Institute/ School Name	School of Engineering and Technology		
Department Name	Department of Computer Science & Engineering		
Program Name	Bachelor of Engineering (Computer Science & Engineering): B.E (CSE)		
Course Code	26CS013	Course Name	Data Structures using Java
L-T-P (Per Week)	3-0-4	Course Credits	04
Academic Year	2025-26	Semester/Batch	4th/2024-2028
Pre-requisites (if any)	None		
NHEQF Level	5	SDGs	4,9
Course Coordinator	Dr. Osho Sharma		

1. Scope and Objective of the Course:

This course provides a strong foundation in Java programming with an emphasis on the principles and implementation of fundamental data structures. The course develops a theoretical understanding of advanced programming and data-structuring concepts essential for efficient problem solving. It enables students to analyze real-world computational problems and to design, implement, and test data-structure-based solutions. Students are equipped with the skills required to build efficient, robust, and scalable software using appropriate data structures and algorithms in Java.

2. Programme Outcomes (POs):

At the end of the programme, students will be able to achieve knowledge about the following:	
PO 1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO 2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
PO 6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	**Individual and teamwork**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning**: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

3. Course Learning Outcomes (CLO):

After completing the course, the students will be able to:

CLO01: Understand Java programming concepts, including operators, classes, objects, inheritance, packages, and exception handling.

CLO02: Develop the skill to analyze real-world computational problems and select appropriate data structures and algorithms using abstraction, recursion, and complexity analysis.

CLO03: Apply suitable advanced Java programming constructs (multithreading, generics, and the Collection Framework) for designing efficient, modular, and scalable software solutions.

CLO04: Analyse and evaluate the database-driven Java applications by integrating relational databases using JDBC and assessing performance and correctness.

4. CLO-PO Mapping Matrix:

Course Learning Outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	NHEQF Level Descriptor
CLO1	H	M			M						L		Q1
CLO2	M	H			L						L		Q2
CLO3		M	H		H						L		Q3
CLO4			H	M	H						L		Q2, Q4

5. ERISE Grid Mapping:

Feature Enablement	Level (1-5, 5 being highest)
Entrepreneurship	1
Research/Innovation	3
Skills	5
Employability	3

6. Recommended Books (Reference Books/Text Books):

B01: Sierra, K., & Bates, B. Head First Java (2nd Edition), O'Reilly Media, 2005.
B02: Eckel, B. Thinking in Java (4th Edition), Prentice Hall, 2006.
B03: Schildt, H. Java: A Beginner's Guide (9th Edition), McGraw-Hill Education, 2022.
B04: Schildt, H. Java: The Complete Reference (7th Edition), McGraw Hill Education, 2017.
B05: Deitel, P. J., & Deitel, H. M. Java for Programmers, Prentice Hall, 2009.
B06: Lafore, R. Data Structures and Algorithms in Java (2nd Edition), Sams Publishing, 2002 (or around 2002–2003).

7. Other readings and relevant websites:

Resources	Link of Journals, Magazines, Websites and Research Papers
R1	https://www.javatpoint.com/method-overloading-vs-method-overriding-in-java
R2	https://www.javatpoint.com/wrapper-class-in-java
R3	https://www.geeksforgeeks.org/java-exercises/#pattern-programs-in-java
R4	https://www.baeldung.com/java-control-structures
R5	https://www.javaspring.net/blog/java-array-tutorial
R6	https://www.edureka.co/blog/inheritance-in-java
R7	https://techvidvan.com/tutorials/java-abstract-class
R8	NPTEL – Data Structure and Algorithms Using Java (Prof. Debasis Samanta, IIT Kharagpur)
Resources	**Link of Audio-Video resources**
V1	https://www.youtube.com/watch?v=sqdOAjjBMFc&list=PLYlskOUYCZsjfHeNJuBd4na3cb%20xzNWw2Q

V2	https://www.codecademy.com/resources/blog/advanced-java-code-challenges/
V3	https://www.youtube.com/watch?v=yNpRDGGZtO4
V4	https://www.youtube.com/watch?v=SmWl-zm97Vc
V5	https://www.youtube.com/watch?v=3xuJIaP3C4g&t=17s
V6	https://www.youtube.com/watch?v=rzA7UJ-hQn4
V7	https://www.youtube.com/watch?v=sifEAuiVUac

8. Recommended Tools and Platforms:

Code Quotient, Visual Studio, Eclipse

9. Course Plan:

Lecture Number	Topics	Weightage in ETE (%)	Instructional Resources
1-2	History and Features of Java, Java Virtual Machine (JVM), JRE, and JDK	15%	B03, V4
3-4	Practice Problems: • Setting up JAVA environment for (MacOS/Window/Linux). • Working on Integrated Development Environment (IDE)-Eclipse/NetBeans/Visual Studio code		
5-6	Practice Problems: • Compiling and Interpreting Java Program • Understanding (public static void main(string[] arg) • Method Command Line Arguments.		
7-8	Java Basics: Identifiers, Keywords, Java Data Types & Operators		B03, V1
9-10	Practice Problems: • Write first program in Java for basic Input and Output • Aggregate and Percentage Marks of a student		
11-14	Control Statements in Java: Decision Constructs, loop constructs, jump statement e.g. break, continue and return		B03, R4
15-16	Practice Problems: • Is the number Kaprekar or not • Generate the first n Prime numbers		
17-18	Nested Loops and conditional statements, involving patterns		B03, R3
19-20	Practice Problem: • Write a Java program that prints the following pattern for a given integer n using nested loops: For n = 5 1 2 3 4 5 1 2 4 5 1 5 1 2 4 5 1 2 3 4 5 • Write a Java program that reads n numbers and prints the frequency of each digit (0–9) appearing in all numbers.		
21-22	Array: Introduction, Representation of Linear Arrays in Memory Traversing Linear Arrays, Insertion and Deletion in arrays	10%	B03, R5
23-24	Practice Problems: • Find the Second Largest Element in an Array • Given an array of size n and an integer k, perform left rotation of the array by k positions.		

25-26	Multi-Dimensional Array		
27-28	Practice Problems: • Matrix Multiplication • Spirally traversing a matrix • Rotate a 2-D array by 90 degrees		B03
29-30	Classes & Objects: Classes, objects and methods: defining a class, Access Control, Method overloading		
31-34	Constructing Objects through Constructors, Default constructors, Parameterized constructors, Copy constructors, Wrapper Classes		
35-36	Practice Problem: • Convert given number of seconds to days, hours, minutes, and seconds, Class- Timespan		B03, R2, R1
37-38	Practice Problem: • Class – TollBooth, methods of class: Display of Strings		
39-40	Class variables (static keyword), Instance variables and methods, this keyword		
41-42	Practice Problems: • Create a Student class with the following specifications: o Class Variables (static): ▪ collegeName (String) – same for all students ▪ studentCount (int) – keeps track of the number of Student objects created o Instance Variables: ▪ rollNo (int) ▪ name (String) ▪ marks (double) Requirements: • Initialize collegeName using a static block. • Increment studentCount each time a new Student object is created. • Create an instance method displayDetails() to print student information. • Create a static method displayStudentCount() to print total students.	20%	B03
43-44	Inheritance: Working with Inheritance: Inheritance Basics & Types, using super, Method Overriding		
45-46	Dynamic method dispatch, final keyword		B03, R6
47-48	Practice Problem: class – Box, BookCD, Marketer		
49-50	Abstract Methods & Classes, Packages & Interfaces: Built-In Packages and User-Defined Packages		
51-52	Practice Problem: • Design a Payment Processing System for an e-commerce platform. The system should support multiple payment methods, such as Credit Card and UPI, ensuring a common structure and enforceable behavior across payment types.	10%	B03, R7
53-54	Interfaces: Declaration, Implementation, Extending Classes and Interfaces		
55-56	Practice Problems: • Class - Customer and BankAccount • Class - MovablePoint and MovableCircle • Class: Circle and ResizableCircle		B03
57 – 58	Strings, StringBuffer, StringBuilder & StringTokenizer: Introduction, Immutable String	10%	B03, V5

59 – 60	Methods of String class, String Buffer class & StringBuilder class, toString method, StringTokenizer class		
61-62	Practice Problem: • Capitalize the first letter of each word. • Swap the first and last characters of each word in a string. • Find occurrences of palindrome words in a string.		
63-66	Revisit - Linked List (Singly, Doubly, Circular)	20%	B06, R8
67-68	Practice Problem: Build a Linked List Manager that supports different types of linked lists and performs common operations such as insertion, deletion, traversal, and search.		
69-72	Queue, Stack and Applications (Evaluation of Prefix and Postfix expressions, Conversion from Infix to prefix/postfix)		B06, R8
73-74	Practice Problem: Implement a Stack class using an array with the following operations: push(int element), pop(), peek(), isEmpty(), isFull()		
75-76	Collections Framework: Introduction, Collection Interfaces-List, Queue		B04, V6
77-78	Practice Problems • Duplicate the Queue elements • Number is a Happy number or not		
79-80	Collection Interfaces: Set(Overview), Collection Classes: ArrayList, LinkedList	10%	B04, V6
81-82	Collections Interfaces- Iterator, Working with Maps (Overview), Comparable & Comparator		
83-84	Practice Problems • Remove duplicates from a vector • Implement Own ArrayList in Java		
85-86	Exception Handling: Exception handling fundamentals, Exception types, try and catch		B03
87-88	Multiple catch clauses, nested try, throw, throws and finally, Creating custom Exception.		
89-90	Practice Problems: Password too short Exception, Valid email address, Valid index of array		
90-93	Multithreading: Java thread model, main thread, creating thread by implementing Runnable and extending thread class		B03
94-95	Creating multiple threads, using isAlive() and join(), thread Priorities,		
96-99	Synchronization		
100-101	Practice Problems • Multithreading Java: Greetings in reverse • Transactions in Account with and without Synchronization		
102-103	JDBC Connectivity: Introduction, Architecture of JDBC, Database Connection.	5%	B04, V7
104-105	JDBC Connectivity: Establishing JDBC Database, Connection.		

10. Industry Interventions:

Industry curated course "Data Structures" at the link below:

- https://codequotient.com

11. Innovative Pedagogies:

- Case study-based learning (Annexure-II)

12. Action plan for different types of learners

Slow Learners	Average Learners	Advanced Learners
Remedial Classes	Assignments (Annexure-III)	Advance Level Problem Assignment (Annexure-IV)

13. Evaluation Scheme & Components:

Evaluation Component	Type of Component	No. of Assessments	Weightage of Component	Mode of Assessment (Offline/ Online)
Internal Component 1	Formative Assessments (FAs)	01*	20%	Offline
Internal Component 2	Sessional Tests (STs)	03**	30%	
External Component	End Term Examination	01	50%	
Total		**100%**		

* Formative Assessment (FA) is mandatory.
** Out of 03 STs, the best 02 STs marks will be considered to evaluate final marks.

14. Details of Evaluation Components:

Evaluation Component	Description	Syllabus Covered	Timeline of Examination	Weightage (%)
Internal Component 1	FA	Lab Assessments	Will be intimated in due course	20%
Internal Component 2	ST 01	Up to 40% (Lectures 1-42)		30%
	ST 02	40% - 80% (Lectures 43-84)		
	ST 03	100% (Lectures 1-105)		
External Component	End Term Examination*	100%		50%
Total				100%

* Minimum 75% attendance is required to become eligible for appearing in the End Semester Examination

15. Format of Evaluation Components:

Type of Assessment	Total Marks	Hands-on Assessments	1 Mark MCQ	2 Marks MCQ	5 Marks Coding	10 Marks Coding
Formative Assessments	20	20	-	-	-	-
Sessional Tests	40	-	10	5	2	1
End Term Examination	60	-	10	10	4	1

16. Revision (if any):

Academic Year of Previous Version	2024-25	Percentage of Revision	70%
Topics:			
Data Structure implementation with Java has been proposed in this course, whereas in the previous course, DS was implemented using C++. • Java Basics: Keywords, Operators, Control Statements, Loops • Java: Constructors • Inheritance, Abstract Methods & Classes, Packages & Interfaces: Built-In Packages and User Defined Packages • Interfaces: Declaration, Implementation, Extending Classes and Interfaces • Collection Frameworks and Interfaces • Exceptional Handling • Multithreading			

17. <u>This Document is.</u>

Designation	Name	Signature
Prepared by Course Coordinator	Dr. Osho Sharma	
Verified by Assistant Dean	Dr. Hakam Singh	
Date	12/01/2026	

Annexure-I

S. No	Lab Exercises
1.	• Setting up JAVA environment for (MacOS/Windows/Linux). • Working on Integrated Development Environment (IDE)- Eclipse/NetBeans/Visual Studio code
2.	Write a Java program that calculates the gross salary of an employee based on basic salary and applicable allowances using proper class structure and conditional logic.
3.	Write a program in Java to determine the next closest Fibonacci number that is greater than or equal to a given integer using efficient iterative logic.
4.	Given a 4-digit positive integer, write a program to add 1 to each of its digits and form a new number using the updated digits. If adding 1 to a digit, results in 10, replace that digit with 0
5.	Write a Java program to print a diamond star pattern of height n (where n is an odd number). For n = 5 * *** ***** *** *
6.	An index i of an array is called an equilibrium index if the sum of elements at indices less than i is equal to the sum of elements at indices greater than i. Write a program to find the first equilibrium index in the array. If no such index exists, print -1.
7.	Write a Java program that manages employee profiles in an organization. The program should allow creating employee objects using different sets of initial data, achieved through constructor overloading. Class Requirements Create a class named Employee with the following instance variables: • empId (int) • empName (String) • department (String) • salary (double) Constructors to Implement 1. Default Constructor ○ Initializes: ▪ empId = 0 ▪ empName = "Not Assigned" ▪ department = "General" ▪ salary = 0.0 2. Constructor with ID and Name ○ Initializes empId and empName ○ Assigns default values to remaining variables 3. Constructor with ID, Name, and Department ○ Initializes empId, empName, and department ○ Sets salary to 0.0 4. Constructor with All Details ○ Initializes all instance variables Program Requirements 1. Use constructor overloading to define multiple constructors with different parameter lists. 2. Create a method displayEmployeeDetails() to display all employee information. 3. In the main method: ○ Create at least four employee objects, each using a different constructor. ○ Display the details of all employees.

8.	Write a Java program to read a 2D array (matrix) of size m × n and calculate: • Sum of each row • Sum of each column
9.	Create a base class Vehicle with attributes brand and speed. Also create a derived class Car that adds fuelType and overrides a method displayDetails().
10.	Create an abstract class Shape with an abstract method calculateArea(). Create subclasses Circle and Rectangle that provide implementations for area calculation.
11.	Write a Java program that accepts a string and prints the frequency of each character.
12.	Implement a Circular Linked List that supports: • Insert at end • Delete a given value • Display the list (one complete cycle) Requirements: • Handle empty and single-node cases • Avoid infinite loops during traversal
13.	Write a Java program to convert a prefix expression into its equivalent postfix expression using a stack.
14.	Create a program that performs withdrawal from a bank account. Throw a user-defined exception InsufficientBalanceException if withdrawal amount exceeds balance.
15.	Store student names in an ArrayList. Traverse the list using an Iterator and remove names shorter than 4 characters.
16.	Create a HashMap<Integer, String> to store employee IDs and names. Perform the following operations: • Insert entries • Search employee by ID • Display all entries Requirements: • Use entrySet() • Demonstrate basic Map operations
17.	Create a Student class with rollNo, name, and marks. Sort students: • By roll number using Comparable • By marks using Comparator Requirements: • Implement Comparable • Create a separate Comparator class
18.	Create two threads: • One prints even numbers • Another prints odd numbers (from 1 to 20) Requirements: • Extend Thread or implement Runnable • Demonstrate concurrent execution

Annexure-II

S. No	Topics
1	Case study-based learning: OOPs Concept

CASE STUDY: Library Management System Using OOP Concepts in Java

1. Introduction

A Library Management System (LMS) is required to manage books, members, and transactions such as issuing and returning books. The system should be scalable, modular, and easy to maintain, which makes Object-Oriented Programming (OOP) an ideal approach. This case study demonstrates how core OOP principles in Java Encapsulation, Inheritance, Polymorphism, Abstraction, and Association can be applied to design an efficient Library Management System.

2. Problem Statement

Design a Java-based Library Management System that:

- Maintains details of books and members
- Allows issuing and returning of books
- Calculates late fine for overdue books
- Ensures data security and code reusability

3. OOP Concepts Applied

3.1 Encapsulation

- Data members are declared private
- Access is controlled using getter and setter methods

3.2 Inheritance

- Common attributes of users are inherited from a base class

3.3 Abstraction

- Abstract classes define common behavior without implementation

3.4 Polymorphism

- Method overriding is used for different member types

3.5 Association

- Objects interact with each other (Library → Book → Member)

4. System Design (Class Structure)

4.1 Class: Book

Purpose: Represents a book in the library.

Attributes:

- bookId
- title
- author
- isIssued

Methods:

- issueBook()
- returnBook()
- displayBookDetails()

4.2 Abstract Class: Member

Purpose: Represents a library member.

Attributes:

- memberId
- name

Abstract Method:

- calculateFine(int daysLate)

Concrete Method:

- displayMemberDetails()

4.3 Derived Classes

StudentMember

- Inherits Member
- Lower fine rate

FacultyMember

- Inherits Member

- Higher fine-free duration

Polymorphism:
calculateFine() behaves differently based on member type.

4.4 Class: Library

Purpose: Manages books and members.
Responsibilities:

- Add books
- Register members
- Issue and return books
- Display records

Create a class diagram using tools such as Mermaid and implement the above-designed system using Java

Key Objectives:

- To understand the problem and design a solution for same

Annexure-III

S. No	Topics
1	Assignments

Assignment 1:

Problem I: String Analysis Tool
Concepts Covered:
String handling, loops, conditionals
Problem Statement:
Write a Java program that accepts a string and counts:

- Number of vowels
- Number of consonants
- Number of digits
- Number of spaces

Requirements:

- Use charAt() method
- Ignore special characters

Problem II: Student Management System Using OOP
Concepts Covered:
Classes, objects, constructors, encapsulation
Problem Statement:
Create a Student class with:

- Roll number, name, marks
- Parameterized constructor
- Method to calculate grade

Requirements:

- Use private data members
- Display student details and grade

Annexure-IV

S. No	Topics
1	Assignments on Advance Level Problems

Assignment 1:

1. Write a Java program to reverse a string without using the reverse method of Java's String class.

This exercise will teach you about how Java handles strings. First, you should create a class with a main method that accepts a string and then returns a string where the characters are in reverse order. But, you can't use the reverse method of the String class because that'd be cheating.

Even without doing that, there are at least three ways you can solve this exercise. Can you solve it in three different ways?

2. Write a Java program to determine whether a number is prime or not.

This program should accept an array of numbers. It should iterate through the array of numbers and determine if each is prime or not. Once it's done processing, it should return only those numbers that are prime in an array.

3. Create a Java Singleton class.

The singleton pattern is a design pattern that restricts the instantiation of an object to only one instance: To do this, you'll need to create a Singleton class that has a non-parameterized constructor.

The class should have one public variable called str. It should also have a static method called getSingleInstance that'll return the single instance of the class.

4. Write a Java program that returns an MD5 hash.

MD5 (Message-Digest algorithm 5) is a cryptographic function for creating 128-bit hashes from strings. It was designed by Professor Ronald Rivest of MIT, and although the function is considered cryptographically broken, it's still useful as a programming exercise.

Research the algorithm and write a Java class from scratch that accepts an alphanumeric string and returns the MD5 hash for that string.

5. Write three Java lambda expressions.

This exercise will test your ability to write lambda expressions in Java. Java lambda expressions take in parameters and return a value. They're similar to methods but don't need a name and can be implemented in the body of a method.

In this challenge, you'll write three of them:

- One that returns true if a number passed to it is odd and false if it's even.
- One that returns true if a number is prime and false if it's not.
- One that returns true if the parameter is a palindrome and false if it's not.