**Course Title:**

**Object-Oriented Analysis and Design**

**Course Code: CCS 2212**

**Course Description:**

This course introduces students to the principles and practices of Object-Oriented Analysis and Design (OOAD). Through this course, students will learn how to model, design, and develop robust, scalable, and maintainable software systems using object-oriented approaches. The course emphasizes the use of Unified Modeling Language (UML) for system design, and covers design patterns, software architecture, and principles of object-oriented programming.

Key areas of focus include object-oriented design principles such as encapsulation, inheritance, and polymorphism, as well as the importance of design patterns in solving common design problems. Students will also engage in real-world case studies and practical applications that involve analyzing requirements, developing UML diagrams, and implementing object-oriented systems.

By the end of the course, students will be equipped with the skills to design high-quality software using object-oriented approaches and modern software engineering practices.

**Course Objectives:**

By the end of this course, students will be able to:

1. Understand the fundamental concepts of Object-Oriented Programming (OOP) and its significance in software development.
2. Apply Object-Oriented Analysis and Design techniques to solve real-world software problems.
3. Utilize UML for modeling and designing software systems.
4. Implement design patterns to enhance code reuse and maintainability.
5. Apply principles of software architecture, including coupling and cohesion, to ensure system scalability and maintainability.

**Course Outline:**

**Week 1: Introduction to Object-Oriented Concepts**

* Overview of Object-Oriented Programming (OOP) principles.
* Difference between Object-Oriented Programming and Procedural Programming.
* Key concepts: classes, objects, methods, attributes.
* Introduction to Encapsulation, Inheritance, and Polymorphism.

**Week 2: Object-Oriented Analysis**

* Introduction to Object-Oriented Analysis (OOA).
* Understanding system requirements and user stories.
* Identifying actors, use cases, and relationships.
* Use case modeling and system boundaries.

**Week 3: Unified Modeling Language (UML)**

* Overview of UML diagrams.
* Use case diagrams, class diagrams, sequence diagrams, and activity diagrams.
* Practical: Drawing UML diagrams for a sample system (e.g., a library management system).

**Week 4: Classes and Objects in OOP**

* Detailed discussion on classes and objects.
* Constructors, destructors, and class members (attributes and methods).
* Access control: public, private, and protected members.
* Example project: Building a basic class-based system.

**Week 5: Advanced UML Diagrams**

* Sequence diagrams and interaction modeling.
* Collaboration diagrams vs. sequence diagrams.
* State diagrams and modeling object behavior.
* Practical: Design and implement an online shopping system’s UML diagrams.

**Week 6: Inheritance and Polymorphism**

* Inheritance: extending classes and overriding methods.
* Polymorphism: method overloading and overriding.
* Practical examples of inheritance and polymorphism in software design.

**Week 7: Design Patterns Overview**

* Introduction to design patterns.
* Types of design patterns: creational, structural, and behavioral patterns.
* Factory method, Singleton, and Observer pattern in detail.
* Practical: Implementing basic design patterns in a project.

**Week 8: Architectural Design Principles**

* Understanding the importance of software architecture.
* Coupling and cohesion: minimizing dependencies and ensuring modularity.
* Layered architecture and client-server architecture.
* Practical: Designing the architecture for a web-based application.

**Week 9: Separation of Concerns & SOLID Principles**

* Concept of Separation of Concerns in object-oriented design.
* Introduction to SOLID principles and their importance in software design.
* Applying SOLID principles to improve code quality.
* Practical: Refactoring code to align with SOLID principles.

**Week 10: Aggregation, Composition, and Association**

* Understanding relationships between objects.
* Differences between aggregation, composition, and association.
* Designing systems with appropriate object relationships.
* Practical: Developing a Student-Course registration system.

**Week 11: Software Design Case Study**

* Detailed case study of a real-world software system.
* Analyzing system requirements and constraints.
* Designing the solution using object-oriented principles and UML diagrams.
* Presentations and group discussions.

**Week 12: Object-Oriented Design Best Practices**

* Best practices in OOAD.
* Avoiding common design pitfalls.
* Ensuring system maintainability and scalability.
* Review of key course concepts.

**Week 13: Project Work**

* Students work in teams to design and implement an object-oriented software system.
* Complete documentation of the design process, including UML diagrams, class relationships, and code implementation.
* Peer review and project presentations.

**Week 14: Final Review and Examination**

* Review of major topics and concepts covered in the course.
* Final examination preparation.
* Practical assessment of project work.

**Assessment Criteria:**

* **Practical Labs and Case Studies**: 10%
* **CAT Exam**: 10%
* **Final Project**: 10%
* **Final Exam**: 70%

**Recommended Texts & Resources:**

* **"Object-Oriented Analysis and Design with Applications"** by Grady Booch, Robert A. Maksimchuk, Michael W. Engel.
* **"Design Patterns: Elements of Reusable Object-Oriented Software"** by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.
* **"Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development"** by Craig Larman.

Prepared by Nick Gachui, BSc. Computer Science (JKUAT), MSc. Software Engineering (JKUAT)Top of Form