K L Deemed to be University	**CSE-1 -- KLVZA** **Course Handout** **2025-2026, Odd Sem**

Course Title	**: PROBLEM SOLVING THROUGH PROGRAMMING (JAVA)**
Course Code	**: 25SC1105E**
L-T-P-S Structure	**: 2-0-2-6**
Pre-requisite	:
Credits	**: 4.5**
Course Coordinator	**: Satya Gouri Arunasri Pabbisetti**
Team of Instructors	:
Teaching Associates	:

Syllabus :	• Problem-solving methodology and flowchart design• Introduction to Java programming model (syntax, variables, data types, type casting, operators)• Input/output operations (Scanner, BufferedReader)• Control flow: if-else, nested if, switch-case• Iterative constructs: for, while, do-while• Logical reasoning and flow tracing through dry runs• Pattern printing and arithmetic-based problem logic• Debugging and error tracing techniques • Concept of arrays and memory representation• 1D array operations: creation, traversal, insertion, deletion, rotation, merging• 2D arrays: matrix representation and manipulation• Searching techniques – Linear & Binary search with complexity analysis• Sorting techniques – Bubble, Selection, Insertion, Merge, Quick Sort• Optimization strategies: two-pointer technique, prefix sum, sliding window• Matrix algorithms – transpose, rotation, diagonal operations• CodeChef-style problem solving using arrays and loops • String handling and immutability• String vs StringBuilder vs StringBuffer• Common string problems: palindrome, anagram, substring, frequency count, character manipulation• Regular Expressions (regex) – pattern matching and text validation• Bitwise operators (AND, OR, XOR, NOT, shifts, masks)• Bit manipulation tricks for optimization (checking even/odd, power of 2, swapping, subset generation)• Recursion fundamentals, base & recursive cases, tracing stack frames• Recursive problem-solving: factorial, Fibonacci, backtracking (n-Queens, subset sum)• Quantitative and mathematics-based logical problems • Transition from procedural to object-oriented logic• Defining and using classes & objects• Methods, parameters, return types• Method overloading, constructors, and use of this keyword• Access specifiers and encapsulation• Static data and methods• Design of simple OOP-based applications (banking system, student result system, etc.)• Modularization of logic through classes • Concept of inheritance and types• Method overriding, super, and final keyword• Abstract classes and abstract methods• Interfaces and multiple inheritance in Java• Polymorphism (compile-time and runtime)• Dynamic binding and late method resolution• Reflection API – introspecting class members at runtime• OOP mini-project integrating multiple classes• Intro to design patterns – factory, strategy, and template • Types of exceptions, hierarchy, try-catch-finally, throw and throws• Custom exception classes• File handling: byte stream and character stream• Reading/writing files using FileInputStream, FileOutputStream, FileReader, FileWriter, and buffered streams• Serialization and Deserialization• Generics – parameterized classes and methods• Java Collections Framework – List, Set, Map, Queue and their implementations (ArrayList, HashSet, HashMap, PriorityQueue, etc.)• Functional Programming in Java – Lambda expressions, Stream API• Capstone mini-project integrating file I/O, collections, and exception handling
Text Books :	Text Books : 1. Java: The Complete Reference, 13th Edition, Herbert Schildt, McGraw-Hill

COURSE OUTCOMES (COs):

CO NO	Course Outcome (CO)	PO/PSO	Blooms Taxonomy Level (BTL)
CO1	Apply fundamental programming constructs such as data types, operators, conditional and iterative statements in Java to develop logic-based solutions for basic computational problems. Students will learn to design simple algorithms, trace execution, and validate logic through hands-on coding tasks.	PO2,PO3,PO1	3
CO2	Design, trace, and optimize algorithms using one-dimensional and two-dimensional arrays to solve mathematical, quantitative, and real-world problems efficiently through search, sort, and matrix manipulation techniques. Students will also analyze the efficiency and accuracy of algorithmic approaches.	PO1,PO2,PO4	4
CO3	Construct and evaluate advanced problem-solving logic using strings, recursion, and bitwise operations for solving complex mathematical, pattern-based, and combinatorial problems relevant to competitive coding platforms. Students will be able to integrate mathematical reasoning and pattern recognition into coding strategies.	PO2,PO4,PO5	4
CO4	Develop structured and modular programs by applying object-oriented programming principles such as encapsulation, abstraction, and modularization using Java classes, methods, and constructors. Students will transition from procedural to modular design thinking.	PO3,PO9,PO5	3
CO5	Design extensible and reusable Java programs employing inheritance, polymorphism, abstract classes, interfaces, and reflection API to solve domain-oriented problems with clarity and maintainability. Students will model real-world entities and relationships through effective OOP architecture.	PO3,PO5,PO11	5
CO6	Implement robust, scalable, and generic Java applications integrating exception handling, file I/O, generics, and collections framework, along with functional programming constructs to handle real-world data-driven tasks. Students will demonstrate ability to write production-level, fault-tolerant programs.	PO3,PO5,PO11	4

COURSE OUTCOME INDICATORS (COIs)::

Outcome No.	Highest BTL	COI-2	COI-3	COI-4	COI-5
CO1	3	**Btl-2** Understand the basic Java syntax, data types, variables, and operators to perform simple computations.	**Btl-3** Apply conditional statements (if, if-else, switch) to develop logic for decision-making in programs and Use iterative statements (for, while, do-while) to design and implement logic-based solutions for repetitive		

Outcome No.	Highest BTL	COI-2	COI-3	COI-4	COI-5
			computational problems.		
CO2	4	**Btl-2** Understand the usage of one-dimensional arrays to solve mathematical and logical problems effectively.	**Btl-3** Apply the various searching and sorting algorithms using arrays	**Btl-4** Analyze two-dimensional array-based algorithms for matrix operations and quantitative problem-solving.	
CO3	4	**Btl-2** Understand the recursion flow and base cases in problem-solving.	**Btl-3** Apply bitwise operations to optimize computational logic.	**Btl-4** Decompose complex problems into modular, logical components using recursion and strings.	
CO4	3	**Btl-2** Explain OOP principles: class, object, encapsulation, abstraction and initialization of objects using constructors.	**Btl-3** Design and implement Java classes applying encapsulation, constructors and modularize the programs with the appropriate methods & class structures -	**Btl-4** Analyze class hierarchies for reusability and coupling.	
CO5	5	**Btl-2** Explain the concepts of inheritance, polymorphism, abstract class, interfaces in OOP.	**Btl-3** Apply Inheritance, polymorphism, abstract classes and interfaces to develop modular components.	**Btl-4** Analyze class hierarchies for reusability and coupling.	**Btl-5** Integrate inheritance, polymorphism, abstract classes and interfaces to model complex domain systems.
CO6	4	**Btl-2** Understand the exception handling hierarchy ,Java I/O stream classes and their purposes.	**Btl-3** Apply generics , collection classes ,lambda expressions and Stream API to build modular and reusable code	**Btl-4** Decompose large data-driven problems into modular, fault-tolerant components	

PROGRAM OUTCOMES & PROGRAM SPECIFIC OUTCOMES (POs/PSOs)

Po No.	Program Outcome
PO1	Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.
PO2	Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)
PO3	Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public