CS23322 - Object Oriented Programming-Lab Manual

II/ECE/3rd Semester

Name of the Student:

Register Number:

Class/Sec:

Vision of the Institute

- To set a benchmark in the field of engineering education by providing quality technical education that fosters the spirit of learning, research and globally competent professionalism.

Mission of the Institute

- To impart education that caters to the growing challenges of the industry and social needs of our nation.
- To constantly upgrade the standards of teaching and learning in the field of engineering and technology while promoting a healthy research atmosphere.
- To foster a healthy symbiosis with the industry through meaningful and dynamic interactions.

Vision of the Department

- To initiate high quality technical education and to nurture young minds towards creative thinking that inspires them to undertake innovations in the field of Electronics and Communication Engineering (ECE) and be competent in the global arena.

Mission of the Department

- Constantly upgrade engineering pedagogy that caters to the growing challenges of the Industry.
- Develop conceptual learning that leads towards critical and innovative thinking.
- Establish good harmony with industry that fills the gap between academia and the outside world enabling the students to prepare for diverse and competitive career paths.
- To endorse higher studies and pursue research in the ECE discipline with sensitivity towards societal requirements.

PROGRAMME EDUCATIONAL OBJECTIVES (PEO)

- PEO I: To enable graduates to pursue research, or have a successful career in academia or industries associated with Electronics and Communication Engineering, or as entrepreneurs.
- PEO II: To develop skills for applying necessary computing techniques and communication tools among the students, for catering to the current industrial needs in broader domain of Electronics and Communication Engineering.
- PEO III: To inculcate ethical values, leadership qualities and team spirit for promoting innovations and entrepreneurship qualities among students in addressing societal concerns.

PROGRAMME OUTCOMES (PO)

Engineering Graduates will be able to:

- Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 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.
- 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.
- 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.
- Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- 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.

- Environment and Sustainability: Understand the impact of the professional engineering solutions to societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 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.
- 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.
- Lifelong learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

At the end of the programme, a student will be able to:

PSO1: To analyze, design and develop solutions by applying foundational concepts of electronics and communication engineering.

PSO2: To apply design principles and best practices for developing quality products for scientific and business applications.

PSO3: To adapt to emerging information and communication technologies (ICT) to innovate ideas and solutions to existing/novel problems.

CS23322 OBJECT ORIENTED PROGRAMMING LABORATORY

COURSE OBJECTIVES:

- To develop foundational programming skills through algorithm implementation.
- To understand and implement inheritance and polymorphism, utilizing data structures and collections.
- To develop skills in exception handling, file management, multithreading, and generic programming.
- To acquire knowledge and skills in Java networking and JDBC.
- To gain proficiency in GUI programming using Swing.

LIST OF EXERCISES:

Simple Exercises

1. Find the sum of all numbers from 1 to 100.
2. Develop a program that takes user input for a number and prints whether it's prime or not.
3. Fibonacci sequence up to the nth term using recursion.

Classes and Objects

4. Define a class "Car" with attributes like model, color, and methods to start and stop the car.
5. Implement inheritance by creating a base class "Shape" and derived classes like "Circle" and "Rectangle."
6. Develop a program that uses interfaces to model a simple banking system with classes like "Account" and "Transaction."

Exception Handling

7. Write a Java program to handle exceptions for dividing a number by zero.
8. Implement a multi-threaded program to simulate a race between two threads.
9. Develop a generic class for a binary tree and implement depth-first and breadth-first traversal algorithms.
10. Create a simple Java program to establish a client-server connection using sockets.

Application

11. Develop a Java application to connect to a MySQL database and retrieve information using JDBC.
12. Implement a multi-threaded server using Java RMI to handle concurrent client requests.
13. Design a simple GUI application to convert temperature between Celsius and Fahrenheit.
14. 14. Create a JavaFX application for a basic media player with play, pause, and stop functionalities.
15. Develop an interactive graphical application using JavaFX that involves real-time data visualization.

COURSE OUTCOMES:

At the end of the course, the students will be able to:

- CO1: Construct programs with effective control structures for decision-making and looping.
- CO2: Formulate programs using inheritance and polymorphism concepts, utilizing data structures and collections.
- CO3: Devise programs with effective exception handling, file management, multithreading, and generic programming.
- CO4: Establish Java programs that communicate over a network, and connect to databases using JDBC.
- CO5: Generate Graphical User Interfaces (GUIs) using Swing in Java.

COs-POs & PSOs MAPPING

CO	PO												PSO		
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
1	3	3	3	2	2	2	2	2	3	3	2	2	2	2	2
2	3	3	3	2	3	1	2	2	2	2	1	2	3	2	1
3	1	2	3	2	3	1	1	2	2	2	2	3	2	3	1
4	2	2	2	3	2	2	1	2	3	2	2	2	1	1	2
5	2	1	1	1	1	2	1	1	1	1	2	1	1	1	3
Avg.	2.2	2.2	2.4	2.0	2.2	1.6	1.4	1.8	2.2	2.0	1.8	2.0	1.8	1.8	1.8

1 - Low, 2 - Medium, 3 - High, '-' - No correlation

Library Management System - Project Based Learning

1. **Find the sum of all numbers from 1 to 100.**
   - **Library System Application**: Initialize book inventory with 100 books.
   - **Concept**: Simple calculations to get started with basic coding.
2. **Develop a program that takes user input for a number and prints whether it's prime or not.**
   - **Library System Application**: Check if a given ISBN number is prime (for fun/validation).
   - **Concept**: User input handling and basic algorithm.
3. **Fibonacci sequence up to the nth term using recursion.**
   - **Library System Application**: Generate a unique ID for books using a modified Fibonacci series.
   - **Concept**: Recursion and unique identification.
4. **Define a class "Car" with attributes like model, color, and methods to start and stop the car.**
   - **Library System Application**: Define a class "Book" with attributes like title, author, and methods to borrow and return the book.
   - **Concept**: Classes and Objects.
5. **Implement inheritance by creating a base class "Shape" and derived classes like "Circle" and "Rectangle."**
   - **Library System Application**: Implement inheritance by creating a base class "LibraryItem" and derived classes like "Book" and "Magazine."
   - **Concept**: Inheritance.
6. **Develop a program that uses interfaces to model a simple banking system with classes like "Account" and "Transaction."**
   - **Library System Application**: Use interfaces to model library transactions with classes like "Borrowable" and "Reservable."
   - **Concept**: Interfaces and polymorphism.
7. **Write a Java program to handle exceptions for dividing a number by zero.**
   - **Library System Application**: Handle exceptions for scenarios like borrowing a book that is not available.
   - **Concept**: Exception handling.
8. **Implement a multi-threaded program to simulate a race between two threads.**
   - **Library System Application**: Simulate multiple users borrowing books simultaneously using multi-threading.
   - **Concept**: Multi-threading.

9. **Develop a generic class for a binary tree and implement depth-first and breadth-first traversal algorithms.**
    - **Library System Application**: Create a generic class for a catalog tree and implement search algorithms to find books.
    - **Concept**: Generic classes and tree traversal.
10. **Create a simple Java program to establish a client-server connection using sockets.**
    - **Library System Application**: Establish a client-server connection for remote access to the library database.
    - **Concept**: Networking and sockets.
11. **Develop a Java application to connect to a MySQL database and retrieve information using JDBC.**
    - **Library System Application**: Connect to a library database using JDBC to retrieve and update book information.
    - **Concept**: Database connectivity.
12. **Implement a multi-threaded server using Java RMI to handle concurrent client requests.**
    - **Library System Application**: Implement a multi-threaded library server using Java RMI to handle multiple user requests concurrently.
    - **Concept**: Remote Method Invocation (RMI) and concurrency.
13. **Design a simple GUI application to convert temperature between Celsius and Fahrenheit.**
    - **Library System Application**: Design a simple GUI for the library system to add, search, and manage books.
    - **Concept**: GUI development.
14. **Create a JavaFX application for a basic media player with play, pause, and stop functionalities.**
    - **Library System Application**: Create a JavaFX application for the library system to manage book details with functionalities like add, update, and delete.
    - **Concept**: JavaFX and GUI functionalities.
15. **Develop an interactive graphical application using JavaFX that involves real-time data visualization.**
    - **Library System Application**: Develop a JavaFX application for real-time visualization of library data, such as borrowing trends and book availability.
    - **Concept**: Data visualization and real-time updates.

1. Program Title: Sum of Numbers from 1 to 100

Aim: To calculate the sum of all numbers from 1 to 100.
Algorithm:

1. Initialize sum = 0.
2. Loop i from 1 to 100.
3. Add i to sum in each iteration.
4. Print the value of sum.

Test Cases:

- TC1: Sum from 1 to 100 → Expected: 5050
- TC2: Sum from 1 to 50 → Expected: 1275
- TC3: Sum from 1 to 10 → Expected: 55
- TC4: Sum from 50 to 100 → Expected: 3825
- TC5: Sum from 1 to 1 → Expected: 1

MCQs:

1. What is the result of adding numbers from 1 to 100?
   a) 5000
   b) 5050
   c) 5100
   d) 4950

2. What is the formula to calculate the sum of the first N natural numbers?
   a) N * (N + 1)
   b) N + (N - 1)
   c) N * (N + 1) / 2
   d) $N^2 + 1$

3. Which loop can be used to perform the addition?
   a) if
   b) for
   c) switch
   d) try

4. What data type is suitable for storing the result?
   a) char
   b) String
   c) int
   d) boolean

5. What is the time complexity of this approach?
   a) O(1)
   b) O(N)
   c) O(N^2)
   d) O(log N)

6. What keyword is used to declare a variable in Java?
   a) define
   b) var
   c) int
   d) let

7. How many times will the loop run from 1 to 100?
   a) 99
   b) 100
   c) 101
   d) 50

8. Which symbol is used for addition in Java?
   a) -
   b) =
   c) +
   d) *

9. What will be the output if the loop starts from 0 to 100?
   a) 5050
   b) 5000
   c) 5150

    d) 4950

10. Which of these can replace looping to compute the sum more efficiently?
    a) Recursion
    b) Inheritance
    c) Mathematical formula
    d) Interfaces

Result: Displays the correct sum as 5050.

---

2. Program Title: Prime Number Checker

Aim: To determine whether a user-input number is prime or not.
Algorithm:

1. Input number n.
2. If n <= 1, it's not prime.
3. Check divisibility from 2 to sqrt(n).
4. If divisible by any, it's not prime.
5. Else, it is a prime number.

Test Cases:

- TC1: Input 2 → Output: Prime
- TC2: Input 15 → Output: Not Prime
- TC3: Input 17 → Output: Prime
- TC4: Input 1 → Output: Not Prime
- TC5: Input 0 → Output: Not Prime

MCQs:

1. Which of the following numbers is a prime?
   a) 4
   b) 6
   c) 7
   d) 9

2. What is the square root of 16?
   a) 2
   b) 3
   c) 4

d) 5

3. Which loop is best for checking primes up to N?
   a) while
   b) do-while
   c) for
   d) switch

4. Which number is not considered a prime?
   a) 1
   b) 2
   c) 3
   d) 5

5. What is the remainder when 13 is divided by 2?
   a) 5
   b) 1
   c) 3
   d) 0

6. What is the return type of a method checking for prime?
   a) int
   b) void
   c) boolean
   d) float

7. Can negative numbers be prime?
   a) Yes
   b) No
   c) Sometimes
   d) Only zero

8. What will be the result for 19?
   a) Prime
   b) Not Prime
   c) Composite
   d) Even

9. Which of these methods is used for integer input in Java?
   a) nextLine()

  b) nextFloat()
  c) nextInt()
  d) nextChar()

10. What is the range for checking divisibility?
  a) 2 to n
  b) 2 to n/2
  c) 2 to sqrt(n)
  d) 1 to n

Result: Outputs whether the input number is prime or not.

---

3. Program Title: Fibonacci Sequence using Recursion

Aim: To print the Fibonacci sequence up to the nth term using recursion.

Algorithm:

1. Take input n.
2. Define recursive function:
   - fib(0) = 0
   - fib(1) = 1
   - fib(n) = fib(n-1) + fib(n-2)
3. Loop from 0 to n and print each fib(i).

Test Cases:

- TC1: n = 5 → Output: 0 1 1 2 3
- TC2: n = 7 → Output: 0 1 1 2 3 5 8
- TC3: n = 1 → Output: 0
- TC4: n = 0 → Output:
- TC5: n = 10 → Output: 0 1 1 2 3 5 8 13 21 34

MCQs:

1. What is recursion?
   a) A loop
   b) A function calling itself
   c) A method from superclass
   d) None
2. Which of these is the base case for Fibonacci?
   a) fib(n) = n
   b) fib(0) = 0

c) fib(n-1)
d) fib(n) = fib(n-1)

3. What is fib(3)?
   a) 3
   b) 2
   c) 1
   d) 0

4. What happens without a base case?
   a) Compilation error
   b) Stack overflow
   c) Loop runs forever
   d) Nothing

5. What is the main issue with recursive Fibonacci?
   a) Accuracy
   b) Readability
   c) Efficiency
   d) Scope

6. What data type does fib return?
   a) float
   b) String
   c) int
   d) double

7. What is the time complexity of recursive Fibonacci?
   a) O(1)
   b) O(n)
   c) $O(n^2)$
   d) O(2^n)

8. What is fib(5)?
   a) 5
   b) 3
   c) 8
   d) 4

9. What keyword is used to define a method in Java?
   a) define
   b) void
   c) method
   d) static

10. What's the output for fib(6)?
   a) 5
   b) 6
   c) 8
   d) 13

Result: Prints Fibonacci sequence correctly up to n terms.

---

4. Program Title: Class "Car" with Methods

Aim: To define a class Car with attributes like model, color and methods to start and stop the car.
Algorithm:

1. Define a class Car.
2. Add attributes: model, color.
3. Define constructor to initialize these values.
4. Define start() and stop() methods.
5. Create object and invoke methods.

Test Cases:

- TC1: Car("Tesla", "Red") → Output: Car started
- TC2: Car("BMW", "Black") → Output: Car started
- TC3: Car model empty → Output: Car started (but model not shown)
- TC4: Start method not called → Output: Nothing
- TC5: Call stop method only → Output: Car stopped

MCQs:

1. What is a class in Java?
   a) A variable
   b) A function
   c) A blueprint for objects
   d) A loop

2. What keyword is used to define a class?
   a) object
   b) method
   c) class

d) define

3. How do you create an object of a class?
   a) new Car()
   b) Car.create()
   c) class Car()
   d) object(Car)

4. What is a constructor?
   a) Special method to destroy object
   b) Loop inside a class
   c) Method that initializes object
   d) A keyword in Java

5. What is the access specifier commonly used for attributes?
   a) public
   b) private
   c) static
   d) final

6. Which of the following is not a valid method name?
   a) start
   b) stop
   c) 123method
   d) drive

7. What is the output when start method is called?
   a) Car started
   b) Car stopped
   c) Null
   d) None

8. How do you reference current object in Java?
   a) self
   b) that
   c) this
   d) me

9. How many constructors can a class have?
   a) 1

b) Many
c) Only 2
d) Only 3

10. What does void mean in method definition?
    a) Returns a value
    b) Takes parameters
    c) Does not return a value
    d) Is a constructor

Result: Program creates a Car object and calls its methods successfully.

---

5. Program Title: Inheritance with Shape-Circle-Rectangle

Aim: To implement inheritance using a base class Shape and derived classes Circle and Rectangle.

Algorithm:

1. Create base class Shape.
2. Add method area() in Shape.
3. Create derived classes Circle and Rectangle.
4. Override area() in both.
5. Instantiate objects and call respective methods.

Test Cases:

- TC1: Circle with radius 5 → Area ≈ 78.5
- TC2: Rectangle 10x5 → Area = 50
- TC3: Shape area() not overridden → Output: Default area
- TC4: Zero radius circle → Output: Area = 0
- TC5: Rectangle with width 0 → Area = 0

MCQs:

1. What is inheritance in Java?
   a) Loop concept
   b) One class acquiring another
   c) Overloading functions
   d) Writing multiple main methods

2. Which keyword is used for inheritance?
   a) extend
   b) inherit
   c) extends

d) superclass

3. What is method overriding?
   a) Changing return type
   b) Redefining method in subclass
   c) Creating two methods with same name
   d) Creating private method

4. What does super refer to?
   a) Parent class
   b) Next class
   c) Local variable
   d) Final class

5. Can we create object of abstract class?
   a) Yes
   b) No
   c) Sometimes
   d) Only if final

6. Which class is at the top of Java class hierarchy?
   a) Shape
   b) Object
   c) Class
   d) Main

7. What is the output of Circle area with radius 0?
   a) 1
   b) Error
   c) 0
   d) Null

8. Which of these is not a derived class?
   a) Circle
   b) Rectangle
   c) Shape
   d) Ellipse

9. What is function of @Override annotation?
   a) Overrides final method

b) Hides method
c) Indicates method overrides parent method
d) Makes class final

10. Which of these is not a type of inheritance in Java?
    a) Single
    b) Multilevel
    c) Hybrid
    d) Interface

Result: Inheritance implemented successfully; area() works for each derived class.

6. Program Title: Interface for Banking System

Aim: To model a banking system using interfaces with classes Account and Transaction.

Algorithm:

1. Define interface BankingService with abstract methods.
2. Implement Account class for deposit/withdraw.
3. Implement Transaction class for transaction logs.
4. Create objects and call respective methods.
5. Show polymorphic behavior.

Test Cases:

- TC1: Deposit ₹1000 → Balance = ₹1000
- TC2: Withdraw ₹500 from ₹1000 → Balance = ₹500
- TC3: Withdraw ₹1500 from ₹1000 → Output: Insufficient funds
- TC4: Deposit negative amount → Output: Invalid
- TC5: Log transaction → Output: Transaction recorded

MCQs:

1. What is an interface in Java?
   a) Class with implementation
   b) Class with variables
   c) Collection of abstract methods
   d) Main method

2. How do you declare an interface?
   a) interface
   b) class
   c) extends

d) abstract

3. Can an interface have constructors?
   a) Yes
   b) No
   c) Only default
   d) Only static

4. What keyword is used to implement an interface?
   a) inherit
   b) extends
   c) implements
   d) abstract

5. Which of these is true about interface methods?
   a) They are private
   b) They are static
   c) They are public and abstract
   d) They return void

6. Can a class implement multiple interfaces?
   a) No
   b) Yes
   c) Only one
   d) Only two

7. What will be the result of invalid deposit?
   a) Accepted
   b) Error
   c) Invalid input message
   d) Terminated

8. What is polymorphism in the context of interfaces?
   a) Class calling another class
   b) One method, many forms
   c) Only one object
   d) Static behavior

9. Which of these can be part of an interface?
   a) Constructors

b) Static blocks
c) Final variables
d) Private methods

10. How do you log a transaction in a class?
    a) System.out.print
    b) Call a method
    c) Use Scanner
    d) Use database

Result: Banking operations and logging through interfaces work as expected.

---

7. Program Title: Exception Handling for Division by Zero

Aim: To implement exception handling in Java when a number is divided by zero.
Algorithm:

1. Take two integers as input.
2. Use try block to divide them.
3. Catch ArithmeticException if divisor is zero.
4. Display appropriate message.
5. Print result if division is valid.

Test Cases:

- TC1: 10 / 2 → Output: 5
- TC2: 20 / 0 → Output: Exception handled
- TC3: -5 / 5 → Output: -1
- TC4: 0 / 3 → Output: 0
- TC5: 100 / -10 → Output: -10

MCQs:

1. Which keyword is used to handle exceptions?
   a) catch
   b) try
   c) finally
   d) All of the above

2. What is the exception thrown when dividing by zero?
   a) IOException
   b) NullPointerException
   c) ArithmeticException

d) ArrayIndexOutOfBoundsException

3. What block is executed after try-catch?
   a) final
   b) default
   c) finally
   d) exit

4. Can a program have multiple catch blocks?
   a) No
   b) Yes
   c) Only one
   d) Only inside loops

5. What happens when exception is not caught?
   a) Compile error
   b) Skipped
   c) Program crashes
   d) Ignored

6. Which is the superclass for all exceptions?
   a) Error
   b) Throwable
   c) Exception
   d) RuntimeException

7. What is the result of 10 / 0?
   a) 10
   b) Infinity
   c) ArithmeticException
   d) 0

8. Can exceptions be nested?
   a) No
   b) Yes
   c) Only in classes
   d) Only in constructors

9. What is the purpose of finally block?
   a) Handle logic

    b) Force execution
    c) Exit loop
    d) Replace catch

10. What type of exception is divide-by-zero?
    a) Checked
    b) Unchecked
    c) Compile-time
    d) Syntax

Result: Program gracefully handles division by zero and displays a custom message.

---

8. Program Title: Multi-threaded Race Simulation

Aim: To simulate a race between two threads using Java multithreading.
Algorithm:

1. Create a class that extends Thread.
2. Define run() method to simulate steps in race.
3. Create two thread objects.
4. Start both threads simultaneously.
5. Use sleep() to show progress step-by-step.

Test Cases:

- TC1: Two threads start → Both race together
- TC2: Sleep time different → One finishes first
- TC3: Race between thread A and thread B → Output may vary
- TC4: Add a third thread → Output includes three racers
- TC5: Threads with same sleep → Output may interleave

MCQs:

1. Which method starts a thread in Java?
   a) run()
   b) start()
   c) execute()
   d) call()

2. What is multithreading?
   a) One task
   b) Many classes
   c) Concurrent execution

d) One method

3. What class is used for threading?
   a) Thread
   b) Object
   c) Runnable
   d) Both a and c

4. What does sleep() method do?
   a) Ends thread
   b) Pauses thread
   c) Kills JVM
   d) Skips steps

5. Which method is overridden in thread class?
   a) start()
   b) sleep()
   c) run()
   d) stop()

6. What happens if start() is called twice?
   a) Runs twice
   b) Throws exception
   c) Ignores second call
   d) Resumes thread

7. Which interface must be implemented for threading?
   a) Serializable
   b) Runnable
   c) Comparable
   d) Cloneable

8. Can we use multiple threads for one task?
   a) No
   b) Yes
   c) Only for loops
   d) Only for files

9. What is the main use of multithreading?
   a) Delay

b) Exception
c) Concurrency
d) Debugging

10. What is the output if one thread finishes early?
    a) Stopped
    b) Interruption
    c) Random
    d) Varies

Result: Threads race concurrently and demonstrate interleaved output simulating competition.

---

9. Program Title: Generic Binary Tree with Traversals

Aim: To implement a generic class for binary tree with DFS and BFS traversals.
Algorithm:

1. Define a generic TreeNode<T> class.
2. Implement insert() method.
3. Add inOrder(), preOrder(), postOrder() for DFS.
4. Use queue-based levelOrder() for BFS.
5. Test all traversal methods on sample tree.

Test Cases:

- TC1: Insert 10, 5, 20 → All methods return correctly
- TC2: Level order → Output: 10 5 20
- TC3: In-order → Output: 5 10 20
- TC4: Pre-order → Output: 10 5 20
- TC5: Post-order → Output: 5 20 10

MCQs:

1. What is a binary tree?
   a) Two-rooted tree
   b) Tree with max two children
   c) Tree with only leaves
   d) Unordered data

2. What is generic programming in Java?
   a) Dynamic typing
   b) Abstract methods
   c) Code that works with any type

d) Static binding

3. What does in-order traversal return for 10-5-20?
   a) 10 5 20
   b) 5 10 20
   c) 20 10 5
   d) 10 20 5

4. What data structure is used in BFS?
   a) Stack
   b) Queue
   c) Heap
   d) Array

5. Which traversal visits root first?
   a) Pre-order
   b) In-order
   c) Post-order
   d) BFS

6. Can a binary tree have duplicates?
   a) Yes
   b) No
   c) Only strings
   d) Only leaves

7. What type is used in generics?
   a) T
   b) K
   c) V
   d) All

8. What class helps implement queue in Java?
   a) Stack
   b) ArrayList
   c) LinkedList
   d) HashMap

9. In post-order, which is printed last?
   a) Root

    b) Leaf
    c) Left
    d) Right

10. What is output of DFS if tree has one node?
    a) Null
    b) That node
    c) Root then leaf
    d) Error

Result: Binary tree operations and traversals work correctly using generic implementation.

---

10. Program Title: Client-Server Connection using Sockets

Aim: To establish a simple client-server connection in Java using socket programming.
Algorithm:

1. Create a ServerSocket on a specific port.
2. Accept connection from a Socket object on the client side.
3. Exchange messages via InputStream and OutputStream.
4. Display data received on both ends.
5. Close connection after communication.

Test Cases:

- TC1: Server online, client connects → Output: Connection successful
- TC2: Server not running → Output: Connection refused
- TC3: Client sends message → Server receives correctly
- TC4: Server sends back reply → Client displays reply
- TC5: Multiple clients → Server handles sequentially

MCQs:

1. What package supports socket programming?
   a) java.util
   b) java.io
   c) java.net
   d) java.lang

2. What class is used for creating a server?
   a) Socket
   b) ServerSocket
   c) DatagramSocket

d) ClientSocket

3. Which method waits for a client connection?
   a) listen()
   b) open()
   c) accept()
   d) run()

4. What port range is valid for TCP/IP?
   a) 0 to 65535
   b) 1 to 1024
   c) 1025 to 2048
   d) 1 to 100

5. What exception is common in sockets?
   a) IOException
   b) SQLException
   c) FileNotFoundException
   d) ThreadException

6. Which method closes a socket?
   a) disconnect()
   b) terminate()
   c) shutdown()
   d) close()

7. What is the client's IP address in localhost testing?
   a) 127.0.0.1
   b) 255.255.255.0
   c) 0.0.0.0
   d) 192.168.0.1

8. What happens if client connects before server?
   a) Connection succeeds
   b) Error: Connection refused
   c) Timeout
   d) Crash

9. What is the default transport protocol for Java sockets?
   a) UDP

b) HTTP
c) TCP
d) SMTP

10. Which stream is used to receive data?
    a) OutputStream
    b) InputStream
    c) PrintStream
    d) FileInputStream

Result: Client and server communicate successfully using sockets.

---

11. Program Title: JDBC Connection with MySQL

Aim: To develop a Java application that connects to MySQL and retrieves data using JDBC.
Algorithm:

1. Load JDBC driver.
2. Establish connection using DriverManager.
3. Create Statement or PreparedStatement.
4. Execute query to fetch records.
5. Display result and close resources.

Test Cases:

- TC1: Connect to DB successfully → Output: Connected
- TC2: Execute SELECT query → Output: Records displayed
- TC3: Invalid credentials → Output: Access denied
- TC4: Execute empty query → Output: No results
- TC5: Malformed SQL → Output: SQLSyntaxErrorException

MCQs:

1. What package is needed for JDBC?
   a) java.sql
   b) java.io
   c) java.db
   d) javax.net

2. Which class is used to open a DB connection?
   a) Statement
   b) ResultSet
   c) DriverManager

d) SQLExecutor

3. Which method executes SELECT queries?
   a) executeUpdate()
   b) execute()
   c) executeQuery()
   d) executeFetch()

4. What is returned by executeQuery()?
   a) Boolean
   b) ResultSet
   c) Connection
   d) Integer

5. What does next() method of ResultSet do?
   a) Moves to next record
   b) Closes query
   c) Ends connection
   d) Commits data

6. What type of statement prevents SQL injection?
   a) Statement
   b) PrintStream
   c) PreparedStatement
   d) InputStream

7. Which driver is commonly used for MySQL?
   a) jdbc.oracle.driver
   b) jdbc.mysql.Driver
   c) mysql.jdbc.Driver
   d) com.mysql.cj.jdbc.Driver

8. What exception is thrown for SQL errors?
   a) SQLException
   b) IOException
   c) RuntimeException
   d) ClassNotFoundException

9. What is the port number for MySQL by default?
   a) 1521

b) 3306
c) 1433
d) 8080

10. Which method closes the connection?
    a) exit()
    b) shut()
    c) close()
    d) terminate()

Result: Successfully connects to database and displays queried data.

---

12. Program Title: Multi-threaded Server using Java RMI

Aim: To implement a multi-threaded server using Java RMI to handle multiple clients.
Algorithm:

1. Create remote interface with method signatures.
2. Implement interface in server class.
3. Register server object in RMI registry.
4. Clients lookup and invoke methods.
5. Use threads to manage concurrent client requests.

Test Cases:

- TC1: Start server, connect client → Success
- TC2: Client sends request → Server responds
- TC3: Multiple clients → Handled via threads
- TC4: Server not running → Lookup fails
- TC5: Invalid method call → RemoteException

MCQs:

1. What does RMI stand for?
   a) Remote Message Invocation
   b) Remote Method Invocation
   c) Runtime Method Interface
   d) Remote Memory Interface

2. Which package supports RMI in Java?
   a) java.rmi
   b) java.net
   c) java.util

d) javax.rmi

3. What method is used to bind the remote object?
   a) register()
   b) bind()
   c) connect()
   d) attach()

4. Which class runs the RMI registry?
   a) Registry
   b) LocateRegistry
   c) RMIStarter
   d) RMIConnector

5. What exception is thrown on network failure?
   a) IOException
   b) SQLException
   c) RemoteException
   d) RMIException

6. How is the server object made remotely accessible?
   a) Implement Runnable
   b) Extend Thread
   c) Extend UnicastRemoteObject
   d) Extend RemoteServer

7. What interface must remote class implement?
   a) Serializable
   b) Runnable
   c) Remote
   d) Cloneable

8. Which method allows a client to call a remote object?
   a) send()
   b) run()
   c) lookup()
   d) connect()

9. Can RMI handle multithreading?
   a) No

b) Yes
c) Only with sockets
d) Only with GUIs

10. What command starts RMI registry?
    a) javac
    b) rmic
    c) rmiregistry
    d) javadoc

Result: Server handles concurrent client requests using RMI successfully.

---

13. Program Title: GUI for Temperature Conversion

Aim: To develop a simple GUI to convert temperatures between Celsius and Fahrenheit.
Algorithm:

1. Design GUI with input field, buttons, and result label.
2. Add "Convert to Celsius" and "Convert to Fahrenheit" buttons.
3. On button click, perform respective conversion.
4. Display result in label.
5. Handle invalid or empty inputs gracefully.

Test Cases:

- TC1: Input 100°F → Output: 37.78°C
- TC2: Input 0°C → Output: 32°F
- TC3: Input -40°C → Output: -40°F
- TC4: Blank input → Output: Error/Warning
- TC5: Input string "abc" → Output: Invalid input

MCQs:

1. Which Java package is commonly used for GUI?
   a) java.awt
   b) java.sql
   c) java.rmi
   d) java.lang

2. What is used to create buttons in Swing?
   a) JTextArea
   b) JButton
   c) JDialog

d) JFrame

3. What layout allows components in rows?
   a) BorderLayout
   b) FlowLayout
   c) GridLayout
   d) BoxLayout

4. What event is triggered by button click?
   a) WindowEvent
   b) KeyEvent
   c) ActionEvent
   d) MouseEvent

5. What method adds a component to JFrame?
   a) insert()
   b) push()
   c) add()
   d) mount()

6. Which method reads input from a text field?
   a) getInput()
   b) next()
   c) getText()
   d) readLine()

7. Which temperature scale has a freezing point at 0?
   a) Fahrenheit
   b) Celsius
   c) Kelvin
   d) Newton

8. How do you convert Celsius to Fahrenheit?
   a) C × 2
   b) (C × 9/5) + 32
   c) (C × 5/9) + 32
   d) (C × 9/5)

9. What is used to align components in Swing?
   a) BorderSpacing

b) ComponentLayout
c) LayoutManager
d) SwingAlign

10. What happens on invalid input?
a) Crash
b) Shows result
c) Throws exception
d) Handled using try-catch

Result: User enters temperature and selects conversion type; result is displayed correctly.

---

14. Program Title: JavaFX Media Player

Aim: To create a basic media player in JavaFX with play, pause, and stop functionalities.

Algorithm:

1. Set up JavaFX application structure.
2. Load media file using Media class.
3. Use MediaPlayer to control playback.
4. Create buttons for play, pause, and stop.
5. Bind buttons to control methods.

Test Cases:

- TC1: Click Play → Output: Media starts
- TC2: Click Pause → Output: Media pauses
- TC3: Click Stop → Output: Media stops and resets
- TC4: No file loaded → Output: Error message
- TC5: Play file in unsupported format → Output: MediaException

MCQs:

1. Which class is used to represent media files in JavaFX?
   a) MediaFile
   b) Media
   c) SoundClip
   d) VideoStream

2. What controls media playback?
   a) FilePlayer
   b) AudioManager
   c) MediaPlayer

d) FileReader

3. Which format is supported by JavaFX?
   a) MP3
   b) MP4
   c) WAV
   d) All of the above

4. What layout is used for button alignment?
   a) VBox
   b) FlowPane
   c) GridPane
   d) All of the above

5. Which method starts playback?
   a) start()
   b) run()
   c) play()
   d) launch()

6. What is required to initialize JavaFX application?
   a) main()
   b) init()
   c) start(Stage)
   d) draw()

7. What does stop() method do?
   a) Exits application
   b) Stops media and resets
   c) Pauses video
   d) Closes window

8. What exception is thrown for unsupported media types?
   a) MediaException
   b) FormatException
   c) FileException
   d) InputMismatchException

9. How is the GUI initialized in JavaFX?
   a) JFrame

b) JDialog
c) Stage
d) View

10. How do you load media in JavaFX?
    a) FileReader
    b) new Media(path)
    c) new Video(path)
    d) new Audio(path)

Result: Media file plays with user control via play, pause, and stop buttons.

---

15. Program Title: Real-Time Data Visualization in JavaFX

Aim: To build an interactive JavaFX application for visualizing real-time data using graphs.
Algorithm:

1. Set up JavaFX stage and scene.
2. Create a LineChart or BarChart.
3. Simulate real-time data using thread or timeline.
4. Update chart dynamically with new values.
5. Handle UI refresh and error conditions.

Test Cases:

- TC1: App starts → Chart displays empty
- TC2: Data added every second → Graph updates in real-time
- TC3: Stop updates → Graph freezes
- TC4: Data out of range → Chart scales accordingly
- TC5: Non-numeric input → Handled gracefully

MCQs:

1. Which JavaFX class is used for plotting graphs?
   a) DataView
   b) ChartView
   c) LineChart
   d) GraphView

2. What is the base class for all charts in JavaFX?
   a) Pane
   b) Chart
   c) StackPane

d) GridPane

3. What is used to update UI repeatedly in JavaFX?
   a) Thread
   b) Timer
   c) Timeline
   d) Executor

4. What data structure is used to store chart values?
   a) List
   b) XYChart.Series
   c) Stack
   d) Map

5. What does Platform.runLater() do?
   a) Closes app
   b) Updates UI from background
   c) Stops chart
   d) Adds event handler

6. What is required to display data dynamically?
   a) EventQueue
   b) DataThread
   c) Animation Timer
   d) Timeline

7. What format is used for x and y axis labels?
   a) Strings
   b) Integers only
   c) Doubles only
   d) Dates only

8. Which event triggers chart redraw?
   a) setData()
   b) refresh()
   c) repaint()
   d) update()

9. What happens if chart has too much data?
   a) Shrinks

b) Scrolls
c) Overflows
d) Can slow down

10. What is the best JavaFX layout for charts?
a) VBox
b) HBox
c) BorderPane
d) StackPane

Result: JavaFX chart updates dynamically, effectively visualizing simulated real-time data.