GUJARAT TECHNOLOGICAL UNIVERSITY

Program Name: Engineering

Level: Diploma

Branch: Information & Communication Technology

Course / Subject Code : DI03032021

Course / Subject Name : Java Programming

w. e. f. Academic Year:	2024-25
Semester:	3rd
Category of the Course:	PCC

Prerequisite:	Basic computer programming concepts: Students should have a foundational understanding of programming principles, including variables, control structures, data types, and basic algorithms. Familiarity with any programming language, such as C or Python, is recommended.
Rationale:	This course is designed to teach object-oriented programming concepts, techniques, and applications using the Java programming language. Java is a simple, portable, distributive, robust, secure, dynamic, architecture-neutral, object-oriented programming language that enables development of applications for diverse computing platforms. The aim of this course is for students to master platform-independent object-oriented programming with Java as the foundation for advanced technologies like web applications, cloud computing, and enterprise systems. Students will learn to design, implement, test, and debug Java applications using object-oriented principles. The course emphasizes hands-on programming experience through practical exercises and projects that reinforce theoretical concepts and develop problem-solving abilities.

Course Outcome:

After Completion of the Course, Student will able to:

No	Course Outcomes	RBT Level
01	Write simple Java programs using basic syntax, data types, and control structures.	R,U,A
02	Apply object-oriented programming concepts to solve real-world problems.	R,U,A
03	Develop Java applications using inheritance, interfaces, and packages.	R,U,A
04	Implement exception handling and multithreading in Java programs.	R,U,A
05	Develop Java applications using file handling and collections framework.	R,U,A

**Revised Bloom's Taxonomy (RBT)*

GUJARAT TECHNOLOGICAL UNIVERSITY

Program Name: Engineering

Level: Diploma

Branch: Information & Communication Technology

Course / Subject Code : DI03032021

Course / Subject Name : Java Programming

Teaching and Examination Scheme:

Teaching Scheme (in Hours)			Total Credits L+T+ (PR/2)	Assessment Pattern and Marks				Total Marks
				Theory		Tutorial / Practical		
L	T	PR	C	ESE (E)	PA(M)	PA(I)	ESE (V)	
3	0	2	4	70	30	20	30	150

Course Content:

Unit No.	Content	No. of Hours	% of Weightage
1.	**1. Introduction to Java Programming** 1.1 Explain Java features and platform independence. 1.1.1 Introduction to Java and history 1.1.2 Features of Java and JVM architecture 1.1.3 Java applications and platform independence 1.2 Install and configure Java development environment. 1.2.1 JDK, JRE, and JVM components 1.2.2 Java environment setup and tools 1.3 Write, compile, and execute simple Java programs. 1.3.1 Structure of Java program 1.3.2 Compilation and execution process 1.3.3 Comments and documentation 1.4 Apply data types, variables, and operators in Java programs. 1.4.1 Primitive data types and variables 1.4.2 Type conversion and casting 1.4.3 Operators (arithmetic, relational, logical, bitwise) 1.4.4 Operator precedence and associativity	09	20
2.	**2. Object-Oriented Programming Concepts** 2.1. Explain object-oriented programming principles. 2.1.1. Object-oriented programming fundamentals 2.1.2. Class, object, encapsulation, inheritance, polymorphism 2.1.3. Procedural vs. Object-oriented programming 2.2. Create classes and objects in Java. 2.2.1. Class definition and object creation 2.2.2. Memory allocation for objects 2.2.3. Object references and garbage collection 2.3. Apply encapsulation through access modifiers. 2.3.1. Access modifiers (public, private, protected, default)	09	20

	2.3.2. Data hiding and encapsulation principles 2.3.3. Getter and setter methods 2.4. Implement constructors and methods in Java programs. 2.4.1. Methods and parameter passing 2.4.2. 'this' keyword and its usage 2.4.3. Constructor types and chaining 2.5. Use method overloading and static components. 2.5.1. Method overloading principles 2.5.2. Constructor overloading 2.5.3. Static variables, methods, and blocks 2.5.4. Final keyword with variables, methods, and classes		
3.	**3. Inheritance, Interfaces, and Packages** 3.1. Implement inheritance in Java programs. 3.1.1. Inheritance concepts and types 3.1.2. Extending classes with 'extends' keyword 3.1.3. Super keyword and constructor chaining 3.1.4. Types of inheritance (single, multilevel, hierarchical) 3.2. Apply method overriding for polymorphism. 3.2.1. Method overriding principles 3.2.2. Dynamic method dispatch 3.2.3. Object class and its methods 3.2.4. toString(), equals(), hashCode() methods 3.3. Create and implement interfaces. 3.3.1. Interface definition and implementation 3.3.2. Multiple inheritance through interfaces 3.3.3. Default and static methods in interfaces 3.4. Develop abstract classes for partial implementation. 3.4.1. Abstract classes and methods 3.4.2. Concrete methods in abstract classes 3.4.3. Abstract class vs. Interface 3.5. Organize code using packages. 3.5.1. Package creation and naming conventions 3.5.2. Import statement and package access 3.5.3. Built-in packages in Java	09	20
4.	**4. Exception Handling and Multithreading** 4.1. Implement exception handling in Java applications. 4.1.1. Exception handling fundamentals 4.1.2. Exception hierarchy and types 4.1.3. Try, catch, finally blocks 4.1.4. Multiple catch blocks and exception propagation 4.1.5. Throw and throws keywords 4.2. Create custom exceptions for specific requirements	09	20

	4.2.1. Custom exception creation 4.2.2. Exception handling best practices 4.3. Explain multithreading concepts and benefits. 4.3.1. Multithreading concepts and benefits 4.3.2. Process vs. Thread 4.3.3. Thread lifecycle and states 4.4. Create and manage threads in Java programs 4.4.1. Thread creation approaches 4.4.2. Extending Thread class 4.4.3. Implementing Runnable interface		
5.	**5. Collections & File Handling** 5.1. Implement string manipulation in Java programs. 5.1.1. String class and its methods 5.1.2. String immutability concept 5.1.3. StringBuilder and StringBuffer 5.1.4. String operations and regular expressions 5.2. Apply Java Collections Framework for data management. 5.2.1. Collections Framework overview 5.2.2. List interface and implementations (ArrayList, LinkedList) 5.2.3. Set interface and implementations (HashSet, TreeSet) 5.2.4. Map interface and implementations (HashMap, TreeMap) 5.2.5. Iterator and enhanced for-loop 5.3. Perform file input/output operations. 5.3.1. File handling fundamentals 5.3.2. File class and operations 5.3.3. Byte streams vs. Character streams 5.3.4. Reading and writing text files 5.4. Implement serialization for object persistence. 5.4.1. Object serialization concepts 5.4.2. Serializable interface 5.4.3. ObjectInputStream and ObjectOutputStream	09	20
	Total	**45**	**100**

Suggested Specification Table with Marks (Theory):

Distribution of Theory Marks (in %)					
R Level	**U Level**	**A Level**	**N Level**	**E Level**	**C Level**
20	30	50	--	--	--

Where R: Remember; U: Understanding; A: Application, N: Analyze and E: Evaluate C: Create (as per Revised Bloom's Taxonomy)

References/Suggested Learning Resources:

(a) Books:

1. Java: The Complete Reference by Herbert Schildt, McGraw Hill Education, 12th Edition, 2021, ISBN: 9781260440232
2. Programming with Java by E Balagurusamy, McGraw Hill Education, 6th Edition, 2019, ISBN: 9789353161491
3. Core Java: Volume I - Fundamentals by Cay S. Horstmann, Prentice Hall, 11th Edition, 2018, ISBN: 9780135166307
4. Head First Java by Kathy Sierra and Bert Bates, O'Reilly Media, 3rd Edition, 2022, ISBN: 9781491910771
5. Thinking in Java by Bruce Eckel, Prentice Hall, 4th Edition, 2006, ISBN: 9780131872486

(b) Open source software and website:

- Softwares
  - Java Development Kit – JDK 23
  - IntelliJ IDEA Community Edition
  - Visual Studio Code
  - Eclipse IDE
- Official Documentation
  - Official Java Documentation
- Online Tutorials
  - W3Schools Java Tutorial
  - JavaTpoint - Java Programming Tutorials
  - GeeksforGeeks - Java Programming Language
  - Learn Java Online
  - TutorialsPoint - Java Tutorials and Examples
- Video Courses
  - Introduction to Java Programming (Udacity YouTube Playlist)
  - Programming in Java by Debasis Samanta, IIT Kharagpur (NPTEL YouTube Playlist)
  - Java Tutorials For Beginners In Hindi (CodeWithHarry YouTube Playlist)
- Comprehensive Courses
  - Java Programming Nanodegree by Udacity
  - Core Java Specialization by LearnQuest on Coursera
  - Java Masterclass 2025: 130+ Hours of Expert Lessons by Tim Bulchka on Udemy
  - Introduction to Object-Oriented Programming with Java by Georgia Tech on edX

Suggested Course Practical List:

Sr. No.	Practical Outcomes (PrOs)	Unit No.	Approx. Hrs. required
1	Install JDK, write a simple "Hello World" program, compile, and execute using Java compiler and interpreter.	I	1
2	Write a Java program to implement basic arithmetic operations and demonstrate type casting.	I	1
3	Create a Java program that demonstrates the use of decision-making and loop statements.	I	1
4	Develop a program to perform operations on one-dimensional and two-dimensional arrays.	I	1
5	Write a program that demonstrates the use of enhanced for-loop with arrays and simple lambda expressions.	I	1
6	Create a class to represent a Student with appropriate attributes and methods, then instantiate and manipulate Student objects.	II	1
7	Implement a class with proper encapsulation using private attributes and public getter/setter methods.	II	1
8	Create a program that demonstrates the use of constructors and 'this' keyword.	II	1
9	Develop a program to demonstrate method overloading for different operations.	II	1
10	Implement a class with static methods, variables, and blocks, then demonstrate their behaviour.	II	1
11	Create a program to demonstrate single, multilevel, and hierarchical inheritance.	III	1
12	Implement method overriding and demonstrate dynamic method dispatch.	III	1
13	Develop a program with an abstract class containing both abstract and concrete methods.	III	1
14	Create and implement interfaces for multiple inheritance and define default methods.	III	1
15	Develop a program that organizes classes in packages and demonstrates import statements.	III	1
16	Implement exception handling using try, catch, and finally blocks for different scenarios.	IV	1
17	Create custom exceptions and demonstrate their usage in handling specific error conditions.	IV	1
18	Develop a multithreaded application by extending Thread class to perform concurrent tasks.	IV	1
19	Implement a multithreaded program using Runnable interface and demonstrate thread states.	IV	1

20	Create a program that demonstrates thread synchronization and inter-thread communication.	IV	1
21	Develop a program to demonstrate String class methods and string manipulation operations.	V	1
22	Implement a program using ArrayList and LinkedList to demonstrate List interface capabilities.	V	1
23	Create an application that utilizes HashSet and HashMap for efficient data management.	V	1
24	Develop a program to read, write, and manipulate text files using file streams.	V	1
25	Implement object serialization to save and retrieve object states from files.	V	2
26	Develop a program that demonstrates serialization of collection objects.	V	2
27	Create a simple CRUD application combining collections, file handling, and exception handling.	Multiple	2
	Total		**30**

List of Laboratory/Learning Resources Required:

Sr. No.	Equipment Name with Broad Specifications	PrO. No.
1	Computer with latest configuration with Windows/Linux/Unix Operating System.	All
2	JDK (Java Development Kit) Version 11 or above	All
3	Any IDE - Eclipse, IntelliJ IDEA, NetBeans, Visual Studio Code	All

Suggested Project List:

A suggestive list of micro-projects is given here. Similar micro-projects could be added by the concerned course teacher:

1. **Library Management System**: Implement a system to manage books, members, and borrowing with proper data validation, exception handling, and file-based persistence.
2. **Student Information System**: Create an application that maintains student records, course enrolments, and grade calculations using inheritance for different student types.
3. **E-commerce Cart Management**: Develop a console-based shopping cart system with product categories (using inheritance), order management, and data persistence.
4. **Task Management Application**: Build a task tracker with task categories, priorities, deadlines, and status updates using collections and file handling.
5. **Quiz Management System**: Create an interactive quiz application with question types (using inheritance), scoring, timer functionality (using threads), and result storage.
6. **Personal Finance Manager**: Develop an application to track income, expenses, and budgets, with data visualization, categorization, and persistence.

7. **Address Book Application**: Implement a contact management system with search, sort, and filter capabilities using collections and file handling.

Suggested Activities for Students:

1. **Code Review Sessions**: Organize peer code review sessions where students can present their code and receive constructive feedback from their classmates. This helps in improving coding practices and learning from each other.
2. **Hackathons**: Conduct hackathons where students can work on real-world projects in teams. This encourages collaboration, problem-solving, and application of Java concepts in practical scenarios.
3. **Guest Lectures**: Invite industry experts to give guest lectures on advanced Java topics, current trends, and best practices. This provides students with insights into the professional world and emerging technologies.
4. **Coding Challenges**: Introduce weekly coding challenges that focus on specific Java concepts. These challenges can be solved individually or in groups, promoting continuous learning and practice.
5. **Project Showcases**: Organize project showcase events where students can present their completed projects to the class. This fosters a sense of accomplishment and allows students to learn from each other's work.
6. **Online Forums**: Create online forums or discussion boards where students can ask questions, share resources, and discuss Java-related topics. This provides a platform for continuous engagement and support.

* * * * * * *