Gyanmanjari Innovative University

Course Syllabus
Gyanmanjari College of Computer Application
Semester-I(MCA)

Subject: Object Oriented Programming Using Java - MCA2XX11502

Type of course: Major Core

Prerequisite: Basic knowledge Of Programming

Rationale:

The course is designed to provide students with strong foundations in object-oriented programming, software development, and enterprise-level application design. Java is one of the most widely used programming languages in industry due to its platform independence, security, robustness, scalability, and support for modern technologies.

This syllabus enables students to understand fundamental and advanced programming concepts such as classes, objects, inheritance, polymorphism, exception handling, multithreading, file handling, collections, generics, and lambda expressions.

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks		Total Marks
CI	T	P	C	SEE	CCE	
4	0	2	5	100	50	150

Legends: CI-Class Room Instructions; T – Tutorial; P - Practical; C – Credit; SEE - Semester End Evaluation; CCE-Continuous and Comprehensive Evaluation;

Course Content:

Sr. No	Course content	Hrs	% Weightage
	Java Programming Foundations Introduction of object oriented programming, Programming Platform, Java Buzzwords, History of Java, The Java Programming Environment: installing JDK, using the command line tools, using IDE, Programming Structures in Java: data types, Operators and Control Structures, Arrays, Working with Strings, Scanner class. **Practical:** 1. Implement a Java program illustrating the basic structure of a Java class including package declaration, class definition, main method, and comments.		

1	2. Develop a program to demonstrate the use of all primitive data types in Java with memory size and default values. 3. Create a Java application to perform arithmetic, relational, logical, bitwise, assignment, and unary operations using different operators 4. Create a program to accept user input using the Scanner class and display formatted output. 5. Develop a program to demonstrate decision-making statements using if, if-else, nested if, and switch-case. 6. Create a Java application to implement looping structures using for, while, do-while, enhanced for, and nested loops. 7. Implement a menu-driven calculator program using control structures and switch expressions. 8. Develop a Java program to generate patterns using nested loops and conditional statements. 9. Create a program to demonstrate the use of break, continue, and labeled statements in loop control. 10. Implement a Java application to find prime numbers, Armstrong numbers, and palindrome numbers using iterative structures. 11. Develop a program to create, initialize, and manipulate one-dimensional arrays. 12. Create a Java application to perform matrix operations such as addition, subtraction, and multiplication using two-dimensional arrays. 13. Implement a program to search and sort arrays. 14. Develop a Java program to calculate statistical operations like average, maximum, minimum, and standard deviation using arrays. 15. Develop a program to perform string manipulation operations such as concatenation, substring extraction, replacement, splitting, and tokenization. 16. Implement a program to count vowels, consonants, words, and special characters in a given string.	18	20%

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Program Execution & Logic Task** This Task is a practical assessment method used to evaluate students' understanding of Java programming concepts, logical thinking, problem-solving abilities, and coding skills. Students are required to analyze a given problem, develop the appropriate logic, write a Java program, execute it successfully	20	
2	**Java Learning Blog Activity** This Activity is designed to encourage students to actively engage with Java programming concepts beyond the classroom. Students create and maintain a learning blog where they summarize topics covered in class, share coding examples, document their learning journey, and		5

	explore real-world applications of Java concepts.		
3	**Execution Forecast** The faculty provides Java expressions or short program snippets without executing them. Students carefully analyze the code and write the predicted output. Students explain the logic behind their predictions. Faculty evaluates students based on prediction accuracy, logical reasoning, and explanation of the output.		5
	Total	**20**	**10**

Examination Style:

Program Execution & Logic Task (20 Marks):

Each student will be assigned 2–3 unsolved problem statements by the faculty covering: decision making and looping, array, string and Scanner class concepts. Students are required to: Analyze the problem statement ,write a complete java program. Compile and execute the program. Display correct output.

Java Learning Blog Activity (5 Marks):

Students prepare a technical blog explaining important Java concepts using real-world examples, diagrams, and Java programs. Students may choose topics such as OOP concepts, Stream API, Exception Handling, Multithreading, Collections Framework, JVM Architecture, or Design Patterns. The final submission includes a blog article PDF uploaded on the GMIU web portal.

Execution Forecast (5 Marks):

Execution Forecast is a Java activity where students predict the output of code snippets before compiling and executing them. They analyze variables, operators, control statements, and expressions to determine the expected results. After execution, students compare their predictions with the actual output, enhancing their logical thinking, debugging skills, and understanding of Java programming concepts.

2	**Object Modeling with Inheritance and Interfaces** classes, objects, objects and object variables, Defining your own classes, methods & Object Construction, Initializer block and Class Initializer block , Method Overloading , Static fields and methods, Inheritance concepts, Method overriding and polymorphism, Use of Final and Abstract, Wrapper Class and Enumeration, Interface **Practical:** 17. Implement a program demonstrating object creation, object references, and memory allocation in Java. 18. Create a Java application to define user-created classes with multiple attributes and behaviors. 19. Develop a program to demonstrate communication between objects using methods and object variables. 20. Implement a Java application illustrating parameterized methods and return types in class design. 21. Create a program to demonstrate object construction using default	18	20%

and parameterized constructors.
22. Develop a Java application to illustrate constructor overloading with different argument lists.
23. Implement a program demonstrating the use of this keyword for resolving variable ambiguity and constructor chaining.
24. Create a Java application to demonstrate initializer blocks and their execution order during object creation.
25. Create a Java application demonstrating method overloading using different parameter types and counts.
26. Create a program illustrating static methods, static data members.
27. Implement a Java program to demonstrate single inheritance using parent and child classes.
28. Create a Java application illustrating multilevel inheritance and method accessibility.
29. Develop a program demonstrating hierarchical inheritance for real-world entity modeling.
30. Create a program illustrating runtime polymorphism using dynamic method dispatch.
31. Implement a Java program demonstrating the use of super keywords to access parent class members and constructors.
32. Develop a program illustrating the use of final variables, final methods, and final classes.
33. Create a Java program using abstract classes and abstract methods for partial abstraction.
34. Create a Java application illustrating the use of wrapper classes
35. Develop a Java application to define and use enumerations (enum) with constants and methods.
36. Develop a program demonstrating multiple inheritance using interfaces.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**OOP Concept Presentation with Live Coding** The OOP Concept Presentation with Live Coding is an assessment method designed to evaluate students' understanding of Object-Oriented Programming (OOP) concepts and their ability to implement them in Java. Students are required to present a selected OOP concept and demonstrate its practical implementation through live coding.	20	
2	**Java Code Narration** This Activity encourages students to analyze, interpret, and explain Java programs in their own words. Students are provided with a Java program and narrate its functionality by describing the purpose of the code, the role of different statements, program logic, and the expected output.		5
3	**Refactor & Optimize** The faculty provides a Java program with code quality issues. Students analyze the existing		5

		code to identify areas for improvement. Students refactor the program and test the program to ensure the output remains unchanged. Faculty evaluates the submissions based on correctness, optimization, readability, and adherence to Java coding standards.			
		Total	**20**	**10**	

Examination Style:

OOP Concept Presentation with Live Coding (20 Marks):

Each student will be assigned 2–3 unsolved problem statements by the faculty and students perform live coding to implement core Object-Oriented Programming concepts in Java, including inheritance, polymorphism, abstract classes, and interfaces. Students write, execute, and explain Java programs in real time to demonstrate how different OOP principles are applied in software development.

Java Code Narration(5 Marks):

students will select or develop a Core Java program and explain the complete program execution through video narration. Students must describe the purpose and functionality of each line/block of code, including variables, methods, loops, conditional statements, classes, objects, and output generation and submit video on GMIU Web Portal.

Refactor & Optimize (5 Marks):

Refactor & Optimize is a hands-on coding activity designed to help students improve the quality, readability, maintainability, and efficiency of Java programs. Students are provided with a Java program containing issues such as redundant code, poor variable naming, duplicated logic, inefficient loops, or improper formatting. Without changing the program's functionality, students must refactor the code by applying clean coding principles and optimization techniques.

3	**Reliable Programming with Packages and Exceptions** Introduction to Packages, Built in Packages and User Defined Packages, String class, String Buffer and String Builder Class, dealing with errors, catching exceptions, tips for using exceptions, Exception Handling Mechanism, A Simple Generic class & generic methods **Practical:** 37. Implement a Java application to import and utilize predefined classes from packages such as java.util, java.lang, and java.io. 38. Develop a Java application to organize multiple classes into packages for modular programming and code reusability. 39. Implement a Java program demonstrating package hierarchy and access control using public and default access modifiers. 40. Develop a Java program to perform string creation and manipulation using the String class. 41. Implement a Java application to demonstrate immutable behavior of the String class. 42. Develop a Java application to compare strings using equals(), equalsIgnoreCase(), compareTo(), and == operator. 43. Create a Java application to reverse a string and check palindrome.	18	20%

conditions using string methods.

44. Develop a Java program to demonstrate mutable string handling using the StringBuffer class.
45. Implement a Java application to perform append, insert, delete, reverse, and replace operations using StringBuffer.
46. Create a Java application demonstrating exception handling using try, catch, finally, throw, and throws.
47. Develop a Java program to handle arithmetic, array index, null pointer, and number format exceptions.
48. Implement a Java application illustrating multiple catch blocks and nested try-catch structures.
49. Create a Java program to demonstrate the use of finally block for resource management.
50. Develop a Java application to create and use user-defined exceptions for custom error handling.
51. Develop a Java program to validate user input using exception handling mechanisms.
52. Develop a Java application illustrating object storage and retrieval using generic classes.
53. Implement a Java program demonstrating generic methods with different data types.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Robust Java Code Optimization Challenge** This is an assessment method designed to evaluate students' ability to analyze, optimize, and improve existing Java programs for better efficiency, readability, maintainability, and performance. Students are provided with Java code containing redundant statements, inefficient logic, poor coding practices, or performance issues and are required to enhance the code while preserving its functionality.	20	
2	**Java Case Study** This Activity enables students to apply Java programming concepts to analyze and solve real-world problems. Students are provided with a case scenario or business problem and are required to identify the requirements, design a suitable solution, and explain how Java can be used to implement the proposed system.		5
3	**Code Rescue** The faculty provides Java programs containing different types of errors. Students analyze the code to identify the errors and their causes. Students correct the errors without changing the intended functionality of the program.		5
	Total	20	10

	Examination Style: **Robust Java Code Optimization Challenge (20 Marks):** Students are required to develop and optimize a Java application assigned by faculty by applying package creation and exception handling concepts. Students must organize classes into appropriate packages, implement built-in and custom exceptions, handle runtime errors efficiently, and execute the application successfully. **Java Case Study Analysis (5 Marks):** The faculty will assign real-world problem topics to each student for detailed case study analysis. Students are required to prepare a structured case study report including problem analysis, objectives, proposed solution, system design, technologies used, and conclusion. The final report must be converted into PDF format and submitted on GMIU web portal. **Code Rescue (5 Marks):** **Code Rescue** is a practical Java activity in which students identify, analyze, and fix errors in pre-written Java programs. The code may contain syntax, logical, or runtime errors that prevent it from compiling or producing the expected output. Students must debug the program, implement the necessary corrections, and verify the solution by executing the code. This activity strengthens debugging skills, enhances logical reasoning, and promotes systematic problem-solving using Java programming concepts.		
4	**Java Generic Collections and Streamlined Coding** Introduction to Collection Framework, Collection Interface, List Interface, MAP Interface, Arraylist and Linked list, HashSet and TreeSet, Generics in Collection Framework, Introduction to Lambda Expression, Syntax and need for Lambda Expressions, Functional interfaces, method reference, Constructor reference, variable scope, Processing Lambda expression and inner classes **Practical:** 54. Create a Java program to perform insertion, deletion, traversal, and searching operations using collection classes. 55. Implement a Java program to compare arrays and collections for data management efficiency. 56. Create a Java application illustrating the use of the List interface for ordered data storage. 57. Develop a Java program to perform CRUD operations using ArrayList. 58. Create a Java program demonstrating insertion and deletion operations in LinkedList. 59. Develop a Java application to iterate list elements using Iterator, ListIterator. 60. Implement a Java program to sort and reverse elements stored in a List. 61. Develop a Java program illustrating the use of the Map interface for key-value pair management. 62. Implement a Java application to perform insertion, retrieval, update, and deletion operations using HashMap. 63. Implement a Java program demonstrating unique data storage	18	20%

using HashSet.

64. Develop a Java program illustrating automatic sorting and navigation operations using TreeSet.
65. Develop a Java application illustrating functional programming concepts using lambda expressions.
66. Implement a Java application to perform collection sorting and filtering using lambda expressions.
67. Create a Java application demonstrating variable scope and effectively final variables in lambda expressions.
68. Develop a Java program illustrating the relationship between lambda expressions and inner classes.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Interactive Assessment** This faculty will ask questions from the Unit. Students are required to answer the questions interactively to demonstrate their understanding of Java programming concepts. This helps assess clarity of thought, communication skills, and conceptual depth.	20	
2	**Java Brain Booster** Students participate individually in online quizzes to improve analytical thinking, coding accuracy, technical knowledge, and decision-making abilities in a competitive and engaging learning environment. This activity encourages quick recall of concepts, enhances confidence in programming, and supports preparation for technical interviews, coding assessments, and real-world software development tasks		5
3	**Technical Debate** The faculty will announce the debate topic and explain the rules, evaluation criteria, and time limits. Students will be divided into two teams or assigned individual positions: For and Against the given technical topic. Each team will research the topic and prepare technical arguments, supporting facts, examples, and evidence from reliable sources and participate in debate.		5
	Total	20	10

Examination Style:

Interactive Assessment (20 Marks):

Students are required to respond to questions, solve programming problems, explain concepts, and demonstrate their understanding in a dynamic and collaborative environment. The assessment focuses on both theoretical knowledge and practical application of Java programming concepts.

	Java Brain Booster (5 Marks): The activity promotes quick concept recall, strengthens programming confidence, and prepares students for technical interviews, coding assessments, and real-world software development challenges. **Technical Debate (5 Marks):** The Technical Debate is an interactive learning activity designed to enhance students' technical knowledge, critical thinking, analytical reasoning, communication, and presentation skills. Students participate in structured debates on technical topics. Participants present evidence-based arguments, defend their viewpoints, and respond to opposing opinions in a professional and respectful manner.		
5	**File Processing and Concurrent Programming** Introduction to File Handling. Working with Files. Byte Streams and Character Streams. Reading and Writing data to files. Copying, moving and deleting files. Random Access Files. Introduction to Threads. Thread Life Cycle. Runnable Interface. Thread Priorities. Daemon Threads **Practical:** 69. Create a Java application to demonstrate byte stream handling using FileInputStream and FileOutputStream. 70. Implement a Java application to demonstrate character stream handling using FileReader and FileWriter. 71. Implement a Java program to copy data from one file to another using byte streams. 72. Create a Java application to copy text files using character streams and buffered readers/writers. 73. Implement a Java application to count characters, words, and lines from a text file. 74. Implement a Java program to move files from one directory to another programmatically. 75. Create a Java application to rename and delete files using file handling methods. 76. Create a Java program to maintain student or employee records using random access files. 77. Develop a Java program to demonstrate thread creation using the Thread class. 78. Implement a Java application illustrating thread creation using the Runnable interface. 79. Develop a Java application to execute multiple threads concurrently for parallel task processing. 80. Implement a Java program demonstrating thread synchronization for shared resources. 81. Create a Java application illustrating inter-thread communication using wait(), notify(), and notifyAll() methods. 82. Develop a Java program to demonstrate thread priorities and their impact on execution order. 83. Create a Java program illustrating daemon thread behavior in background task execution. 84. Develop a Java application demonstrating thread scheduling using sleep and yield methods. 85. Develop a Java program demonstrating producer-consumer	18	20%

problems using multithreading.

86. Implement a Java application to perform file processing tasks using multithreading.

Evaluation Method:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Code Mastery Assessment** This evaluates students' proficiency in Java programming through hands-on coding tasks and problem-solving exercises. Students are required to develop, execute, and debug Java programs based on given requirements, demonstrating their understanding of programming concepts, logical thinking, and coding practices.	20	
2	**Capstone Integration Activity** This enables students to integrate and apply multiple Java programming concepts learned throughout the course to solve a real-world problem or develop a mini-project. Students are required to design, implement, and present a solution that demonstrates their understanding of Java fundamentals, object-oriented programming principles, and problem-solving techniques.		10
	Total	20	10

Examination Style:

Code Mastery Assessment (20 Marks):

Two 10 marks unsolved problem statements will be given based on a real-time problem-solving scenario aligned with competitive coding practices. The objective is to test students' logical thinking, problem-solving ability, and coding skills in unfamiliar yet practical contexts.

Capstone Integration Activity (10 Marks):

In this activity, the Faculty will assign tasks to students (in groups of four) and students design and develop a real-world Java application by integrating concepts of file handling, exception handling, multithreading, synchronization, and thread management into a single project and submit project reports on GMIU Web Portal.

Suggested Specification :

Distribution of Theory Marks (**Revised Bloom's Taxonomy**)						
Level	Remembrance (R)	Understanding (U)	Application (A)	Analyze (N)	Evaluate (E)	Create (C)
Weightage	10%	15%	30%	15%	10%	20%

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from above table.

Course Outcome:

After learning the course the students should be able to:	
CO1	Apply the principles of object-oriented programming to design and implement Java programs
CO2	Analyze Java class structures and distinguish key object-oriented concepts like inheritance, interfaces, method overloading etc.
CO3	Evaluate Java tools like collections and lambdas for optimal data management.
CO4	Utilize advanced programming concepts to design efficient and scalable solutions for complex tasks.
CO5	Implement Java features into the real world.

Instructional Method:

The course delivery method will depend upon the requirement of content and need of students.

A minimum of 20–30% of the course content shall be suggested to deliver through Flipped Learning, where students study digital learning resources before the classroom session. Classroom time shall be utilized for discussion, problem-solving, practical applications, and higher-order learning activities.

Students shall be encouraged to use AI-powered learning platforms, simulation software, virtual laboratories, digital libraries, research databases, online certification courses, NPTEL, SWAYAM, MOOCs, and other technology-enabled learning resources to strengthen conceptual understanding and practical competence.

Continuous assessment shall be conducted through Active Learning Assignments (ALAs), quizzes, skill demonstrations, presentations, practical exercises, projects, case studies, innovation activities, peer assessment, reflective learning, and industry-oriented tasks instead of relying primarily on conventional written assignments.

Reference Books:

[1] Java: The Complete Reference (13th Edition), by Herbert Schildt and Danny Coward, McGraw-Hill Education.
[2] Programming with Java (7th Edition), by E. Balagurusamy, McGraw-Hill Education.
[3] Head First Java (3rd Edition), by Kathy Sierra and Bert Bates, O'Reilly Media.
[4] Core Java Volume I – Fundamentals (13th Edition), by Cay S. Horstmann, Pearson Education.
[5] Effective Java (3rd Edition), by Joshua Bloch, Addison-Wesley

Suggested Assessment Guidelines:

Module 1: Java Programming Foundations

Program Execution & Logic Task (20 Marks)

Criteria	Description	Marks
Program Logic and Implementation	Correct use of decision making, looping, arrays, strings, and Scanner class concepts	5
Compilation and Execution	Error-free compilation and successful execution of the Java program	5
Output Accuracy	Correctness of output and handling of different test cases	5
Coding Standards and Readability	Proper indentation, meaningful variable names, and code readability	5
Total		**20**

Module 2: Object Modeling with Inheritance and Interfaces

OOP Concept Presentation with Live Coding (20 Marks):

Criteria	Description	Marks
Program Design and Implementation	Correct implementation of classes, objects, inheritance hierarchy, method overriding/overloading, abstract classes, and interfaces	5
Live Coding and Program Execution	Ability to write, compile, debug, and execute the Java program successfully during live coding	5
Output and Functionality	Correct execution flow, expected output, and proper demonstration of OOP behavior	5
Explanation and Technical Understanding	Ability to explain code logic, OOP concepts used, and answer faculty questions confidently	5
Total		**20**

Module 3: Reliable Programming with Packages and Exceptions

Robust Java Code Optimization Challenge (20 Marks)

Criteria	Description	Marks
Package Creation and Code Organization	Proper creation and usage of packages, class organization, and modular programming practices	5
Exception Handling Implementation	Correct use of built-in exceptions, custom exceptions, try-catch-finally blocks, and runtime error handling	5
Program Execution and Debugging	Successful compilation, execution, debugging, and stability of the Java application	5
Code Optimization and Readability	Efficient coding practices, readability, maintainability, and proper documentation/comments	5
Total		**20**

Module 4: Java Generic Collections and Streamlined Coding

Interactive Assessment (20 Marks):

Criteria	Description	Marks
Conceptual Understanding	Ability to understand and explain java programming concepts from Unit 4	5
Accuracy of Answers	Ability to provide correct and relevant answers to faculty questions	5
Problem Solving Approach	Ability to apply programming concepts logically while answering scenario based questions.	5
Communication Skills	Clarity, confidence and effectiveness in explaining concepts and program logic	5
Total		**20**

Module 5: File Processing and Concurrent Programming

Code Mastery Assessment (20 Marks):

Criteria	Description	Marks
Handling Unfamiliar Problems	Ability to adapt programming knowledge to solve new and practical coding challenges.	5
Logical Thinking & Approach	Ability to develop proper logic and an efficient problem-solving approach.	5
Output Accuracy	The program generates correct output for the given test cases and scenarios.	5
Viva / Explanation	Clarity, confidence and effectiveness in explaining concepts and Ability to explain the logic, approach and working of the implemented solution.	5
Total		**20**