modeling Language (UML).

It will include the following information about the course:

- Understand the fundamental concepts of OOP, including objects, classes, attributes, methods, benefits, and challenges.
- Gain an introduction to UML, its building blocks (things, relationships, diagrams), and its role in object-oriented modeling
- Learn the object-oriented analysis process, including use case modeling, building the conceptual model (identifying classes and relationships), and using CRC cards
- Master the transition from analysis to design, designing classes and interfaces, operations (methods), system behavior using state diagrams, and common design patterns.
- Explore design patterns and their benefits, understand how to apply common design patterns, and gain an introduction to object-oriented frameworks.
- Apply OOAD principles to a real-world case study, develop a UML model for the chosen scenario, and present your findings.

2. General Objectives

1. Explain the core principles of object-oriented programming.
2. Apply object-oriented analysis techniques to understand system requirements.
3. Design robust and maintainable object-oriented software systems using UML.
4. Utilize design patterns to solve common software design problems.

3. Method of Instructions

General Instructional Technique: Lecture, Discussion, Readings, Question Answer
Specific Instructional Technique: Practical works, Project Based Learning, Self-Directed Learning, Industry Insights and Case Study

4. Content in Detail with Specific Objectives

Specific Objectives	Content
▪ Understand the concepts of class, objects, attributes, and methods. ▪ Describe and differentiate between association, aggregation, and composition relationships with	**Unit 1: Introduction to Object-Oriented [4 Hrs.]** 1.1. Class, objects, attributes, methods. 1.2. Relation between objects**:** Association, aggregation, and composition

examples. ▪ Identify the benefits and challenges of Object-Oriented Programming (OOP).	relationships with examples. 1.3. Benefits and challenges of OOP
Specific Objectives	**Content**
▪ Understand the fundamentals of requirement gathering. ▪ Identify and apply requirement elicitation techniques. ▪ Perform requirement analysis and specification. ▪ Validate requirements. ▪ Understand the concepts of use cases, scenarios, and actors. ▪ Gain an overview of the Unified Modeling Language (UML) fundamentals and notations.	**Unit 2: Requirement Elicitation and Analysis [7 Hrs.]** 2.1 Requirement gathering 2.1.1 Requirement fundamentals 2.1.2 Requirement elicitation techniques 2.1.3 Requirement analysis 2.1.4 Requirement specification 2.1.5 Requirement validation 2.4 Use cases, Scenario and Actors 2.5 Overview of the Unified Modeling Language: UML Fundamentals and Notations 2.5.1 Behavior Diagram: Use case Diagram and Activity Diagram 2.5.2 Interaction Diagram: Sequence and communication Diagram 2.5.3 Structure Diagram: Class, Package, and Component diagram
Specific Objectives	**Content**
▪ Perform domain modeling and concept modeling, including identifying relationships. ▪ Identify classes, attributes, and methods. ▪ Model behavior using state diagrams and activity diagrams.	**Unit 3: Object oriented analysis [6 Hrs.]** 3.1 Domain modeling and concept modeling relationships 3.2 Identifying classes, attributes, methods 3.3 Behavior modeling using state diagrams and activity diagrams
Specific Objectives	**Content**
▪ Elaborate use cases and create use case diagrams. ▪ Understand system interaction diagrams: Sequence and communication diagrams. ▪ Develop logical architecture and UML package diagrams. ▪ Create class diagrams, including generalization, dependency, constraints, composition over aggregation, singleton class, and template class.	**Unit 4: Object-Oriented Modeling Using UML Notation [10 Hrs.]** 4.1 Elaborating use cases and Use case diagram 4.1.1 Use cases for reuse 4.1.2 Use cases and components 4.1.3 Separation variant behavior 4.1.4 Generalization 4.1.5 Actor as classes 4.1.2 System interaction diagram: Sequence and communication diagram 4.2 Logical architecture and UML package diagram 4.2.1 Model-View separation principal

	4.3 Class diagram 4.3.1 Generalization 4.3.2 Dependency 4.3.3 Constraints 4.3.4 Composition over Aggregation 4.3.5 Singleton class 4.3.6 Template class
	Content
▪ Understand General Responsibility Assignment Software Patterns (GRASP) principles. ▪ Design for visibility. ▪ Understand and apply SOLID principles	**Unit 5: Object Oriented Design principles [6 Hrs.]** 5.1 General Responsibility Assignment Software Patterns (GRASP) 5.1.1 Creator 5.1.2 Information Expert 5.1.3 Controller 5.1.4 Low Coupling 5.1.5 High Cohesion 5.2 Designing for visibility 5.3 SOLID principal 5.3.1 Single Responsibility Principle 5.3.2 Open Closed Principal 5.3.3 Liskov Substitution Principle (LSP) 5.3.4 Interface Segregation Principle (ISP) 5.3.5 Dependency Inversion Principle (DIP)
Specific Objectives	**Content**
▪ Understand and apply Design patterns	**Unit 6: Applying GOF Design Patterns [8 Hrs.]** 6.1 Adaptor 6.2 Factory 6.3 Singleton 6.4 Strategy 6.5 Composite 6.6 Façade 6.7 Observer/Publish-Subscribe/Delegation Event Model
Specific Objectives	**Content**
▪ Apply Object-Oriented Analysis and Design (OOAD) principles to a real-world case study. ▪ To implement the object oriented analysis and design in real world cases	**Unit 7: Case Study and Project [4 Hrs.]** 7.1 Apply OOAD principles to a real-world case study 7.2 Develop a UML model for the chosen case study 7.3 Class presentations and discussions
Specific Objectives	**Content**

5. Laboratory Work

It builds the foundation for analysis of information system and software engineering.

1. Implementing class, objects, attributes, and methods in a simple Java program.
2. Exploring different types of relationships (association, aggregation, composition) between objects through coding examples.
3. Conducting requirement elicitation and analysis for a small software project, using techniques such as interviews and surveys.
4. Creating use case diagrams and specifying requirements for a given scenario using UML notation.
5. Modeling domain concepts and relationships using UML class diagrams.
6. Developing behavior models using state diagrams and activity diagrams for a simple system.
7. Elaborating use cases and system interactions through sequence and communication diagrams.
8. Designing logical architecture and package diagrams for a software system.
9. Implementing SOLID principles in a programming project and analyzing their impact on code quality.
10. Implementing and applying various design patterns (e.g., Factory, Singleton, Observer) in a software project, assessing their effectiveness in addressing design challenges.

Note:

1. ***Motivate students to create small project work integrating all of the above concepts***.
2. ***Each of the above lab session should cover more than 4 hours of practical work.***

6. Evaluation system and Student's Responsibilities

Evaluation System

In addition to the formal exam(s), the internal evaluation of a student may consist of quizzes, assignments, lab reports, projects, class participation, etc. The tabular presentation of the internal evaluation is as follows :

<table>
<tr><th>External Evaluation</th><th>Marks</th><th>Internal Evaluation</th><th colspan="2">Marks</th></tr>
<tr><td rowspan="6">Semester End Examination</td><td></td><td>Theory</td><td></td><td>40</td></tr>
<tr><td rowspan="5">40</td><td>Practical</td><td></td><td rowspan="5">20</td></tr>
<tr><td>Attendance & Class Participation</td><td>5</td></tr>
<tr><td>Lab Report/Project Report</td><td>5</td></tr>
<tr><td>Practical Exam/Project Work</td><td>5</td></tr>
<tr><td>Viva</td><td>5</td></tr>
<tr><td>Total External</td><td>40</td><td>Total Internal</td><td></td><td>60</td></tr>
<tr><td colspan="5">Full Marks: 40 + 40+ 20 = 100</td></tr>
</table>

Student's Requirements

Each student must secure at least 45% marks separately in both internal assessment and practical evaluation with a minimum of 80% attendance in the class in order to appear in the Semester End Examination. Failing to get such score will be given NOT QUALIFIED (NQ) to appear the Semester-End Examinations. Students are advised to attend all the classes, formal exam, test, etc. and complete all the assignments within the specified time period. ***Students are required to complete all the requirements defined for the completion of the course.***

7. Prescribed Books and References

Prescribed Text Books :

1. Larman, C., Applying UML and Patterns, Pearson Education Asia, 2008.

Reference Books:

2. Stevens, P, Pooley, R., Using UML: Software Engineering with Objects and Components, Addision-Wesley, 2009.
3. Fowler, M., Scott, K., UML Distilled: Applying the Standard Object Modeling Language, Addison-Wesley, 2007.
4. Booch, G., Jacobson, I., Rumbaugh, J., The Unified Software Development Process, Addison-Wesely, 2009.
5. Booch, G., Jacobson, I., Rumbaugh, J., The Unified Modeling Language User Guide, Addison-Wesely, 2008.
6. Jacobson I., Object-Oriented Software Engineering – A Use Case Driven Approach, Addison-Wesely, 2009.