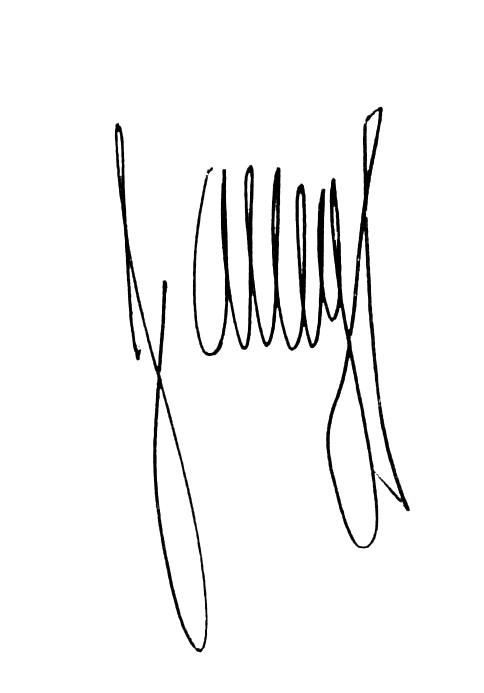
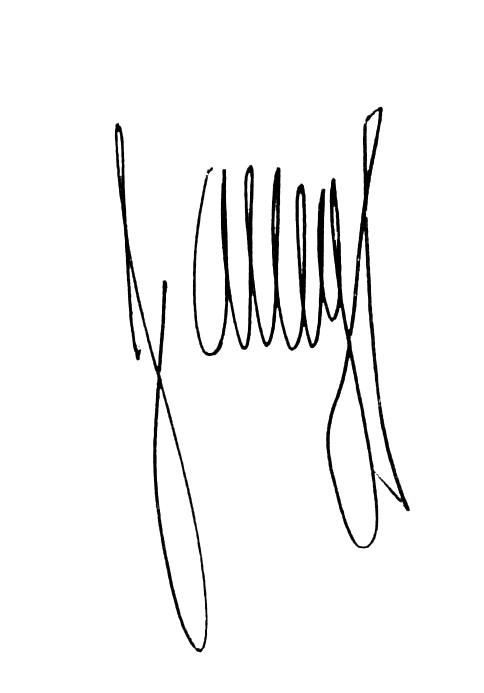
| **COURSE INFORMATION** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Course Code** | **ITP03** | | | | | |
| **Course Title** | **Object-Oriented Programming** | | | | | |
| **Term Offered** | **1st Semester** | | | | | |
| **Course Credit** | **3 units** | | | | | |
| **Class hours per Course Type** | **Lecture Laboratory 90hrs Lecture with laboratory** | | | | | |
| **Pre-Requisite** | **ITC03** | | | | | |
| **Co-requisite** | **NONE** | | | | | |
| **University Philosophy, Mission, Vision, Objectives, School Vision and Mission** | | | | | | |
| **University Philosophy** | The institution believes that through education, man’s God given gifts are discovered and developed for his personal fulfillment and community uplift. | | | | | |
| **University Vision** | It envisions to create a community responsive to the challenges of the changing world. | | | | | |
| **University Mission** | It is tasked to prepare individuals with the best that education can offer in a manner that is consistent with the needs of the society. | | | | | |
| **University Objectives** | To inculcate critical thinking.  To provide competent human resources in various fields.  To uphold discipline, justice and equality; and  To improve man’s quality of life through research and community services. | | | | | |
| **School Vision** | To build a community of technology innovators responsive to the challenges of the computing world. | | | | | |
| **School Mission** | To prepare computer professionals with the best computing science education adaptive to the global environment. | | | | | |
|  | **Institutional, Program and Course Outcomes** | | | | | |
| **Institutional Outcomes** | The Institutional Outcomes align to the course are (is):   1. Meet the standards of both local and global markets. 2. Innovate and contribute to technological advancement and societal development. 3. Exercise leadership grounded on competence and commitment to excellence. | | | | | |
| **Program Outcomes** | By the time of graduation, the students shall be able to:   1. Apply knowledge of computing, science, and mathematics appropriate to the discipline. (IT01) 2. Understand best practices and standards and their applications. (IT02) 3. Design, implement, and evaluate computer-based systems, processes, components, or programs to meet desired needs and requirement under various constraints. (IT05) 4. Integrate IT-based solutions into the user environment effectively. (IT06) 5. Apply knowledge through the use of current techniques, skills, tools and practices necessary for the IT profession. (IT07) 6. Assist in the creation of an effective IT project plan. (IT09) 7. Communicate effectively with the computer community and with society at large about complex computing activities through logical writing, presentations, and clear instructions. (IT10) 8. Function effectively as a member or leader of a development team recognizing the different roles within a team to accomplish a common 9. Meet the standards of both local and global markets. (IO1) 10. Innovate and contribute to technological advancement and societal development. (IO2) 11. Exercise leadership grounded on competence and commitment to excellence. (IO3) | | | | | |
| **Course Description** | This course covers Object-Oriented Programming approach to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts on Object, Class, Inheritance, polymorphism, Abstraction, and Encapsulation. It also covers other terms which are used in Object-Oriented design such as Coupling, Cohesion, Association, Aggregation, and Composition. Along with Exception handling and File input and output processing. | | | | | |
| **Course Outcomes (CO)** | At the end of the course, the students will be able to:   1. Explain core OOP principles in PHP, including classes, objects, and methods. Understand inheritance, polymorphism, abstraction, and interfaces to create modular code. 2. Apply encapsulation, composition, and exception handling in PHP. Utilize namespaces, autoloading, traits, and reflection to enhance code structure. 3. Implement advanced design patterns and optimize PHP code for performance. Integrate these concepts into practical projects. 4. Conduct unit testing, apply debugging techniques, and present a project that demonstrates the effective use of OOP principles and design patterns. | | | | | |
| **Course Learning Plan** | | | | | | |
| Course Outcomes Number | Intended Learning Outcomes | Topic | Time Allotment | Learning Activities | Learning Materials and Platform | Learning Assessment |
|  | **PRELIM PERIOD** | | | | | |
|  | At the end of the lesson/learning experience, the students will be able to**:** |  |  |  |  |  |
| **[1]** | * Observe the classroom Policy such as classroom requirement, grading system and consultation hours, Discuss the University PVMO, specific program objectives. * Explain the key principles of Object-Oriented Programming (OOP) and its importance in software development. * Understand the basic syntax of PHP as it applies to OOP. * Define and instantiate classes and objects in PHP. * Describe and use properties and methods within PHP classes to manage data and behavior. * Explain the purpose and use of access modifiers (public, protected, and private) in PHP classes to control access to properties and methods. * Understand the role of constructors and destructors in initializing and cleaning up objects, and how to implement them in PHP. * Demonstrate method overloading and its application within PHP classes to create more versatile methods. * Understand inheritance by explaining the roles of base and derived classes and their relationship. * Demonstrate method overriding to modify or extend the behavior of inherited methods in derived classes. * Explain and implement polymorphism to enable objects of different classes to be treated as objects of a common base class. * Understand abstract classes and methods by explaining their purpose and how they differ from concrete classes. * Implement interfaces to define a contract for classes, showing how different classes can fulfill the same interface requirements. * Compare abstract classes and interfaces to highlight their differences and appropriate use cases in design. | * Orientation on Classroom Policies and Management and Faculty Consultation hours; Discussion on Grading Policy; University PMVO and the Specific Program Objectives; and EOMS Policy   **Week 1: Introduction to OOP Concepts**   * Overview of Object-Oriented Programming * Basic PHP Syntax for OOP * Classes and Objects: Definition and Instantiation * Properties and Methods   **Week 2: Access Modifiers and Constructors**   * Access Modifiers (public, protected, private) * Constructor and Destructor Methods * Method Overloading   **Week 3: Inheritance and Polymorphism**   * Understanding Inheritance: Base and Derived Classes * Method Overriding * Polymorphism: Concept and Implementation * Using parent:: to Call Parent Methods   **Week 4: Abstraction and Interfaces**   * Abstract Classes and Methods * Interfaces and Their Implementation * Abstract Classes vs. Interfaces * Practical Applications and Examples | **1 Hrs.**  **2 Hrs.**  **3 Hrs.**  **2 Hrs.**  **3 Hrs.**  **2 Hrs.**  **3 Hrs.**  **2 Hrs.**  **3 Hrs.** | * Class discussion on Classroom Policies and Management and Faculty Consultation hours. * Class discussion on the College PVMO, specific program objectives and classroom policies.   Class discussion on:   * Overview of Object-Oriented Programming (OOP) * Basic PHP Syntax for OOP * Classes and Objects: Definition and Instantiation * Properties and Methods   Hands on (Lab Activities)   * Understand the basics of defining and instantiating a class. * Learn how to define and use properties and methods in a class. * Explore the use of public, private, and protected access modifiers. * Apply OOP concepts to model a real-world entity.   Class discussion on:   * Access Modifiers (public, protected, private) * Constructor and Destructor Methods * Method Overloading   Hands on (Lab Activities)   * Create an Employee class with public, protected, and private properties. Test property access and visibility. * Build a Product class with a constructor for initialization and a destructor for cleanup. Instantiate and destroy objects. * Develop a BankAccount class with methods for deposits, withdrawals, and balance checks. Use access modifiers and constructors.   Class discussion on:   * Inheritance * Method Overriding * Polymorphism * Using parent:   Hands on (Lab Activities)   * Create a base class and one or more derived * Write a program demonstrating method overriding by extending a base class. * Develop a scenario where polymorphism is used to treat objects of different derived classes. * Implement a class that uses parent:: to call and utilize methods from its parent class.   Class discussion on:   * Abstract Classes and Methods * Interfaces and Implementation * Abstract Classes vs. Interfaces * Practical Applications   Hands on (Lab Activities)   * Create an abstract class with abstract methods and concrete methods. * Demonstrate how the classes fulfill the contract specified by the interfaces. * Write a program that uses both abstract classes and interfaces. | In – Person Class (F2F) | * Research / Assignment * Seat work * Quiz * Laboratory |
|  | **PRELIM EXAMINATION** | | | | | |
|  | **MID-TERM PERIOD** | | | | | |
| **[2]** | * Understand encapsulation by explaining its principles and practices, including how it helps protect object state and ensure controlled access to data. * Compare composition and inheritance to determine when to use each approach for building complex systems. * Use getter and setter methods to manage access to private data in a class, demonstrating how they encapsulate and control data modifications. * Explain exception handling basics by using try, catch, and finally blocks to manage and resolve errors in code. * Describe how to create and throw custom exceptions to handle specific error conditions effectively. * Design error handling strategies that follow best practices to ensure robustness and maintainability in code. * Develop debugging techniques to identify, diagnose, and fix exceptions and errors efficiently. * Explain the concept and benefits of namespaces in organizing code and avoiding name conflicts. * Demonstrate the use of the namespace keyword to define and manage namespaces in PHP. * Implement autoloading of classes using spl\_autoload\_register to automatically include class files when needed. * Explain traits by defining their purpose and demonstrating how they are used to reuse code across multiple classes. * Demonstrate how to combine traits within classes to enhance functionality and achieve code reuse. * Describe reflection in PHP, including what it is and how it can be used to inspect and manipulate class metadata at runtime. | **Week 5: Encapsulation and Composition**   * Encapsulation: Principles and Practices * Composition vs. Inheritance * Getter and Setter Methods * Real-world Examples   **Week 6: Exception Handling and Error Management**   * Exception Handling Basics: Try, Catch, Finally * Creating and Throwing Custom Exceptions * Error Handling Best Practices * Debugging Techniques   **Week 7: Namespaces and Autoloading**   * Introduction to Namespaces * Using namespace Keyword * Autoloading Classes with spl\_autoload\_register * Practical Exercises   **Week 8: Traits and Reflection**   * Traits: Definition and Usage * Combining Traits in Classes * Reflection: What It Is and How to Use It * Practical Examples and Use Cases | **2 Hrs.**  **3 Hrs.**  **2 Hrs.**  **3 Hrs.**  **2 Hrs.**  **3 Hrs.**  **2 Hrs.**  **3 Hrs.** | Class discussion on:   * Encapsulation Principles * Composition vs. Inheritance * Getter and Setter Methods * Real-World Examples   Hands on (Lab Activities)   * Create a class with private attributes and use getter and setter methods to access and modify these attributes. * Implement a project that uses both composition and inheritance to solve a problem. * Develop a class with private data and implement getter and setter methods to manage access. * Build a small application that demonstrates encapsulation and composition.   Class discussion on:   * Exception Handling Basics * Custom Exceptions * Error Handling Best Practices * Debugging Techniques   Hands on (Lab Activities)   * Write code to practice using try, catch, and finally blocks. * Design and implement custom exceptions for specific error conditions. * Develop a project that incorporates best practices for error handling. * Create a set of exercises where students must use debugging tools to identify and resolve exceptions and errors.   Class discussion on:   * Introduction to Namespaces * Using the namespace Keyword * Autoloading Classes   Hands on (Lab Activities)   * Create a PHP project that uses namespaces to organize classes. * Write a set of exercises where students define and use namespaces with the namespace keyword. * Implement class autoloading using spl\_autoload\_register.   Class discussion on:   * Traits * Combining Traits * Reflection   Hands on (Lab Activities)   * Implement traits in a PHP project. * Create a class that uses multiple traits. * Write code to use PHP’s Reflection API to inspect classes, methods, and properties at runtime. * Develop a project that incorporates both traits and reflection. | In – Person Class (F2F) | * Research / Assignment * Seat work * Quiz |
|  | **MID-TERM EXAMINATION** | | | | | |
|  | **PRE-FINAL PERIOD** | | | | | |
| **[3]** | * Explain the concept of design patterns, including their purpose and benefits in software design. * Describe and implement common design patterns such as Singleton, Factory, and Strategy in PHP, demonstrating their usage and advantages. * Apply design patterns by developing practical examples in PHP. * Explain the Decorator pattern, including its purpose and how it enhances or extends object functionality dynamically. * Describe the Observer pattern, focusing on how it facilitates communication between objects in a way that maintains loose coupling. * Understand the Prototype pattern, demonstrating how it allows for creating objects based on a template or prototype. * Explain performance considerations specific to Object-Oriented Programming (OOP) in PHP, including common bottlenecks and how to address them. * Describe various code optimization techniques to enhance the efficiency and speed of PHP applications. * Demonstrate the use of profiling and benchmarking tools. * Initiate a mini project, setting objectives and planning tasks to apply Object-Oriented Programming (OOP) concepts and design patterns. * Implement OOP concepts and design patterns in the mini project, demonstrating how to integrate and apply learned principles to solve real-world problems. | **Week 9: Design Patterns Overview**   * Introduction to Design Patterns * Common Patterns: Singleton, Factory, Strategy * Implementing Patterns in PHP   **Week 10: Advanced Design Patterns**   * Decorator Pattern * Observer Pattern * Prototype Pattern * Practical Examples and Implementation   **Week 11: Performance Optimization**   * Performance Considerations in PHP OOP * Code Optimization Techniques * Profiling and Benchmarking PHP Code   **Week 12: Practical Application and Project Work**   * Start of Mini Project * Implementing OOP Concepts and Design Patterns | **2 Hrs.**  **3 Hrs.**  **2 Hrs.**  **3 Hrs.**  **2 Hrs.**  **3 Hrs.**  **2 Hrs.**  **3 Hrs.** | Class discussion on:   * T Introduction to Design Patterns * Common Patterns * Implementing Patterns in PHP   Hands on (Lab Activities)   * Implement the Singleton pattern in a PHP class. * Create a Factory pattern to instantiate different types of objects based on input. * Design and implement the Strategy pattern to define a family of algorithms and make them interchangeable.   Class discussion on:   * Decorator Pattern * Observer Pattern * Prototype Pattern   Hands on (Lab Activities)   * Implement the Decorator pattern by creating a base class and a series of decorators. * Develop a system using the Observer pattern. * Create a class that uses the Prototype pattern to clone objects.   Class discussion on:   * Performance Considerations in PHP OOP * Code Optimization Techniques. * Profiling and Benchmarking.   Hands on (Lab Activities)   * Use profiling tools to analyze a PHP application. * Optimize a provided PHP script by applying various techniques. * Develop a small project that requires performance optimization.   Class discussion on:   * Mini Project Kickoff * Implementing OOP Concepts and Design Patterns   Hands on (Lab Activities)   * Begin the mini project by defining requirements, creating a project plan, and setting up the development environment. * Implement various OOP principles and design patterns in the mini project. | In – Person Class (F2F) | * Research / Assignment * Seat work * Quiz * Laboratory |
|  | **PRE-FINAL EXAMINATION** | | | | | |
|  | **FINAL PERIOD** | | | | | |
| **[4]** | * Explain the basics of PHPUnit, including its role and benefits in unit testing for PHP applications. * Demonstrate how to write and run unit tests using PHPUnit, * Describe the principles of Test-Driven Devaelopment (TDD) and how to apply them. | **Week 13: Unit Testing with PHPUnit**   * Introduction to PHPUnit * Writing and Running Unit Tests * Test-Driven Development (TDD) Basics   **Week 14: Final Project Presentation**   * Final Project Presentation | **9 Hrs.**  **10 Hrs.**  **10 Hrs.** | Class discussion on:   * Introduction to PHPUnit * Writing and Running Unit Tests * Test-Driven Development (TDD) Basics   Hands on (Lab Activities)   * Create a simple test case to understand the framework's structure and basic functionalities. * Write and execute unit tests for various components of a PHP application**.**   Presentation of the final project application (System)  Presentation of the final project application (Document) | In – Person Class (F2F) | * Research / Assignment * Seat work * Quiz * Laboratory |
|  | **Final Examination** | | | | | |
|  |  |  |  |  |  |  |
|  |  |  | **Total Number of Hours = 90 Hrs.** |  |  |  |
| **Course References** | Books:   * Schwartz, M. (2021). *PHP objects, patterns, and practice* (3rd ed.). Apress. * Bergmann, S. (2021). PHPUnit: Test-Driven Development for PHP. Packt Publishing. * Bergmann, S., & Priebsch, S. (2020). Test-Driven Development with PHPUnit. O'Reilly Media. * McConnell, S. (2018). PHP Unit Testing: A Beginner's Guide. Apress. | | | | | |
| **Course Requirements** | Student portfolio | | | | | |
|  |  |  |  |  |  |  |
| **Assessment and Grading** |  | | | | | |
|  |  |  |  |  |  |  |
| **Course Policies** | Late Submissions of Assignment / Research / Case Study.  Late submissions will be permitted past a due date but with a 10%-point reduction for each missed day. Discuss any circumstances with the instructor if a late submission becomes necessary.  Attendance: you are expected to attend every class. Excessive absences may result in lowering your grade and even failing the course.  Tardiness: Class starts at the specified hour. Three (3) consecutive lates will be equivalent to 1 absence, the maximum number of absences for lecture class is 12 hours; for lecture and laboratory classes a maximum of 20 hours of absences. Maximum number of absences will automatically mean a “dropped” mark in the course. | | | | | |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  | **Revision History** | | | | | |
| **Revision Number** | **Date of Revision** | **People Involved in the Revision** | **Date of Effectivity** | **Highlights of Revision** |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Prepared by:** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Name of Faculty** | **Rochelle S. Lanto** | | | | | |
| **Email Address** | [**rochellelanto.faculty@cdd.edu.ph**](mailto:rochellelanto.faculty@cdd.edu.ph) | | | | | |
| **Consultation Schedule (Day and Time)** | Friday: 9:30-11:00 Student consultation | | | | | |
| **Location of Office** | Faculty Office - 2nd Floor LCA Bldg. | | | | | |

****



**Checked by: Recommending Approval: Approved:**

**JANN ALFRED A. QUINTO, MSIB DR. CARIDAD O. ABUAN JANN ALFRED A. QUINTO, MSIB**

Dean Vice President for Academics, Research, & Extension Chief Operating Officer 