Course Contents
Course Title: [CS-104] Object Oriented Programming (3-1)

Instructor Information	
Instructor:	Dr. Syed Adeel Ali Shah
Email:	adeel@uetpeshawar.edu.pk
Office Location:	Department of Computer Science (2nd Floor, Old Block)
Office Hours:	11:00 am. to 1:00 pm. (Wednesday to Friday)

Course Information

Course Pre-Requisite:

Participants are expected to have passed the following course;

- Programming Fundamentals

Reference Books/Material:

- Java: How to Program, 9th Edition by Paul Deitel
- Beginning Java 2, 7th Edition by Ivor Horton
- An Introduction to Object Oriented Programming with Java, 5th Edition by C. Thomas
- Starting Out with C++ from Control Structures to Objects, 9th Edition, Tony Gaddis
- C++ How to Program, 10th Edition, Deitel & Deitel.
- Object Oriented Programming in C++, 3rd Edition by Robert Lafore

Grading Policy

- Sessional: 25%
- Mid Term: 25%
- Final Term: 50%

Learning Outcomes

- Understand and apply encapsulation, inheritance, polymorphism, and abstraction.
- Design and implement classes and objects effectively.
- Apply inheritance and polymorphism to enhance code reusability.
- Model and solve problems using OOP techniques.
- Implement exception handling and file I/O in OOP.

Class Information

Class Timings:

- Sections A
  - Thursday (08:00 am to 10:30 am)
- Section B
  - Thursday (8:00 am to 10:30 am)

Class Policy:

- Students are required to take **100%** of the lectures. Students may be allowed a 25% deficiency in attendance based on legitimate documentary evidence of emergency and/or sickness etc. However, Under **NO** circumstances, students shall be allowed to sit in the final exam if their attendance is below **75%.**
- For every hour of lecture, you are required to put at least 2 hours' effort outside the class.
- Assignments must be submitted in time; late submission will NOT be accepted.
- Assignment must be submitted via *Turnitin* only.
- Assignments and quizzes are pre-announced (see the weekly schedule below) and no separate announcement will be made.

Course Outline
Introduction to object oriented design, history and advantages of object oriented design, introduction to object oriented programming concepts, classes, objects, data encapsulation, constructors, destructors, access modifiers, const vs non-const functions, static data members & functions, function overloading, operator overloading, identification of classes and their relationships, composition, aggregation, inheritance, multiple inheritance, polymorphism, abstract classes and interfaces, generic programming concepts, function & class templates, standard template library, object streams, data and object serialization using object streams, exception handling.

Weekly Breakdown of Contents

Week	Topics to be Covered
1	**Introduction to Object-Oriented Design** • Overview of object-oriented design • History and evolution of object-oriented programming • Advantages of object-oriented design over procedural programming • Introduction to key object-oriented programming (OOP) concepts: classes and objects
2	**Classes, Objects, and Data Encapsulation** • Understanding classes and objects in detail • Data encapsulation and abstraction • Implementation of classes and objects in a programming language • Practical exercises on defining classes and creating objects

3	**Constructors and Destructors** • Introduction to constructors and their types • Role of constructors in initializing objects • Destructors and their importance in resource management • Hands-on exercises with constructors and destructors
4	**Access Modifiers and Member Functions** • Access modifiers: private, public, protected • Const vs non-const member functions • Static data members and static member functions • Practical implementation and examples
5	**Function Overloading and Operator Overloading** • Concept of function overloading • Implementation of function overloading in OOP • Introduction to operator overloading • Practical examples of operator overloading for custom operations
6	**Identification of Classes and Their Relationships** • Techniques for identifying classes in problem domains • Understanding relationships between classes: association, dependency • Implementing class relationships in code • Examples and exercises on class identification
7	**Composition and Aggregation** • Understanding composition and aggregation in OOP • Differences between composition and aggregation • Implementation of composition and aggregation in programming • Practical exercises and examples
Mid Term Examination	
8	**Inheritance and Multiple Inheritance** • Introduction to inheritance: base and derived classes • Advantages of inheritance in code reuse • Understanding and implementing multiple inheritance • Exercises on creating and using derived classes

9	**Polymorphism** • Understanding polymorphism: compile-time vs runtime polymorphism • Implementation of polymorphism using virtual functions • Advantages of polymorphism in flexible code design • Practical examples and exercises
10	**Abstract Classes and Interfaces** • Concept of abstract classes and their role in OOP • Introduction to interfaces and their use in designing flexible systems • Implementation of abstract classes and interfaces • Practical exercises on abstract classes and interfaces
11	**Generic Programming Concepts** • Introduction to generic programming • Understanding templates: function templates and class templates • Benefits of generic programming in code reusability • Implementing templates in a programming language
12	**Standard Template Library (STL)** • Overview of the Standard Template Library • Key components of STL: containers, iterators, algorithms • Using STL in practical programming scenarios • Hands-on exercises with STL components
13	**Object Streams and Data Serialization** • Introduction to object streams and their use in I/O operations • Understanding data serialization and deserialization • Implementing object serialization using object streams • Practical examples of data and object serialization
14	**Exception Handling** • Importance of exception handling in robust software design • Basics of try, catch, and throw in exception handling • Custom exception classes and their implementation • Practical exercises on error handling using exceptions

15	**Advanced Topics and Course Review** • Discussion on advanced OOP topics • Review of key concepts and their applications in real-world scenarios • Final project discussion and guidance • Q&A session and preparation for final assessments
Final Term Examination	