Gyanmanjari Innovative University

Syllabus
Gyanmanjari Diploma Engineering College
Semester-2 (Diploma)

Subject: Java Application Development - DET1CE12202

Type of course: Major (Core)

Prerequisite: Programming fundamental, logic and problem-solving skill, mathematical logic.

Rationale:

Object-Oriented Programming (OOP) enables modular, reusable, and scalable software development. This course introduces students to Java, starting from basic syntax and advancing through OOP principles such as classes, objects, inheritance, polymorphism, abstraction, and encapsulation. It further covers exception handling, collections, and multithreading to build robust and concurrent applications. The skills gained will prepare students to design effective solutions for real-world problems and serve as a foundation for advanced areas like software engineering, mobile, and enterprise development.

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; MSE- Mid Semester Examination; LWA - Lab Work Assessment; V – Viva voce; CCE-Continuous and Comprehensive Evaluation; ALA- Active Learning Activities.

Course Content:

Sr. No	Course Content	Hrs.	% Weightage
1	**Theory Topics** - History & features of Java, JVM, JDK, JRE, Structure of Java program, Data types, variables, operators, Input/Output using Scanner, Control statements (if, if-else, nested if, switch-case), Loops: for, while, do-while break & continue.	T:06 P:12	20%

Practical:

1. Hello World: Write a program to print 'Hello World'.
2. Variables & Data Types: Program demonstrating declaration and initialization of variables of all data types.
3. Arithmetic Operations: Program to perform addition, subtraction, multiplication, and division.
4. Area of Shapes: Program to calculate area of circle, rectangle, and triangle.
5. Temperature Conversion: Convert Celsius to Fahrenheit and vice versa.
6. Swapping Numbers: Swap two numbers using third variable and without using third variable.
7. Even/Odd Check: Program to check whether a number is even or odd.
8. Largest of Three Numbers: Find the largest of three given numbers.
9. Menu-driven Calculator: Program to implement a calculator using switch-case.
10. Factorial: Program to find factorial of a number using loops.
11. Fibonacci Series: Print Fibonacci series up to n terms using loops.
12. Palindrome Number: Check whether the given number is palindrome or not.
13. Prime Number: Check whether a number is prime or not.
14. Pattern Printing: Print patterns such as pyramid, diamond, etc.

Examination Style:

Sr. No.	Evolution Methods	SEE	CCE
1	Debugging Task.	20	00
2	Code Optimization.	00	10
	Total	20	10

1. **Debugging Task (10 Marks)**:
   In this task, students will be four programs each carrying 05 marks containing logical or syntactical errors. The question will be unsolved, practical, or competitive coding type, and will be provided by the faculty. This task is designed to assess students' debugging and problem-solving abilities.

	2. **Code Optimization (10 Marks):** In this task, students will be provided with two programs, each carrying 05 marks. Students are expected to improve the logic, reduce time and/or space complexity, and minimize the number of lines of code without affecting the functionality. The questions will be practical in nature and designed by the faculty to assess students' ability to write efficient and concise code.		
2	**Theory Topics**: 1D & 2D arrays, Searching & sorting (linear, binary, bubble sort), String handling (String, String Buffer, StringBuilder), Methods in Java, Method overloading, Recursion. **Practical:** 15. Array Input & Display: Program to input and display elements of an array. 16. Max & Min Element: Find maximum and minimum element in an array. 17. Linear Search: Program to search for an element in an array using linear search. 18. Binary Search: Program to implement binary search on a sorted array. 19. Bubble Sort: Program to sort elements using bubble sort algorithm. 20. Matrix Addition: Add two 2D matrices. 21. Matrix Multiplication: Multiply two 2D matrices. 22. String Reversal: Reverse a string without using built-in functions. 23. Palindrome String: Check whether the string is palindrome or not. 24. Count Vowels/Consonants: Program to count vowels and consonants in a string. 25. Factorial using Method: Find factorial of a number using a method. 26. Fibonacci using Recursion: Print Fibonacci series using recursion. 27. GCD & LCM: Find GCD and LCM of two numbers using methods. 28. Tower of Hanoi: Solve Tower of Hanoi problem using recursion.	T:07 P:11	20%

Examination Style:

Sr. No.	Evolution Methods	SEE	CCE
1	Error Hunt Activity.	00	10
2	Skill Builder.	20	00
	Total	20	10

1. **Error Hunt Activity (10 Marks)**:
   In this task, students will be given two programs, each carrying 10 marks, containing logical or syntactical errors. The questions will be of unsolved, practical, or competitive coding type and will be provided by the faculty. The task aims to assess the students' debugging skills, logical reasoning, and ability to correct code to produce the desired output.

2. **Skill Builder (20 Marks)**:
   The practical task will consist of four questions carrying 05, 05, and 03 marks respectively, totalling 00 marks. Questions will be based on practical implementation from the unit. Students are required to write, execute, and submit the program with correct logic and output. All questions will be framed and provided by the faculty to assess students' practical understanding and coding skills.

3	**Theory Topics:** Classes & Objects, Constructors & Destructors, this & static keyword, Method overloading & overriding, Inheritance, Polymorphism, Encapsulation & Abstraction. **Practical:** 29. Class & Object Example: Program to create a class Student with fields and methods, then create objects. 30. Constructor Demo: Demonstrate default and parameterized constructors. 31. Static Keyword: Use static variable and static method in a program. 32. Method Overloading: Demonstrate method overloading with different parameter lists. 33. Example: Program to show single inheritance (class Employee, subclass Manager). 34. Multilevel Inheritance: Program to demonstrate multilevel inheritance. 35. Hierarchical Inheritance: Program where multiple classes inherit from a base class. 36. Method Overriding: Demonstrate method overriding in inheritance.	T:06 P:12	20%

37. Super Keyword: Use super to call parent class method and constructor.
38. Polymorphism Example: Demonstrate runtime polymorphism using base class reference.
39. Encapsulation: Create a class with private fields and public getters/setters.
40. Abstraction using Abstract Class: Program with abstract class Shape and subclasses Circle, Rectangle.
41. Abstraction using Interface: Program with interface Vehicle implemented by Car and Bike classes.
42. Final Keyword: Demonstrate final variable, final method, and final class.

Examination Style:

Sr. No.	Evolution Methods	SEE	CCE
1	ALA:- Core2App: From Concepts to Application.	00	10
2	Code Acceleration.	15	00
3	Difference and Comparison Questions.	05	00
	Total	20	10

1. **ALA Core2App: From Concepts to Application (10 Marks):**
   Students will create a working Java application that integrates OOP principles into a real-world scenario. They will gain skills in designing modular, reusable, and maintainable code, and learn to apply all OOP concepts cohesively.

2. **Code Acceleration (15 Marks):**
   This section consists of three questions, each carrying 05 marks, focusing on code optimization. Students are expected to improve the logic, reduce time and/or space complexity, and minimize the number of lines of code without affecting functionality. The questions will be framed by the faculty and aim to assess the student's ability to write clean, efficient, and optimized code.

	3. Difference and Comparison Questions (05 Marks): The difference and comparison question carries 05 marks and will be asked from the units covered in the syllabus. Students will be required to clearly differentiate or compare two related concepts, terms, or techniques. This task is designed to test conceptual clarity and the ability to highlight precise distinctions.		
4	**Theory Topics:** Packages & Access Modifiers, Wrapper classes & Autoboxing/Unboxing, Exception handling (try, catch, finally, throw, throws), File Handling in Java, Collections Framework: List, Set, Map, Generics. **Practical:** 43. Package Creation: Program to create and use custom packages. 44. Access Modifiers: Demonstrate public, private, protected, and default access specifiers. 45. Wrapper Classes: Convert primitive data types into objects using wrapper classes. 46. Try-Catch Example: Program to handle division by zero exception. 47. Multiple Catch: Program to handle multiple exceptions using multiple catch blocks. 48. Finally Block: Program demonstrating use of finally block. 49. Throw & Throws: Demonstrate use of throw and throws in exception handling. 50. Custom Exception: Create a user-defined exception class and use it. 51. File Read/Write: Program to read from and write into a text file. 52. File Copy: Copy content from one file to another. **Examination Style:** **Sr. No. | Evolution Methods | SEE | CCE** 1 | ALA:- Java Data Vault Project | 00 | 10 2 | Code Optimization. | 10 | 00 3 | Viva/Oral Examination. | 10 | 00 **Total** | 20 | 10	T:07 P:11	20%

	1. **ALA: Java Data Vault Project (10 Marks):** Students will create a fully functional application that applies advanced Java features for data storage, retrieval, and manipulation using file. They will gain expertise in structured coding, error handling, persistence, and efficient data management. The final Code will be submitted on the GMIU Web Portal. 2. **Code Optimization (10 Marks):** In this task, students will be provided with two programs, each carrying 05 marks. Students are expected to improve the logic, reduce time and/or space complexity, and minimize the number of lines of code without affecting the functionality. The questions will be practical in nature and designed by the faculty to assess students' ability to write efficient and concise code. 3. **Viva/Oral Examination Style (10 Marks):** The viva/oral test carries 05 marks and will be conducted individually to evaluate the student's understanding of practical concepts. Questions will be asked from the units covered in the syllabus, including logic explanation, code reasoning, and real-time application relevance. This test helps assess clarity of thought, communication skills, and conceptual depth		
5	**Theory Topics:** Multithreading concepts, Thread lifecycle, Creating threads (Thread class & Runnable interface), Thread priorities & synchronization, Inter-thread communication, **Practical:** 53. Thread Creation (Extending Thread): Program to create a thread by extending Thread class. 54. Thread Creation (Runnable): Program to create a thread by implementing Runnable interface. 55. Thread Sleep: Program demonstrating sleep method in threads. 56. Thread Priority: Program to set and get thread priorities. 57. Thread Synchronization: Program to demonstrate synchronization in threads. 58. Producer-Consumer: Program demonstrating inter-thread communication using wait () and notify (). 59. Multiple Threads: Program to run multiple threads simultaneously. 60. Deadlock Example: Program demonstrating deadlock situation. 61. Thread-safe Banking: Bank account transaction program using synchronization.	T:06 P:12	20%

Examination Style:

Sr. No.	Evolution Methods	SEE	CCE
1	ALA:-IntelliManage – Smart Java Application Suite.	00	10
2	Code Troubleshooting Task.	10	00
3	ProCoder Challenge.	10	00
	Total	20	10

1. **ALA: IntelliManage – Smart Java Application Suite (10 Marks):**
   This activity requires students to collaboratively design and implement a comprehensive Java-based application for record management. The project integrates concepts from all units: control structures for login modules and menu-driven operations; arrays, strings, and recursion for validation, searching, sorting, and reporting; OOP principles for modular design of classes; advanced Java features for secure file-based data handling with collections and generics; and multithreading for simulating concurrent tasks such as processing, calculations, and inter-thread communication.

2. **Code Troubleshooting Task (10 Marks):**
   In this task, students will be given a two-program containing logical or syntactical errors, carrying 05 marks. The question will be unsolved, practical, or competitive coding type, and will be provided by the faculty. This task is designed to assess students' debugging and problem-solving abilities.

3. **ProCoder Challenge (10 Marks):**
   One 10 marks question will be based on a real-time problem-solving scenario aligned with competitive coding practices, not directly from the syllabus. The objective is to test students' logical thinking, problem-solving ability, and coding skills in unfamiliar yet practical contexts. The question will be provided by the faculty.

Suggested Specification table with Marks (Theory):100

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%

Course Outcome:

After learning the course, the students should be able to:	
CO1	Understand Java history, features, JVM/JDK/JRE, and apply basic syntax, data types, operators, control statements, and loops
CO2	Implement arrays, strings, methods, recursion, and apply searching and sorting techniques in Java
CO3	Demonstrate OOP principles—classes, objects, constructors, inheritance, polymorphism, encapsulation, and abstraction—for real-world problem solving.
CO4	Apply advanced Java features—packages, access modifiers, wrapper classes, exception handling, file handling, collections, and generics—to build secure applications
CO5	Design and develop modular, reusable, multithreaded Java applications integrating OOP and advanced concepts cohesively

Instructional Method:

The course delivery method will depend upon the requirement of content and need of students. The teacher in addition to conventional teaching method by black board, may also use any of tools such as demonstration, role play, Quiz, brainstorming, MOOCs etc.

From the content 10% topics are suggested for flipped mode instruction.

Students will use supplementary resources such as online videos, NPTEL/SWAYAM videos, e-courses, Virtual Laboratory.

The internal evaluation will be done on the basis of Active Learning Assignment.

Practical/Viva examination will be conducted at the end of semester for evaluation of performance of students in laboratory.

Reference Books:

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

Suggested Rubrics:

Suggested Assessment Guidelines	
1	• **Debugging Task:** • **5 Marks:** Fully correct + well-explained + complete. • **4 Marks:** Mostly correct + minor gaps. • **3 Marks:** Partially correct + average explanation. • **2 Marks:** Limited correctness + major gaps. • **1 Mark:** Minimal relevant content. • **0 Marks:** Incorrect / Empty.
2	• **Code Optimization:** • **5 Marks:** Fully optimized, clean, efficient code. • **4 Marks:** Mostly optimized with minor improvement possible. • **3 Marks:** Partially optimized, basic effort. • **2 Marks:** Minimal optimization, many issues. • **1 Mark:** Very poor attempt, no improvement. • **0 Mark:** Not attempted / irrelevant.
3	• **Error Hunt Activity:** • **5 Marks:** All errors found + correct explanations + proper corrections. • **4 Marks:** Most errors found + mostly correct fixes. • **3 Marks:** Some errors found + basic corrections. • **2 Marks:** Few errors found + incorrect or incomplete corrections. • **1 Mark:** Very poor attempt, minimal correctness. • **0 Mark:** No attempt / irrelevant answer.
4	• **Skill Builder:** • **5 Marks:** Fully correct, skill applied effectively. • **4 Marks:** Mostly correct, small mistakes. • **3 Marks:** Partially correct, basic skill shown. • **2 Marks:** Limited skill, major gaps. • **1 Mark:** Very poor attempt. • **0 Mark:** No attempt / irrelevant.
5	• **Code Acceleration:** • **5 Marks:** Code highly accelerated + efficient + clean. • **4 Marks:** Mostly accelerated with small inefficiencies.

	• **3 Marks:** Partial improvement with some inefficiencies. • **2 Marks:** Minimal speed improvement. • **1 Mark:** Very poor acceleration attempt. • **0 Mark:** No attempt / irrelevant.
6	• **Code Acceleration:** • **5 Marks:** All issues found + correct fixes + clear reasoning. • **4 Marks:** Most issues found + mostly correct fixes. • **3 Marks:** Some issues found + partial fixes. • **2 Marks:** Few issues found + incorrect/incomplete fixes. • **1 Mark:** Very poor troubleshooting attempt. • **0 Mark:** No attempt / irrelevant answer.
7	• **ProCoder Challenge:** • **10 Marks:** Fully correct + optimized + clean + complete. • **9 Marks:** Mostly correct + minor improvements. • **8 Marks:** Very good logic + small mistakes. • **7 Marks:** Good attempt + some errors. • **6 Marks:** Average + partially correct. • **5 Marks:** Limited correctness. • **3–4 Marks:** Poor code + many errors. • **1–2 Marks:** Very poor understanding. • **0 Mark:** No attempt.